百度智能云

BML 文档



【版权声明】

版权所有©百度在线网络技术（北京）有限公司、北京百度网讯科技有限公司。 未经本公司书面许可，任何单位和个人不得擅自摘抄、复制、传播本文档内容，否则本公司有权依法追究法律责任。

【商标声明】



和其他百度系商标，均为百度在线网络技术（北京）有限公司、北京百度网讯科技有限公司的商标。 本文档涉及的第三方商标，依法由相关权利人所有。未经商标权利人书面许可，不得擅自对其商标进行使用、复制、修改、传播等行为。

【免责声明】

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导。 如您购买本文档介绍的产品、服务，您的权利与义务将依据百度智能云产品服务合同条款予以具体约定。本文档内容不作任何明示或暗示的保证。

目录

目录 2
版本发布记录 6
产品简介 11
BML平台升级公告 11
什么是BML 11
产品功能 14
产品优势 15
文心大模型 16
平台重点升级介绍 17
产品定价 20
模型训练计费说明 20
公有云部署计费说明 21
通用小型设备部署价格说明 22
服务器部署价格说明 22
专项适配硬件部署价格说明 22
批量预测计费说明 23
平台管理 24
在BML平台使用容器镜像服务CCR 24
在BML使用外部镜像 31
外部镜像添加步骤 31
使用外部镜像 32
在BML平台使用并行文件系统PFS和对象存储BOS 32
镜像管理 38
镜像管理简介 38
自定义镜像 39
镜像使用 43
常见问题 44
资源管理 44
资源池管理简介 44
资源池使用简介 48
项目空间管理 51
权限管理 54
快速开始 64
用BML实现图片分类 64
用BML实现物体检测 75
用BML实现实例分割 90
用BML实现文本分类 98
用BML评价短文本相似度 105
用BML实现文本实体抽取 112
用BML实现序列标注 118

用BML实现表格预测 125
用BML实现开源大模型的预训练（Post-pretrain） 141
数据服务 146
数据服务简介 146
管理视觉数据 147
图像分类数据 147
物体检测数据 155
实例分割数据 164
管理文本数据 170
文本分类数据 170
短文本匹配数据 175
序列标注数据 177
文本实体抽取数据 182
智能标注 185
图像智能标注简介 185
文本智能标注简介 188
公有云服务调用数据反馈 192
智能数据API 194
模型训练 213
预置模型调参建模 213
预置模型调参简介 213
开发视觉模型 214
开发自然语言处理模型 232
错误码 237
开发表格预测模型 243
开发文字识别模型 250
神经网络训练搜索 254
Notebook建模 256
Notebook从训练到部署快速入门 256
Notebook简介 337
创建并启动Notebook 338
使用Notebook开发模型 340
保存Notebook中的模型 342
配置模型 344
发布模型 345
常见问题 346
Notebook使用参考 348
数据模型可视化功能说明 370
Notebook导入数据集 392
如何使用Notebook SSH 功能 393

自定义作业建模 397
自定义作业简介 397
训练作业 397
自动搜索作业 430
训练作业API 466
可视化建模 471
概述 471
快速入门 472
画布操作说明 480
组件菜单 481
组件说明 487
模型仓库 676
模型仓库简介 676
创建模型 677
从训练任务导入模型 678
从本地导入模型 679
查看模型 679
校验模型 680
服务代码文件示例 680
Sklearn服务代码文件示例 680
Sklearn服务代码文件示例 680
XGBoost服务代码文件示例 682
预测部署 683
公有云部署管理 683
公有云部署简介 683
公有云部署 684
配置休眠策略 684
配置AB测试版本 685
批量预测服务 686
视觉模型部署 687
视觉任务模型部署整体说明 687
公有云部署 688
私有服务器部署 705
设备端SDK部署 748
软硬一体方案部署 793
端云协同服务部署 843
智能边缘控制台-单节点版 849
智能边缘控制台-多节点版 859
NLP模型部署 869
自然语言处理任务模型部署整体说明 869
公有云部署 869

私有服务器部署 885
表格预测模型部署 892
整体说明 892
公有云部署 893
公有云部署简介 893
通用模型部署 897
公有云部署 897
公有云部署简介 897
Paddle框架API调用文档 899
Tensorflow框架API调用文档 900
Pytorch框架API调用文档 902
sklearn框架API调用文档 902
XGBoost框架API调用文档 904
错误码 905
通用类模型API参考 906
标准接口规范参考 906
文字识别模型部署 910
文字识别任务公有云部署 910
文字识别任务API参考文档 912
批量预测(用户资源池)API 915
语音技术模型部署 915
声音分类API调用文档 915

版本发布记录

2025年1月

序号	功能模块	功能描述
1	资源管理	支持专属资源池，管理员可在bml一站式购买专属隔离集群；支持notebook、作业建模等任务调度到专属资源池上
2	notebook	notebook支持SSH，支持内网直连
3	notebook	预置llamafactory镜像案例，可加载HF开源大模型进行调试
4	notebook	基于公共资源池创建notebook服务支持配置运行时长，防止因遗忘产生额外扣费
5	预置模型调参	xgboost升级1.6.2
6	作业建模	预置megatron加速镜像
7	模型仓库	新增模型导入失败异常提示，可辅助定位问题
8	用户资源池	K8S升级到1.24版本

2024年2月

序号	功能模块	功能描述
1	Notebook	集成codelab 1.6.5
2	作业建模	集成百舸AIAK训练加速镜像
3	作业建模	支持环境变量
4	在线服务	集成百舸AIAK推理加速镜像
5	用户资源池	用户资源池支持K8s 1.20版本
6	可视化	可视化建模支持使用个人BOS数据

2023年12月

序号	功能模块	功能描述
1	BML	BML平台支持区域（region划分），支持项目空间，角色设置和权限细分
2	模型仓库	模型仓库支持BOS导入
3	交互优化	操作台新增快速跳转至资源池管理页面入口，优化项目空间创建交互流程

2023年8月

序号	功能模块	功能描述
1	作业建模	作业建模日志支持下载
2	操作台	BML前端控制台迁入百度智能云console
3	批量预测	批量预测支持提交至用户资源池

2023年7月

序号	功能模块	功能描述
1	资源池管理	支持一键挂载PFS实例
2	资源池管理	镜像仓库支持添加外部镜像
3	作业建模	新增支持灵活使用BOS存储和PFS存储数据源提交任务，点击了解
4	镜像管理模块	"环境管理"功能升级，并变更为"镜像管理"。支持构建的镜像用于作业建模、在线服务、批量预测

2023年4月

序号	功能模块	功能描述
1	资源池管理	新增PFS引入
2	作业建模	新增支持使用PFS存储数据源提交任务

2023年3月

序号	功能模块	功能描述
1	作业建模	新增新增支持从CCR镜像仓库拉取镜像提交作业建模任务，点击了解
2	作业建模	新增支持BOS token的renew操作
3	作业建模	新增支持开放训练作业API，通过销售申请即可使用

2023年2月

序号	功能模块	功能描述
1	作业建模	使用用户资源池时，支持调用RDMA网络能力（需要用户资源池具备RDMA能力）

2022年12月

序号	功能模块	功能描述
1	资源池管理	新增资源池管理功能，支持用户挂载自有CCE集群，并可用于作业建模、可视化建模任务

2022年11月

序号	功能模块	功能描述
1	飞桨套件专区	新增飞桨模型库
2	Codelab Notebook	新增飞桨套件镜像配置环境
3	预置模型调参	新建任务流程优化
4	平台功能	填写个人信息及创建任务流程优化

2022年9月

序号	功能模块	功能描述
1	环境管理	新增支持环境管理模块，支持用户管理自己的训练环境以及自定义新的训练环境
2	Codelab Notebook	新增支持自定义镜像部署功能，用户可以在Notebook中的通用类型使用自定义的环境来进行模型的部署
3	Codelab Notebook	新增较大数据文件上传路径，支持用户快速上传5G以内的大文件
4	Codelab Notebook	创建Notebook任务时允许用户选择多套件镜像环境
5	飞桨套件专区	新增飞桨套件专区

2022年8月

序号	功能模块	功能描述
1	预置模型调参	图像分类单标签新增InceptionV4、DPN131模型
2	预置模型调参	实例分割新增支持MaskRCNN+ResNet101_vd、SOLOv2+R50-FPN、Cascade+ResNet50vd模型
3	预置模型调参	新增语义分割方向，提供PPLiteSeg、FastSCNN、STDCSeg等一系列语义分割模型

2022年7月

序号	功能模块	功能描述
1	可视化建模	平台提供可视化的实验开发环境，开发人员和业务人员根据场景和业务需求能够在交互式画布上直观地连接数据处理、特征工程，算法，模型预测和模型评估等组件，基于无代码方式实现人工智能模型开发。可视化建模在降低模型开发门槛的同时提升了建模的效率。内置数百个成熟的机器学习算法，支持多种算法框架，覆盖了机器学习和深度学习场景，满足用户不同程度的需求。点击了解详情

2022年5月

序号	功能模块	功能描述
1	Codelab Notebook	原Notebook建模更名为Codelab Notebook，并新增飞桨行业、官方Notebook案例，用户可以直接复制到自己的项目进行使用
2	Codelab Notebook	NLP通用模版升级至文心3.0
3	Codelab Notebook	新增支持PaddlePaddle2.3.0框架
4	预置模型调参	物体检测方向新增支持小目标检测能力，用户可以选择开启小目标检测获得针对小目标更优的精度表现
5	预置模型调参	图像分类、物体检测方向新增支持数据不平衡优化能力

2022年4月

序号	功能模块	功能描述
1	Notebook	Notebook升级交互样式，操作体验更加丝滑
2	预置模型调参	新增行业模型，并新增车辆检测、卡车识别等模型，长期实践沉淀，垂类场景效果好
3	预置模型调参	新增智能归因能力，深入洞悉模型问题、推荐调优对策，助力开发者突破训练精度瓶颈

2022年2月

序号	功能模块	功能描述
1	预置模型调参	视觉模型支持H5能力
2	官网	改动版面

2021年11月

序号	功能模块	功能描述
1	预置模型调参	新增OCR文字识别-自然场景文字识别，同时支持数据标注、模型训练和模型部署
2	计费模块	新增支持训练资源和公有云部署资源的小时包

2021年10月

序号	功能模块	功能描述
1	预置模型调参	新增引入PP-YOLO系列优质物体检测模型，提升模型精度
2	预置模型调参	建模体验升级为表单式，且支持10+参数调整，提升建模效率

2021年08月

序号	功能模块	功能描述
1	预置模型调参	视觉方向训练任务新增支持增量训练，用户在开启训练任务时，可以选择加载原有任务的模型参数作为基准，并新增数据进行重新训练，提升训练效率
2	预置模型调参	视觉方向模型新增支持以H5形式发布，用户可以在经过验证发布后，在手机上快速体验公有云API服务的效果
3	预置模型调参	视觉方向训练任务-模型评估新增支持混淆矩阵功能，用户在发起新的训练任务并完成后，将可以查看模型的混淆矩阵，并与Badcase分析、热力图查看结合，为模型优化提供思路
4	自定义作业	新增支持自动搜索作业功能，用户可以上传本地代码文件，利用平台进行自动超参搜索，优化模型效果
5	自定义作业	新增支持PaddlePaddle2.1.1、Blackhole1.0.0，欢迎体验

2021年07月

序号	功能模块	功能描述
1	个人中心	新增用户中心、财务中心，以及合同管理、查看余额、优惠券码等多个快速入口，优化用户体验
2	模型管理	上线动物识别、行人检测、车辆检测、以及通用图像分类、物体检测等多个优质直接可用的开源模型，无需训练可直接部署为推理服务，并支持在本地服务器及边缘设备端离线运行

2021年06月

序号	功能模块	功能描述
1	预置模型调参	原『脚本调参』模块更新为『预置模型调参』模块，优化了模型训练、进度提示等用户体验
2	Notebook	Notebook开发环境升级，新增丰富代码库、资源监控等功能，内置PaddleX、Paddlehub、文心套件及海量函数库，提升开发体验
3	公有云部署	公有云部署新增『批量预测』功能，支持对大规模数据定时批量预测

2021年04月

序号	功能模块	功能描述
1	脚本调参	视觉任务支持自定义验证集
2	Notebook	新增“通用”型，支持PaddlePaddle、TensorFlow、Pytorch、Sklearn、XGBoost、Blackhole等开源框架或库
3	模型仓库	支持导入本地模型，支持基于深度学习框架：PaddlePaddle、TensorFlow、Pytorch、MXNet、Caffe、Darknet、ONNX的图像分类、物体检测模型以及基于Sklearn、XGBoost的机器学习模型，并支持在线部署或端部署
4	公有云部署	支持基于流量的自动启停
5	计费	私有API支持计费
6	全局优化	规范Python版本以及AI框架描述

2021年01月

序号	功能模块	功能描述
1	脚本调参	新增表格数据预测，支持基于结构化数据创建二分类、多分类和回归模型，且支持AutoML和专家两种模式
2	脚本调参	视觉任务增加错误示例
3	脚本调参	NLP任务支持Python3、VisualDL以及热启动
4	Notebook	新增支持PaddlePaddle2.0.0-Python3.7环境
5	自定义作业	PaddlePaddle、TensorFlow、PyTorch深度学习框架版本升级且提供分布式训练方式
6	模型仓库	新增支持对表格预测模型的管理能力
5	模型部署	私有SDK、私有API的部署功能迁移到操作台内
6	计费能力	公有云部署调整为quota计费，支持多实例配置、自助启停服务等操作
7	计费能力	离线服务上线计费功能，服务器SDK、设备端SDK、专项硬件SDK支持在线购买并获取永久有效序列号
8	全局优化	脚本调参、Notebook、自定义作业、模型、模型版本均支持修改名称与描述

2020年12月

序号	功能模块	功能描述
1	脚本调参	优化运行中任务提醒，任务配置中选择数据集增加标注状态等
2	脚本调参	支持自动发布模型到模型仓库
3	脚本调参	优化任务配置顺序

2020年11月

EasyDL专业版升级为全功能AI开发平台BML。

序号	功能模块	功能描述
1	脚本调参	支持发布模型到模型仓库
2	Notebook	支持发布模型到模型仓库
3	自定义作业	新增训练方式
4	模型仓库	新增模块
5	模型部署	前端交互修改

2020年9月

序号	功能模块	功能描述
1	数据管理	新增支持VOC标注格式
2	脚本调参视觉任务	新增支持实例分割、支持VisualDL
3	脚本调参文本任务	新增文本实体抽取、支持多机训练
4	Notebook	接入VisualDL、支持性能监控、支持文件打包下载、Python文件编辑集成LSP，支持Paddle1.8.4

2020年8月

序号	功能模块	功能描述
1	Notebook	优化启动速度
2	脚本调参视觉任务	上线预训练网络与部署方式适配性查询页面
3	视觉模型部署	支持华为达芬奇NPU

2020年7月

序号	功能模块	功能描述
1	脚本调参视觉任务	上线超参搜索功能
2	脚本调参文本任务	上线文本分类-多标签任务

2020年5月

序号	功能模块	功能描述
1	脚本调参视觉任务	上线超大规模预训练模型、新增12种网络类型
2	Notebook	优化交互
3	视觉模型部署	发布EasyDL-Jetson软硬一体方案

2020年4月

序号	功能模块	功能描述
1	数据管理	新增公开数据集
2	Notebook	上线Notebook功能，支持图像类模型开发
3	视觉类模型训练	新增支持Paddle-fluid-1.6.2框架、新增支持Python3、物体检测新增支持RetinaNet-ResNet50-FPN网络、上线训练计费、支持GPU V100、P40、P4机型
4	文本类模型训练	文本类任务后端升级Text One框架

2019年12月

序号	功能模块	功能描述
1	数据管理	新增支持NLP序列标注功能
2	云服务数据	支持图像分类、物体检测云服务数据管理

2019年10月

首次发布

产品简介

BML平台升级公告

亲爱的开发者您好！为全面提升平台用户体验，方便用户更好的进行开发管理，BML平台现全面升级，引入项目，区域（region），权限，角色等相关概念。相关升级给您带来的不便，敬请谅解。本次升级同样适用于EasyData和EasyEdge独立平台。

区域（region）

历史数据统一迁移至华北-北京区域（region）下，在使用cce集群，BOS，PFS等基础资源时，需要在同一区域下才能正常使用。

项目

项目可以用来实现管理资产、资源和子用户，按项目（工作空间）实现资源和数据的隔离。
（1）对于用户的历史训练任务和数据，将统一迁移至default项目（北京）下。
（2）新用户注册时系统自动分配默认项目，主账户可以创建多个项目，在创建子用户后可以分配该子用户到对应项目中。

权限

针对BML平台各个功能进行了权限细分，不同角色系统默认分配了相关权限。

角色

平台目前的默认角色包括：超级管理员，资源管理员，项目管理员，开发者，标注员。主账户默认为超级管理员，拥有平台所有权限。主账户可以为子账户绑定不同的角色，不同角色对应同的平台权限

超级管理员：拥有当前区域下平台所有权限

资源管理员：拥有当前区域下资源管理、分配等权限。

项目管理员：拥有授权项目下数据、训练、部署等平台功能权限及项目管理角色授权权限。

开发者：拥有授权项目下数据，训练，部署等平台功能权限。

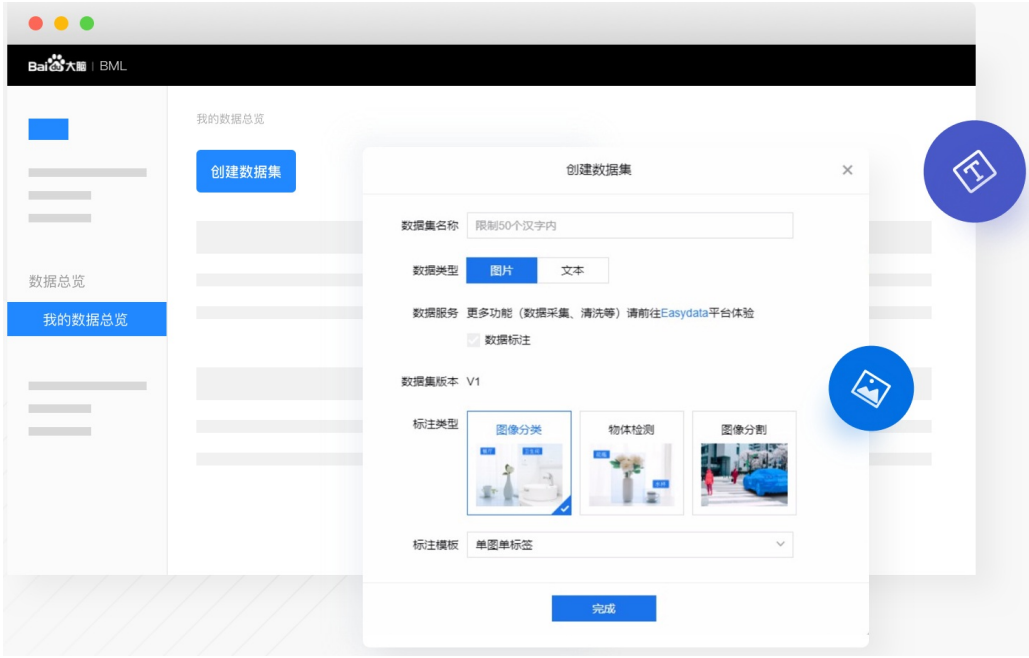
标注成员：拥有授权项目下多人标注任务接收的权限

什么是BML

BML 全功能 AI 开发平台是一个面向企业和个人开发者的机器学习集成开发环境，为经典机器学习和深度学习提供了从数据处理、模型训练、模型管理到模型推理的全生命周期管理服务，帮助用户更快的构建、训练和部署模型。

智能的数据管理方式

将智能引入数据管理，自动化的完成数据清洗、数据标注等工作，为模型训练提供高质量数据。



适用于不同人员的AI开发方式

- 预置模型调参

低代码开发方式，无需或仅需编写极少代码即可完成模型的构建与训练。



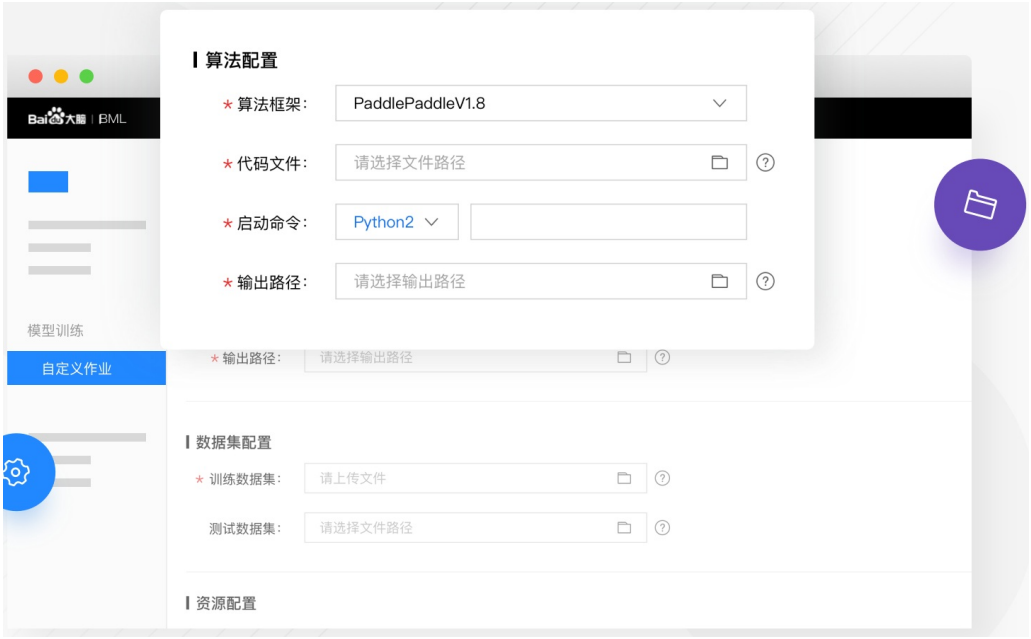
- Notebook 建模

在托管的 Notebook 中编写和运行自己的代码，也可以使用预置代码快速构建模型。



- 自定义作业建模

在云端运行已编写的模型训练代码，可扩展的算力加快模型训练过程。



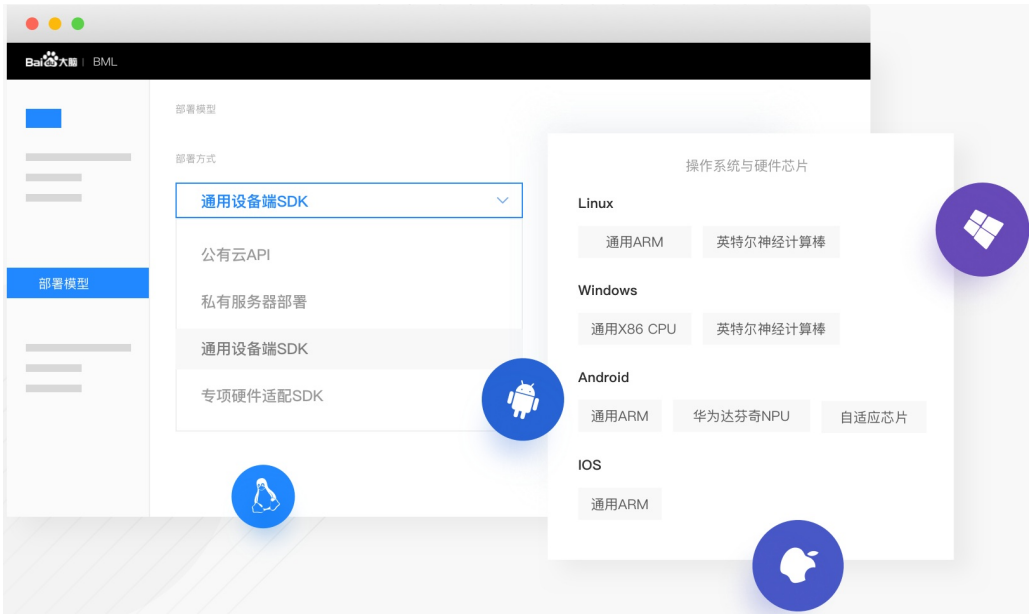
开放的模型管理服务

对内外部模型提供统一的管理能力，不断扩充支持的框架以及模型格式。



适用于不同场景的推理形式

一次训练，支持多种部署方式；或将已有模型转换为适用于端部署的形式。



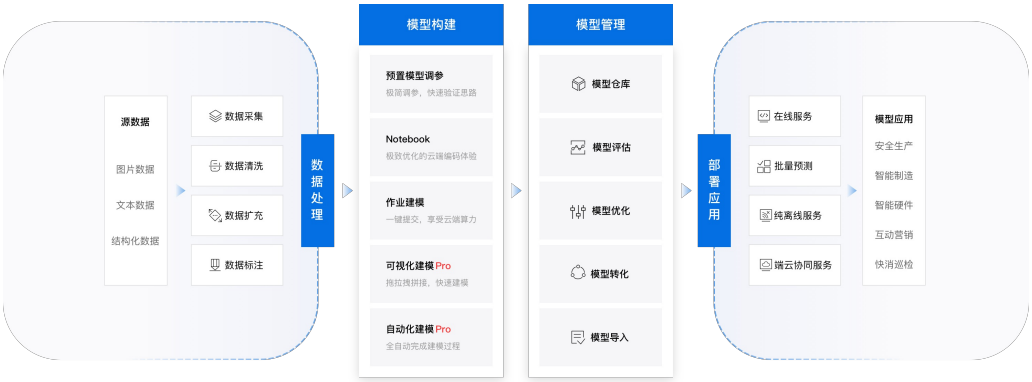
产品功能

产品功能

BML 将 AI 模型开发过程中依赖的各种工具、框架、库集成在产品中，并通过简单易用的方式交付给开发者。

功能总览

BML 提供兼具广度与深度的机器学习栈，开发者无需关注底层资源管理、运维和开发环境准备，通过开箱即用的云端机器学习开发环境，提升开发者的效率，加速各领域的智能化进程。



特色功能

- 数据管理

支持智能标注、数据清洗、线上数据回流等功能，提供数据集版本管理，数据与模型的关系更清晰。

- 模型构建

基于百度超大规模数据集的 CV 预训练模型、基于文心 ERNIE 的 NLP 预训练模型以及飞桨的优质预训练模型，帮助开发者更快地获得精度更高的模型。

- 云边端全场景推理

将智能应用到所需的地方，无论云端、私有化服务器或设备端。

产品优势

产品优势

BML 是一种基于云的 AI 开发环境，提供兼具效果与性能的卓越 AI 开发体验。

高性能的算力支持

- 支持多机多卡分布式训练。
- 提供丰富算力机器资源选型。
- 按需获取资源，灵活便捷。

强大的数据处理服务

- 业界首个开放的数据采集和清洗服务。
- 支持自动数据增强，助力模型效果有效提升。
- 智能标注与多人标注双管齐下，标注效率提升80%。

高精度训练效果

- 预置百度超大规模视觉预训练模型。
- 预置业界领先的 NLP 预训练模型文心 ERNIE。
- 支持自动超参搜索、不平衡数据处理等先进训练机制。
- 支持视觉、文本、机器学习等丰富任务类型。

丰富完善的开发方案

- 全面超越原生 Jupyter Notebook 的云端开发环境。
- 提供适合 AI 初学者的脚本调参开发方式，快速上手。
- 支持开放框架多语言的作业建模，高度灵活。
- 提供本地客户端BML CodeLab，预置海量丰富插件。

广泛适配的模型推理服务

- 云服务：SLA 99.99%，弹性扩缩容。
- 本地服务器部署：多平台多语言支持。
- 设备端SDK：适配15+硬件芯片，预测速度最快可提升10倍。
- 软硬一体方案：6款软硬一体方案，高中低全矩阵覆盖。

全面丰富的交付方案

- 3大交付方案，完善丰富。
- 公有云交付，即刻上手体验。

- 私有化交付，保障企业核心生产数据私密性和安全性。
- 支持一体机交付，满足不同算力需求。

文心大模型

文心大模型是什么

文心大模型是百度发布的全球首个知识增强千亿大模型。具有技术领先、全场景覆盖、零门槛使用、真实可落地的四大特点，目标成为百行千业AI 开发的首选基座大模型。目前文心大模型包括：

文心·NLP 大模型

面向 NLP 任务场景，具备超强语言理解能力以及写小说、歌词、诗歌、对联等文学创作能力。创新性地将大数据预训练与多源丰富知识相结合，通过持续学习技术，不断吸收海量文本数据中词汇、结构、语义等方面的新知识，实现模型效果不断进化。

文心·CV 大模型

基于领先的视觉技术，利用海量的图像/视频等数据，为企业/开发者提供强大的视觉基础模型，以及一整套视觉任务定制与应用能力。

文心·跨模态大模型

基于知识增强的跨模态语义理解关键技术，可实现跨模态检索、图文生成、图片文档的信息抽取等应用的快速搭建，落实产业智能化转型的AI助力

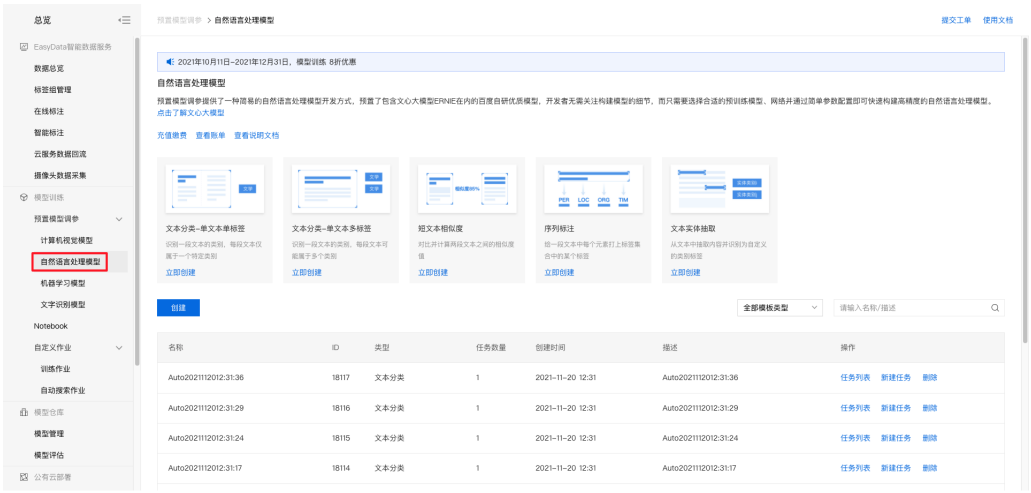
如何在 BML 上使用文心大模型

文心·NLP 大模型

文心·NLP 大模型已接入预置模型调参和Notebook建模技术方向。现为您介绍相关使用路径。

预置模型调参

1、点击进入 BML 操作台后，点击预置模型调参-自然语言处理模型

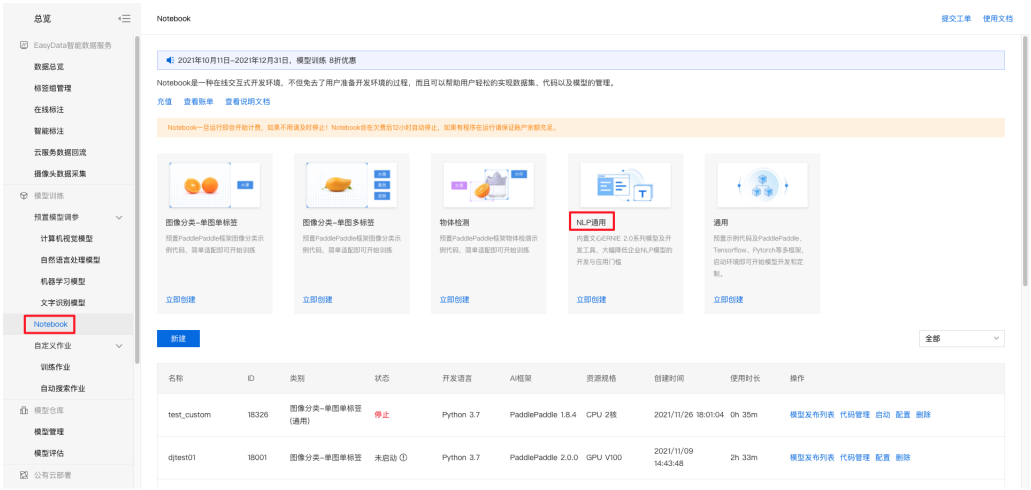


2、点击新建任务后，即可选择文心·NLP 大模型旗下的 ERNIE2.0 系列模型进行训练



Notebook建模

点击进入 BML 操作台后，点击 Notebook ，并点击 NLP 通用即可使用预置文心 ERNIE 2.0系列模型及开发工具的 Notebook 环境进行开发。



文心·CV 大模型

文心·CV 大模型即将接入，敬请期待！

文心·跨模态大模型

文心·跨模态大模型即将接入，敬请期待！

平台重点升级介绍

近期 BML 平台针对大模型训练加速和模型监控方面有了较大的功能升级，将逐渐在私有化和公有云的平台解决方案中提供给客户。

大模型训练加速服务

BML 平台针对大模型和主流的深度学习模型的训练都提供了加速服务：

1. 在大模型训练场景中，支持了大模型参数下的混合并行加速，支持主流大模型，如ERNIE, GPT, VIT, SWIN-TRANSFORMER等的并行训练加速。
2. 在主流深度学习模型的训练场景中，支持了在大规模数据量下的数据并行加速需求。

其核心的技术优化点为：

1. 基础网络优化。提供高吞吐，低时延的、低负载的RDMA网络，显著提升云服务之间的通信效率。
2. 基础单机性能优化。 提供GPU高性能定制算子，配合混合精度训练机制显著提升单机性能吞吐。
3. 分布式梯度通信优化。 提供高压缩比的梯度通信优化技术，可以显著降低节点间(卡间)通信量，在节点间(卡间)通信量成为瓶颈的情况下有显著收益 。
4. 并行策略自动调优。通过对并行策略及参数进行自动化调优，可以无需人工调优介入，在较短时间内找到与专家调优持平甚至更优的并行策略组合。
5. 自动网络结构搜索。 通过网络结构参数的自动化搜索，可以搜索到精度基本无损、训练吞吐上能有大幅提升的最佳性能子网。

下面围绕自动网络结构搜索和并行策略自动调优，介绍整体的效果提升。

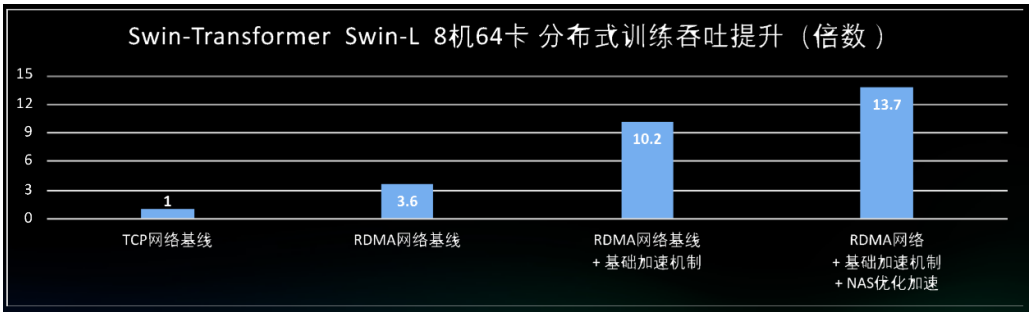
优势项1:自动网络结构搜索

通过网络结构参数的自动化搜索，搜索到精度基本无损、训练吞吐上能有大幅提升的最佳性能子网。例如在swin transformer swin-large的的吞吐提升优化中，自动网络结构搜索后可以将训练吞吐从原来的10.2倍提升到13.7倍。

自动网络结构搜索示意图：



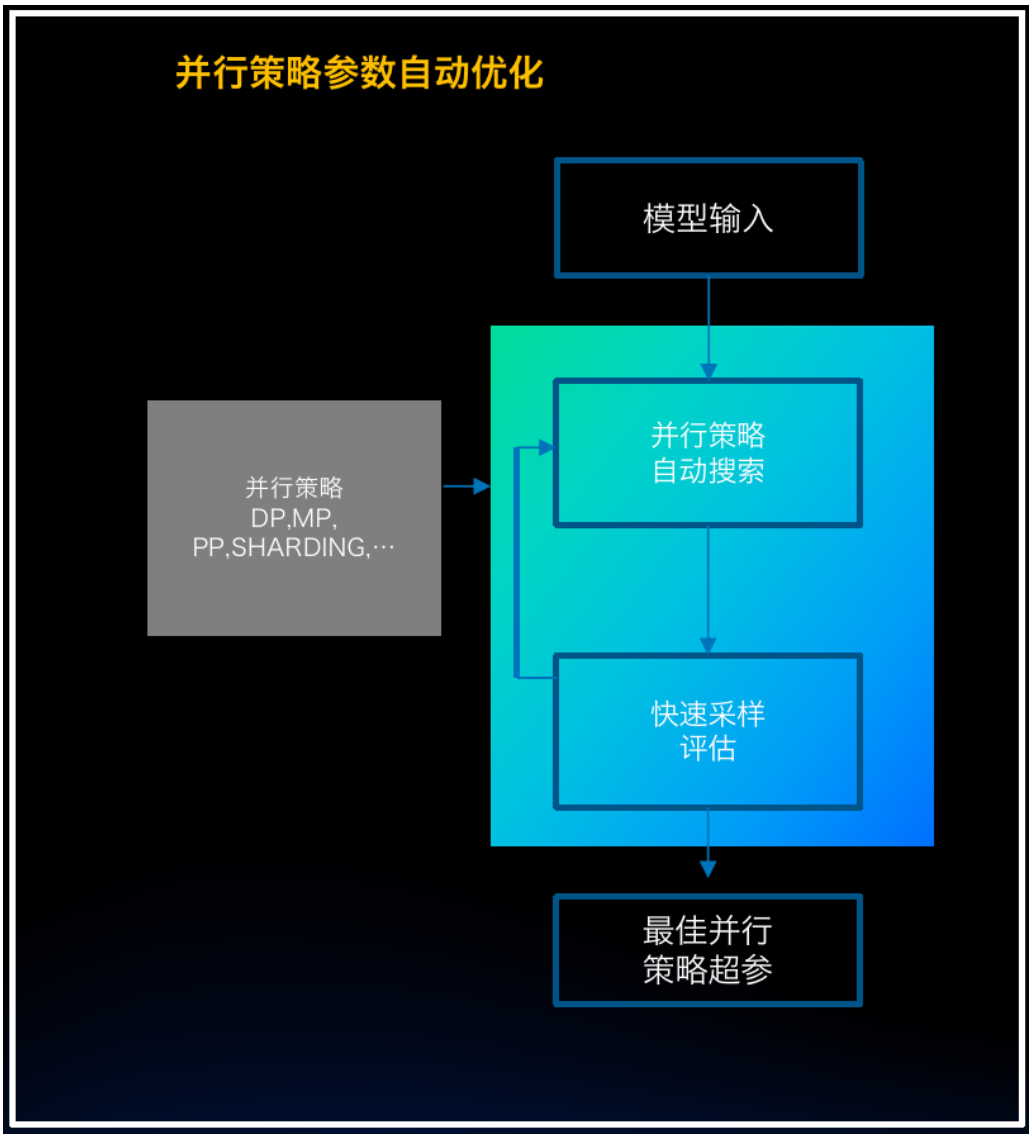
效果提升明显，训练吞吐提升13.7倍，算力成本可节省90%：



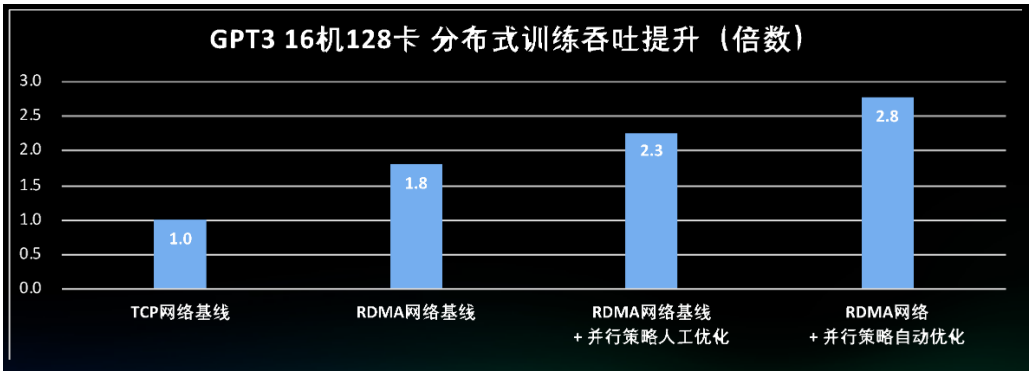
优势项2:并行策略自动调优

通过对并行策略进行自动化调优，可以无需人工调优介入，在较短时间内找到与专家调优持平甚至更优的并行策略组合。例如在GPT3 16机128卡分布式训练吞吐提升优化中，并行策略自动优化可以将人工优化的2.3倍提升到2.8倍。

并行策略自动调优示意图：



效果提升明显，训练性能提升2.8倍，训练算力成本可节省60%+：



生产环境的模型在线监控

生产环境的模型在线监控旨在使用数据科学和统计技术来持续评估生产中数据的质量、模型的质量，从而在早期发现模型不稳定，了解模型指标下降的方式和原因，对模型衰减做出提前预警，并推动模型的重训。

在经历开发阶段一系列调试、验证之后，模型被正式部署到生产环境中，此时模型研发人员面临的挑战才刚刚开始。在开发阶段研发人员使用静态样例集进行训练，但在生产环境中模型需要应对来自真实世界的动态数据。当模型部署在生产环境之后，它可能会在毫无预兆的情况下，业务指标快速下降，开发中的静态训练数据与生产中的动态数据之间的差异，或是特征与输出数据之间变量关系的变化是导致指标下降的主要可能因素。

在生产中主要通过以下两种方法监控模型性能的下降：模型质量评估和漂移检测。

模型质量评估

针对模型评价指标或业务指标开展的评估作业，旨在反映模型在生产中的实际表现，例如欺诈检测模型是否准确标记某条特定交易是欺诈交易。

- 评价指标：通用，与领域无关，如准确度、AUC等。由于模型设计者可能基于其中一个指标筛选的最佳模型，因此它是重要的监控项。
- 业务指标：基于特定领域定义的指标，如信用评分业务制定的A/B卡。

模型质量评估功能已可以在私有化平台解决方案上进行使用。

漂移检测

- 数据漂移检测

基于输入数据的特征值进行监控，时效性更高，无需等待预测数据标签。它利用模型的认知局限性来预判模型衰减的可能，即预测数据和训练数据的分布表现出明显差异时，模型的预测能力可能会下降。

- 概念漂移检测

基于输入数据的标签值进行监控，旨在观测输入数据和目标变量之间的关系随时间的变化。它可以用于在线学习系统，为分析学习准确率下降提供思路。

产品定价

模型训练计费说明

BML平台模型训练模块均已开放计费，支持按量计费（后付费）的计费方式
具体计费规则如下：

- 未开通付费时，仅支持使用免费额度，免费额度用完即停止训练任务。
- 开通付费后，优先消耗免费额度，免费额度用完后自动转为按量后付费
- 按分钟计费，不足1分钟按1分钟计。
- **预置模型调参与自定义作业模块**在任务结束后统一扣费，任务运行中欠费不会中断任务。
- **Notebook**启动后实时计费，请保证账户余额充足，如果不用请及时停止。
- 使用 BML 前需保证账户无欠款。
  计费公式 费用=训练机型单价×节点数×使用时长
  时长计量方法：只包括任务状态为“运行”时的统计时间。

价格详情

预置模型调参+自定义作业 在预置模型调参，自定义作业两种建模方式任务配置过程中，您可以选择运行环境及节点数。目前自定义作业暂时只支持单节点。

训练方式	运行环境	定价	免费额度
自定义作业	CPU 4核16GB内存	0.025元/分钟/节点	100（小时*节点）
自定义作业	CPU 16核64GB内存	0.092元/分钟/节点	-
自定义作业	CPU 64核256GB内存	0.36元/分钟/节点	-
自定义作业 预置模型调参	TeslaGPU_P4_8G显存单卡_12核CPU_40G内存	0.28（元/分钟/节点）	100（小时*节点）
自定义作业 预置模型调参	TeslaGPU_P40_24G显存单卡_12核CPU_40G内存	0.36（元/分钟/节点）	50（小时*节点）
自定义作业 预置模型调参	TeslaGPU_V100_16G显存单卡_12核CPU_56G内存	0.45（元/分钟/节点）	-

> **说明1**：扣费发生的时间点为任务训练结束（包含手动暂停训练或自动停止训练）后，如果因BML系统异常导致训练任务运行失败，则相应训练任务的全部耗时在账单中会做扣减，不会参与计费。
> **说明2**：为确保训练任务的正常进行，建议您在开通付费后确保账户余额不低于100元。
> **说明3**：预置模型调参参与计费的时长计算方式：
> 若用户选择数据增强—自动搜索，则计费时长为实际训练过程耗时（**即状态为运行中-训练阶段的持续时间）+ 自动数据增强耗时相加总时长**。 若用户选择数据增强-模型配置或手动配置，或未选择数据增强，则计费时长**仅为实际训练过程耗时（即状态为运行中-训练阶段持续时**

> 间）

Notebook+Codelab 在Notebook与codelab中，提供训练资源：

运行环境	定价	免费额度
CPU 2核8GB内存	**限时免费**	-
TeslaGPU_V100_32GB显存单卡_8核CPU_32GB内存	0.5元/分钟	50小时

> **说明1**：Notebook后付费资源为实时扣费，
> **说明2**：Codelab中，资源启动即开始计费，停止后终止计费
> **说明3**：为确保训练任务的正常进行，建议您在开通付费后确保账户余额不低于100元。
> **说明4**：如果您已经开通训练计费，免费资源用完后，将会**自动转为按量后付费**。
> **说明5**：如果您未开通训练计费，免费资源用完后，任务将自动停止，为确保训练任务的正常进行，建议您及时开通付费。
> **说明6**：开通付费后，如果您在任务运行过程中账户欠费，为保证任务不受影响，**平台将允许您的Notebook任务继续运行12小时**，请您在此期间及时充值。

余额不足提醒与欠费处理

余额不足提醒
根据您历史的账单金额，判断您的账户余额（含可用代金券）是否足够支付未来的费用，若不足以支付，系统将在欠费前三天、两天、一天发送续费提醒短信，请您收到短信后及时前往控制台财务中心充值。

欠费处理

- 北京时间整点检查您的账户余额是否足以支付本次账单的费用（如北京时间11点整检查账户余额是否足以支付10点至11点的账单费用），若不足以支付，即为欠费，欠费时系统会发送欠费通知。
- 欠费后您开通付费的产品将进入欠费状态，只能使用免费额度，并且无法发起新的训练任务。

公有云部署计费说明

BML公有云部署费用为模型部署时使用资源产生的费用：

- 所有模型都需要您配置机器资源。**计次试用期间部署的模型需重新配置机器资源后，才能再次使用，详细操作步骤请前往公有云部署说明查看**
- 系统会根据您选择的机型与配置自动计算价格
- 按分钟计费，不足1分钟按1分钟计。
- 使用 BML 前需保证账户无欠款。

计费规则

配额：系统将运行公有云部署的机器资源1核cpu和4G内存折算为1配额。配额越高，单实例性能越高

实例：运行公有云服务的副本数，与QPS数成正比

计费方式：按量后付费

计费配额= 配额x实例

费用= 服务机型单价x计费配额数x使用时长

时长计量方法：公有云部署中，计费从公有云部署服务的状态变为『运行中』，开始计时计费，服务的状态变为『已停止』，停止计时计费。公有云服务的停止，开启均需在平台上手动控制。

价格详情

机型	价格	免费额度
CPU	0.4元/小时/配额	100（小时*配额）

余额不足提醒与欠费处理

余额不足提醒 根据您历史的账单金额，判断您的账户余额（含可用代金券）是否足够支付未来的费用，若不足以支付，系统将在欠费前三天、两天、一天发送续费提醒短信，请您收到短信后及时前往控制台财务中心充值。

欠费处理 北京时间整点检查您的账户余额是否足以支付本次账单的费用（如北京时间11点整检查账户余额是否足以支付10点至11点的账单费用），若不足以支付，即为欠费，欠费时系统会发送欠费通知。 欠费后您开通付费的产品将进入欠费状态，只能使用免费额度，**若免费额度用完，将不能提交新的公有云部署服务，且所有运行中的公有云部署服务将自动停止，并且充值后才能重启**。

通用小型设备部署价格说明

每个模型发布通用小型设备SDK后需通过序列号激活使用，每发布一个模型即可申请2个测试序列号供3个月内免费试用。

目前已支持在控制台在线按设备购买授权或按产品线购买。

价格参考

按设备购买永久授权

授权终端设备数量	单设备授权价格（基础版SDK）	单设备授权价格（加速版SDK）
1≤设备数≤10	300元	2000元
11≤设备数≤100	250元	1500元
101≤设备数	200元	1000元

说明：

- 设备授权的购买是「分段阶梯计费」，例如您购买了9个授权，因为这9个是在1~10的区间，所以付费300元/个×9个；当您继续购买2个时，累计购买量已经达到了11个，则付费1×300+1×250，因为后1个已经在11~100的区间内

按产品线购买永久授权

单个序列号可激活设备数上限为1万台，目前已支持按年购买授权：基础版SDK每年10万元，加速版SDK每年30万元。

如有授权年限、设备上限数的更多需求，请提交工单联系我们。

> 说明：
> 1、按产品线授权的序列号适用于开发手机APP，序列号仅限在绑定的包名下使用
> 2、如同一模型需同时购买iOS和Android包名的授权，需按2个产品线购买

服务器部署价格说明

目前BML脚本调参任务已支持将定制模型部署在本地服务器上，只需在发布模型时提交本地服务器部署申请，通过审核后即可获得一个月免费试用。

目前已支持在控制台在线按设备使用年限购买授权。

价格参考

部署类型	按年授权价格（每个模型每年在每个设备）	永久授权价格（每个模型在每个设备）
本地服务器SDK	10000元	50000元
本地服务器API	10000元	50000元

注：如果您需要部署的规模较大，欢迎点击合作咨询留下您的相关信息，我们将会联系您进一步的商务洽谈，予以专项支持。

说明：

- 脚本调参视觉模型支持SDK，API两种部署方式。自然语言处理模型支持API部署方式。
- 本地服务器部署的授权粒度为每个模型每年每台设备，例如购买了2个模型3年在1台设备的授权，价格为2（个模型）x3（年）x1（台）x10000元=60000元。可在一台设备上激活2个有效期三年的模型。

专项适配硬件部署价格说明

BML脚本调参图像分类，物体检测模型支持多种专项适配硬件方案，请前往专题页面对比不同方案的性能与价格，选择与业务场景最匹配的方

案。

方案获取流程

- Step 1：在BML训练专项适配硬件的图像分类/物体检测模型，迭代模型至效果满足业务需求
- Step 2：发布模型时选择专项适配硬件的专用SDK，并在AI市场购买方案
- Step 3：获得硬件和用于激活专用SDK的专用序列号，参考文档集成后，即可实现离线AI预测

如有其他硬件方案需求，欢迎提交工单咨询。

专用SDK价格参考

按设备购买永久授权

授权终端设备数量	单设备授权价格
1≤设备数≤100	300元
101≤设备数≤1000	280元
1001≤设备数	240元

说明：

- 设备授权的购买是「单笔订单阶梯计费」，例如您需购买200个授权，如果分2次购买，每次购买100个，需要付费2×100×300元；若您1次购买，则需要付费200×280元，因为200已经在101-1000的阶梯内。

批量预测计费说明

批量预测服务的计费方式类似于模型训练模块，具体计费规则如下：

- 未开通付费时，仅支持使用免费额度，免费额度用完即停止训练任务。
- 开通付费后，优先消耗免费额度，免费额度用完后自动转为按量后付费
- 按分钟计费，不足1分钟按1分钟计。
- 在任务结束后统一扣费，任务运行中欠费不会中断任务。
- 使用 BML 前需保证账户无欠款。
  计费公式：费用=训练机型单价×节点数×使用时长
  时长计量方法：只包括任务状态为“运行”时的统计时间。

价格详情

在批量预测的任务配置过程中，您可以选择运行环境及节点数。

运行环境	定价	免费额度
4核CPU 16GB内存	0.025元/分钟/节点	100（小时*节点）
16核CPU 64GB内存	0.092元/分钟/节点	-
64核CPU 256GB内存	0.36元/分钟/节点	-
TeslaGPU P4 8G显存单卡 12核CPU 40G内存	0.28（元/分钟/节点）	100（小时节点）
TeslaGPU P40 24G显存单卡 12核CPU 40G内存	0.36（元/分钟/节点）	50（小时节点）
TeslaGPU V100 16G显存单卡 12核CPU 56G内存	0.45（元/分钟/节点）	-

> **说明1**：扣费发生的时间点为批量预测任务结束（包含手动暂停或自动停止）后，如果因BML系统异常导致训练任务运行失败，则相应训练任务的全部耗时在账单中会做扣减，不会参与计费。
> **说明2**：为确保批量预测任务的正常进行，建议您在开通付费后确保账户余额不低于100元。
> **说明3**：计费时长**仅为实际训练过程耗时（即状态为运行中-训练阶段持续时间）**

余额不足提醒与欠费处理

余额不足提醒

根据您历史的账单金额，判断您的账户余额（含可用代金券）是否足够支付未来的费用，若不足以支付，系统将在欠费前三天、两天、一天发送续费提醒短信，请您收到短信后及时前往控制台财务中心充值。

欠费处理

- 北京时间整点检查您的账户余额是否足以支付本次账单的费用（如北京时间11点整检查账户余额是否足以支付10点至11点的账单费用），若不足以支付，即为欠费，欠费时系统会发送欠费通知。
- 欠费后您开通付费的产品将进入欠费状态，只能使用免费额度，并且无法发起新的训练任务。

平台管理

在BML平台使用容器镜像服务CCR

在BML平台使用容器镜像服务CCR

平台支持用户在用户资源池上关联容器镜像服务CCR作为资源池的镜像仓库，在使用用户资源池提交任务时，可以使用镜像仓库中的镜像。

当前支持的容器镜像服务CCR类型：

- 容器镜像服务CCR-企业版
- 容器镜像服务CCR-个人版

当前支持使用容器镜像服务CCR提交的任务：

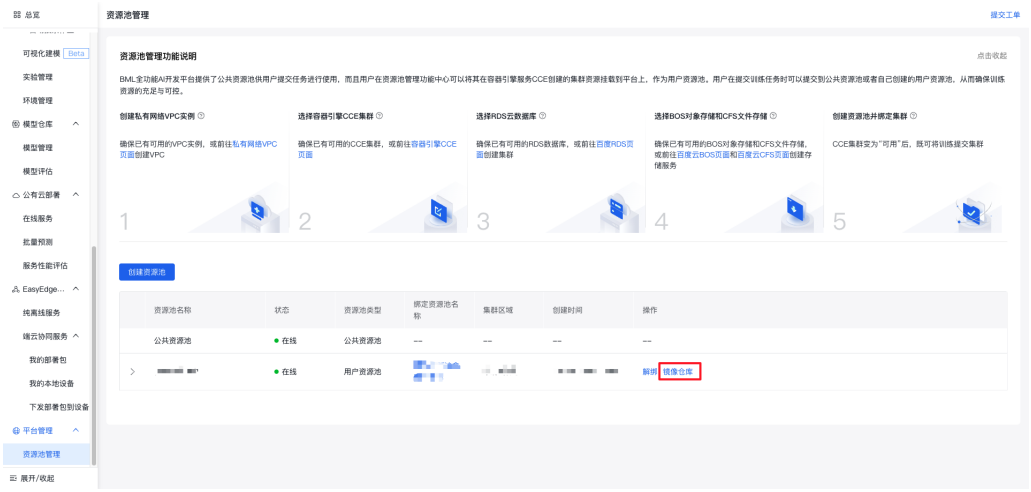
- 自定义作业-训练作业任务、自动搜索作业任务

前提条件

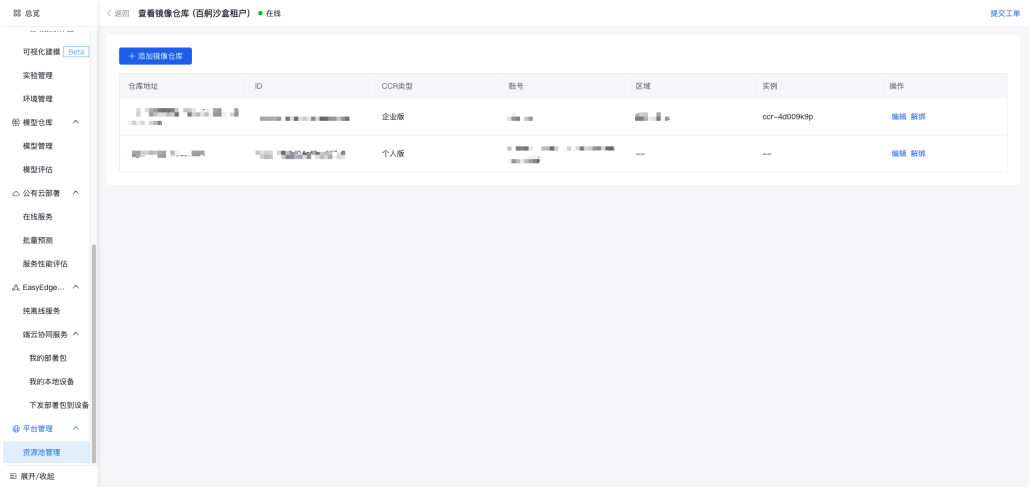
1. 用户在平台上已经挂载了容器引擎CCE资源作为用户资源池，点击了解容器引擎CCE；
2. 用户已经创建了容器镜像服务CCR，点击了解容器镜像服务CCR。
3. 容器镜像服务CCR能够被容器引擎CCE资源访问到，也即能被对应的VPC访问到。

创建镜像仓库

- Step1:进入平台管理-资源池管理，已挂载并运行正常的用户资源池支持“镜像仓库”的操作项，点击即可开始查看镜像仓库。



- Step2:点击镜像仓库，即可进入镜像仓库列表。



- Step3：点击添加镜像仓库，即可进入添加流程。

企业版：支持选择资源池对应区域和VPC下的，归属于主账号的容器镜像服务CCR-企业版的实例，并填写账号密码进行添加。



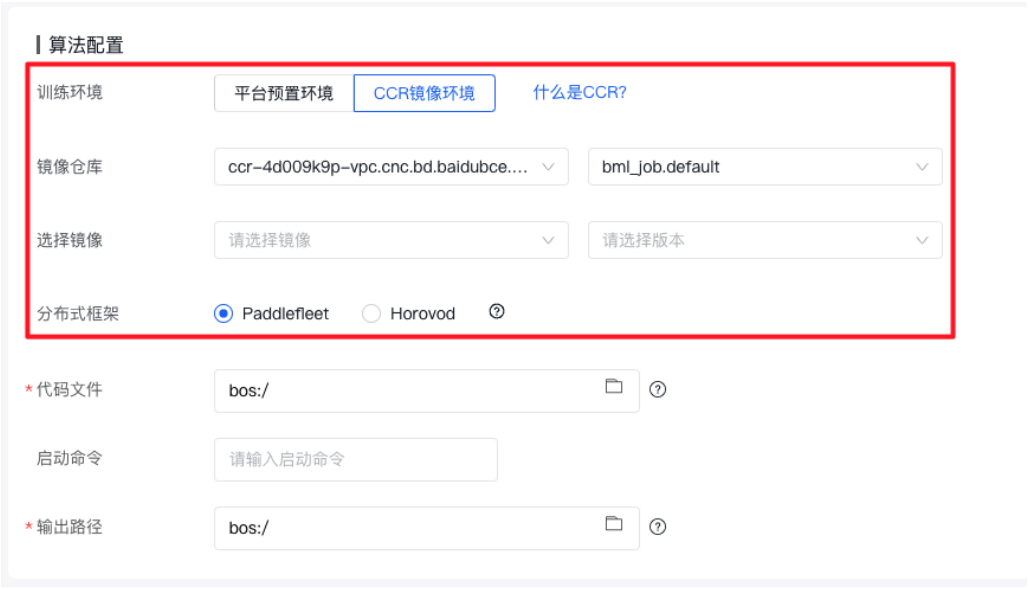
个人版：支持选择归属于主账号的容器镜像服务CCR-个人版的实例，并填写账号密码进行添加。



使用镜像提交自定义作业任务

在算法配置阶段，如果用户选择了用户资源池，即支持选择该资源池所关联的CCR镜像环境提交任务。（训练作业和自动搜索作业任务的提交过程一致）

- 企业版：支持依次选择镜像仓库-命名空间-镜像-版本，从而选中一个唯一确定的镜像用于提交任务。



- 个人版：支持依次选择镜像仓库-镜像-版本，从而选中一个唯一确定的镜像用于提交任务。



在选择完镜像后，需要根据镜像中的深度学习框架来选择分布式框架，其中：

- Paddlefleet：PaddleFleet是PaddlePaddle推出的分布式图引擎以及大规模参数服务器，用于支撑飞桨框架大规模分布式训练能力。
- Horovod：分布式训练框架，支持tensorflow、pytorch、ray、mxnet等业界著名的开源的机器学习框架。通过对底层tensorflow等框架进行上层分布式调度、分布式通信、梯度计算等封装，完成大规模集群下模型的训练。horovod主要支持在GPU资源集群上的大规模分布式的同步训练。

后续填写代码文件、启动命令、输出路径等信息后，即可提交自定义作业任务。

附录：自定义镜像规范

Paddle镜像

- python版本要求：python3.7及以上
- 安装bos：dockerfile实现

```
RUN /bin/bash -c 'mkdir /home/bos && \
  cd /home/bos && \
  wget --no-check-certificate https://sdk.bce.baidu.com/console-sdk/linux-bcecmd-0.3.0.zip && \
  unzip linux-bcecmd-0.3.0.zip && \
  echo "export LANG=en_US.UTF-8" >> ~/.bashrc && \
  echo "export PATH="/home/bos/linux-bcecmd-0.3.0:${PATH}"" >> ~/.bashrc && \
  source ~/.bashrc'
```

或者自行在镜像中安装到/home/bos/linux-bcecmd-0.3.0目录下，确保 “/home/bos/linux-bcecmd-0.3.0/bcecmd” 这句命令在命令行能弹出相关bcecmd帮助信息，即命令能被系统识别

- 安装搜索需要的 sdk
- 用户自行安装wheel包：rudder_autosearch-1.0.0-py3-none-any.whl，点击下载。
- 安装 jq 解析sts需要：apt-get install jq
- 安装 curl 解析sts需要：apt-get install curl
- 安装protobuf 搜索sdk需要：建议安装3.20.1版本 pip install protobuf==3.20.1
- 设置虚拟化环境变量初始值：dockerfile实现

```
ENV NVIDIA_VISIBLE_DEVICES ""
```

Sklearn镜像

```
 FROM ubuntu16.04-python3

#### Configure time zone
RUN apt-get update && \
    apt-get install -y tzdata && \
    ln -snf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime && \
    dpkg-reconfigure -f noninteractive tzdata && \
    apt-get clean

RUN apt-get install -y --no-install-recommends\
    build-essential \
    libopencv-dev \
    libssl-dev \
    dnsutils \
    unzip \
    vim \
    jq \
    curl \
    wget && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

#### py3
RUN bash -c 'cd /tmp && \
    wget --no-check-certificate https://repo.anaconda.com/miniconda/Miniconda3-py37_4.8.3-Linux-x86_64.sh && \
    bash Miniconda3-py37_4.8.3-Linux-x86_64.sh -b -p ~/miniconda3 && \
    echo "source ~/miniconda3/bin/activate" >> ~/.bashrc && \
    echo "export PATH="~/miniconda3/bin:${PATH}"" >> ~/.bashrc && \
    source ~/.bashrc  && \
    rm -rf Miniconda3-py37_4.8.3-Linux-x86_64.sh' && \
    /root/miniconda3/bin/python -m pip config set global.index-url https://pypi.douban.com/simple/ && \
    /root/miniconda3/bin/python -m pip install --upgrade pip && \
    /root/miniconda3/bin/python -m pip install --upgrade setuptools && \
    /root/miniconda3/bin/python -m pip install numpy==1.17.4 && \
    /root/miniconda3/bin/python -m pip install albumentations==0.4.3 && \
    /root/miniconda3/bin/python -m pip install Cython==0.29.16 && \
    /root/miniconda3/bin/python -m pip install pycocotools==2.0.0 && \
    /root/miniconda3/bin/python -m pip install ruamel.yaml && \
    /root/miniconda3/bin/python -m pip install ujson && \
    /root/miniconda3/bin/python -m pip install scipy==1.5.3 && \
    /root/miniconda3/bin/python -m pip install scikit-learn==0.23.2 && \
    /root/miniconda3/bin/python -m pip install pandas

RUN /root/miniconda3/condabin/conda clean -p && \
    /root/miniconda3/condabin/conda clean -t
RUN rm -rf ~/.cache/pip
RUN rm -rf /usr/bin/python3 && ln -s /root/miniconda3/bin/python /usr/bin/python3

#### sklearn
RUN /root/miniconda3/bin/python -m pip install xgboost==1.3.1

#### 安装bos
RUN /bin/bash -c 'mkdir /home/bos && \
    cd /home/bos && \
    wget --no-check-certificate https://sdk.bce.baidu.com/console-sdk/linux-bcecmd-0.3.0.zip && \
    unzip linux-bcecmd-0.3.0.zip && \
    echo "export LANG=en_US.UTF-8" >> ~/.bashrc && \
    echo "export PATH="/home/bos/linux-bcecmd-0.3.0:${PATH}"" >> ~/.bashrc && \
    source ~/.bashrc'

#### 添加搜索作业SDK install 3.20.1 protobuf for searchjob
COPY rudder-autosearch-1.0.0-py3-none-any.whl /home/rudder-autosearch-1.0.0-py3-none-any.whl
RUN pip install /home/rudder-autosearch-1.0.0-py3-none-any.whl && \
    pip install protobuf==3.20.1

ENV NVIDIA_VISIBLE_DEVICES ""

ENTRYPOINT ["/bin/bash"]
```

如果是pytorch/tensorflow的多机分布式作业，则需要安装额外的依赖包

- 安装openmpi

```
 #### 安装openmpi
RUN mkdir /tmp/openmpi && \
    cd /tmp/openmpi && \
    wget https://download.open-mpi.org/release/open-mpi/v4.0/openmpi-4.0.0.tar.gz && \
    tar zxf openmpi-4.0.0.tar.gz && \
    cd openmpi-4.0.0 && \
    ./configure --enable-orterun-prefix-by-default && \
    make -j $(nproc) all && \
    make install && \
    ldconfig && \
    rm -rf /tmp/openmpi
#### Create a wrapper for OpenMPI to allow running as root by default
RUN mv /usr/local/bin/mpirun /usr/local/bin/mpirun.real && \
    echo '#!/bin/bash' > /usr/local/bin/mpirun && \
    echo 'mpirun.real --allow-run-as-root "$@"' >> /usr/local/bin/mpirun && \
    chmod a+x /usr/local/bin/mpirun

#### Configure OpenMPI to run good defaults:
#### --bind-to none --map-by slot --mca btl_tcp_if_exclude lo,docker0
RUN echo "hwloc_base_binding_policy = none" >> /usr/local/etc/openmpi-mca-params.conf && \
    echo "rmaps_base_mapping_policy = slot" >> /usr/local/etc/openmpi-mca-params.conf && \
    echo "btl_tcp_if_exclude = lo,docker0" >> /usr/local/etc/openmpi-mca-params.conf
```

- 安装horvod（MPI形式的作业，需要安装horovod，并进行相关配置。horovod安装需要cmake3.13+，可以apt-get install 指定版本安装）

```
 RUN ldconfig /usr/local/cuda/targets/x86_64-linux/lib/stubs && \
    HOROVOD_GPU_ALLREDUCE=NCCL HOROVOD_WITH_PYTORCH=1  pip install --no-cache-dir horovod && \
    ldconfig
```

- 安装并配置nccl

```
 #### Set default NCCL parameters
RUN echo NCCL_DEBUG=INFO >> /etc/nccl.conf
```

- 安装openssh并配置

```
 #### Install OpenSSH for MPI to communicate between containers
RUN apt-get install -y --no-install-recommends openssh-client openssh-server && \
    mkdir -p /var/run/sshd

#### Allow OpenSSH to talk to containers without asking for confirmation
RUN cat /etc/ssh/ssh_config | grep -v StrictHostKeyChecking > /etc/ssh/ssh_config.new && \
    echo "    StrictHostKeyChecking no" >> /etc/ssh/ssh_config.new && \
    mv /etc/ssh/ssh_config.new /etc/ssh/ssh_config
```

pytorch/tf 镜像示例

```
 FROM cuda11.0.3-cudnn8-devel-ubuntu18.04

#### Configure time zone
RUN rm /etc/apt/sources.list.d/cuda.list && rm /etc/apt/sources.list.d/nvidia-ml.list
RUN apt-get update && \
    DEBIAN_FRONTEND="noninteractive" TZ="Asia/Shanghai" \
    apt-get install -y tzdata && \
    ln -snf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime && \
    dpkg-reconfigure -f noninteractive tzdata && \
    apt-get clean

#### cmake需要大于3.13版本
RUN apt-get install -y \
    build-essential \
    libopencv-dev \
```

```
    libssl-dev \
    dnsutils \
    unzip \
    vim \
    git \
    jq \
    curl \
    cmake \
    wget \
    ca-certificates \
    libjpeg-dev \
    libpng-dev \
    wget && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

#### py3
RUN bash -c 'cd /tmp && \
    wget --no-check-certificate https://repo.anaconda.com/miniconda/Miniconda3-py37_4.8.3-Linux-x86_64.sh && \
    bash Miniconda3-py37_4.8.3-Linux-x86_64.sh -b -p ~/miniconda3 && \
    echo "source ~/miniconda3/bin/activate" >> ~/.bashrc && \
    echo "export PATH="~/miniconda3/bin:${PATH}"" >> ~/.bashrc && \
    source ~/.bashrc  && \
    rm -rf Miniconda3-py37_4.8.3-Linux-x86_64.sh' && \
    /root/miniconda3/bin/python -m pip config set global.index-url https://pypi.douban.com/simple/ && \
    /root/miniconda3/bin/python -m pip install --upgrade pip && \
    /root/miniconda3/bin/python -m pip install --upgrade setuptools && \
    /root/miniconda3/bin/python -m pip install numpy==1.17.4 && \
    /root/miniconda3/bin/python -m pip install albumentations==0.4.3 && \
    /root/miniconda3/bin/python -m pip install Cython==0.29.16 && \
    /root/miniconda3/bin/python -m pip install pycocotools==2.0.0 && \
    /root/miniconda3/bin/python -m pip install ruamel.yaml && \
    /root/miniconda3/bin/python -m pip install ujson && \
    /root/miniconda3/bin/python -m pip install scikit-learn==0.23.2 && \
    /root/miniconda3/bin/python -m pip install pandas

RUN /root/miniconda3/condabin/conda clean -p && \
    /root/miniconda3/condabin/conda clean -t
RUN rm -rf ~/.cache/pip
RUN rm -rf /usr/bin/python3 && ln -s /root/miniconda3/bin/python /usr/bin/python3

#### torch
RUN /root/miniconda3/bin/python -m pip install torch==1.7.1+cu110 torchvision==0.8.2+cu110 torchaudio===0.7.2 \
    -f https://download.pytorch.org/whl/torch_stable.html

#### Install Open MPI 4.0.0
RUN mkdir /tmp/openmpi && \
    cd /tmp/openmpi && \
    wget https://download.open-mpi.org/release/open-mpi/v4.0/openmpi-4.0.0.tar.gz && \
    tar zxf openmpi-4.0.0.tar.gz && \
    cd openmpi-4.0.0 && \
    ./configure --enable-orterun-prefix-by-default && \
    make -j $(nproc) all && \
    make install && \
    ldconfig && \
    rm -rf /tmp/openmpi

#### Install Horovod, temporarily using CUDA stubs
#### /usr/local/cuda links to /usr/local/cuda-10.1
RUN ldconfig /usr/local/cuda/targets/x86_64-linux/lib/stubs && \
    HOROVOD_GPU_ALLREDUCE=NCCL HOROVOD_WITH_PYTORCH=1  pip install --no-cache-dir horovod && \
    ldconfig

#### Create a wrapper for OpenMPI to allow running as root by default
RUN mv /usr/local/bin/mpirun /usr/local/bin/mpirun.real && \
    echo '#!/bin/bash' > /usr/local/bin/mpirun && \
    echo 'mpirun.real --allow-run-as-root "$@"' >> /usr/local/bin/mpirun && \
    chmod a+x /usr/local/bin/mpirun

#### Configure OpenMPI to run good defaults:
```

```
#### --bind-to none --map-by slot --mca btl_tcp_if_exclude lo,docker0
RUN echo "hwloc_base_binding_policy = none" >> /usr/local/etc/openmpi-mca-params.conf && \
    echo "rmaps_base_mapping_policy = slot" >> /usr/local/etc/openmpi-mca-params.conf && \
    echo "btl_tcp_if_exclude = lo,docker0" >> /usr/local/etc/openmpi-mca-params.conf

#### Set default NCCL parameters
RUN echo NCCL_DEBUG=INFO >> /etc/nccl.conf

#### Install OpenSSH for MPI to communicate between containers
RUN apt-get install -y --no-install-recommends openssh-client openssh-server && \
    mkdir -p /var/run/sshd

#### Allow OpenSSH to talk to containers without asking for confirmation
RUN cat /etc/ssh/ssh_config | grep -v StrictHostKeyChecking > /etc/ssh/ssh_config.new && \
    echo "    StrictHostKeyChecking no" >> /etc/ssh/ssh_config.new && \
    mv /etc/ssh/ssh_config.new /etc/ssh/ssh_config

#### 安装bos
RUN /bin/bash -c 'mkdir /home/bos && \
    cd /home/bos && \
    wget --no-check-certificate https://sdk.bce.baidu.com/console-sdk/linux-bcecmd-0.3.0.zip && \
    unzip linux-bcecmd-0.3.0.zip && \
    echo "export LANG=en_US.UTF-8" >> ~/.bashrc && \
    echo "export PATH="/home/bos/linux-bcecmd-0.3.0:${PATH}"" >> ~/.bashrc && \
    source ~/.bashrc'

#### 添加搜索作业SDK install 3.20.1 protobuf for searchjob
COPY rudder-autosearch-1.0.0-py3-none-any.whl /home/rudder-autosearch-1.0.0-py3-none-any.whl
RUN pip install /home/rudder-autosearch-1.0.0-py3-none-any.whl && \
    pip install protobuf==3.20.1

ENV NVIDIA_VISIBLE_DEVICES ""

ENTRYPOINT ["/bin/bash"]
```

在BML使用外部镜像

资源池中除了可以关联容器镜像服务CCR外，用户也可以在资源池的镜像仓库中管理外部镜像。支持使用外部镜像的服务包括：

- 作业建模
- 在线服务
- 预测服务

外部镜像添加步骤

点击平台管理，进入资源池管理。



点击列表页的镜像仓库，进入镜像仓库列表中。



点击添加镜像仓库，用户输入外部镜像的镜像地址和账号密码后，即保存在该资源池下的镜像仓库列表中。



使用外部镜像

在训练作业、自动搜索作业、在线服务、批量预测中，选择用户资源池后，可使用外部镜像发起训练/预测。





在BML平台使用并行文件系统PFS和对象存储BOS

在BML平台使用并行文件系统PFS

平台支持用户在用户资源池上关联并行文件存储PFS作为建模任务时的数据存储，当前支持使用并行文件系统PFS提交的任务：

- 自定义作业-训练作业任务、自动搜索作业任务

前提条件

1. 用户在平台上已经挂载了容器引擎CCE资源作为用户资源池，点击了解容器引擎CCE；

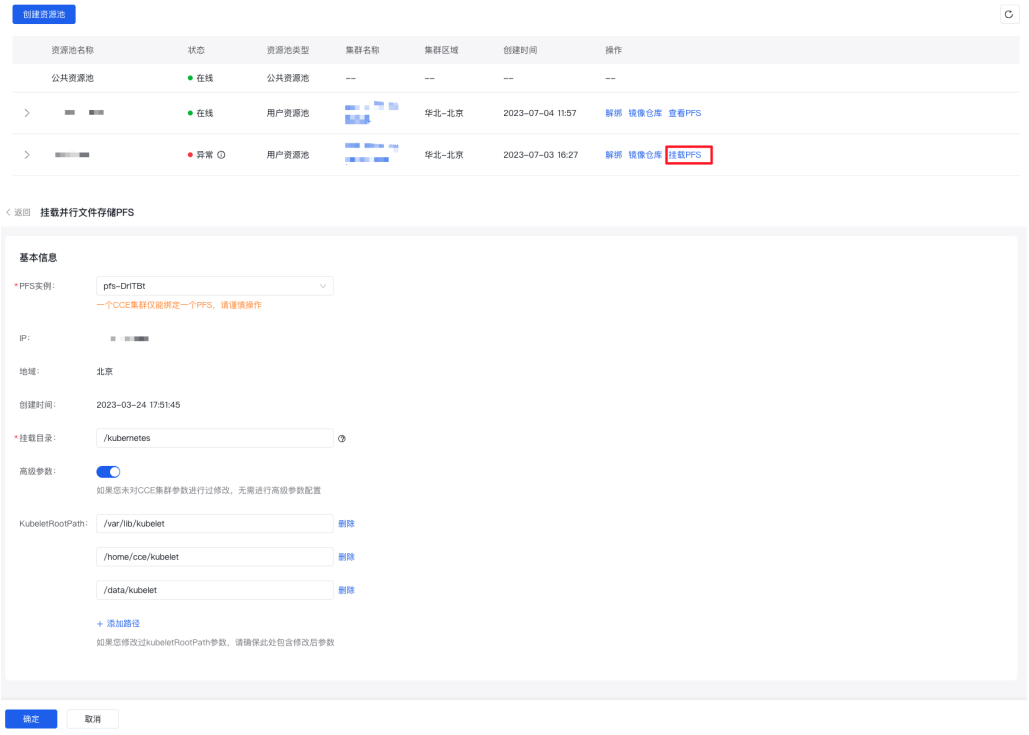


2. 用户已经创建了并行文件系统PFS，点击了解并行文件系统PFS。

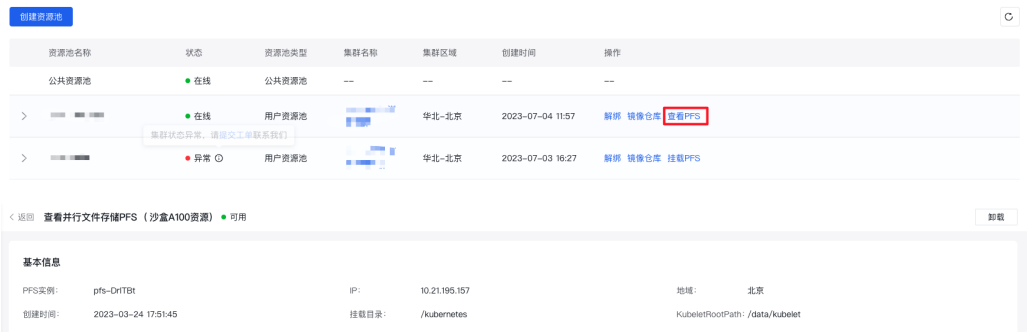
3. 并行文件系统PFS能够被容器引擎CCE资源访问到，也即能被对应的VPC访问到。

在用户资源池中挂载PFS实例

- Step1:进入平台管理-资源池管理，已挂载运行正常的用户资源池支持“挂载PFS”的操作项，点击即可选择挂载的PFS实例。



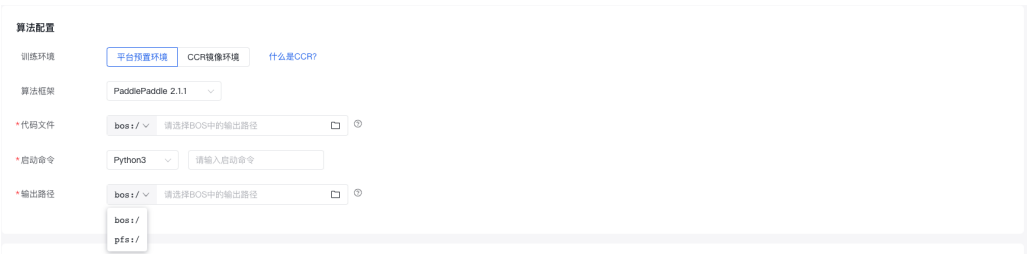
- Step2:完成引入PFS后，支持点击“查看PFS”，并支持卸载操作。



使用PFS作为数据来源提交作业建模任务

在提交作业建模任务时，选择用户资源池后，支持选择该资源池中引入的BOS存储或者PFS存储作为数据来源。

挂载数据和编辑代码时需要注意：平台仅能使用挂载路径下的文件，请确保您的相关文件在此挂载路径之下。



操作及代码示例

前提条件

平台上的用户资源池已经成功挂载PFS。

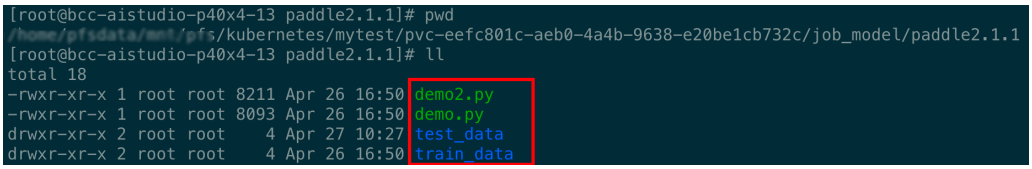
操作

依次点击 自定义作业 -> 训练作业 -> 新建

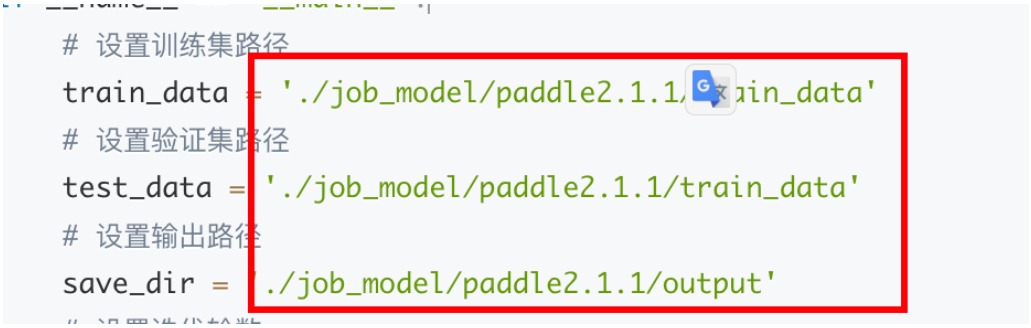
选择了用户资源池之后，算法配置信息填写如下：



以代码文件为例： 找到机器中pfs的挂载点，将代码及数据集上传到pfs挂载中，代码文件中只需要填写基于挂载路径的相对路径即可。



demo2.py内容附后，为官网提供的多节点paddle2.1.1的demo案例，我们只需要修改输出路径为pfs中基于挂载路径的相对路径即可。



代码示例

```
 #### !/usr/bin/env python
#### -*- coding: utf-8 -*-
"""
"""
import os
import gzip
import struct
import numpy as np
from PIL import Image
import time
import paddle
import paddle.distributed.fleet as fleet
```

```
import paddle.static.nn as nn
import paddle.fluid as fluid
from paddle.io import Dataset
TEST_IMAGE = 't10k-images-idx3-ubyte.gz'
TEST_LABEL = 't10k-labels-idx1-ubyte.gz'
TRAIN_IMAGE = 'train-images-idx3-ubyte.gz'
TRAIN_LABEL = 'train-labels-idx1-ubyte.gz'
class MNIST(Dataset):
    """
    MNIST
    """
    def __init__(self,
                 data_dir=None,
                 mode='train',
                 transform=None,
                 backend=None):
        assert mode.lower() in ['train', 'test'],                 "mode should be 'train' or 'test', but got {}".format(mode)
        if backend is None:
            backend = paddle.vision.get_image_backend()
        if backend not in ['pil', 'cv2']:
            raise ValueError(
                "Expected backend are one of ['pil', 'cv2'], but got {}"
                .format(backend))
        self.backend = backend
        self.mode = mode.lower()
        if self.mode == 'train':
            self.image_path = os.path.join(data_dir, TRAIN_IMAGE)
            self.label_path = os.path.join(data_dir, TRAIN_LABEL)
        else:
            self.image_path = os.path.join(data_dir, TEST_IMAGE)
            self.label_path = os.path.join(data_dir, TEST_LABEL)
        self.transform = transform
        # read dataset into memory
        self._parse_dataset()
        self.dtype = paddle.get_default_dtype()
    def _parse_dataset(self, buffer_size=100):
        self.images = []
        self.labels = []
        with gzip.GzipFile(self.image_path, 'rb') as image_file:
            img_buf = image_file.read()
            with gzip.GzipFile(self.label_path, 'rb') as label_file:
                lab_buf = label_file.read()
                step_label = 0
                offset_img = 0
                # read from Big-endian
                # get file info from magic byte
                # image file : 16B
                magic_byte_img = '>IIII'
                magic_img, image_num, rows, cols = struct.unpack_from(
                    magic_byte_img, img_buf, offset_img)
                offset_img += struct.calcsize(magic_byte_img)
                offset_lab = 0
                # label file : 8B
                magic_byte_lab = '>II'
                magic_lab, label_num = struct.unpack_from(magic_byte_lab,
                                                          lab_buf, offset_lab)
                offset_lab += struct.calcsize(magic_byte_lab)
                while True:
                    if step_label >= label_num:
                        break
                    fmt_label = '>' + str(buffer_size) + 'B'
                    labels = struct.unpack_from(fmt_label, lab_buf, offset_lab)
                    offset_lab += struct.calcsize(fmt_label)
                    step_label += buffer_size
                    fmt_images = '>' + str(buffer_size * rows * cols) + 'B'
                    images_temp = struct.unpack_from(fmt_images, img_buf,
                                                     offset_img)
                    images = np.reshape(images_temp, (buffer_size, rows *
                                                      cols)).astype('float32')
                    offset_img += struct.calcsize(fmt_images)
```

```
                for i in range(buffer_size):
                    self.images.append(images[i, :])
                    self.labels.append(
                        np.array([labels[i]]).astype('int64'))
    def __getitem__(self, idx):
        image, label = self.images[idx], self.labels[idx]
        image = np.reshape(image, [28, 28])
        if self.backend == 'pil':
            image = Image.fromarray(image.astype('uint8'), mode='L')
        if self.transform is not None:
            image = self.transform(image)
        if self.backend == 'pil':
            return image, label.astype('int64')
        return image.astype(self.dtype), label.astype('int64')
    def __len__(self):
        return len(self.labels)
def mlp_model():
    """
    mlp_model
    """
    x = paddle.static.data(name="x", shape=[64, 28, 28], dtype='float32')
    y = paddle.static.data(name="y", shape=[64, 1], dtype='int64')
    x_flatten = paddle.reshape(x, [64, 784])
    fc_1 = nn.fc(x=x_flatten, size=128, activation='tanh')
    fc_2 = nn.fc(x=fc_1, size=128, activation='tanh')
    prediction = nn.fc(x=[fc_2], size=10, activation='softmax')
    cost = paddle.fluid.layers.cross_entropy(input=prediction, label=y)
    acc_top1 = paddle.metric.accuracy(input=prediction, label=y, k=1)
    avg_cost = paddle.mean(x=cost)
    res = [x, y, prediction, avg_cost, acc_top1]
    return res
def train(epoch, exe, train_dataloader, cost, acc):
    """
    train
    """
    total_time = 0
    step = 0
    for data in train_dataloader():
        step += 1
        start_time = time.time()
        loss_val, acc_val = exe.run(
        paddle.static.default_main_program(),
        feed=data, fetch_list=[cost.name, acc.name])
        if step % 100 == 0:
            end_time = time.time()
            total_time += (end_time - start_time)
            print(
                    "epoch: %d, step:%d, train_loss: %f, train_acc: %f, total time cost = %f, speed: %f"
                % (epoch, step, loss_val[0], acc_val[0], total_time,
                1 / (end_time - start_time) ))
def test(exe, test_dataloader, cost, acc):
    """
    test
    """
    total_time = 0
    step = 0
    for data in test_dataloader():
        step += 1
        start_time = time.time()
        loss_val, acc_val = exe.run(
        paddle.static.default_main_program(),
        feed=data, fetch_list=[cost.name, acc.name])
        if step % 100 == 0:
            end_time = time.time()
            total_time += (end_time - start_time)
            print(
                    "step:%d, test_loss: %f, test_acc: %f, total time cost = %f, speed: %f"
                % (step, loss_val[0], acc_val[0], total_time,
                1 / (end_time - start_time) ))
def save(save_dir, feed_vars, fetch_vars, exe):
```

```
    """
    save
    """
    path_prefix = os.path.join(save_dir, 'model')
    if fleet.is_first_worker():
        paddle.static.save_inference_model(path_prefix, feed_vars, fetch_vars, exe)
if __name__ == '__main__':
    # 设置训练集路径
    train_data = './job_model/paddle2.1.1/train_data'
    # 设置验证集路径
    test_data = './job_model/paddle2.1.1/train_data'
    # 设置输出路径
    save_dir = './job_model/paddle2.1.1/output'
    # 设置迭代轮数
    epochs = 10
    # 设置验证间隔轮数
    test_interval = 2
    # 设置模型保存间隔轮数
    save_interval = 2
    paddle.enable_static()
    paddle.vision.set_image_backend('cv2')
    # 训练数据集
    train_dataset = MNIST(data_dir=train_data, mode='train')
    # 验证数据集
    test_dataset = MNIST(data_dir=test_data, mode='test')
    # 设置模型
    [x, y, pred, cost, acc] = mlp_model()
    place = paddle.CUDAPlace(int(os.environ.get('FLAGS_selected_gpus', 0)))
    # 数据加载
    train_dataloader = paddle.io.DataLoader(
        train_dataset, feed_list=[x, y], drop_last=True,
        places=place, batch_size=64, shuffle=True, return_list=False)
    test_dataloader = paddle.io.DataLoader(
        test_dataset, feed_list=[x, y], drop_last=True,
        places=place, batch_size=64, return_list=False)
    # fleet初始化
    strategy = fleet.DistributedStrategy()
    fleet.init(is_collective=True, strategy=strategy)
    # 设置优化器
    optimizer = paddle.optimizer.Adam()
    optimizer = fleet.distributed_optimizer(optimizer)
    optimizer.minimize(cost)
    exe = paddle.static.Executor(place)
    exe.run(paddle.static.default_startup_program())
    prog = paddle.static.default_main_program()
    for epoch in range(epochs):
        train(epoch, exe, train_dataloader, cost, acc)
        if epoch % test_interval == 0:
            test(exe, test_dataloader, cost, acc)
        # save model
        if epoch % save_interval == 0:
            save(save_dir, [x], [pred], exe)
```

资源配置 （依据代码情况填写即可，上文中的案例用到了多节点和GPU资源）

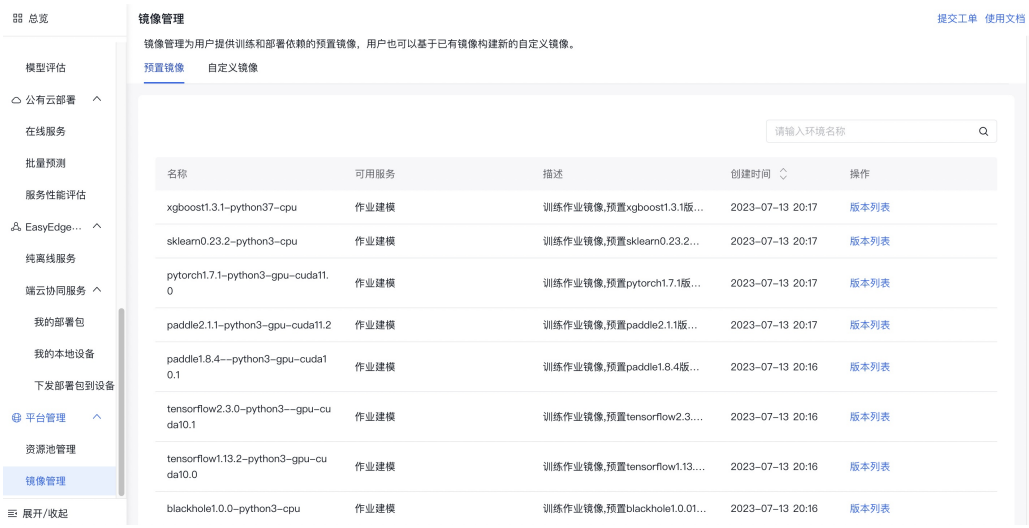


镜像管理

镜像管理简介

镜像管理简介 BML镜像管理为用户预置多种框架的训练镜像。当预置镜像不满足需求，需要进行扩展时，镜像管理提供自定义镜像的能力。

目前BML镜像管理已预置作业建模和Notebook建模的镜像，点击"预置镜像"可查看镜像的名称以及镜像可用服务。

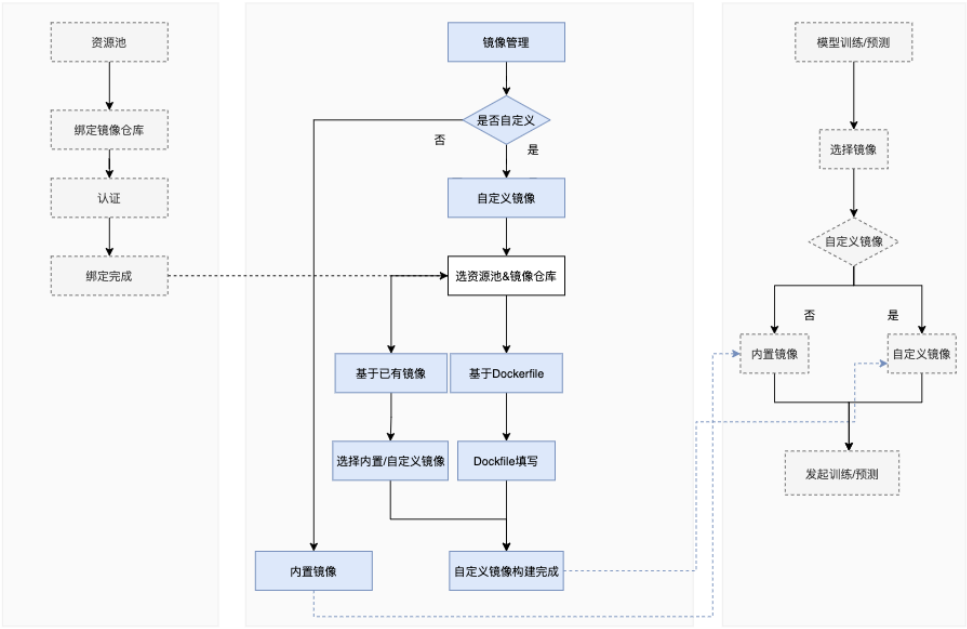


如果平台预置镜像不满足训练或预测需求，可基于平台构建自定义镜像。 点击"自定义镜像"，可查看自定义镜像列表。并可以新建镜像。



镜像使用流程

- 第一步：在资源池中绑定镜像仓库；
- 第二步：在镜像管理中构建镜像；
- 第三步：在训练或预测中使用自定义镜像。



关联产品

需要注意的是：构建的镜像存储在百度容器镜像服务CCR中。您在镜像管理中构建镜像之前，需要先在百度CCR镜像服务中开通企业版镜像服务，创建实例。并在资源池管理中绑定镜像仓库。

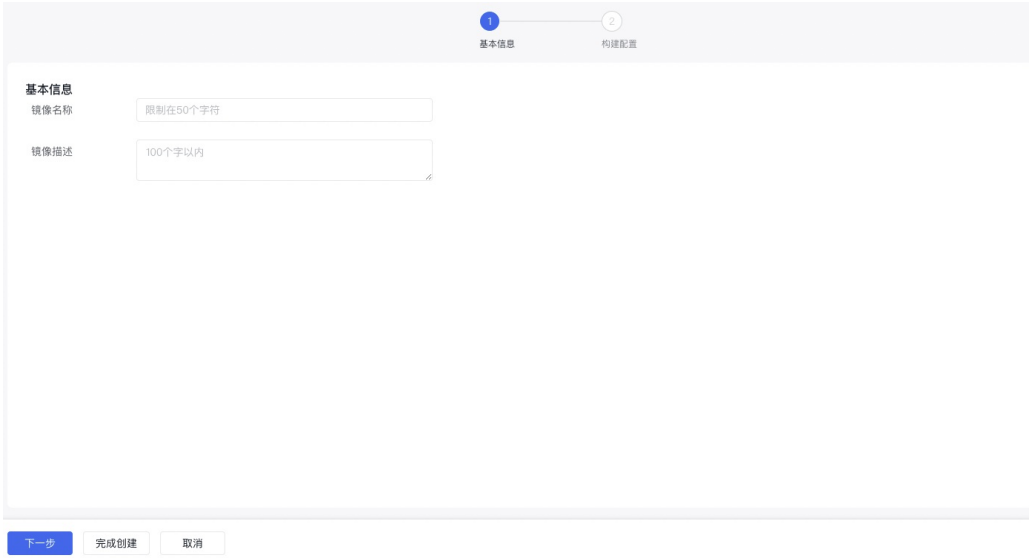
百度CCR：百度智能云容器镜像服务（Cloud Container Registry，简称CCR）是分布式全托管的容器镜像管理服务。您可以在CCR中创建自己专属的命名空间，用于对Docker镜像或Helm Charts进行上传、存储、下载、版本管理、安全扫描等操作。

自定义镜像

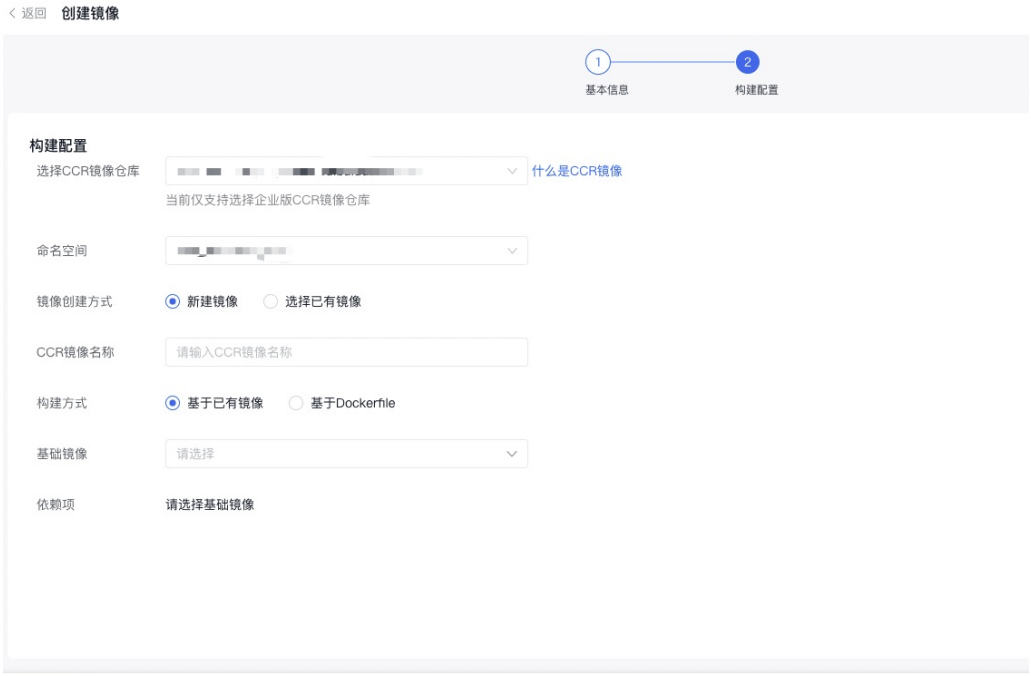
如果平台预置的镜像不满足您的开发需求，需要新增依赖或引入外部镜像时，您可通过自定义镜像构建新的镜像。

构建自定义镜像

点击"自定义镜像"，填写镜像名称和镜像描述。



点击下一步，配置镜像的构建信息。





其中，填写的字段及含义如下：

字段	含义
选择CCR镜像仓库	选择的是在资源池中绑定的企业版CCR镜像仓库
命名空间	镜像仓库的命名空间
新建镜像	在命名空间下新增一条镜像
选择已有镜像	在命名空间的已有镜像中新增一个版本
CCR镜像名称	新建或选择已有镜像的镜像名称
构建方式	包括基于已有镜像和基于Dockfile两种方式
基于已有镜像	基于平台的预置镜像及基于预置镜像构建的自定义镜像。目前暂不支持基于预置的Notebook镜像构建自定义镜像
基于Dockfile	通过填写Dockfile内容构建新的镜像。目前暂不支持基于Dockfile构建的镜像用于Notebook
依赖项	可添加pip3、apt依赖。pip3默认源地址：https://pypi.douban.com/simple/ apt默认源地址：http://mirrors.tuna.tsinghua.edu.cn/ubuntu

基于已有镜像

选择基于已有镜像时，需要选择基础镜像和依赖项。 其中，依赖项内容支持输入框填写，或者上传requirment文件。输入框输入时，可直接写包名，或者指定包的版本，并换行添加下一个依赖包。平台已有默认包的地址，如果要更换地址，可直接 -i https://pypi.xxxxx.com/simple/ 指定地址即可。



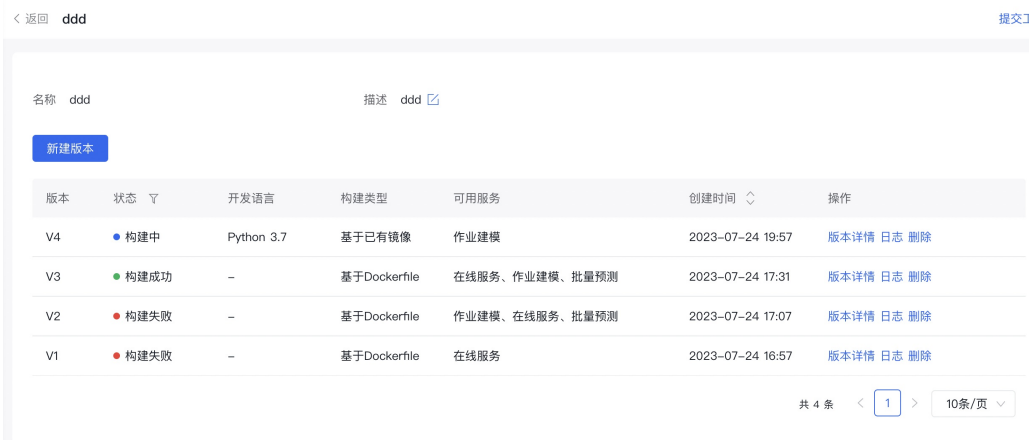
基于Dockerfile

如果内置的镜像不满足需求，或需要使用Dockfile的方式构建镜像时，可选择基于Dockerfile。 可选服务包括作业建模、在线服务、批量预测三种。勾选对应服务后，创建的镜像可以在对应服务中使用。

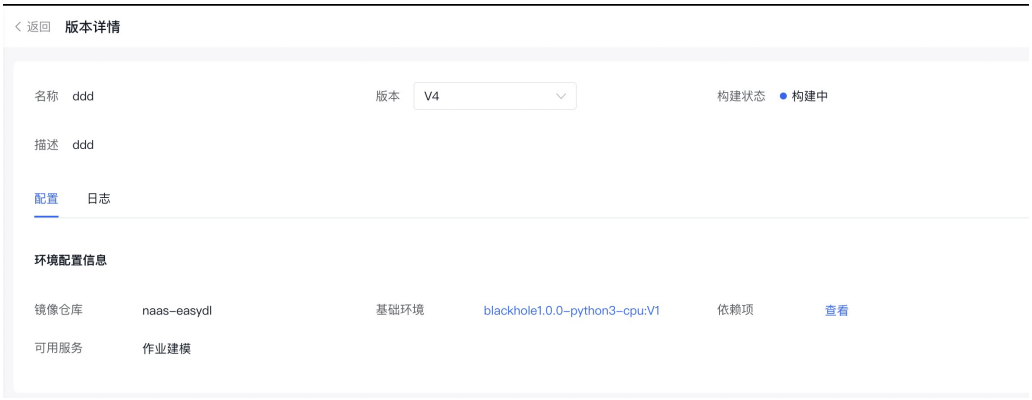


其中，如果构建的镜像需要用于作业建模，需要遵循作业建模的镜像构建规范。

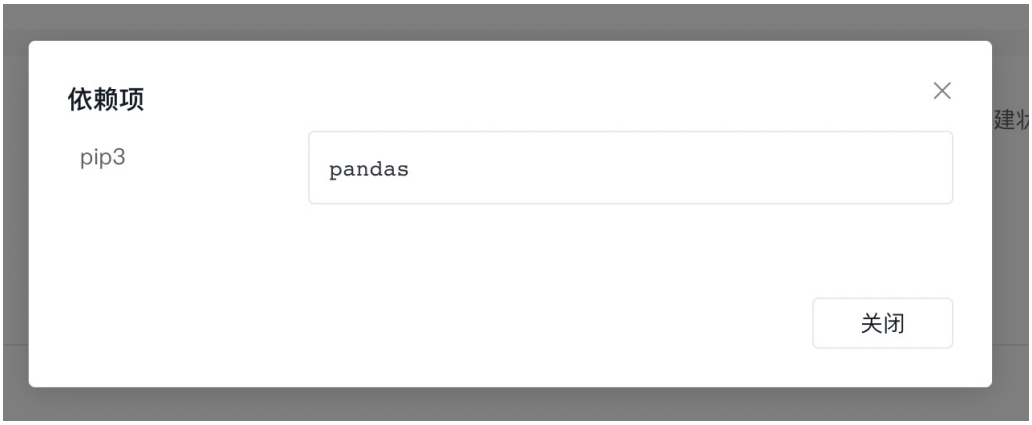
点击完成构建，则提交镜像构建的任务。 **镜像版本列表** 在版本列表中，可查看历史构建的各版本镜像。



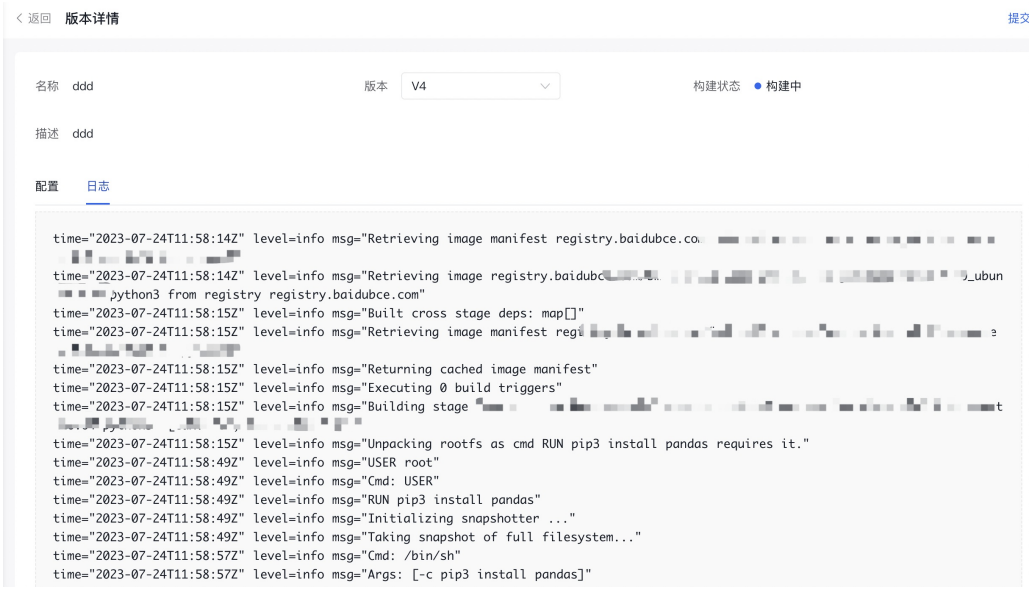
点击"版本详情"，可查看镜像的配置信息。



点击"依赖项"，可查看改镜像添加的依赖。



点击"日志"，可查看镜像构建过程的日志。



镜像使用

镜像构建后，可以在作业建模、在线服务、批量预测中使用。

作业建模中使用 创建作业建模任务，选择用户资源池时，可以选择自定义镜像。

选择预置镜像：



选择自定义镜像，可以看到可使用的镜像列表。其中自定义镜像需要按照镜像规范进行构建。



其中，可选择的自定义镜像，是和选择的用户资源池中的区域相同的镜像仓库中的镜像。

在线服务中使用

部署在线服务时，资源池类型选择用户资源池后，可选择自定义镜像。



批量预测中使用

新建批量预测任务时，资源池类型选择用户资源池后，可选择自定义镜像。



常见问题

1、如何构建Notebook的自定义镜像？

目前暂不支持构建Notebook的自定义镜像。

2、构建的作业建模镜像，创建作业建模任务时显示失败，怎么办？

如果是通过Dockfile构建的，请先排查是否按照作业建模的镜像规范构建的自定义镜像。

3、基于已有镜像构建新镜像时，要安装的包为什么获取失败？

对于作业建模，pip3默认源地址：https://pypi.douban.com/simple/， apt默认源地址：http://mirrors.tuna.tsinghua.edu.cn/ubuntu。 需要确认该源下是否有要安装的包。

4、如果需要apt-get update，怎么处理？

基于已有镜像安装apt依赖时，不支持apt-get update。如果需要使用，请选择Dockerfile构建方式。

资源管理

资源池管理简介

功能说明

BML全功能AI开发平台提供了公共资源池、专属资源池及用户资源池供您使用！您或者平台资源管理员相关角色可在资源管理模块进行管理。

公共资源池

公共资源池作为平台提供的默认资源池，可用于在训推场景提交CPU&GPU任务；支持按量后付费和资源包方式预付费购买；详见模型训练计费说明 公有云部署计费说明



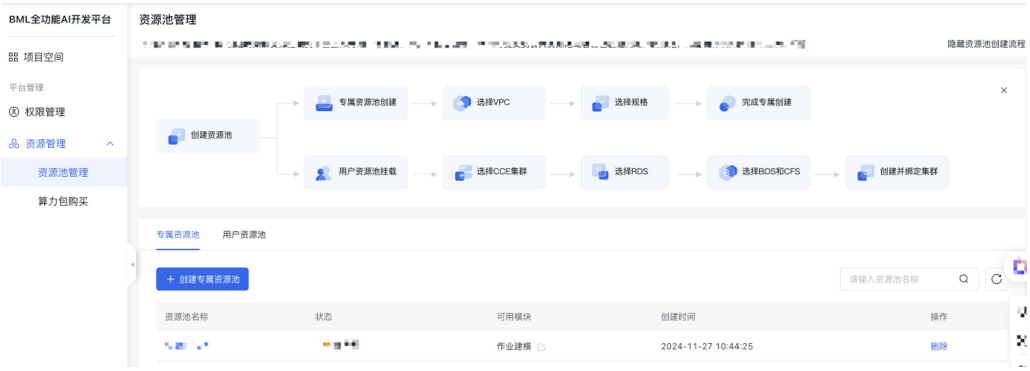


注：平台会为所有的用户提供一个公共资源池，以确保用户可以快速的上手和使用平台，公共资源池不可操作。

专属资源池

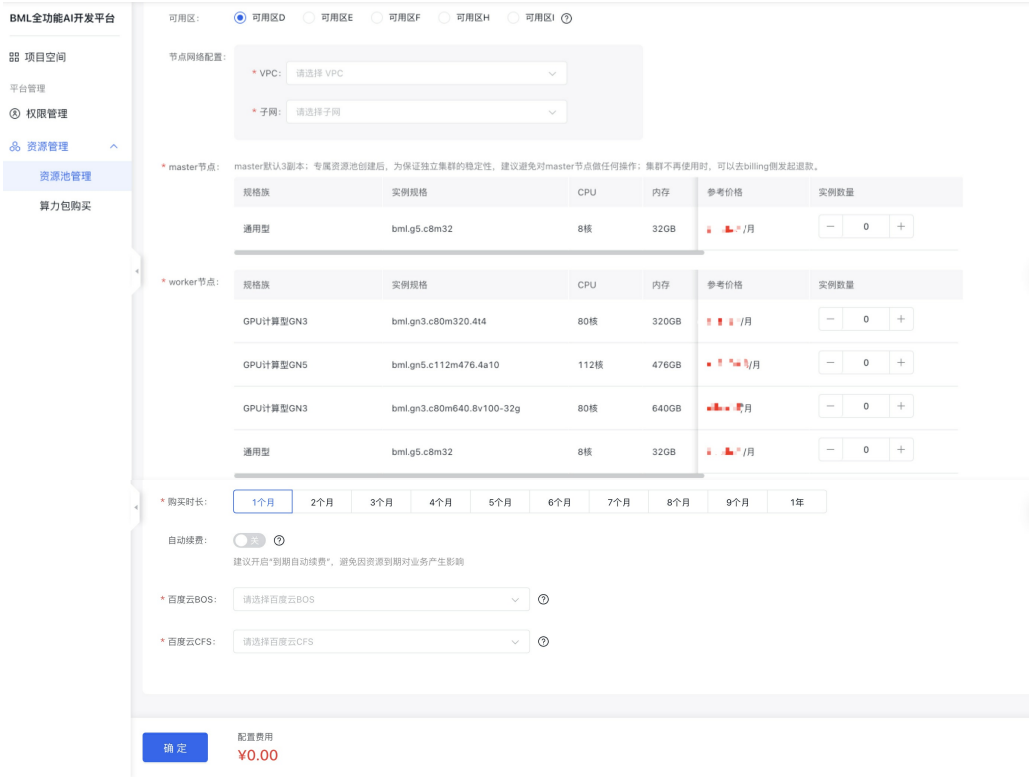
提供专属资源池，当您需要弹性、稳定且弱运维场景，可以在平台购买专属资源池供您使用，来确保训练资源的充足与可控。

专属资源池可支持在平台查看该资源池下资源利用率情况，可在创建好后通过列表操作-监控查看。



创建专属资源池





用户资源池

用户资源池是将您在容器引擎服务CCE创建的集群资源挂载到平台上，作为用户资源池。在提交训练任务时可以提交到自己创建的用户资源池。



相关依赖

资源池管理功能对一些相关的云产品有依赖，用户在使用之前需要事先准备好。

1. 私有化VPC网络

私有网络VPC，是基于百度智能云构建的安全隔离的网络环境，支持用户可自定义网络地址空间、多VPC之间（同城、跨城）对等高速连接，通过VPN/专线的方式，与用户的数据中心构建安全、定制的混合云网络，实现原有业务轻松、安全的迁移到云端。

CCE集群对于VPC网络有依赖性，所以用户需要事先创建私有化VPC网络实例，才可使用CCE集群。

2. 容器引擎CCE集群

容器引擎CCE集群可以提供Docker容器生命周期管理、大规模容器集群运维管理、业务应用一键式发布运行等功能，无缝衔接百度智能云其他产品。弹性、高可用的云端Kubernetes容器运行平台，助力系统架构微服务化、DevOps运维、AI应用深度学习容器化等场景。

资源池管理所使用的资源池即是由CCE所提供的，所以用户需要事先创建容器引擎CCE集群，才可使用资源池管理进行挂载。

注：CCE集群有地域之分，平台暂仅支持中国大陆的CCE集群。另外，CCE集群的地域需要与所使用的数据库等的地域保持一致，否则无法在平台上进行协同。且BML与CCE的协同当前仅支持Kubernetes的1.18.9版本。

3. RDS云数据库

RDS云数据库提供专业的高性能、高可靠关系型云数据库服务，提供简易方便的 Web 界面管理、可靠的数据备份和恢复功能、完备的安全管理

和监控等服务。

CCE集群在BML平台使用时需要的paddleflow组件对于RDS云数据库有依赖性，所以用户需要事先创建RDS云数据库，才可使用CCE集群。

4. BOS对象存储

百度智能云对象存储BOS是一款稳定、安全、高效、高可拓展的云存储服务，支持标准、低频、冷和归档存储等多种存储类型满足多场景的存储需求。用户可以将任意数量和形式的非结构化数据存入BOS，并对数据进行管理和处理。

在挂载资源池后，用户在使用资源池时，所使用的数据集需要保存在BOS中进行调用，故而用户需要事先创建自己的BOS对象存储，才可以在平台上提交相关的训练任务。

注：BOS有地域之分，BOS所在的地域需要与所使用的CCE集群的地域保持一致，否则无法在平台上进行协同。

5. CFS文件存储

文件存储CFS(Cloud File Storage)是百度智能云提供的安全、可扩展的文件存储服务。通过标准的文件访问协议，为云上的虚机、容器等计算资源提供无限扩展、高可靠、全球共享的文件存储能力。可用于内容管理及Web应用、大数据分析、泛娱乐和媒体应用等业务场景。

用户在使用资源池时，平台会将训练过程中的中间文件存储在CFS文件存储中，故而用户需要事先创建自己的CFS文件存储，才可以在平台上提交相关的训练任务。

6. 百度云账户

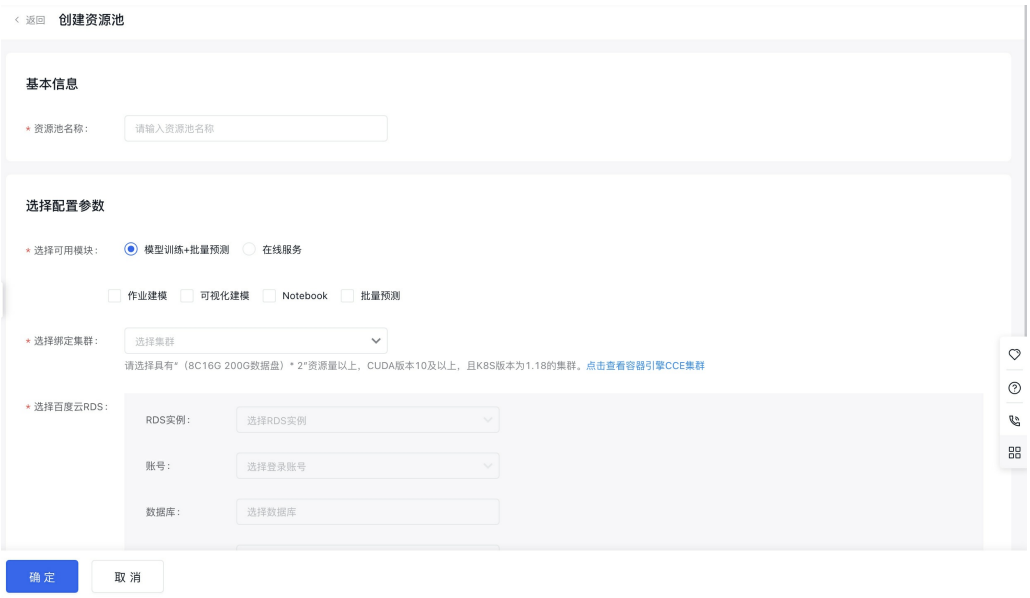
由于 VPC、CCE、RDS、BOS、CFS 均需要注册百度云账户才可使用。另外，如果用户希望多人使用，需要在百度云中创建子账号，子账号也可在 BML 平台上使用主账号所拥有的资产。

创建资源池

点击创建资源池，进入创建流程。



填写相关信息，并点击创建。



解绑资源池

点击即可解绑资源池。



状态说明

创建资源池后可能会出现如下的状态：

- 初始化：点击创建后，平台将根据用户所填写的信息进行初始化，这个过程会持续几分钟或者十几分钟，请稍事等待。
- 在线：资源池已经创建成功，用户可以正常使用。
- 异常：资源池创建过程中出现异常，或者资源池使用过程中出现异常，原因可能是所依赖的 VPC、CCE、RDS、BOS、CFS 地域不一致，或者某一个服务中断等，由于此处发生异常的可能性比较多，如果出现异常状态，请用户提交工单找到我们的工程师协助进行排查。
- 解绑中：用户对某个资源池进行解绑操作后显示此状态。



资源池使用简介

功能说明

资源池管理中所挂载的资源池可以在平台上通过资源分配的方式进行使用。

资源分配说明

入口：资源池管理下已挂载且处于在线状态的用户资源池操作栏

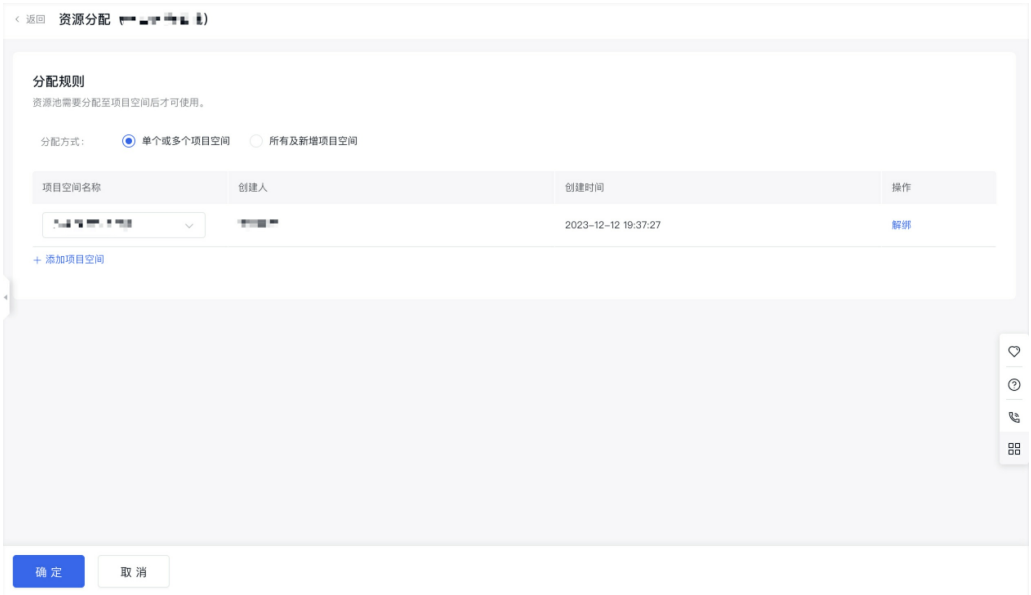
可操作角色：当前区域（region）下的超级管理员、资源管理员



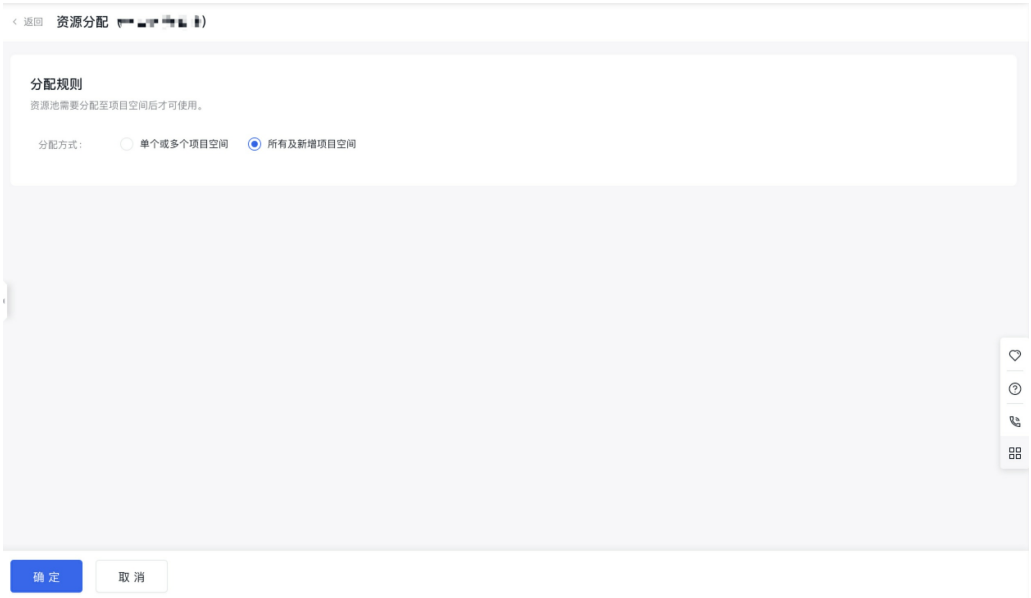
点击资源分配，进入到资源分配页面；资源池需要分配至项目空间后才可使用。

资源分配分为两种方式：单个或多个项目空间、所有及新增项目空间，具体说明如下：

1、单个或多个项目空间：当选择此分配方式时，需要先选择要分配的项目空间，选定后并确定，后续已选择项目在提交任务时将可使用此用户资源池进行调度。见下方业务使用说明。



2、所有及新增项目空间：当选择此分配方式时，代表该用户资源池对当前区域下的所有项目及后续新增的项目都生效，所有项目在任务提交时均可选择该资源池，并进行相应任务调度。因该可使用范围比较大，请谨慎分配。



业务使用说明

作业建模

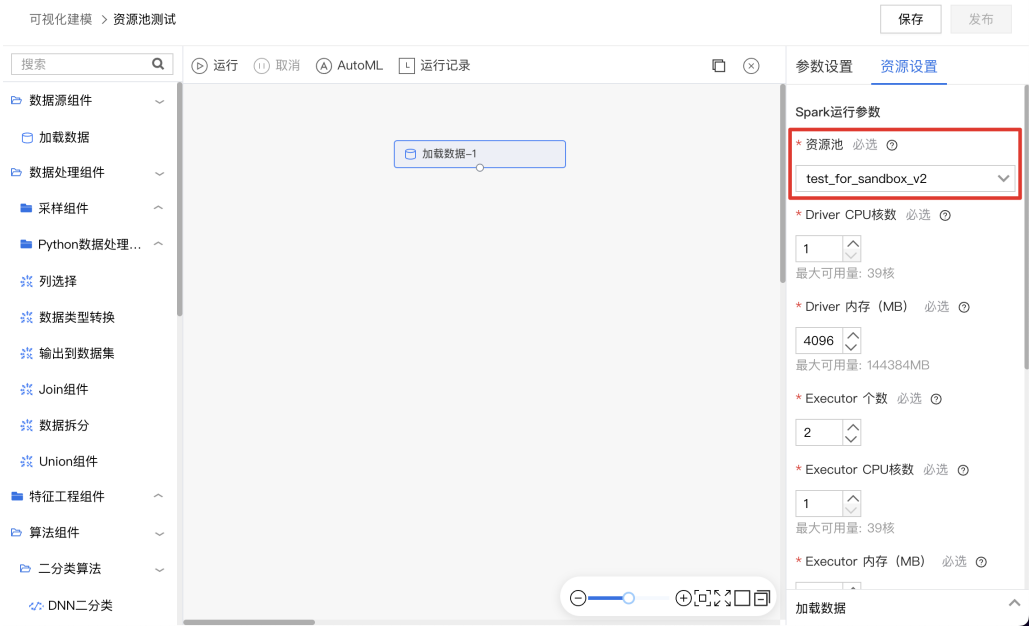
点击进入作业建模（训练作业和搜索作业都一致），在创建过程中支持选择已分配的用户资源池进行创建。需要注意的是，创建作业时选定的BOS地址需要与所使用的资源池处于同一个区域（Region）。



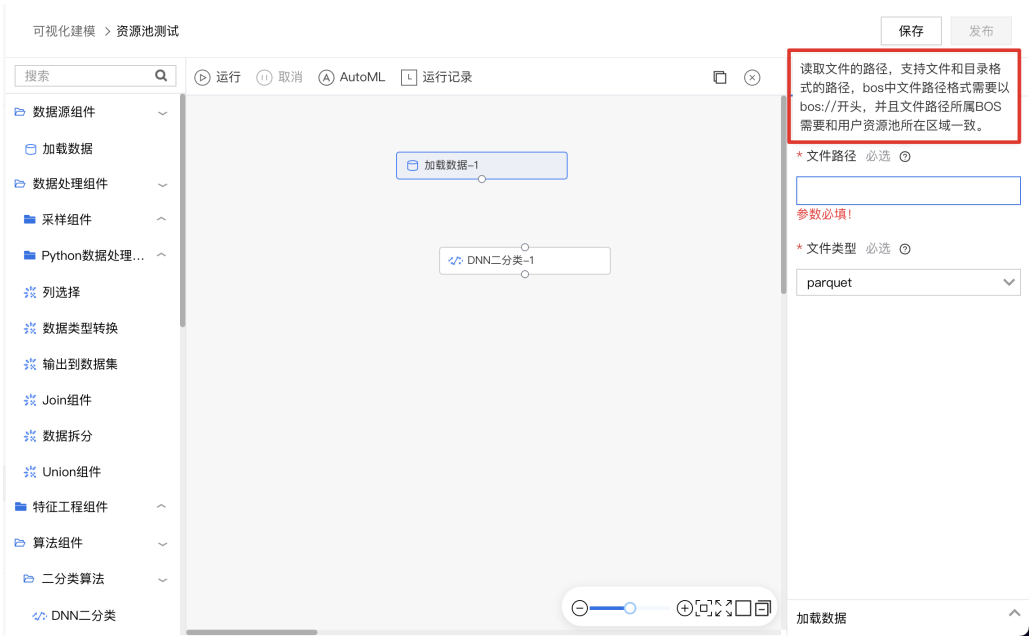
可视化建模

新建可视化建模画布时选择资源池，在编辑画布的过程中即可在相应的资源池下提交任务。需要注意的是，创建任务时选定的BOS地址需要与所使用的资源池处于同一个区域（Region）。





选择用户资源池后，训练数据需要指定自己的BOS路径。

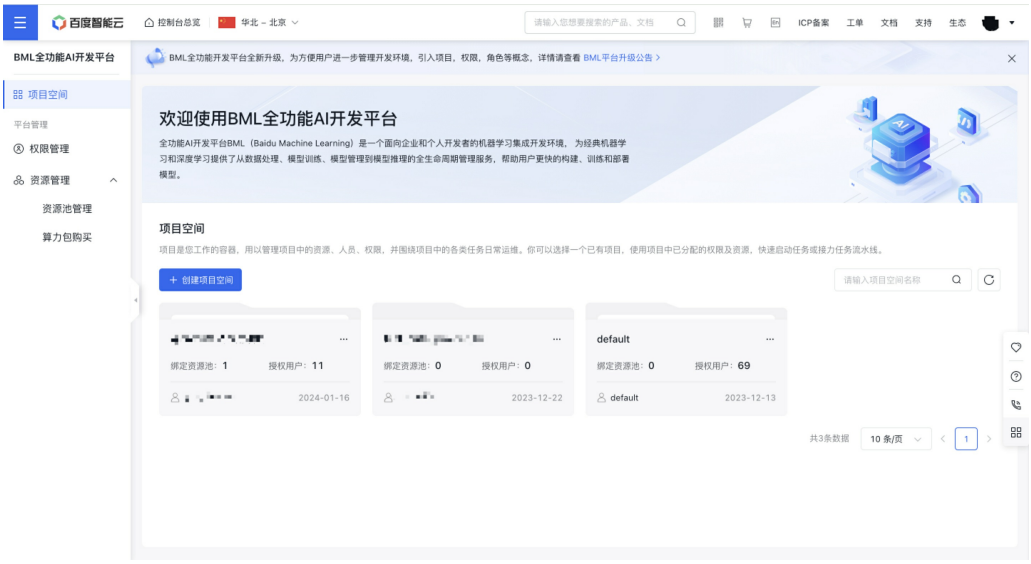


项目空间管理

项目空间

功能说明

项目空间是区域下为了完成同一AI目标而进行的团队协作任务的（数据处理-模型训练-模型管理-模型服务）集合；可按工作范围或目标划分；计算资源通过分配方式授权项目使用，企业/团队成员以某种角色加入到项目中。



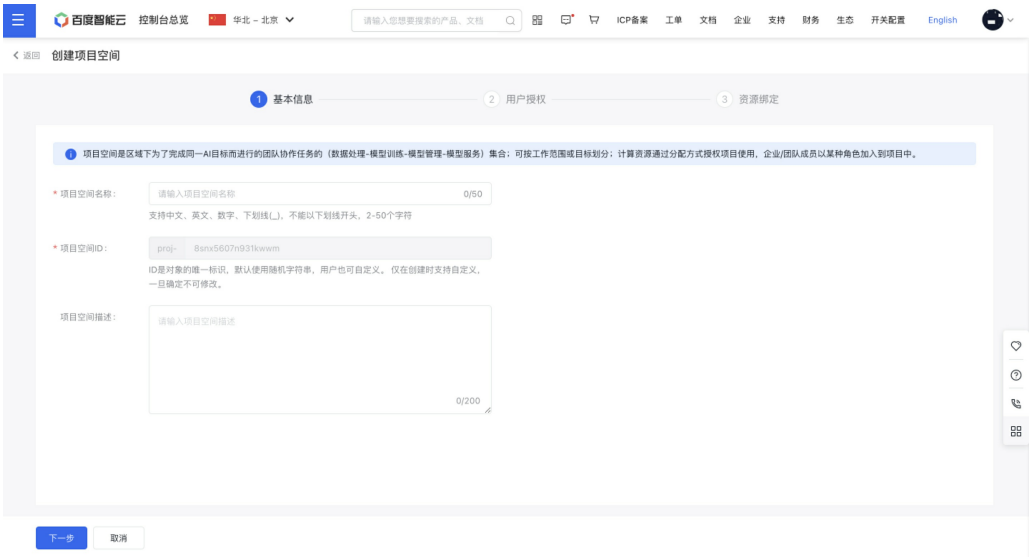
您可以根据资源分配情况或区域要求，切换到对应区域下进行项目空间管理。

华北-北京区域下提供默认项目空间default供开发者角色子用户快速使用。

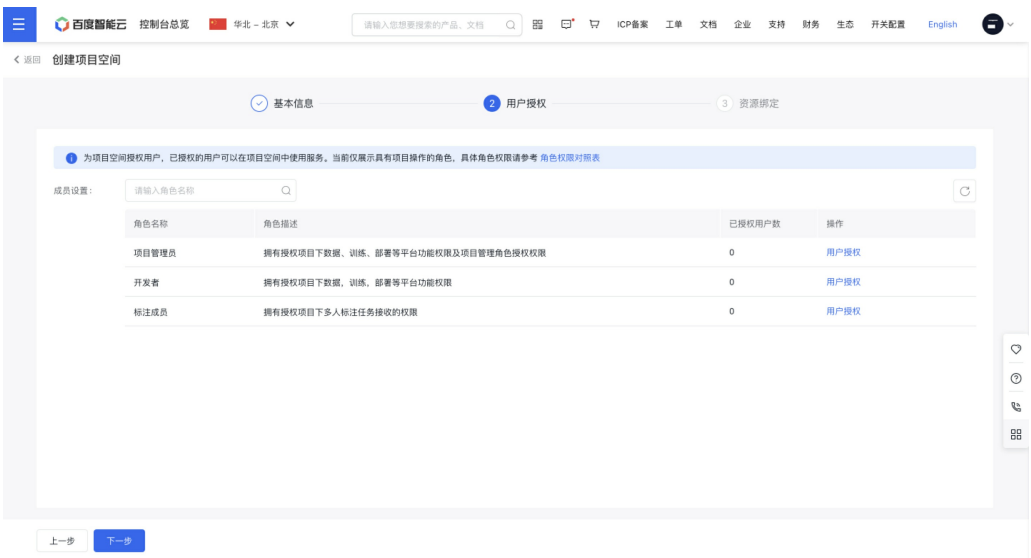
创建项目空间

点击项目空间列表上方【创建项目空间】，可进行创建操作；创建流程可分为以下三步：

Step1：基本信息填写；包含创建项目空间所需要的名称和描述等。

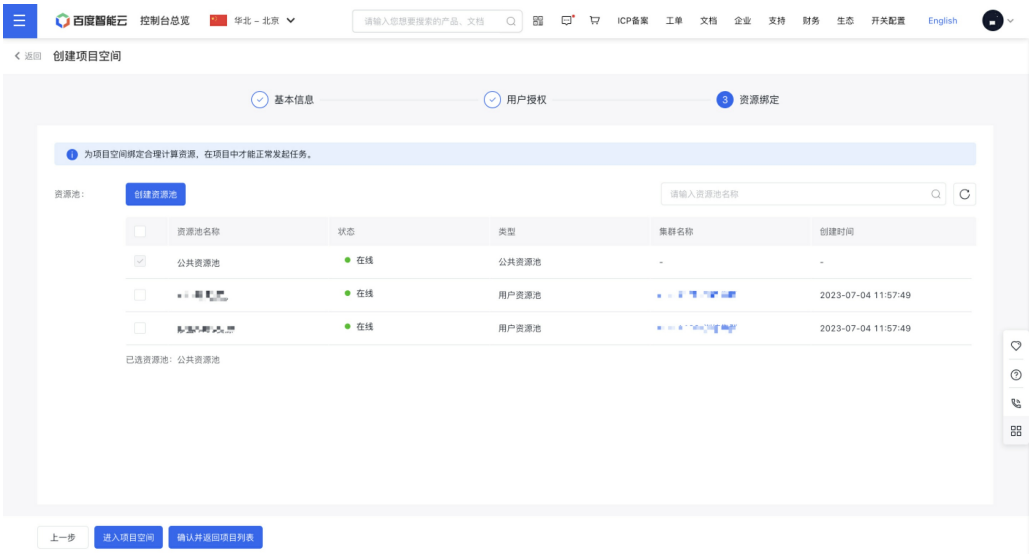


Step2：用户授权；可按内置角色权限进行项目空间内用户访问授权，在项目空间创建流程里可快速对项目管理员、开发者、标注成员授权，详见权限管理



Step3：资源绑定；可支持对当前region（区域）下已创建资源池进行绑定分配操作，绑定后即可在项目空间下进行任务提交时选到，以此来实现不同项目空间的资源隔离需求。

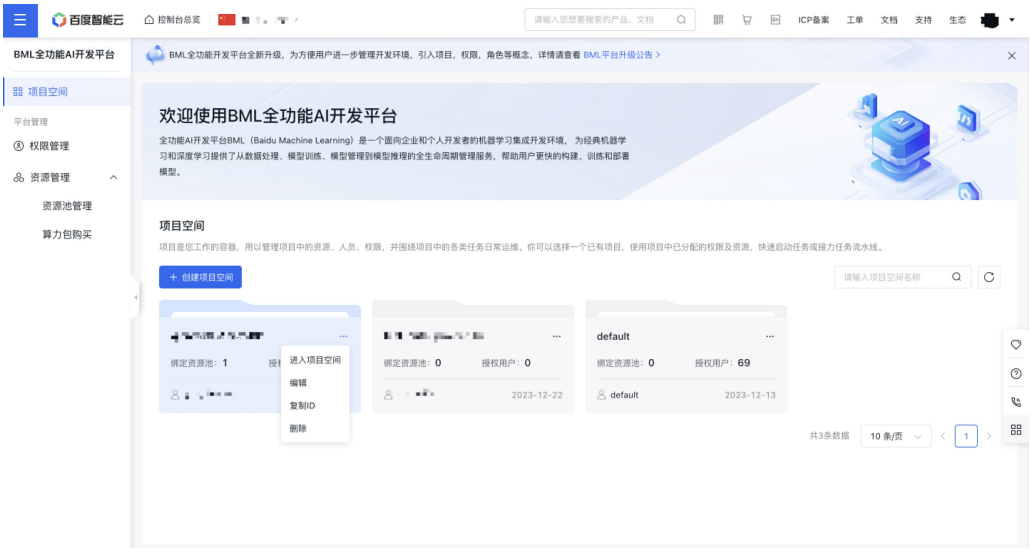
> PS：公共资源池默认绑定。



完成以上配置后，可通过点击下方【进入项目空间】【确认并返回项目列表】操作完成创建并进入到下一步流程。点击【进入项目空间】进入到对应项目空间概览页，点击【确认并返回项目列表】反馈项目空间列表页，可继续进行其他平台管理操作。

编辑项目空间

鼠标悬浮到项目空间卡片右上方下拉，点击【编辑】，可修改项目空间相关配置，编辑操作同创建，可对项目名称、描述、成员授权、资源绑定进行相应修改



授权关系访问

鼠标悬浮到项目空间上方绑定资源池和授权用户数，可查看当前项目里绑定关系；

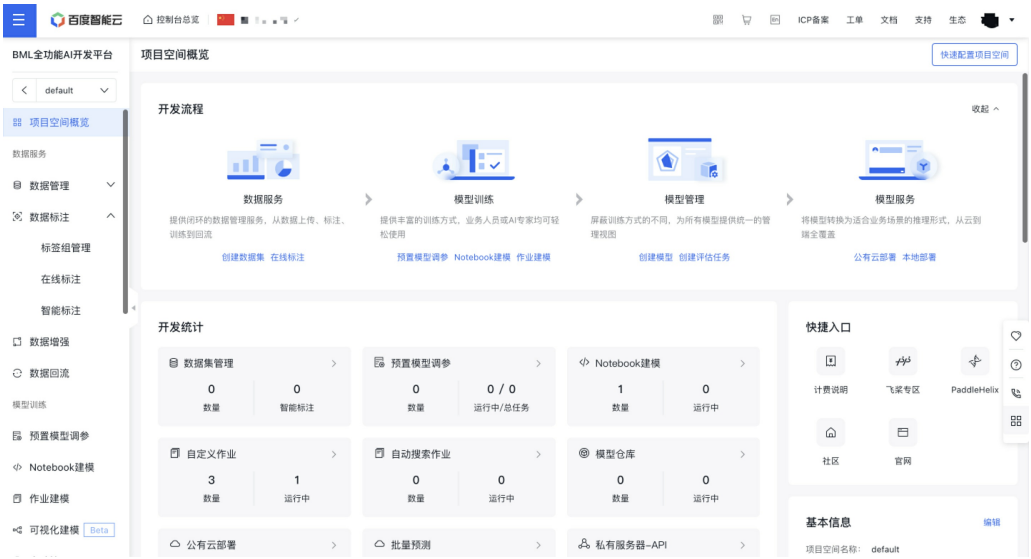


注意，绑定/授权内容统计不包含当前区域授权，授权用户数实际统计为授权关系

访问项目空间

查看项目详情

可点击default默认项目或自定义项目空间进入到BML控制台项目概览页，查看项目详情。



成员配置

在项目空间详情页点击成员列表【成员管理】，右侧弹窗添加成员页面，可进行相关角色授权操作。

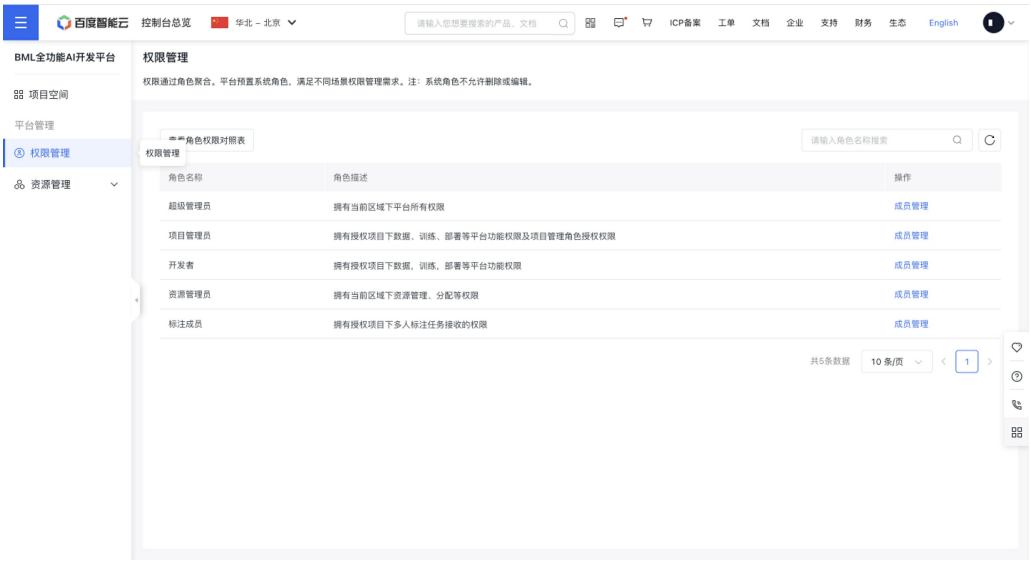
权限管理

权限管理

功能说明

平台支持支持进行细粒度权限管控，可根据不同子用户分工不同，在平台进行相应授权操作。平台内置超级管理员、项目管理员、资源管理员、开发者、标注成员5类角色，权限通过角色聚合，子用户通过角色授权的方式来满足不同场景权限管理需求。

主账户默认为超级管理员，拥有平台所有权限。主账户可以为子用户进行角色授权，授权可叠加，不同角色对应不同的平台权限；系统角色不允许编辑或删除。



前置概念说明

概念	说明
区域（region）	区域(Region)是指分布在全球范围内的各个物理节点，可以根据客户群体分布的地理位置，选择在不同区域部署产品。BML目前提供服务区域为： 华北-北京 、 华北-保定 、华东-苏州、华东-上海、金融华中-武汉、华南-广州。更多见 区域
主账户	是资金、资源管理的主体，负责管理组织，主账户可以创建子账户，或邀请其他账号成为组织中的成员，也可以将成员账户从企业组织中移除，将服务策略应用到组织层级中，从而控制成员账户的访问权限。一个常规注册用户流程即可为一个主账户
子账户	又称成员账户，可以被主账户邀请或创建。子账户和主账户间资源隔离，主账户拥有子账户的资源管理员权限，可集中管理资源。子账户可作为独立账户使用，具有单独出账功能。
子用户	用户级别，IAM 身份体系下一种用户，用于共享或协作主用户的云上资源。用户名在账户下唯一。子用户可以是人、也可以是服务或应用程序，即子用户可以通过账号密码的方式登录控制台，也可以使用API通过编程访问的方式访问账户内云资源

内置权限说明

内置角色权限

- 超级管理员：拥有当前区域下平台所有权限
- 资源管理员：拥有当前区域下资源管理、分配等权限
- 项目管理员：拥有授权项目下数据、训练、部署等平台功能权限及项目管理角色授权权限
- 开发者：拥有授权项目下数据，训练，部署等平台功能权限
- 标注成员：拥有授权项目下多人标注任务接收的权限

查看角色权限对照表

点击【查看角色权限对照表】，可查看基础管控、智能数据、模型开发、模型仓库、公有云部署、本地部署、模型应用、基础资产的权限信息。



角色权限对照表

产品/模块权限			超级管理员	资源管理员	项目管理员	开发者	标注成员
基础管控	项目管理	项目空间创建、编辑、删除	✅	-	-	-	-
		项目空间概览查看	✅	-	✅	✅	✅
	资源池管理	资源池查看	✅	✅	✅	-	-
		资源池创建、挂载存储资源解绑、挂载权限	✅	✅	-	-	-
		资源池分配、镜像仓库编辑、PFS操作	✅	✅	-	-	-
	算力包购买	算力包查看	✅	✅	✅	✅	-
		算力包购买	✅	✅			
	角色管理	角色列表查看	✅	-	✅	-	-
		角色授权成员、成员管理	✅	-	✅	-	-
智能数据	数据集管理	数据集查看	✅	-	✅	✅	-
		新建/删除数据集	✅	-	✅	✅	-
	标签组管理	标签组列表查看	✅	-	✅	✅	-
		新建/删除标签组	✅	-	✅	✅	-
	在线标注	标注页查看	✅	-	✅	✅	✅
		在线标注操作	✅	-	✅	✅	✅
	智能标注	查看智能标注列表、智能标注数据、进度等	✅	-	✅	✅	-
		确认难例等操作	✅	-	✅	✅	-
		新建智能标注任务、中止任务、重新启动	✅	-	✅	✅	-
	多人标注-发起	查看多人标注任务、成员	✅	-	✅	✅	-
		创建多人标注任务、管理标注团队成员	✅	-	✅	✅	-
	多人标注-接收	查看待标注任务和待审核任务状态列表	✅	-	✅	✅	✅
		管理待标注任务和待审核任务	✅	-	✅	✅	✅
	寻求标注支持	查看寻求标注支持页面表单信息	✅	-	✅	✅	-

		提交寻求标注支持页面表单信息	✅	-	✅	✅	-
	数据清洗	查看数据清洗任务列表、查看任务详情	✅	-	✅	✅	-
		新建清洗任务、终止任务	✅	-	✅	✅	-
	数据增强	查看数据增强任务列表、查看任务详情	✅		✅	✅	
		新建增强任务、终止任务	✅	-	✅	✅	-
	数据回流	查看回流数据	✅	-	✅	✅	-
		回流图片保存到数据集	✅	-	✅	✅	-
		回流配置创建、编辑和删除	✅	-	✅	✅	-
	摄像头采集	查看摄像头采集数据	✅	-	✅	✅	-
		摄像头采集数据保存到数据集	✅	-	✅	✅	-
		数据采集配置管理	✅	-	✅	✅	-
模型开发	作业建模	查看作业建模列表	✅	-	✅	✅	-
		新建，克隆，发布，删除作业	✅	-	✅	✅	-
	自动搜索作业	查看自动搜索作业	✅	-	✅	✅	-
		新建，克隆，发布，删除作业	✅	-	✅	✅	-
	预置模型调参-通用模型	查看通用模型列表	✅	-	✅	✅	-
		创建任务，新建运行，删除任务	✅	-	✅	✅	-
	预置模型调参-行业模型	查看行业模型列表	✅	-	✅	✅	-
		创建任务，新建运行，删除任务	✅	-	✅	✅	-
	可视化建模	查看列表，查看历史记录	✅	-	✅	✅	-
		新建、删除、复制、重命名、开始训练、取消、automl、定时、导入、导出、 日志清理、算子操作， 算子配置， 新建发布删除自定义组件、保存、发布	✅	-	✅	✅	-
	notebook建模	查看notebook建模列表	✅	-	✅	✅	-
		创建，访问codelab，重启，停止，删除	✅	-	✅	✅	-
	实验管理	查看实验，查看任务	✅	-	✅	✅	-
		删除实验，删除任务	✅	-	✅	✅	-
模型仓库	模型管理	查看我的模型列表，查看版本列表，版本对比，查看开源模型列表，开源模型文档	✅	-	✅	✅	-

		开源模型文档链接，部署详情					
		创建模型，删除，新建版本，开源模型导入，校验，部署	✅	-	✅	✅	-
	模型评估	模型评估列表查看，详情，查看报告	✅	-	✅	✅	-
		新建评估任务，删除，停止	✅	-	✅	✅	-
公有云部署	在线预测	查看在线预测列表，详情	✅	-	✅	✅	-
		新建部署，设置休眠策略，重启，停止任务	✅	-	✅	✅	-
	批量预测	查看批量预测列表，详情	✅	-	✅	✅	-
		新建预测，删除，停止，定时设置	✅	-	✅	✅	-
	性能评估	查看评估列表，报告	✅	-	✅	✅	-
		创建任务，创建评估，删除	✅	-	✅	✅	-
本地部署	纯离线服务	离线服务，列表查看，服务详情查看，服务搜索，序列号列表查看	✅	-	✅	✅	-
		发布新服务，序列号获取，部署包下载，版本更新，部署，重新发布。 购买正式授权，新增测试序列号，管理序列号，批量离线激活	✅	-	✅	✅	-
	端云协同服务	查看我的部署包列表，查看我的本地设备，本地设备详情查看	✅	-	✅	✅	-
		新建发布部署包（下发到设备，发布新版本），下发部署包到设备	✅	-	✅	✅	-
模型应用	应用接入	应用列表查看，详情和报表数据查看	✅	-	✅	✅	-
		新建应用，编辑，删除	✅	-	✅	✅	-
	权限管理	服务列表查看	✅	-	✅	✅	-
		服务权限配置	✅	-	✅	✅	-
	调用统计	用量统计查看	✅	-	✅	✅	-
基础资产	镜像管理/环境管理	查看预置环境列表、查看预置环境详情、查看自定义环境列表、查看自定义环境详情、复制环境内容、搜索环境	✅	-	✅	✅	-
		新建环境、新建环境版本、删除环境、修改环境描述	✅	-	✅	✅	-

成员管理

点击操作栏【成员管理】，右侧抽屉弹出，需先选择授权范围，目前分为项目和当前区域；根据不同角色可授权范围不同；

对于区域级角色如超级管理员、资源管理员等，可授权范围为当前区域（region），对于项目级角色如项目管理员、开发者、标注成员等，可授权范围即可选择项目，也可以选择当前区域（region），选择区域代表授权范围为当前区域下所有项目，请谨慎选择。



用户授权子用户来源IAM，可先前往IAM控制台新建子用户后方可授权；

各产品权限说明

智能数据

数据集管理

权限	操作
只读权限	数据集列表页查看、数据集全部版本查看、数据集名称和id查询、数据详情页查看和筛选、标签栏查看筛选、查看数据大图、数据质检报告
写权限	新建数据集、新增数据集版本、数据导入
管理权限	数据导出、删除数据集、删除数据组、删除数据

标签组管理

权限	操作
只读权限	标签组列表查看、标签列表查看
写权限	新增修改删除标签组、新增修改删除标签
管理权限	无

在线标注

权限	操作
只读权限	标注页查看
写权限	标注页的所有操作
管理权限	无

智能标注

权限	操作
只读权限	查看智能标注列表查看、查看进度、查看记录、查看数据集
写权限	确认难例操作，以及确认难例页面的所有操作
管理权限	新建智能标注任务、中止任务、重新启动

多人标注-发起

权限	操作
只读权限	查看多人标注任务列表和标注团队成员
写权限	创建多人标注任务、管理标注团队成员
管理权限	无

多人标注-接收

权限	操作
只读权限	查看待标注任务和待审核任务状态列表
写权限	管理待标注任务和待审核任务
管理权限	无

寻求标注支持

权限	操作
只读权限	查看寻求标注支持页面表单信息
写权限	提交寻求标注支持页面表单信息
管理权限	无

数据清洗

权限	操作
只读权限	查看数据清洗任务列表、查看任务详情
写权限	新建清洗任务、终止任务
管理权限	无

数据增强

权限	操作
只读权限	查看数据增强任务列表、查看任务详情
写权限	新建增强任务、终止任务
管理权限	无

数据回流

权限	操作
只读权限	查看和筛选回流数据
写权限	回流图片保存到数据集
管理权限	查看回流配置，回流配置的创建，编辑和删除

摄像头采集

权限	操作
只读权限	查看和筛选摄像头采集数据
写权限	摄像头采集数据保存到数据集
管理权限	查看数据采集配置，配置删除

模型开发

作业建模

权限	操作
只读权限	查看列表、详情，配置，日志，查看概览页开发统计相关模块，概览页开发流程button可点击，查看相关日志，相关监控，相关操作审计
写权限	新建，克隆，发布，删除
管理权限	无

自动搜索作业

权限	操作
只读权限	查看列表、详情，配置，日志，查看概览页开发统计相关模块，查看相关日志，相关监控，相关操作审计
写权限	新建，克隆，发布，删除
管理权限	无

预置模型调参-通用模型

权限	操作
只读权限	查看列表、详情，配置，日志，版本对比，查看概览页开发统计相关模块，概览页开发流程button可点击，查看相关日志，相关监控，相关操作审计
写权限	创建任务，新建运行，删除
管理权限	无

预置模型调参-行业模型

权限	操作
只读权限	查看列表、详情，配置，日志，版本对比，查看概览页开发统计相关模块，相关日志查看，相关监控查看，相关操作审计查看
写权限	创建任务，新建运行，删除
管理权限	无

可视化建模

权限	操作
只读权限	查看列表，历史记录，查看概览页开发统计相关模块，相关日志查看，相关监控查看，相关操作审计查看
写权限	新建、删除、 复制、重命名、开始训练、取消、automl、定时、导入、导出、 日志清理、算子操作， 算子配置， 新建发布删除自定义组件、保存、发布
管理权限	无

notebook建模

权限	操作
只读权限	查看列表，详情，查看概览页开发统计相关模块，概览页开发流程button可点击，相关日志查看，相关监控查看，相关操作审计查看
写权限	创建，访问codelab，重启，停止，删除
管理权限	无

实验管理

权限	操作
只读权限	查看实验，查看任务，相关日志查看，相关监控查看，相关操作审计查看
写权限	删除实验，删除任务
管理权限	无

模型仓库

模型管理

权限	操作
只读权限	查看我的模型列表，查看版本列表，版本对比，查看开源模型列表，开源模型文档链接，部署详情，查看概览页开发统计相关模块，概览页开发流程button可点击，相关日志查看，相关监控查看，相关操作审计查看
写权限	创建模型，删除，新建版本，开源模型导入，校验，部署
管理权限	无

模型评估

权限	操作
只读权限	模型评估列表查看，详情，查看报告，概览页开发流程button可点击
写权限	新建评估任务，删除，停止
管理权限	无

公有云部署

在线预测

权限	操作
只读权限	查看列表，详情，查看概览页开发统计相关模块，概览页开发流程button可点击，相关日志查看，相关监控查看，相关操作审计查看
写权限	新建部署，设置休眠策略，重启，停止任务
管理权限	无

批量预测

权限	操作
只读权限	查看列表，详情，查看概览页开发统计相关模块
写权限	新建预测，删除，停止，定时设置
管理权限	无

性能评估

权限	操作
只读权限	查看评估列表，报告，查看概览页开发统计相关模块
写权限	创建任务，创建评估，删除
管理权限	无

本地部署

纯离线服务

权限	操作
只读权限	离线服务，列表查看，服务详情查看，服务搜索，序列号列表查看，查看概览页开发统计相关模块，概览页开发流程button可点击，相关日志查看，相关监控查看，相关操作审计查看
写权限	发布新服务，序列号获取，部署包下载，版本更新，部署，重新发布。 购买正式授权，新增测试序列号，管理序列号，批量离线激活
管理权限	无

端云协同服务

权限	操作
只读权限	查看我的部署包列表，查看我的本地设备，本地设备详情查看，查看概览页开发统计相关模块
写权限	新建发布部署包（下发到设备，发布新版本），下发部署包到设备
管理权限	无

模型应用

应用接入

权限	操作
只读权限	应用列表查看，详情和报表数据查看
写权限	新建应用，编辑，删除
管理权限	无

权限管理

权限	操作
只读权限	服务列表查看
写权限	服务权限配置
管理权限	无

调用统计

权限	操作
只读权限	用量统计查看
写权限	无
管理权限	无

基础资产

镜像管理/环境管理

权限	操作
只读权限	查看预置环境列表、查看预置环境详情、查看自定义环境列表、查看自定义环境详情、复制环境内容、搜索环境
写权限	新建环境、新建环境版本、删除环境、修改环境描述
管理权限	无

基础管控

项目空间管理

权限	操作
只读权限	项目空间概览查看
写权限	管理项目，区分管理员能看到的项目和普通角色看到的项目
管理权限	项目空间创建、编辑、删除

资源池管理

权限	操作
只读权限	查看资源池列表，日志查看，监控查看
写权限	资源池分配，镜像仓库挂载操作，PFS操作
管理权限	资源池新建、挂载，存储资源解绑、挂载权限

算力包管理

权限	操作
只读权限	算力包列表访问
写权限	算力包购买
管理权限	无

权限管理

权限	操作
只读权限	角色列表查看，详情查看
写权限	角色授权
管理权限	无

快速开始

用BML实现图片分类

目录

1.图片分类简介
2.平台入口
3.准备数据
3.1 抽油机工况分类数据介绍
3.2 创建及导入数据集
4.训练模型
5.模型分析和调优
6.部署模型

用BML实现图片分类：以抽油机工况分类为例

图片分类简介

亲爱的开发者您好，欢迎使用百度BML全功能AI开发平台开启您的AI开发之旅！

图像分类是经典的计算机视觉任务，也是所有计算机视觉任务的基础。图像分类的主要功能为识别一张图中是否是某类物体/状态/场景，适用于图片内容单一、需要给整张图片分类的场景。其主要应用场景有：

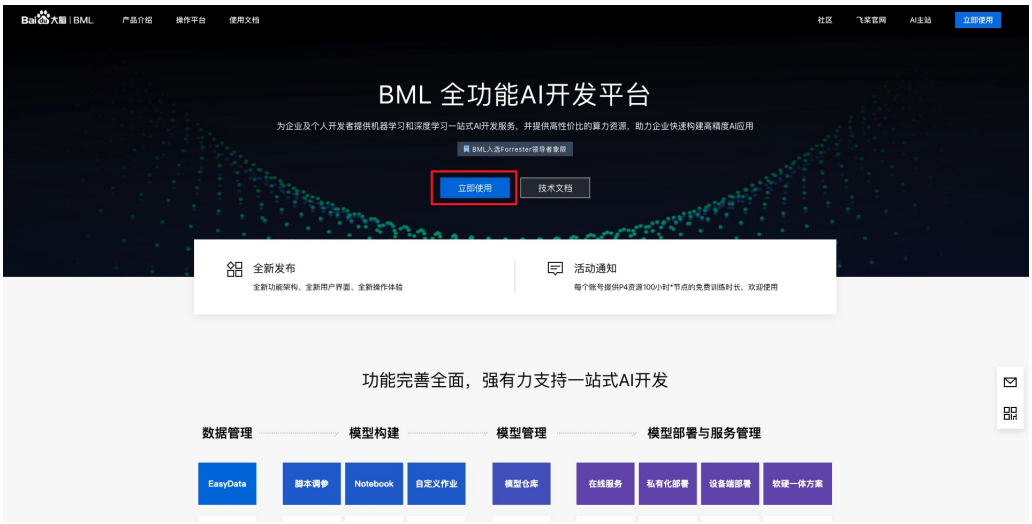
- 图片内容检索：定制训练需要识别的各种物体，并结合业务信息展现更丰富识别结果。
- 图片审核：定制图像审核规则，如训练直播场景中抽烟等违规现象。
- 制造业分拣或质检：定制生产线上各种产品识别，进而实现自动分拣或者质检。
- 医疗诊断：定制识别医疗图像，辅助医生肉眼诊断。

下文中将以抽油机工况分类任务为例，分步骤向您详细介绍如何使用百度BML全功能AI开发平台开发您自己的图像分类模型。

抽油机工况分类任务简介：
抽油机井采油是目前油田开发中普遍应用的方式，抽油机井的管理水平的好坏，关系到油田整体经济效益的高低。要做好抽油机井的生产管理工作，必须取准取全各项生产资料，制定抽油机井合理的工作制度，不断进行分析，适应不断变化的油藏动态，加强并提高抽油机井的日常管理水平。而分析和解释示功图，就是直接了解深井泵工作状况好坏的一个主要手段，不但深井泵工作中的一切异常现象可以在示功图上比较直观的反映出来，而且，还可以结合有关资料，来分析判断油井工作制度是否合理，抽油设备与油层和原油性质是否适应，还可以通过“示功图法”对低产、低能井制定出合理的开关井时间，减少设备的磨损和电能的浪费等。

平台入口

BML全功能AI开发平台为企业及个人开发者提供机器学习和深度学习一站式AI开发服务，并提供高性价比的算力资源，助力企业快速构建高精度AI应用，进入官方网站点击【立即使用】。



准备数据

准备数据是AI模型开发的关键一环，训练数据的质量决定了训练所得模型效果可达到的上限，下面来介绍数据规范与相关操作步骤。

抽油机工况分类数据介绍

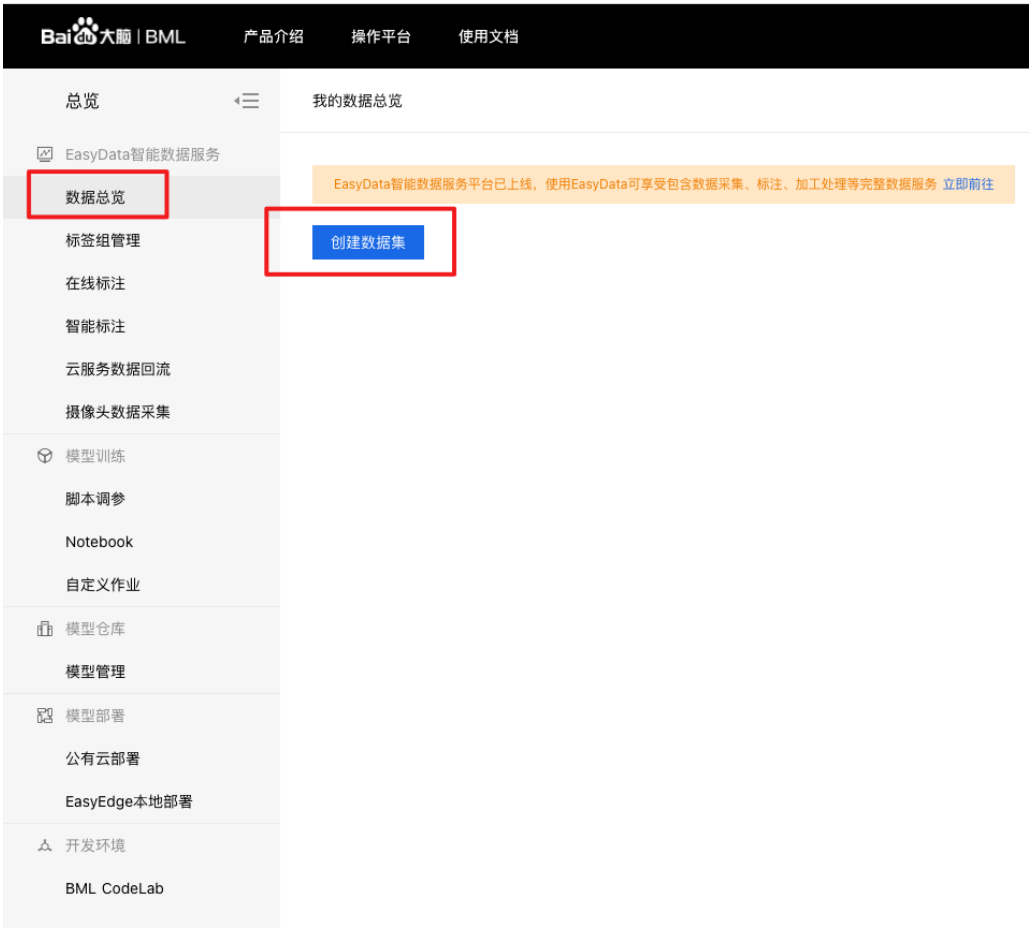
抽油机工况分类数据集来自于百度与昆仑数智合作的『中石油·第二届梦想云创新大赛』比赛数据，是中石油实际业务场景采集的真实数据，其中训练集包含12059个样本数据，每个样本数据包含1个json格式标注数据以及1张对应的png格式图片。其中包含12类典型工况：

序号	类别名	序号	类别名
A01	工作正常	A07	游动阀关闭迟缓
A02	供液不足	A08	柱塞脱出泵工作筒
A03	气体影响	A09	游动阀漏
A04	气锁	A10	固定阀漏
A05	上碰泵	A11	砂影响+供液不足
A06	下碰泵	A12	惯性影响+工作正常

导入平台即可用，数据下载链接：抽油机工况分类-json格式

创建及导入数据集

1、在官网界面点击【数据总览】，进入数据集操作界面，点击【创建数据集】。



2、进入创建数据集界面，填写相关信息，选择数据和标注类型（注意训练集、验证集、测试集需要分开创建）。



3、数据集创建完成后，可以在数据总览界面看到刚才创建好的数据集ID，点击【导入】，将自己要训练的数据集导入。



以本地导入-上传压缩包为例：导入方式选择【本地导入】，选择标注格式，点击【上传压缩包】。



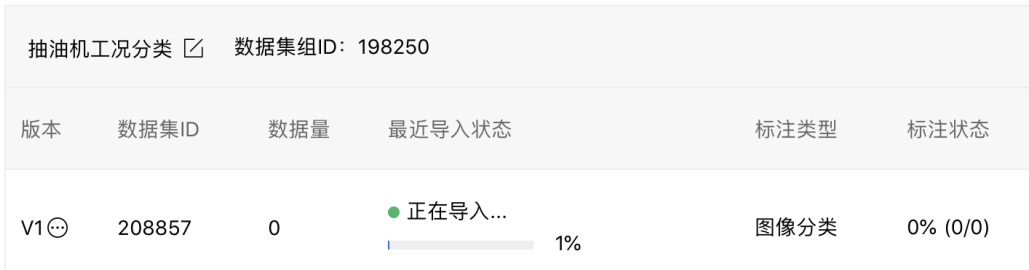
仔细阅读上传压缩包格式要求，可点击【下载示例压缩包】确认格式：



确认格式无误后，点击【已阅读并上传】，注意上传时不要关闭网页：



点击【确认并返回】后自动开始导入：



可看到【标注状态】为100%，如果数据集没有全部标注，可使用平台【智能标注】功能。

训练模型

BML上提供了预置模型调参、NoteBook建模、自定义作业三种开发模式，开发难度和开发的灵活性程度不一，分别满足不同水平和需求的开发者。

本文以使用者最多的预置模型调参开发模式为例，示意训练模型的基本步骤。

1、进入bml官方平台点击【预置模型调参】-【计算机视觉模型】，点击【创建】。



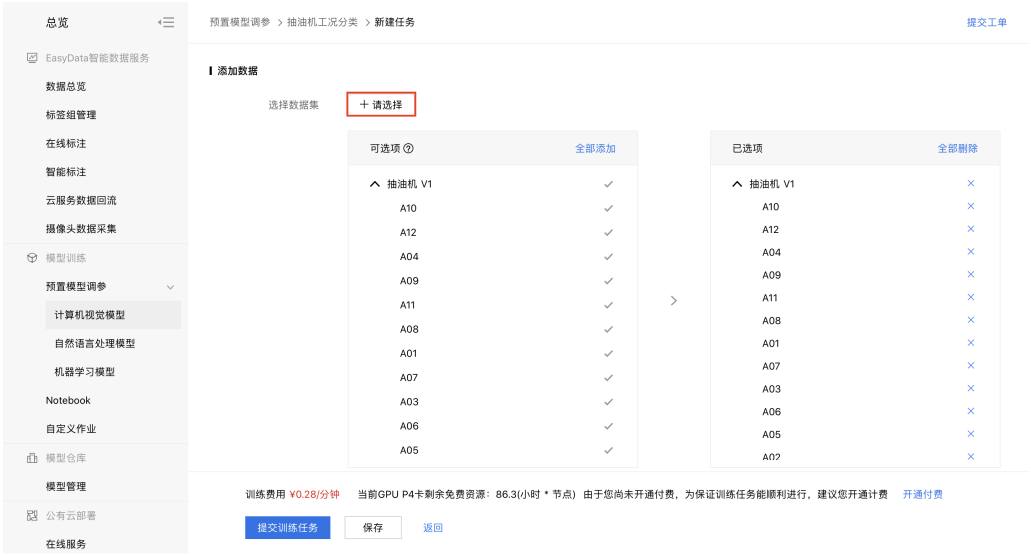
2、填写项目信息并点击【新建】。



3、点击【新建任务】。



4、点击【+请选择】，勾选刚刚上传数据集下的所有标签类，点击右下角【确定】。



5、可以选择上传验证集和测试集。验证集用来确定模型训练过程中超参数的调整。测试集用来获得更客观的模型效果评估结果。如果选择不上传，系统也会自动从已上传的训练集中分割出验证和测试集。



6、配置网络，如果选择网络选型参考：网络选型介绍。



7、配置超参数。如果选择脚本编辑为超参来源， 可在脚本编辑部分代码框内自定义超参数。超参数配置参考：超参数选择。



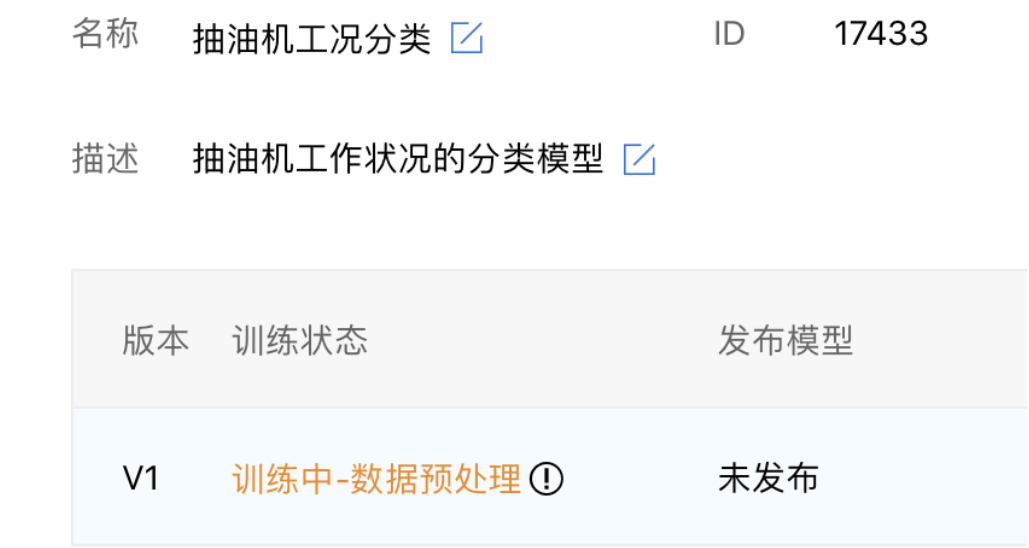
8、可填写相关信息，并发布模型。也可以模型训练完成后再根据训练结果决定是否发布。



9、根据自身的周期和经费安排，配置计算资源。



10、最后点击【提交训练任务】，进入模型训练。



模型分析和调优

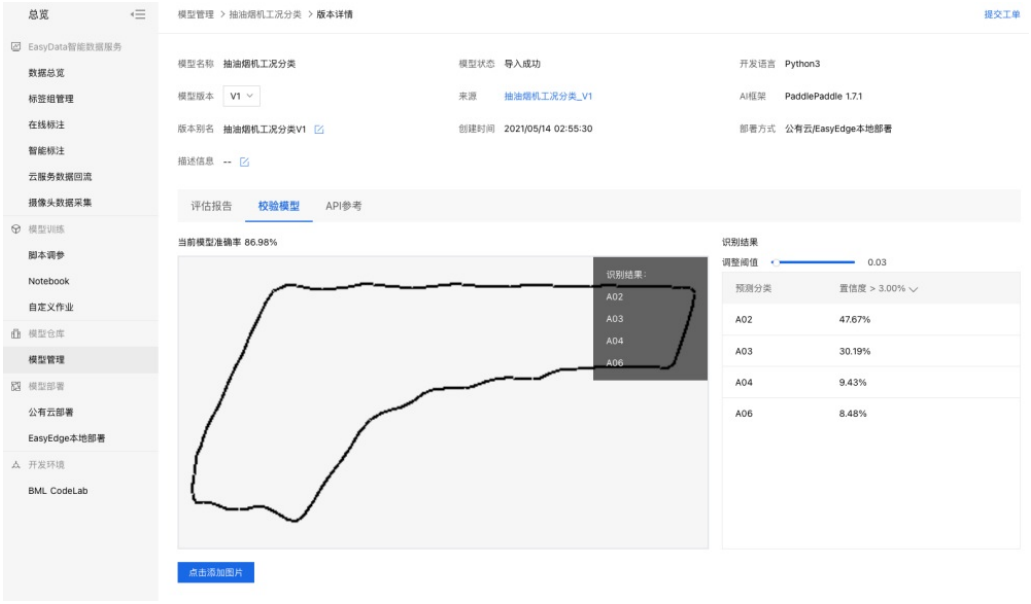
1、获取评估报告：点击【模型仓库】-【模型管理】，点击对应任务的【版本列表】查看训练好的模型，点击【评估报告】。



评估报告如下所示：

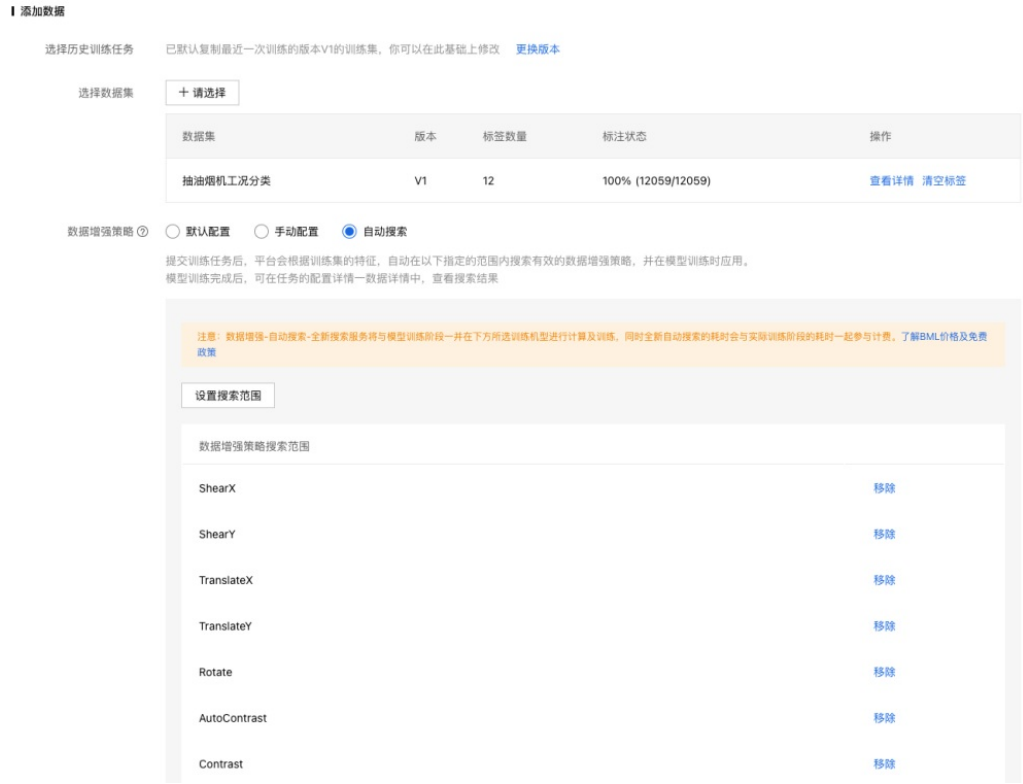


2、点击【校验模型】。



3、模型调优：新建模型时添加如下配置，可提高模型效果。

策略一：数据增强策略（数据增强算子参考）



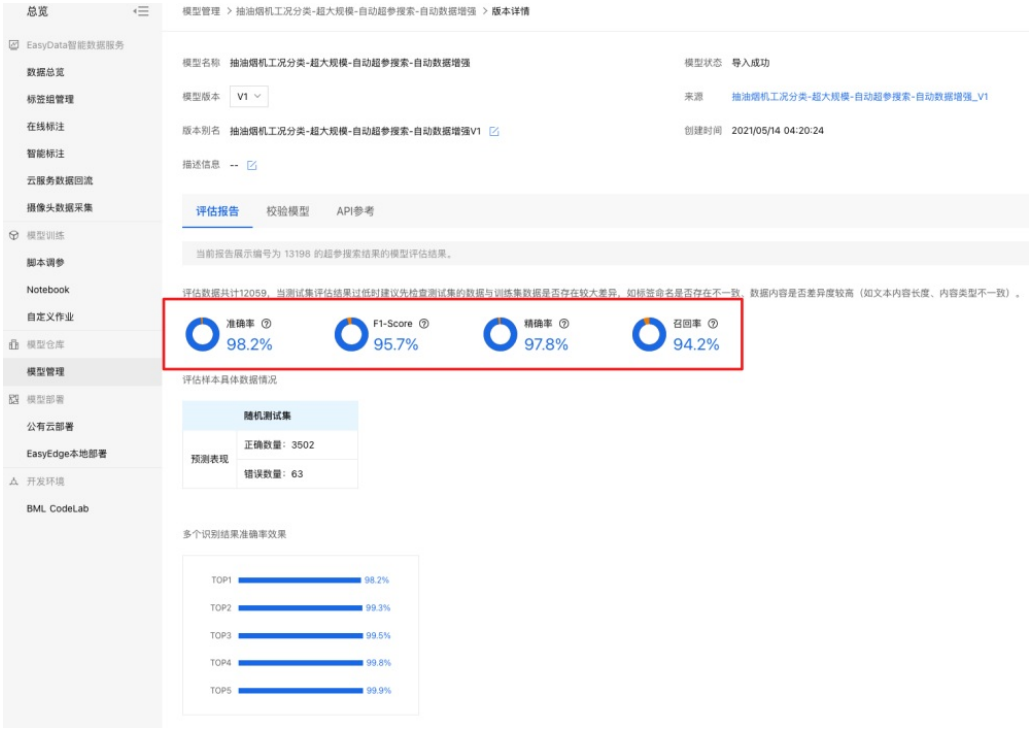
策略二：百度超大规模数据集预训练（预训练模型参考）



策略三：自动超参搜索（自动超参配置参考）



采用以上优化策略之后，重新打开评估报告，可以看到效果有明显的提升。

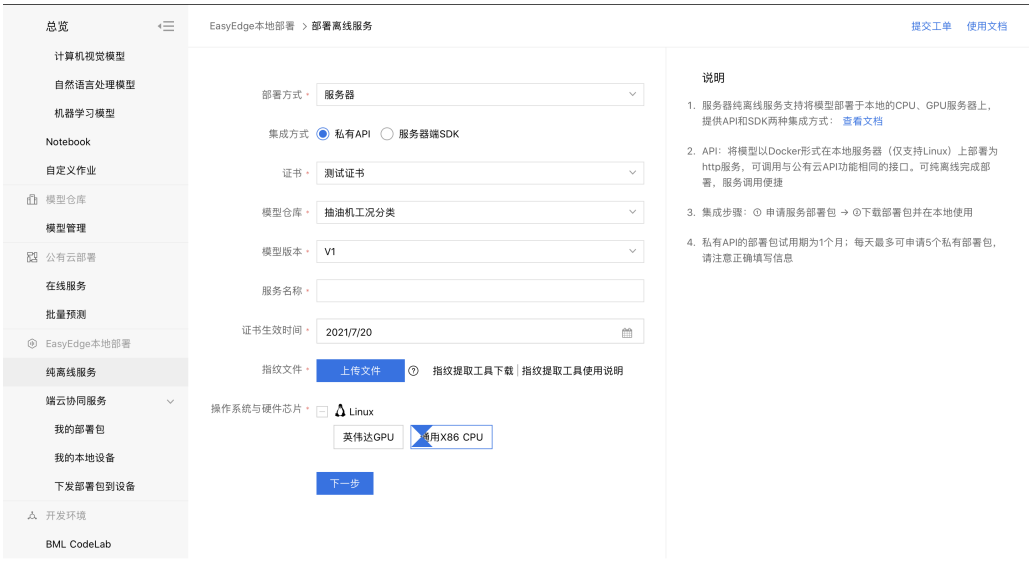


部署模型

1、在模型管理中，可选择公有云部署，端云协同服务，批量预测，和纯离线服务四种方式部署模型。具体参考：如何选择部署方式



2、在模型部署中，用户按照自己情况填写信息完成模型部署。下图以本地部署纯离线服务为例。



用BML实现物体检测

目录

1.物体检测简介
2.平台入口
3.准备数据
3.1 道路交通电子眼检测数据介绍
3.2 创建及导入数据集
4.训练模型
5.模型分析和调优
6.部署模型
7.公有云调用
7.1 使用流程

用BML实现物体检测：以道路交通电子眼检测为例

物体检测简介

亲爱的开发者您好，欢迎使用百度BML全功能AI开发平台开启您的AI开发之旅！

物体检测任务是当前深度学习应用最广的计算机视觉应用任务，主要是检测图中每个物体的位置、名称。适合图中有多个主体要识别、或要识别主体位置及数量。其主要应用场景有：

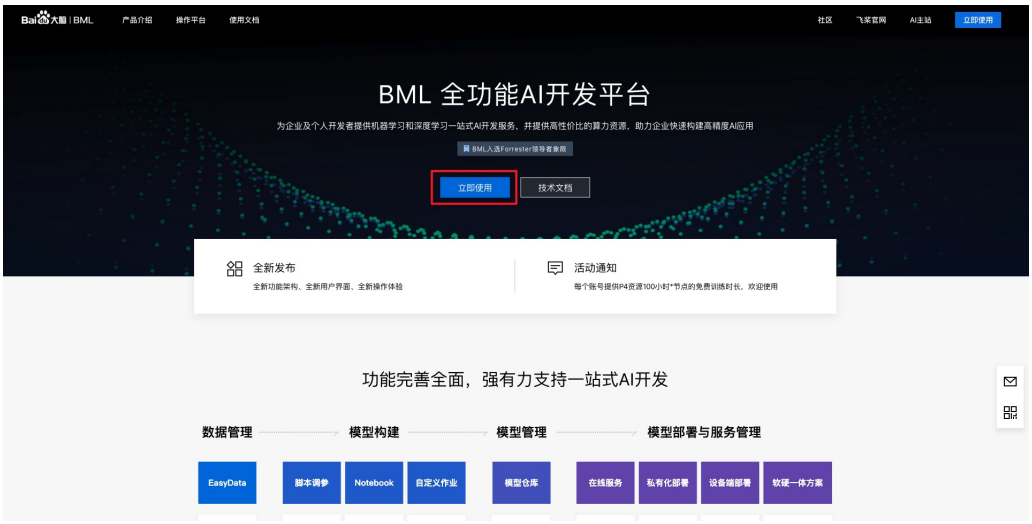
- 视频监控：如检测是否有违规物体、行为出现。
- 工业质检：如检测图片里微小瑕疵的数量和位置。
- 医疗诊断：如医疗细胞计数、中草药识别等。

下文中将以道路交通电子眼检测任务为例，分步骤向您详细介绍如何使用百度BML全功能AI开发平台开发您自己的物体检测模型。

道路交通电子眼检测任务简介：
在实时地图导航场景下，对于不同种类的交通电子提示装置的实时提醒，在规范车主驾驶行为、保障用户出行等方面至关重要。而各类电子提示装置分布于道路大街小巷，这类物体普通目标小，不易发现，通过常规的数据采集和人力手段很难识别，为实现导航规划增加了额外成本。对以电子眼为代表的道路交通要素进行检测，有着较大的价值。

平台入口

BML全功能AI开发平台为企业及个人开发者提供机器学习和深度学习一站式AI开发服务，并提供高性价比的算力资源，助力企业快速构建高精度AI应用，进入官方网站点击【立即使用】。



准备数据

准备数据是AI模型开发的关键一环，训练数据的质量决定了训练所得模型效果可达到的上限，下面来介绍数据规范与相关操作步骤。

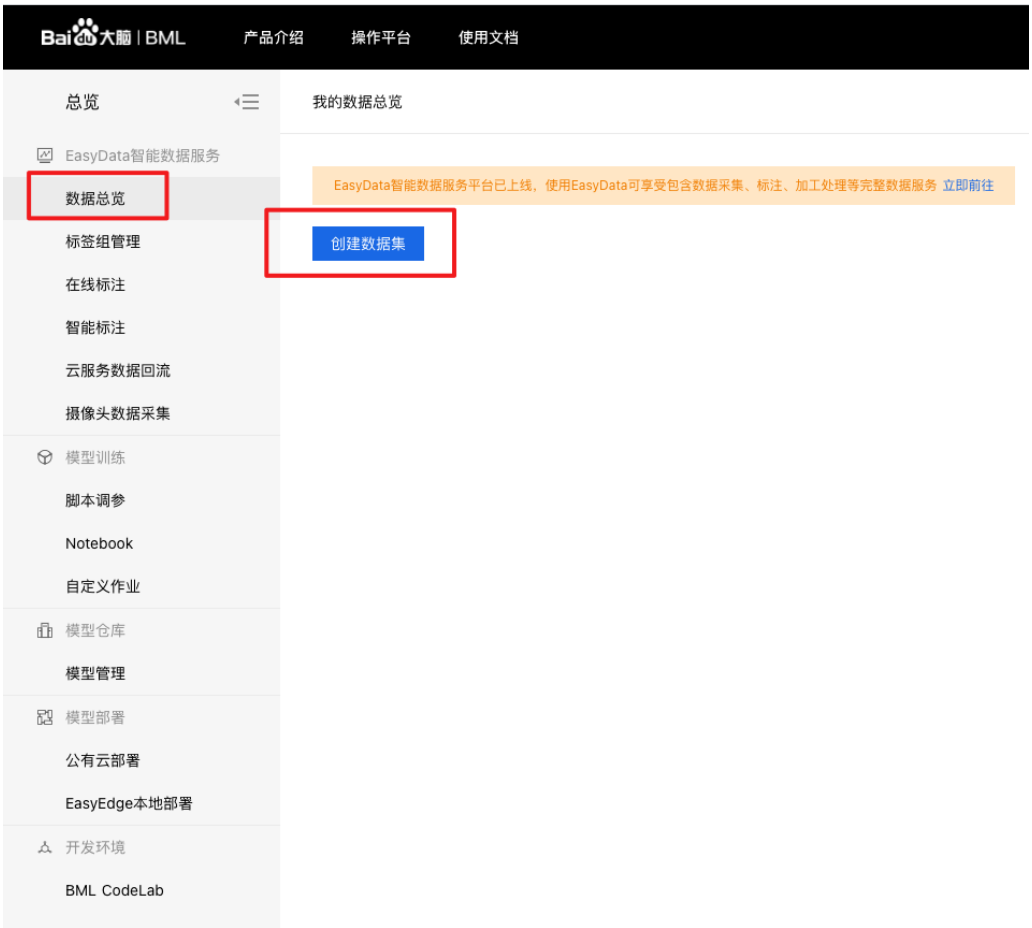
道路交通电子眼检测数据介绍

道路交通电子眼检测数据集来自于百度与曙光合作的『先导杯·计算应用大奖赛』的比赛数据，采集自百度地图的实际业务数据。训练集包括近2W样本，其中包含三类目标：rect_eye、sphere_eye、box_eye。

导入平台即可使用，数据下载链接：道路交通电子眼检测数据-json格式

创建及导入数据集

1、在官网界面点击【数据总览】，进入数据集操作界面，点击【创建数据集】。



2、进入创建数据集界面，填写相关信息，选择数据和标注类型（注意训练集、验证集、测试集需要分开创建）。



3、数据集创建完成后，可以在数据总览界面看到刚才创建好的数据集ID，点击【导入】，将自己要训练的数据集导入。



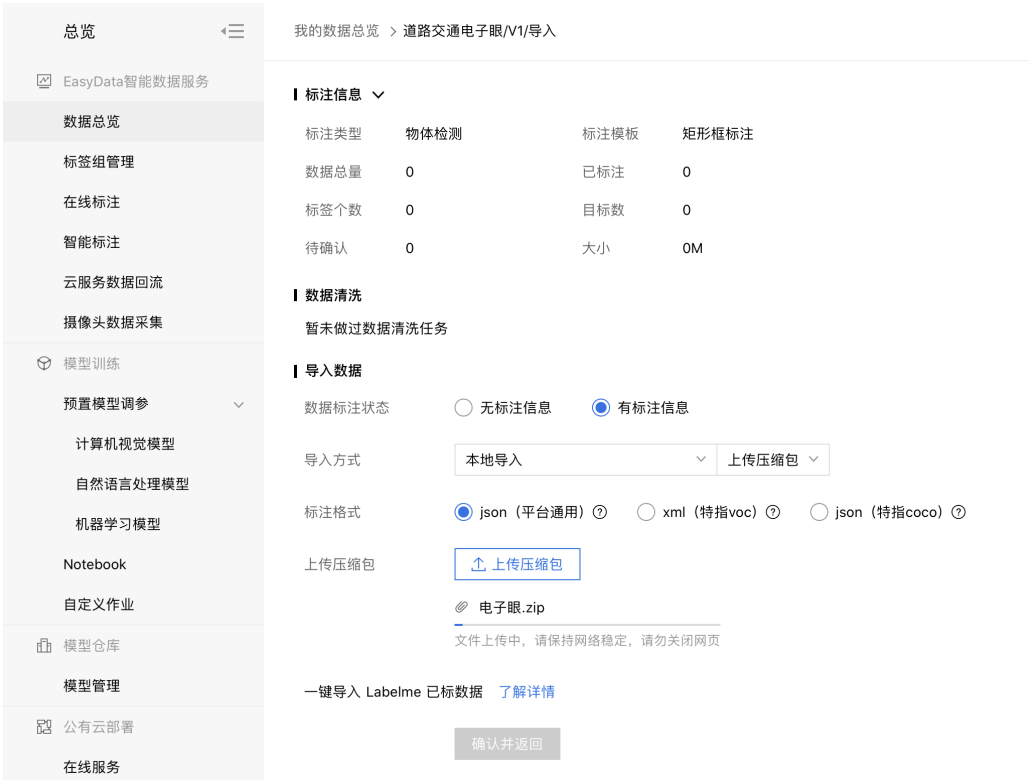
以本地导入-上传压缩包为例：导入方式选择【本地导入】，选择标注格式，点击【上传压缩包】。



仔细阅读上传压缩包格式要求，可点击【下载示例压缩包】确认格式：



确认格式无误后，点击【已阅读并上传】，注意上传时不要关闭网页：



点击【确认并返回】后自动开始导入：



可看到【标注状态】为100%，如果数据集没有全部标注，可使用平台【智能标注】功能。

训练模型

BML上提供了预置模型调参、NoteBook建模、自定义作业三种开发模式，开发难度和开发的灵活性程度不一，分别满足不同水平和需求的开发者。

本文以使用者最多的预置模型调参开发模式为例，示意训练模型的基本步骤。

1、进入bml官方平台点击【预置模型调参】-【计算机视觉模型】，点击【创建】。



2、填写项目信息并点击【新建】。



3、点击【新建任务】。



4、点击【+请选择】，勾选刚刚上传数据集下的所有标签类，点击右下角【确定】。



5、可以选择上传验证集和测试集。验证集用来确定模型训练过程中超参数的调整。测试集用来获得更客观的模型效果评估结果。如果选择不上传，系统也会自动从已上传的训练集中分割出验证和测试集。



6、配置网络，网络选型参考：网络选型介绍。



7、配置超参数。如果选择脚本编辑为超参来源，可在脚本编辑部分代码框内自定义超参数。超参数配置参考：超参数选择



8、可填写相关信息，并发布模型。也可以模型训练完成后再根据训练结果决定是否发布。



9、根据自身的周期和经费安排，配置计算资源。

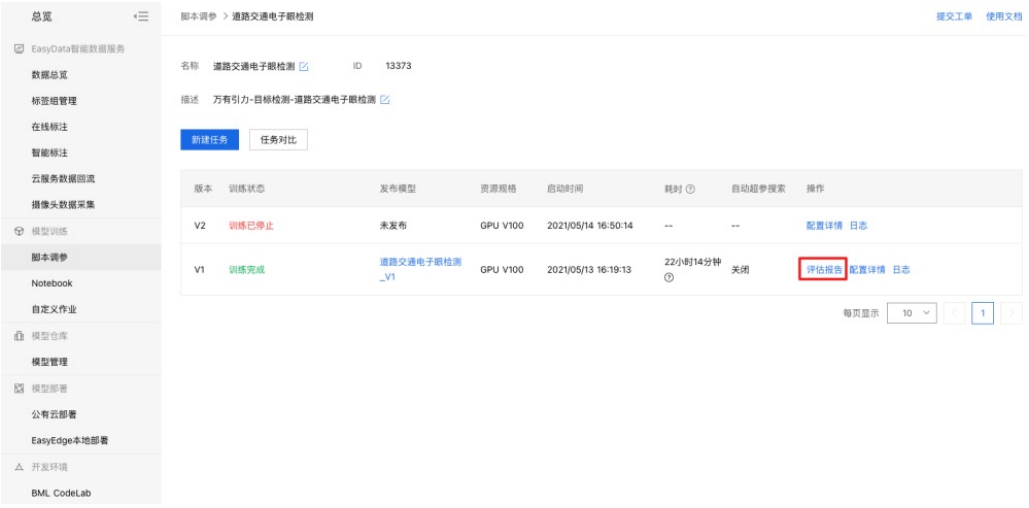


10、最后点击【提交训练任务】，进入模型训练。



模型分析和调优

1、获取评估报告：点击【模型仓库】-【模型管理】，点击对应任务的【版本列表】查看训练好的模型，点击【评估报告】。



评估报告如下所示：



2、点击【校验模型】。



3、模型调优：新建模型时添加如下配置，可提高模型效果。

策略一：数据增强策略（数据增强算子参考）



策略二：百度超大规模数据集预训练（预训练模型参考）



策略三：自动超参搜索（自动超参配置参考）



采用以上优化策略之后，重新打开评估报告，可以看到效果有明显的提升。

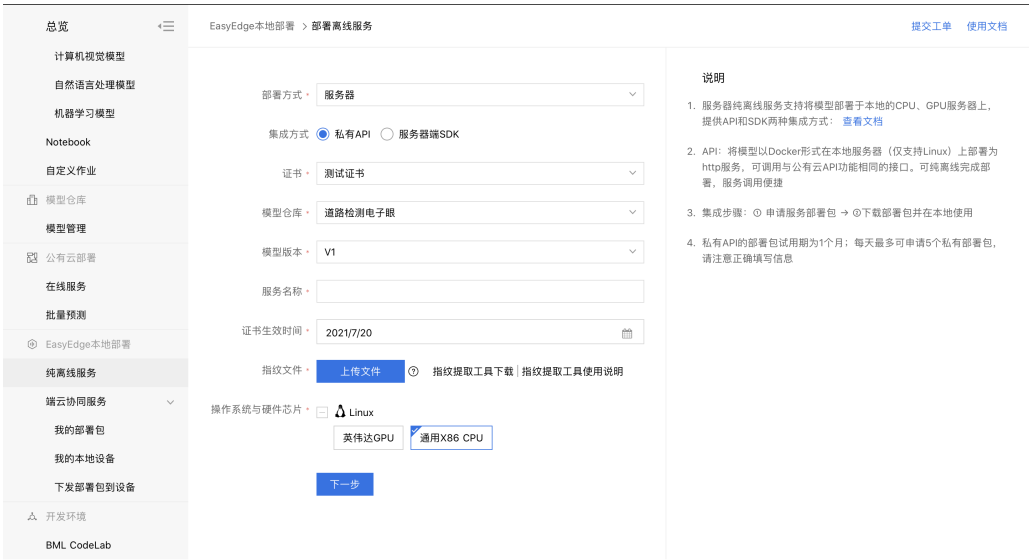


部署模型

1、在模型管理中，可选择公有云部署， 端云协同服务，批量预测，和纯离线服务四种方式部署模型。具体参考：如何选择部署方式



2、在模型部署中，用户按照自己情况填写信息完成模型部署。下图以本地部署纯离线服务为例。



公有云调用

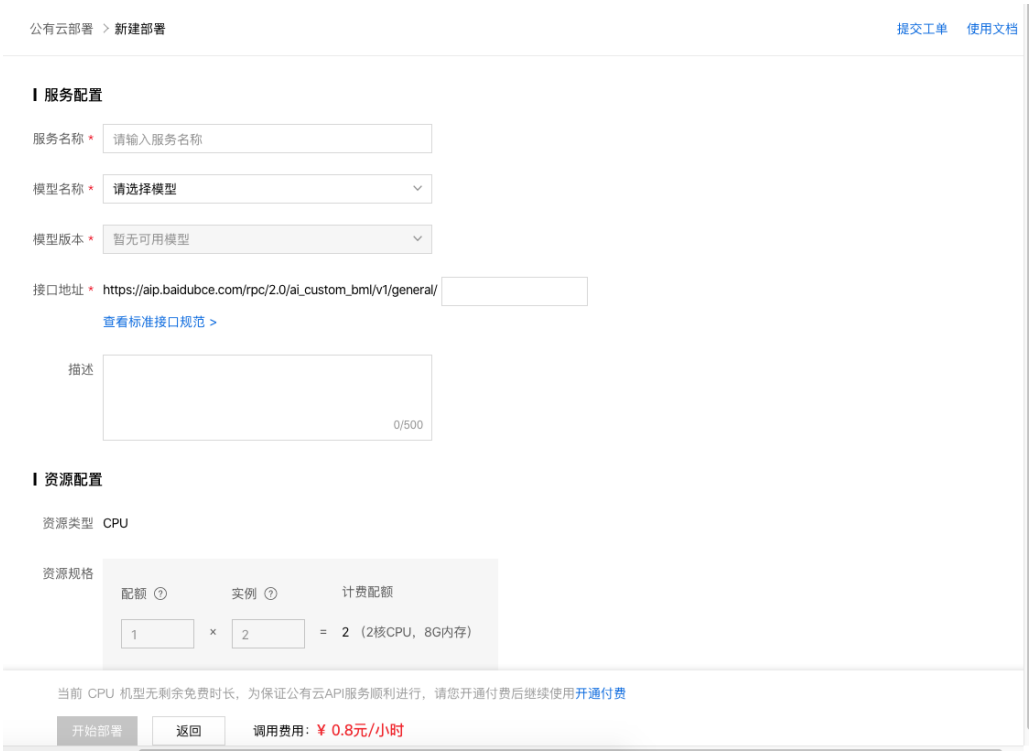
模型仓库中的视觉模型，发布为公有云部署时储在云端，可通过独立Rest API调用模型，实现AI能力与业务系统或硬件设备整合。BML具有完善的鉴权、流控等安全机制，并配置丰富的资源集群稳定承载高并发请求。 并且支持查找云端模型识别错误的数据，纠正结果并将其加入模型迭代的训练集，不断优化模型效果。

使用流程

1、在BML平台公有云部署的操作页面，点击『部署模型』，新建模型部署的任务。

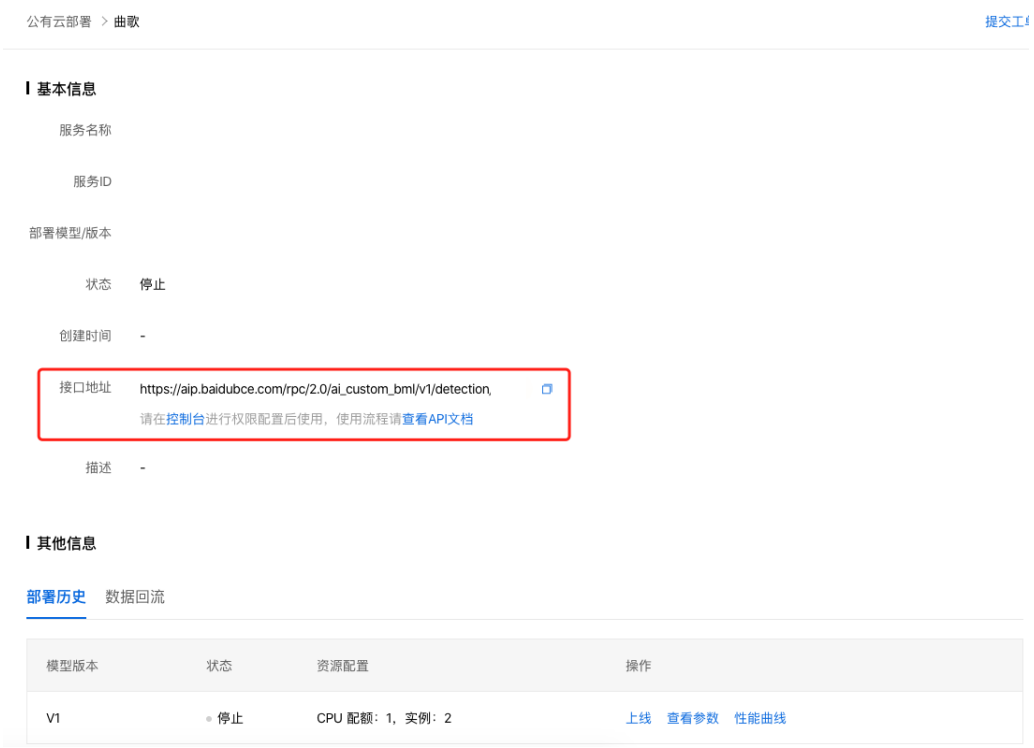


2、填写相关信息，完成创建。

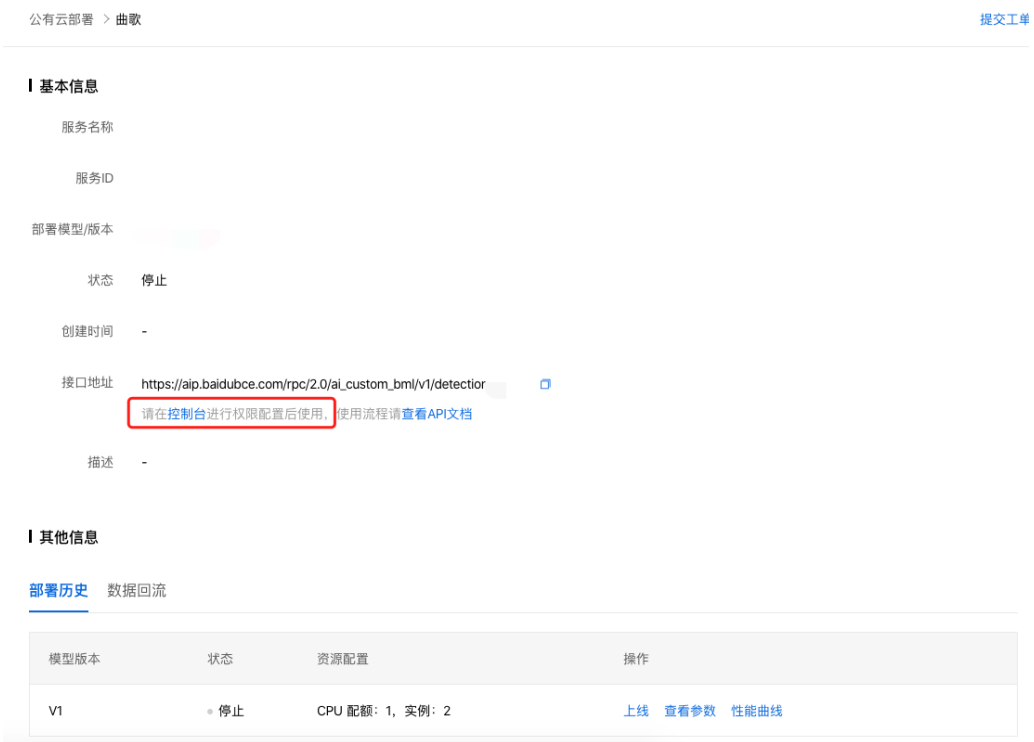


2、完成创建后回到操作页面，点击『查看详情』，查看配置信息，并获取『接口地址』信息。

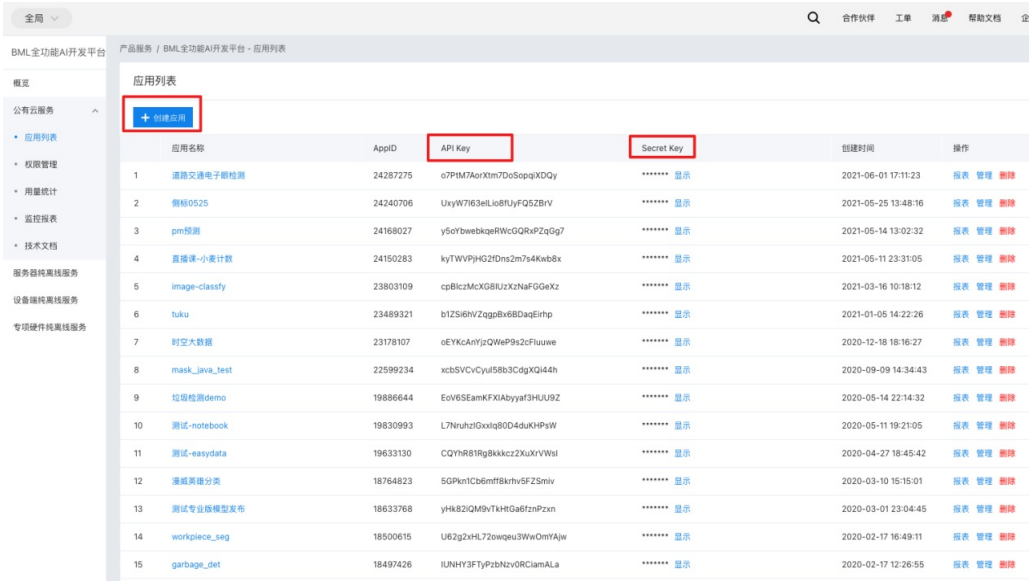




3、点击控制台，进入百度智能云-BML控制台，创建应用。



4、创建应用后，在应用列表页可获取『API Key』和『Secret Key』两个信息。



5、请求示例如下。

HTTP方法：POST

请求URL： 请首先进行自定义模型训练，完成训练后申请上线，上线成功后可在服务列表中查看并获取url。

URL参数如下：

参数	值
access_token	通过API Key和Secret Key获取的access_token

Header如下：

参数	值
Content-Type	application/json

注意：如果出现336001的错误码很可能是因为请求方式错误，与其他图像识别服务不同的是定制化图像识别服务以json方式请求。

6、请求参数注意事项如下。

参数	是否必选	类型	可选值范围	说明
image	是	string	-	图像数据，base64编码，要求base64编码后大小不超过4M，最短边至少15px，最长边最大4096px，支持jpg/png/bmp格式，注意请去掉头部
threshold	否	number	-	阈值，默认值为推荐阈值（0-1之间），具体值可在我的训练任务列表-模型效果查看

7、按照下述示例代码，填入相关信息即可完成调用API服务。

以python3.7为例：

```
 #### coding:utf-8
"""
 Copyright (c) 2021 Baidu.com, Inc. All Rights Reserved
 Author: test@baidu.com
"""

import requests
import json
import base64
import cv2 as cv


def data_form(image_path):
    """
    Convert data format.
    """
```

```
    """
    with open(image_path, 'rb') as f:
        base64_data = base64.b64encode(f.read())
        data = base64_data.decode()
    return data


def show_result(image_path, result, out_path):
    """
    Draw results on image.
    """
    image = cv.imread(image_path)
    for target in result:
        box = target['location']
        label = target['name']
        score = target['score']
        first_point = (box['left'], box['top'])
        last_point = (box['left'] + box['width'], box['top'] + box['height'])
        cv.rectangle(image, first_point, last_point, (0, 255, 0), 2)
        cv.putText(image, label + ':' + '%.4f' % score, first_point, cv.FONT_HERSHEY_SIMPLEX, 0.5,(255,0,0), 1, cv.LINE_AA)
    cv.imwrite(out_path, image)
    img = cv.resize(image, (600,600))


def detection(image_path, header, request_url, out_path):
    """
    Processing detection results.
    """
    data = data_form(image_path)
    request_payload = {
        "image": data,
        "threshold": "0.7"}
    response = requests.post(request_url,data=json.dumps(request_payload),headers=header).text
    print('response : ', response)
    tt = json.loads(response)
    result = tt['results']
    show_result(image_path, result, out_path)
    # 打印预测结果
    print('result : ', result)


#### 调用鉴权接口获取token
def get_access_token(host):
    """
    get access_token.
    """
    response = requests.get(host)
    if response:
        return response.json()['access_token']
    else:
        print('fail to get access_token')
        return None


if __name__ == "__main__":
    # 获取API Key、Secret Key（需要修改）
    AK = '【控制台应用列表处复制】'
    SK = '【控制台应用列表处复制】'
    host = 'https://aip.baidubce.com/oauth/2.0/token?grant_type=client_credentials&client_id={}&client_secret={}'.format(AK, SK)
    # 获取access_token
    access_token = get_access_token(host)
    # 本地数据集存放地址 （需要修改）
    image_path = './检测/道路交通电子眼检测/道路交通电子眼检测_测试集/image/19530.jpg'
    # Header
    header = {"Content-Type": "application/json"}
    # API接口地址 （需要修改）
    request_address = 'https://aip.baidubce.com/rpc/2.0/ai_custom_bml/v1/detection/trafficV1'【配置详情页处复制】
    # 拼接API接口地址和access_token
    request_url = '{}?access_token='.format(request_address) + access_token
    # 预测输出路径 （需要修改）
```

```
out_path = './output/detection_result.jpg'
# 调用API
detection(image_path, header, request_url, out_path)
```

用BML实现实例分割

目录

1.实例分割简介
2.平台入口
3.准备数据
3.1 自动驾驶道路实例分割数据介绍
3.2 创建及导入数据集
4.训练模型
5.模型分析和调优
6.部署模型

用BML实现实例分割：以自动驾驶道路实例分割为例

实例分割简介

亲爱的开发者您好，欢迎使用百度BML全功能AI开发平台开启您的AI开发之旅！

实例分割是主流的计算机视觉任务，对比物体检测，支持用多边形标注训练数据，模型可像素级识别目标。适合图中有多个主体、需识别其位置或轮廓的场景。其主要应用场景有：

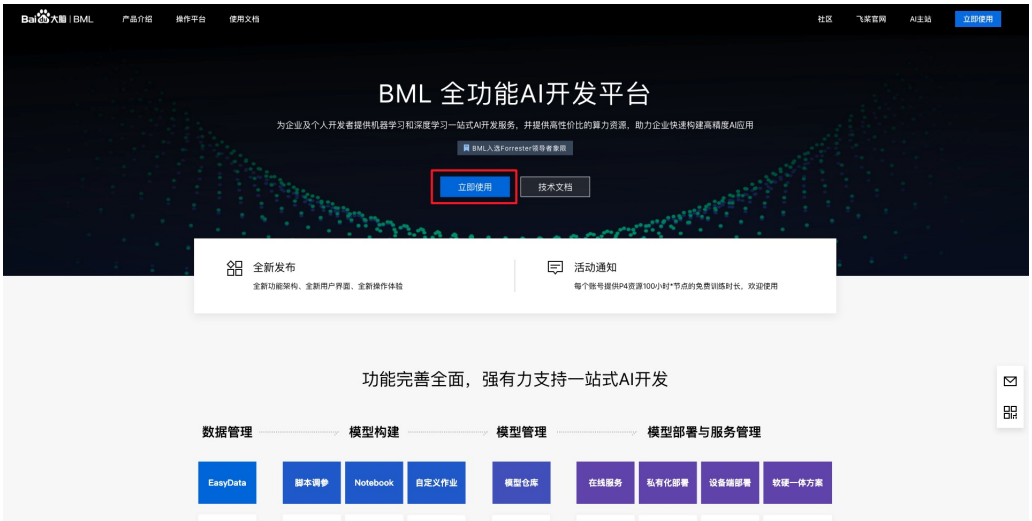
- 专业检测：应用于专业场景的图像分析，比如在卫星图像中识别建筑、道路、森林，或在医学图像中定位病灶、测量面积等。
- 智能交通：识别道路信息，包括车道标记、交通标志等。

下文中将以自动驾驶道路实例分割任务为例，分步骤向您详细介绍如何使用百度BML全功能AI开发平台开发您自己的实例分割模型。

自动驾驶道路实例分割任务简介：
道路场景分割在自动驾驶领域至关重要，主要应用在自动驾驶中的可行驶区域划分、自动驾驶路径规划、高精地图构建以及辅助驾驶的AR导航。

平台入口

BML全功能AI开发平台为企业及个人开发者提供机器学习和深度学习一站式AI开发服务，并提供高性价比的算力资源，助力企业快速构建高精度AI应用，进入官方网站点击【立即使用】。



准备数据

准备数据是AI模型开发的关键一环，训练数据的质量决定了训练所得模型效果可达到的上限，下面来介绍数据规范与相关操作步骤。

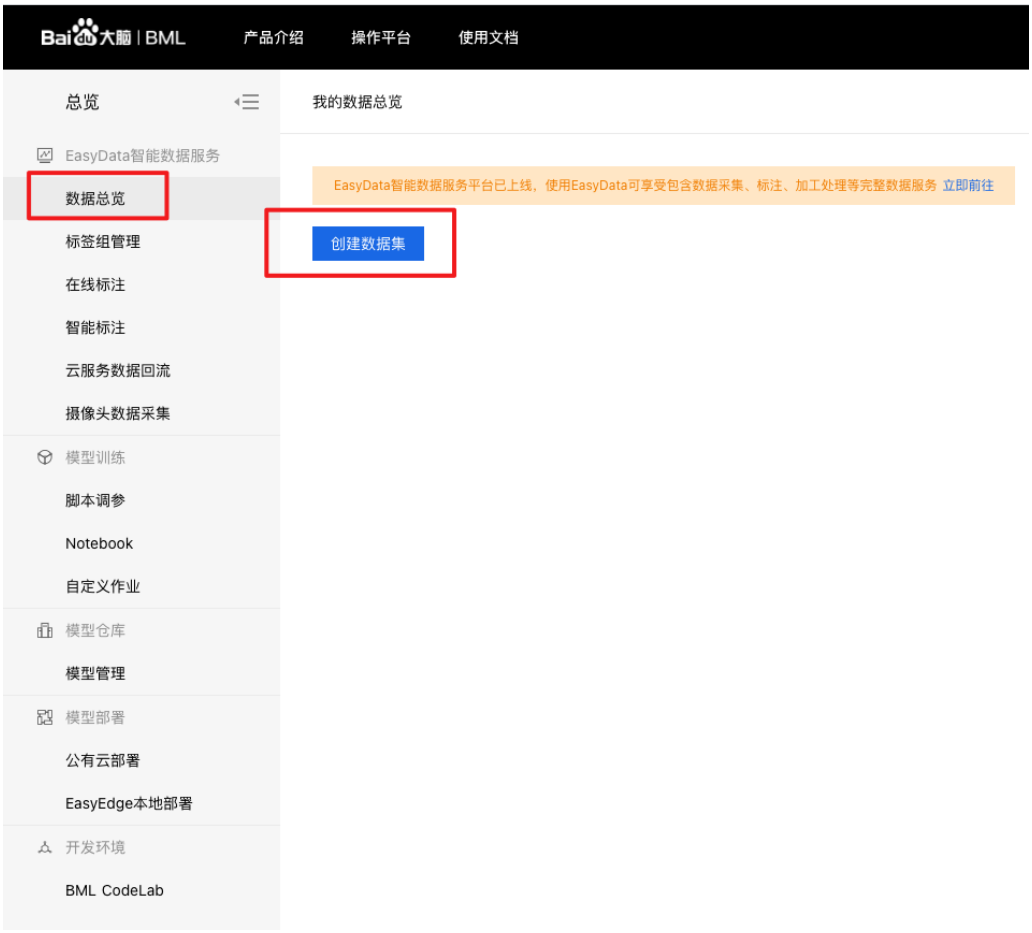
自动驾驶道路实例分割数据介绍

本文采用来自apollo道路实例分割数据集作为示例，总共268张图片，分为rider、bicycle、motorbicycle、truck、person、bicycle_group、bus、car、motorbicycle_group、tricycle十个类别。

导入平台即可使用，数据下载链接：Apollo道路实例分割数据集-json格式。

创建及导入数据集

1、在官网界面点击【数据总览】，进入数据集操作界面，点击【创建数据集】。



2、进入创建数据集界面，填写相关信息，选择数据和标注类型（注意训练集、验证集、测试集需要分开创建）。



3、数据集创建完成后，可以在数据总览界面看到刚才创建好的数据集ID，点击【导入】，将自己要训练的数据集导入。



以本地导入-上传压缩包为例：导入方式选择【本地导入】，选择标注格式，点击【上传压缩包】。



仔细阅读上传压缩包格式要求，可点击【下载示例压缩包】确认格式：



确认格式无误后，点击【已阅读并上传】，注意上传时不要关闭网页：



点击【确认并返回】后自动开始导入：



可看到【标注状态】为100%，如果数据集没有全部标注，可使用平台【智能标注】功能。

训练模型

BML上提供了预置模型调参、NoteBook建模、自定义作业三种开发模式，开发难度和开发的灵活性程度不一，分别满足不同水平和需求的开发者。

本文以使用者最多的预置模型调参开发模式为例，示意训练模型的基本步骤。

1、进入bml官方平台点击【预置模型调参】-【计算机视觉模型】，点击【创建】。



2、填写项目信息并点击【新建】。



3、点击【新建任务】。



4、点击【+请选择】，勾选刚刚上传数据集下的所有标签类，点击右下角【确定】。



5、可以选择上传验证集和测试集。验证集用来确定模型训练过程中超参数的调整。测试集用来获得更客观的模型效果评估结果。如果选择不上传，系统也会自动从已上传的训练集中分割出验证和测试集。



6、配置网络，网络选型参考：网络选型介绍。



7、配置超参数。 如果选择脚本编辑为超参来源，可在脚本编辑部分代码框内自定义超参数。超参数配置参考：超参数选择



8、可填写相关信息，并发布模型。也可以模型训练完成后再根据训练结果决定是否发布。



9、根据自身的周期和经费安排，配置计算资源。



10、最后点击【提交训练任务】，进入模型训练。



模型分析和调优

1、获取评估报告：点击【模型仓库】-【模型管理】，点击对应任务的【版本列表】查看训练好的模型，点击【评估报告】，评估报告如下图所示。



2、点击【校验模型】。

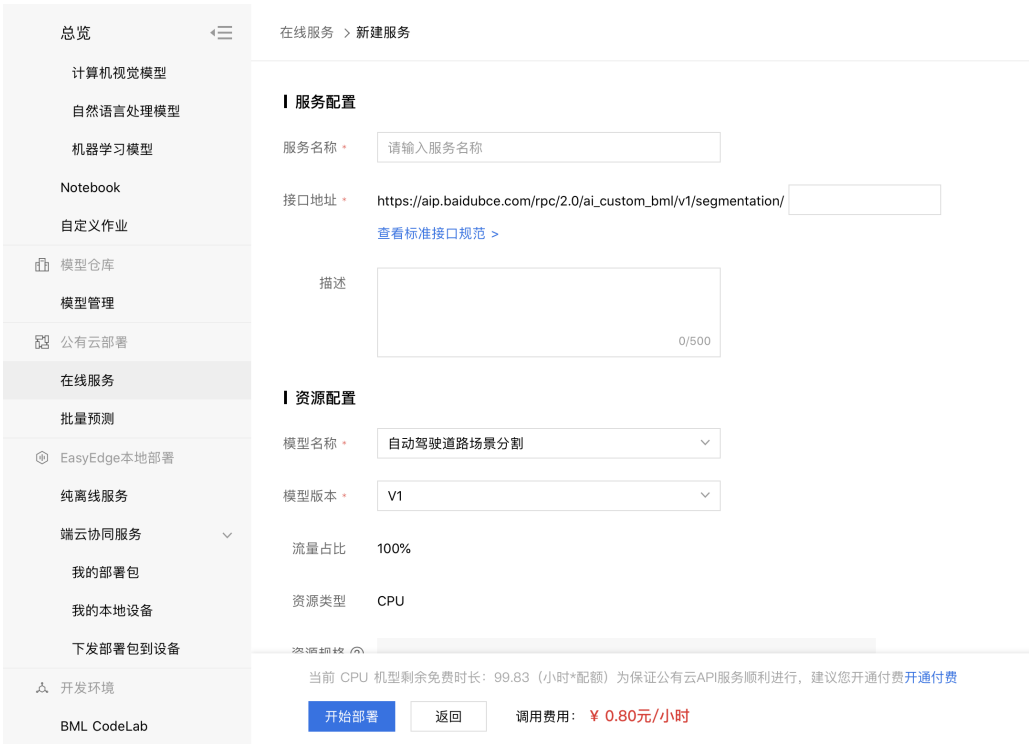


部署模型

1、在模型管理中，可选择公有云部署，端云协同服务，和批量预测三种方式发布模型。具体参考：如何选择部署方式



2、在模型部署中，用户按照自己情况填写信息完成模型部署。下图以公有云部署为例。



用BML实现文本分类

目录

1.文本分类简介
2.平台入口
3.准备数据
3.1 数据规范
3.2 创建及导入数据集

4.训练模型
5.校验模型
6.部署模型

用BML实现文本分类：以中文新闻文本标题分类为例

文本分类简介

亲爱的开发者您好，欢迎使用百度BML全功能AI开发平台开启您的AI开发之旅！

在我们的生活和工作中，很多事情都可以转化为一个分类问题来解决，比如“上班坐公交还是坐地铁”、“吃米饭还是吃面条”等等可以转化为二分类问题。自然语言处理领域也是这样，大量的任务可以用文本分类的方式来解决，比如垃圾文本识别、涉黄涉暴文本识别、意图识别、文本匹配、命名实体识别等，有着极其广泛的应用场景：

- 投诉信息分类：训练客服投诉信息的自动分类，将每个用户投诉的内容进行分类管理，节省大量客服人力。
- 媒体文章分类：训练网络媒体文章的自动分类，进而实现各类文章的自动分类。
- 文本审核：定制训练文本审核的模型，如训练文本中是否含有违规/偏激性质的描述。
- 其他：尽情脑洞大开，训练你希望实现的文本分类(单标签)模型。

定制文本分类的模型，是基于自建分类体系的机器学习方法，可实现文本按内容类型做自动分类。平台目前提供的文本分类模型包括：文本分类（单标签）和文本分类（多标签）两种模型类型，请您根据自己的业务场景来选择合适的模型。

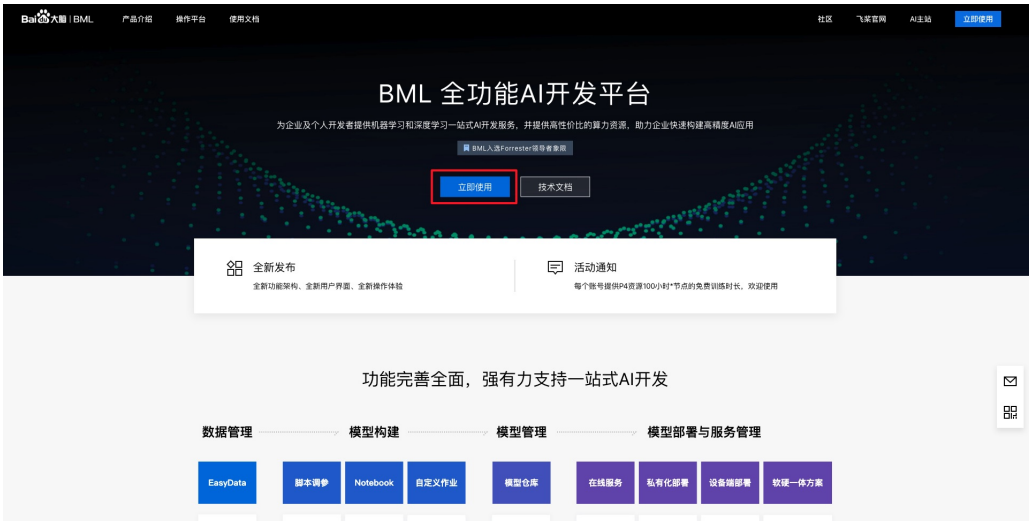
- 文本分类（单标签）场景：如您对网络文章进行舆情分析，判断舆情是正向评价还是负向评价，此问题属于单标签的文本分类场景；
- 文本分类（多标签）场景：如您对网络文章进行板块划分，文章可能属于娱乐、国际、生活等多个标签，则可使用多标签的文本分类模型；

下文中将以中文新闻文本标题分类为例，分步骤向您详细介绍如何使用百度BML全功能AI开发平台开发您自己的文本分类模型。

中文新闻文本标题分类任务简介：
新闻分类是文本分类中常见的应用场景。在传统分类模式下，往往是通过人工对新闻内容进行核对，从而将新闻划分到合适的类别中。这种方式会消耗大量的人力资源，并且效率不高。采用深度学习的方法可以取得较高的分类精度，是新闻推荐等场景下的基础任务。

平台入口

BML全功能AI开发平台为企业及个人开发者提供机器学习和深度学习一站式AI开发服务，并提供高性价比的算力资源，助力企业快速构建高精度AI应用，进入官方网站点击【立即使用】。



准备数据

准备数据是AI模型开发的关键一环，训练数据的质量决定了训练所得模型效果可达到的上限。
本文采用中文新闻文本标题分类数据集进行示例，数据链接：中文新闻文本标题分类
下面来介绍数据规范与相关操作步骤。

数据规范

本地上传数据规范：

- 可支持单个txt文本文件上传、Excel上传，或将多个文本文件以压缩包的方式统一上传。

- 压缩包内的一个文本文件将作为一个样本上传，压缩包格式为.zip格式，压缩包内文件类型支持txt，编码仅支持UTF-8
- 上传过程中存在文本内容完全一样的样本，将会做去重处理。
- 文本文件类型为txt，单次上传限制100个文本文件。
- 单个文本大小限制在4M以内，文本文件大小限制长度最大4096个UTF-8字符。
- 文本文件内数据格式要求为"文本内容\n"（即每行一个未标注样本，使用回车换行），每一行表示一组数据，每组数据的字符数建议不超过512个，超出将被截断。
- 单个数据集大小限制为10万文本文件，超出后会被忽略。

创建及导入数据集

1、在官网界面点击【数据总览】，进入数据集操作界面。



2、进入创建数据集界面，选择好数据类型和标注类型等信息，点击完成。



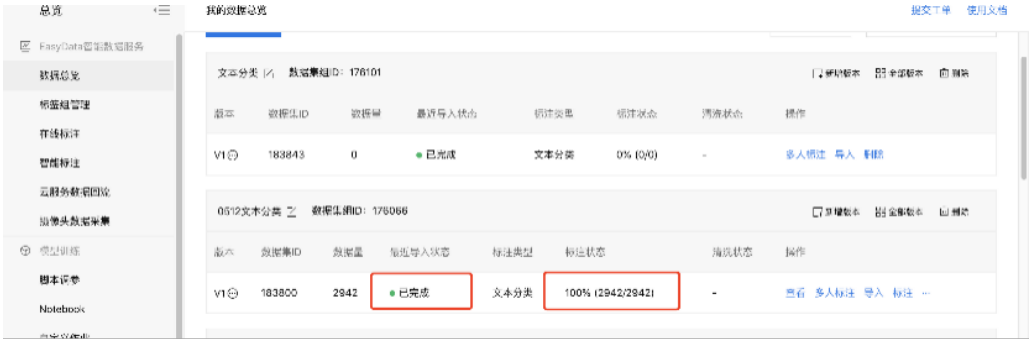
3、数据集创建完成后，可以在数据总览界面看到刚才创建好的数据集ID。



4、点击【导入】，将自己要训练的数据集导入，如这里选择本地导入Excel文件方式导入数据集，点击添加文件，然后确认并返回，完成数据集的导入。



5、回到数据总览界面，可实时查看导入状态信息和标注状态，最终成功则显示已完成。



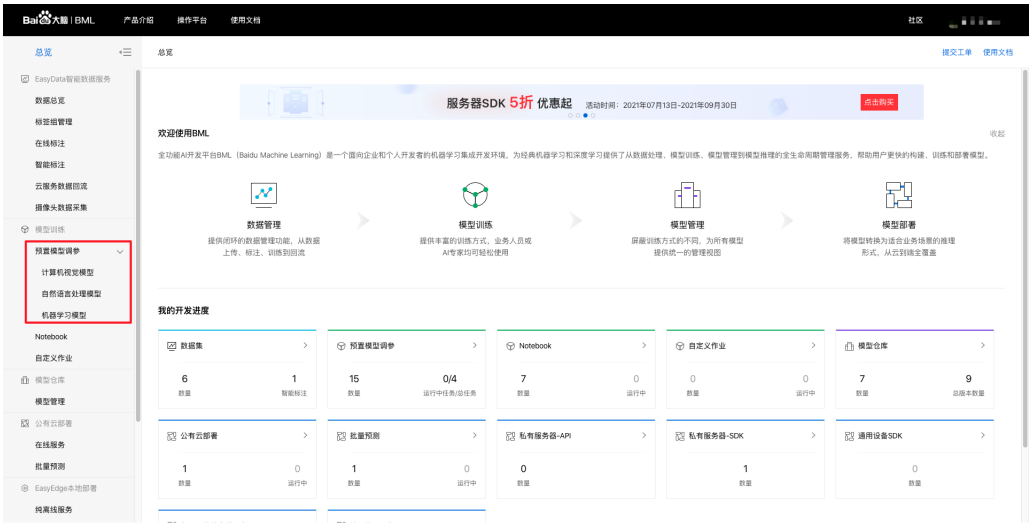
训练模型

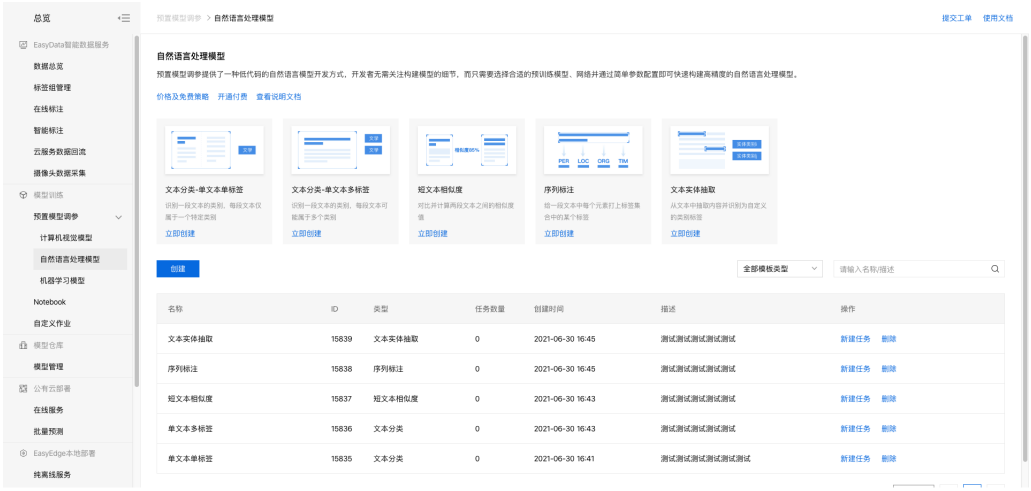
BML上提供了预置模型调参、NoteBook建模、自定义作业三种开发模式，开发难度和开发的灵活性程度不一，分别满足不同水平和需求的开发者。

当前NLP方向仅支持使用者最多的预置模型调参模式，后续将陆续支持NoteBook建模、自定义作业开发模式。

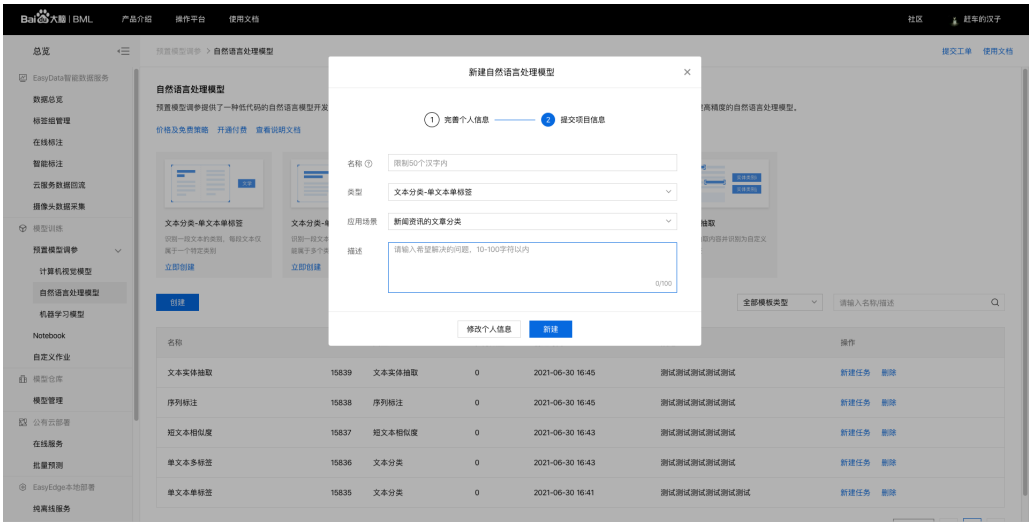
本文将采用预置模型调参模式示意训练模型的基本步骤。

1、进入bml官方平台点击立即使用预置模型调参，点击【预置模型调参】-【自然语言处理模型】，进入操作台。





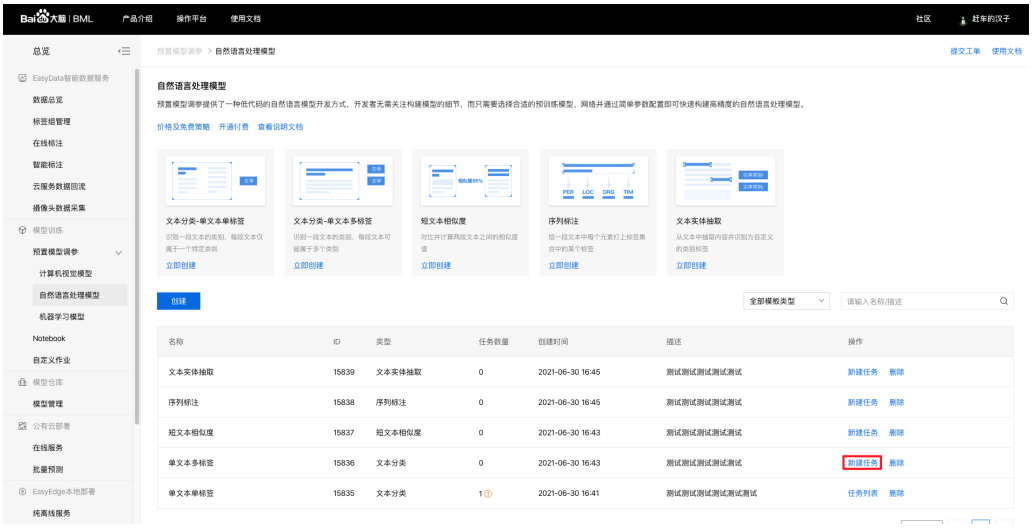
2、在模型列表下点击创建模型。



选择训练类型为文本分类-单文本单标签，填写模型信息（名称、描述信息等）后，点击【创建】。

3、配置NLP分类训练任务。

之前已经建立好训练模型，现在开始配置NLP训练训练，点击【新建任务】。



在新建任务面板中，可查看相关项目“基本信息”、“配置任务类型”、“添加数据”、“配置网络”等操作，在添加任务时添加刚才数据集确定添加。



配置模型网络，选择显卡类型和是否启用分布式训练等。



配置好后，可提交训练任务开始训练。

5、训练完成。

等待训练过程，完成后显示训练完成，用户可查看训练时长，训练结果的评估报告等信息。

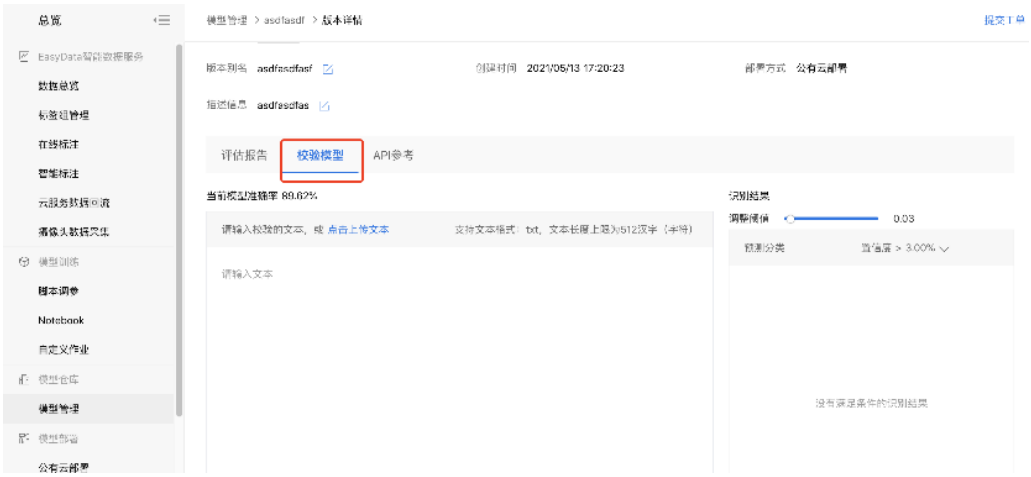


在评估报告中可查看本次训练过程的准确率，精确率等指标报告信息。



校验模型

启动模型校验。



用训练好的模型对本次的新闻类型文本（可实时输入或用户上传）进行分类校验。



部署模型

1、在模型管理中，可选择公有云和本地部署两种方式发布模型。



2、在模型部署中，用户按照自己情况填写信息完成模型部署。



用BML评价短文本相似度

目录

1.短文本相似度简介
2.平台入口
3.准备数据
3.1 数据规范
3.2 创建及导入数据集
4.训练模型
5.校验模型
6.部署模型

用BML评价短文本相似度：以新冠疫情相似句对判定为例

短文本相似度简介

亲爱的开发者您好，欢迎使用百度BML全功能AI开发平台开启您的AI开发之旅！

短文本相似度，即求解两个短文本之间的相似程度；它是文本匹配任务或文本蕴含任务的一种特殊形式，返回文本之间相似程度的具体数值。在工业界中，短文本相似度计算占有举足轻重的地位。

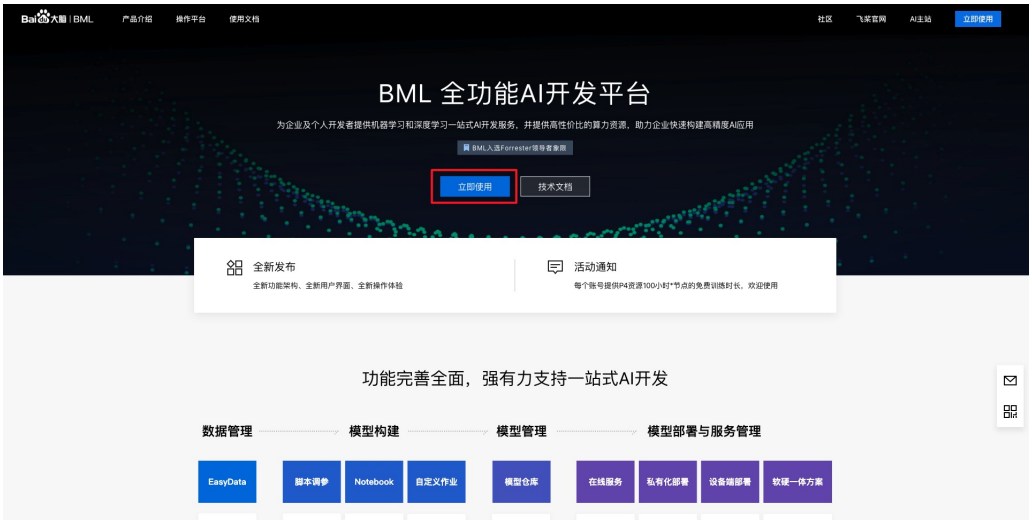
例如：在问答系统任务（问答机器人）中，我们往往会人为地配置一些常用并且描述清晰的问题及其对应的回答，我们将这些配置好的问题称之为“标准问”。当用户进行提问时，常常将用户的问题与所有配置好的标准问进行相似度计算，找出与用户问题最相似的标准问，并返回其答案给用户，这样就完成了一次问答操作。

下文中将以新冠疫情相似句对判定任务为例，分步骤向您详细介绍如何使用百度BML全功能AI开发平台开发您自己的短文本相似度评价模型。

新冠疫情相似句对判定任务简介：
面对疫情抗击，疫情知识问答应用得到普遍推广。如何通过自然语言技术将问答进行相似分类仍然是一个有价值的问题。如识别患者相似问题，有利于理解患者真正诉求，帮助快速匹配准确答案，提升患者获得感；归纳医生相似答案，有助于分析答案规范性，保证疫情期间问诊规范性，避免误诊。

平台入口

BML全功能AI开发平台为企业及个人开发者提供机器学习和深度学习一站式AI开发服务，并提供高性价比的算力资源，助力企业快速构建高精度AI应用，进入官方网站点击【立即使用】。



准备数据

准备数据是AI模型开发的关键一环，训练数据的质量决定了训练所得模型效果可达到的上限.
本文采用新冠疫情相似句对数据集进行示例，数据链接：新冠疫情相似句对判定。
下面来介绍数据规范与相关操作步骤。

数据规范

本地上传数据规范：

- 可支持单个txt文本文件上传、Excel上传，或将多个文本文件以压缩包的方式统一上传。
- 压缩包内的一个文本文件将作为一个样本上传，压缩包格式为.zip格式，压缩包内文件类型支持txt，编码仅支持UTF-8
- 上传过程中存在文本内容完全一样的样本，将会做去重处理。
- 文本文件类型为txt，单次上传限制100个文本文件。
- 单个文本大小限制在4M以内，文本文件大小限制长度最大4096个UTF-8字符。
- 文本文件内数据格式要求为"文本内容\n"（即每行一个未标注样本，使用回车换行），每一行表示一组数据，每组数据的字符数建议不超过512个，超出将被截断。
- 单个数据集大小限制为10万文本文件，超出后会被忽略。

创建及导入数据集

1、在官网界面点击【数据总览】，进入数据集操作界面。



2、进入创建数据集界面，选择好数据类型和标注类型等信息，点击完成。



3、数据集创建完成后，可以在数据总览界面看到刚才创建好的数据集ID。



4、点击【导入】，将自己要训练的数据集导入，如这里选择本地导入Excel文件方式导入数据集，点击添加文件，然后确认并返回，完成数据集的导入。



5、回到数据总览界面，可实时查看导入状态信息和标注状态，最终成功则显示已完成。



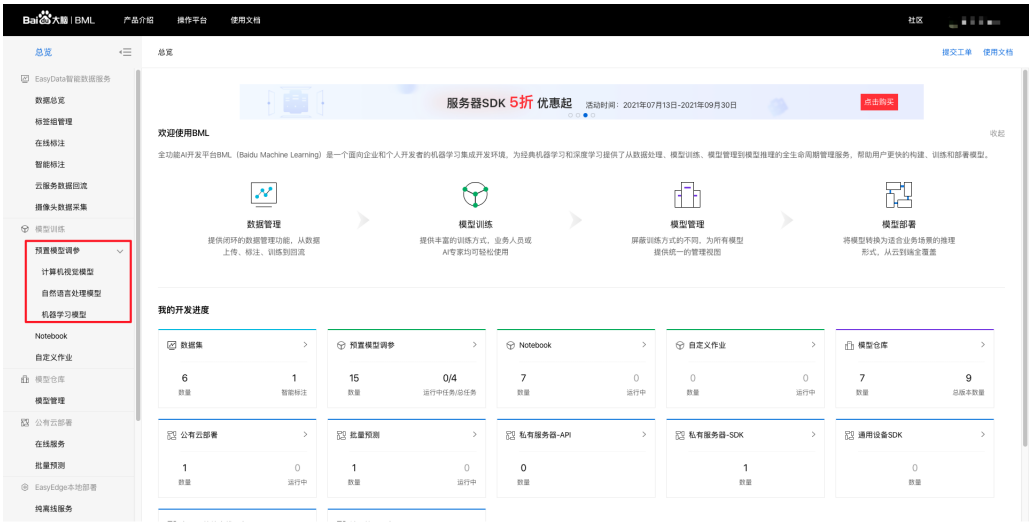
训练模型

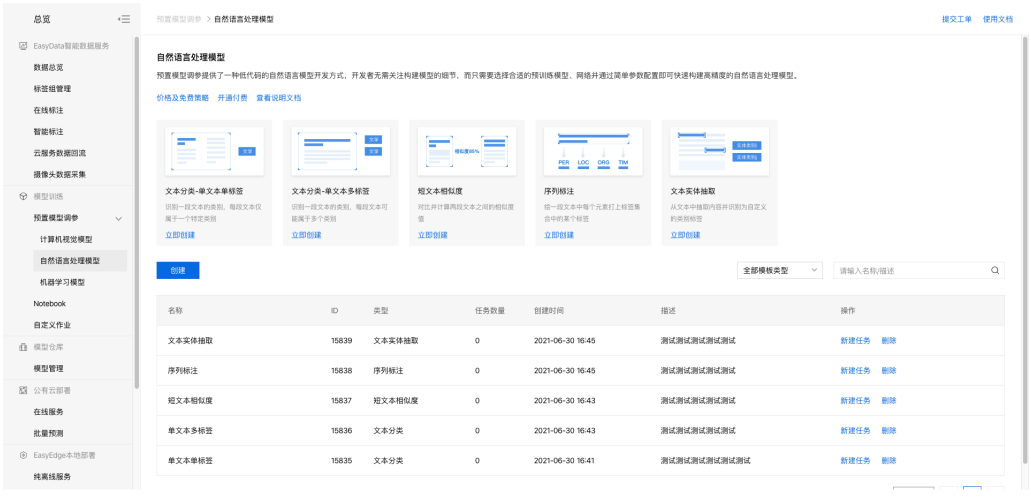
BML上提供了预置模型调参、NoteBook建模、自定义作业三种开发模式，开发难度和开发的灵活性程度不一，分别满足不同水平和需求的开发者。

当前NLP方向仅支持使用者最多的预置模型调参，后续将陆续支持NoteBook建模、自定义作业开发模式。

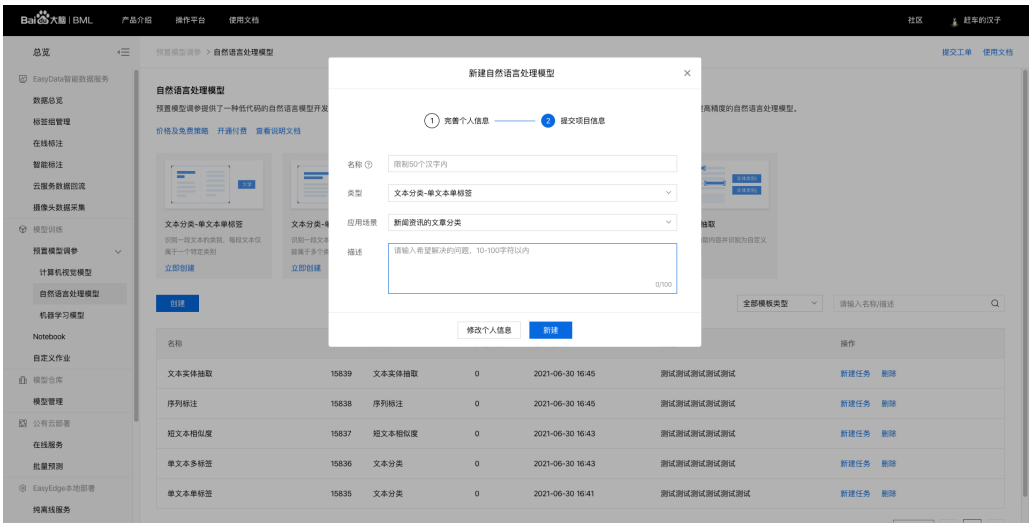
本文将采用预置模型调参开发模式示意训练模型的基本步骤。

1、进入bml官方平台点击立即使用预置模型调参，点击【预置模型调参】-【自然语言处理模型】，进入操作台。



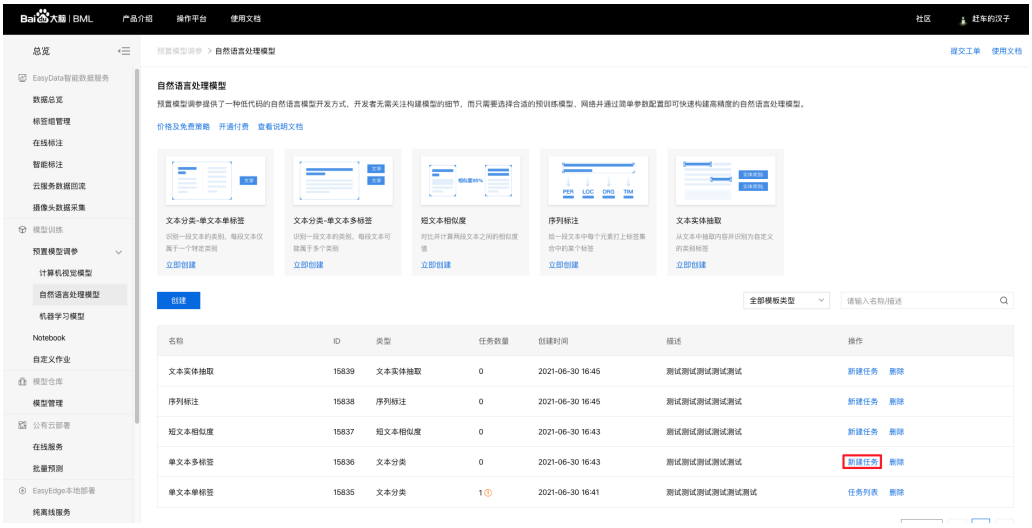


2、点击创建模型。



选择训练类型为短文本相似度，填写模型信息（名称、描述信息等）后，点击【创建】。

3、在模型列表下可看到所建模型信息，若平台已上传好数据集，则点击新建任务跳转至创建训练任务。



4、配置NLP分类训练任务。

之前已经建立好训练模型，现在开始配置NLP训练训练，点击【新建任务】。



在新建任务面板中，可查看相关项目“基本信息”、“配置任务类型”、“添加数据”、“配置网络”等操作，在添加任务时添加刚才数据集确定添加。



配置模型网络，选择显卡类型和是否启用分布式训练等。



配置好后，可提交训练任务开始训练。

5、训练完成。

等待训练过程，完成后显示训练完成，用户可查看训练时长，训练结果的评估报告等信息。

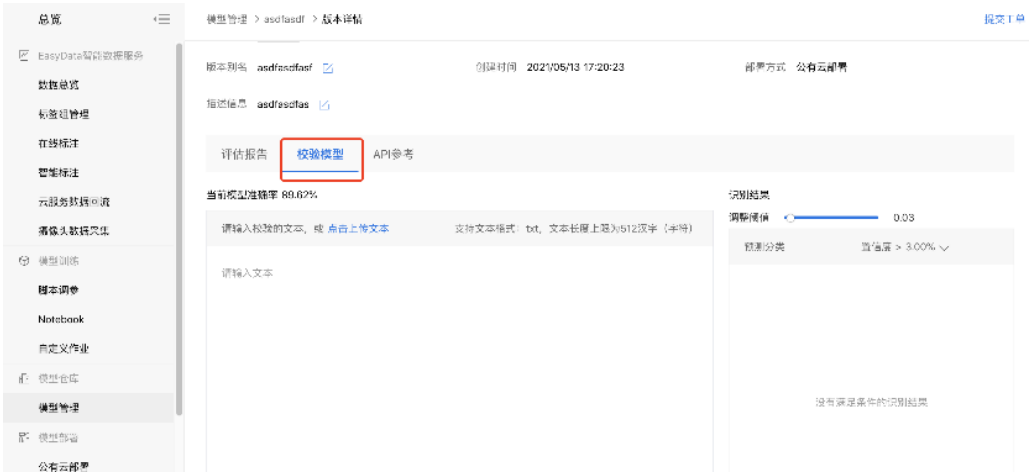


在评估报告中可查看本次训练过程的准确率，精确率等指标报告信息。



校验模型

启动模型校验。

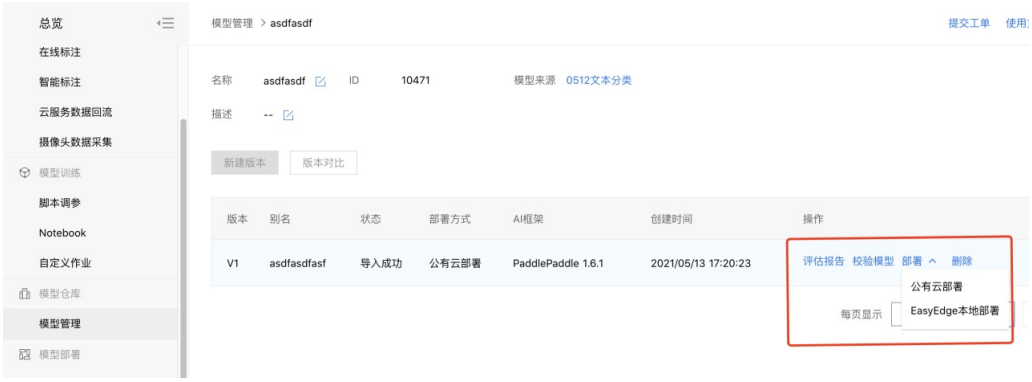


用训练好的模型对输入的文本进行相似度校验。



部署模型

1、在模型管理中，可选择公有云和本地部署两种方式发布模型。



2、在模型部署中，用户按照自己情况填写信息完成模型部署。



用BML实现文本实体抽取

目录

1.文本实体抽取简介
2.平台入口
3.准备数据
3.1 数据规范
3.2 创建及导入数据集
4.训练模型
5.校验模型
6.部署模型

用BML实现文本实体抽取：以简历信息抽取为例

文本实体抽取简介

亲爱的开发者您好，欢迎使用百度BML全功能AI开发平台开启您的AI开发之旅！

文本实体抽取是一项重要的自然语言处理任务，主要是实现对文本进行内容抽取，并识别为自定义的实体类别。

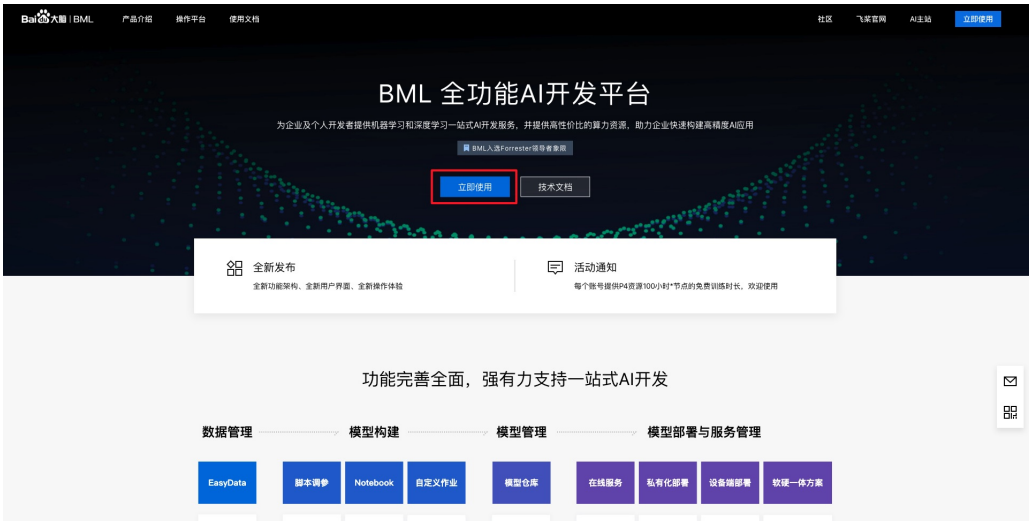
例如：对内容中的关键实体进行识别和抽取，如金融研报信息识别，法律案件文书实体抽取，医疗病例实体抽取等多种应用场景。

下文中将以简历信息抽取为例，分步骤向您详细介绍如何使用百度BML全功能AI开发平台开发您自己的文本实体抽取模型。

简历信息抽取任务简介：
AI简历分析可以帮助HR在海量简历中迅速准确地找到和JD匹配的候选人，在各大招聘网站、各大公司的简历筛选过程中都有着广泛的应用。其中，简历信息抽取任务是简历分析的基础任务之一。

平台入口

BML全功能AI开发平台为企业及个人开发者提供机器学习和深度学习一站式AI开发服务，并提供高性价比的算力资源，助力企业快速构建高精度AI应用，进入官方网站点击【立即使用】。



准备数据

准备数据是AI模型开发的关键一环，训练数据的质量决定了训练所得模型效果可达到的上限。
数据可参考：简历信息抽取。
下面来介绍数据规范与相关操作步骤。

数据规范

本地上传数据规范：

- 可支持Excel上传，Excel文件内首行为表头，每行为一个样本。
- 每个样本字符数不得超过10000个字符（约5000个字），超出将被截断。
- 文件格式支持xlsx格式，单次可上传100个文件。

创建及导入数据集

1、在官网界面点击【数据总览】，进入数据集操作界面。



2、进入创建数据集界面，选择好数据类型和标注类型等信息，点击完成。



3、数据集创建完成后，可以在数据总览界面看到刚才创建好的数据集ID。



4、点击【导入】，将自己要训练的数据集导入，如这里选择本地导入Excel文件方式导入数据集，点击添加文件，然后确认并返回，完成数据集的导入。



5、回到数据总览界面，可实时查看导入状态信息和标注状态，最终成功则显示已完成。



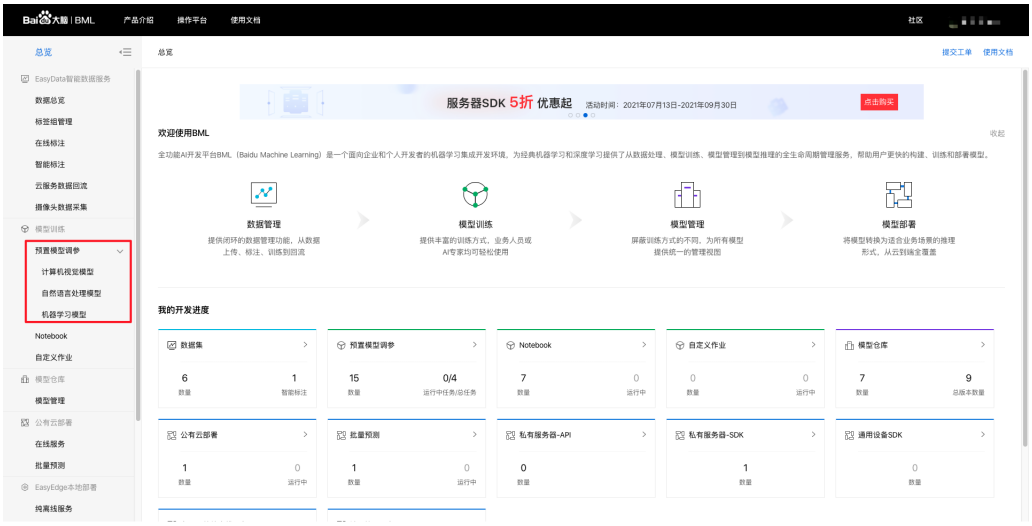
训练模型

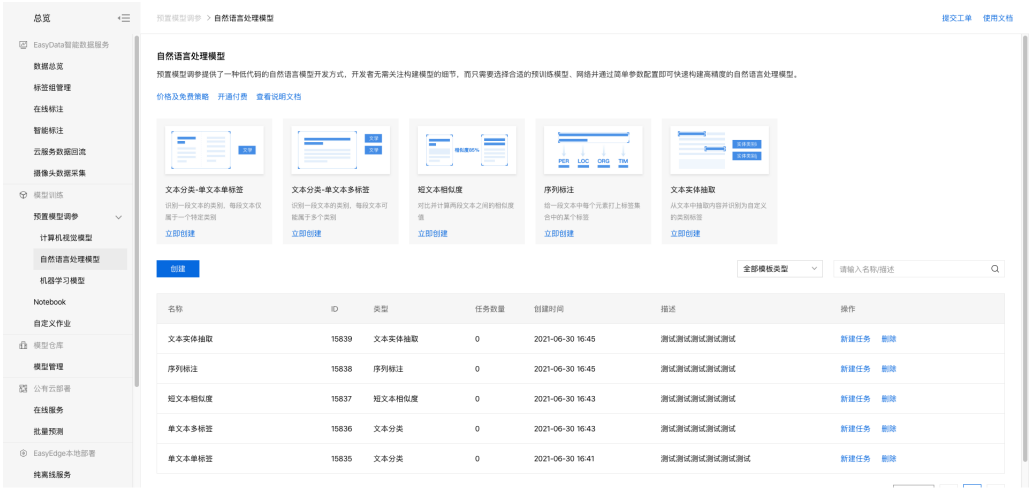
BML上提供了预置模型调参、NoteBook建模、自定义作业三种开发模式，开发难度和开发的灵活性程度不一，分别满足不同水平和需求的开发者。

当前NLP方向仅支持使用者最多的预置模型调参模式，后续将陆续支持NoteBook建模、自定义作业开发模式。

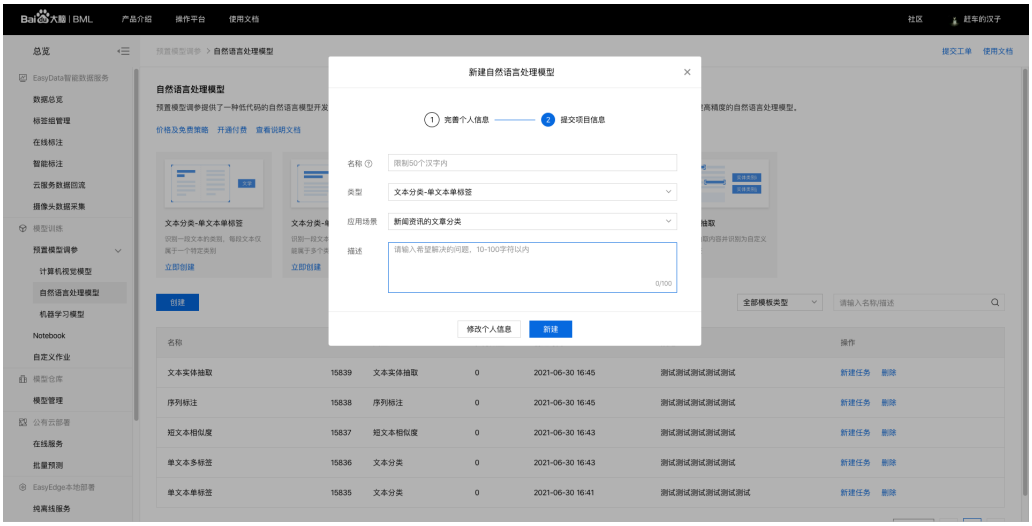
本文将采用预置模型调参开发模式示意训练模型的基本步骤。

1、进入bml官方平台点击立即使用预置模型调参，点击【预置模型调参】-【自然语言处理模型】，进入操作台。



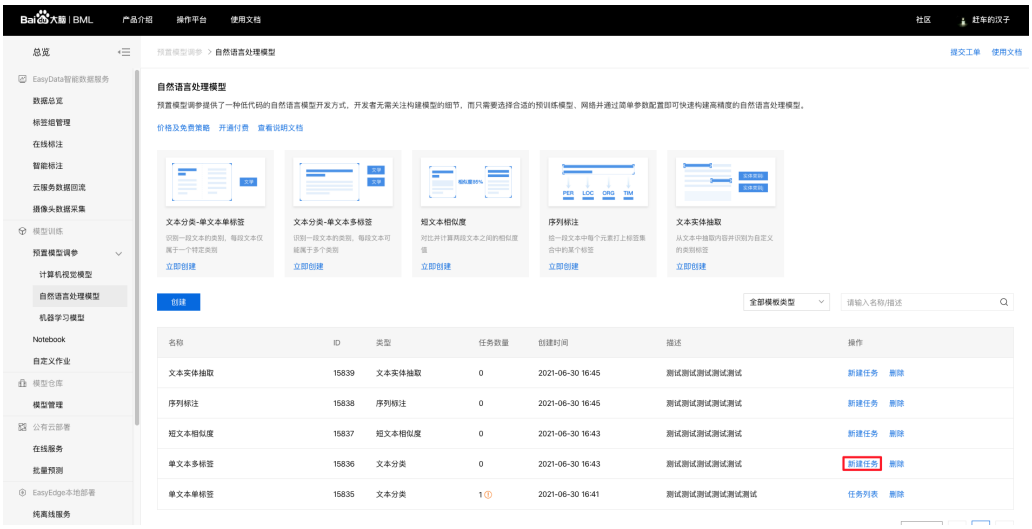


2、在模型列表下点击创建模型。



选择训练类型为文本实体抽取，填写模型信息（名称、描述信息等）后，点击【创建】。

3、在模型列表下可看到所建模型信息，若平台已上传好数据集，则点击新建任务跳转至创建训练任务。



4、配置NLP分类训练任务。

之前已经建立好训练模型，现在开始配置NLP训练训练，点击【新建任务】。



在新建任务面板中，可查看相关项目“基本信息”、“配置任务类型”、“添加数据”、“配置网络”等操作，在添加任务时添加刚才数据集确定添加。



配置模型网络，选择显卡类型和是否启用分布式训练等。



配置好后，可提交训练任务开始训练。

5、训练完成。

等待训练过程，完成后显示训练完成，用户可查看训练时长，训练结果的评估报告等信息。

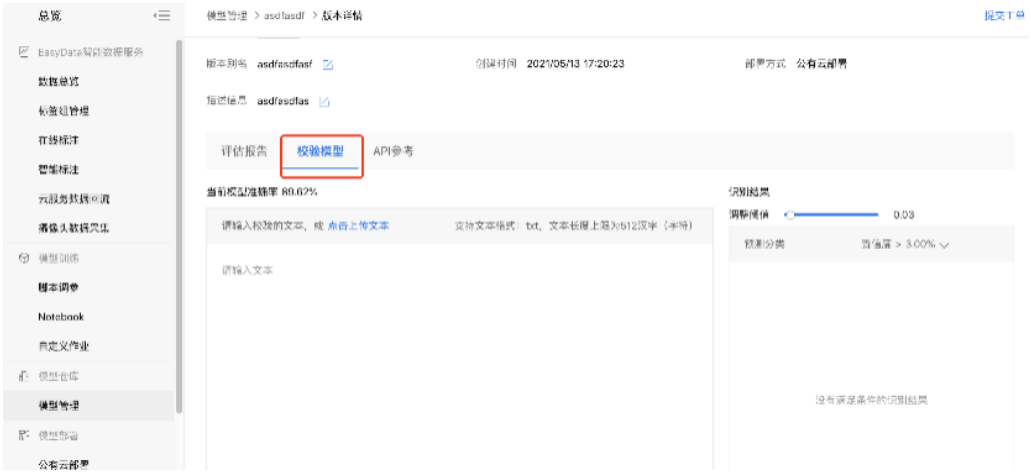


在评估报告中可查看本次训练过程的准确率，精确率等指标报告信息。



校验模型

启动模型校验。



用训练好的模型对输入的文本进行相似度校验。



部署模型

1、在模型管理中，可选择公有云和本地部署两种方式发布模型。



2、在模型部署中，用户按照自己情况填写信息完成模型部署。



用BML实现序列标注

目录

1.序列标注简介
2.平台入口
3.准备数据
3.1 数据规范
3.2 创建及导入数据集
4.训练模型
5.校验模型
6.部署模型

用BML实现序列标注：以快递单信息处理为例

序列标注简介

亲爱的开发者您好，欢迎使用百度BML全功能AI开发平台开启您的AI开发之旅！

序列标注（Sequence Tagging）是NLP中最基础的任务，应用十分广泛，如分词、词性标注（POS tagging）、命名实体识别（Named Entity Recognition，NER）、关键词抽取、语义角色标注（Semantic Role Labeling）、槽位抽取（Slot Filling）等实质上都属于序列标注的范畴。

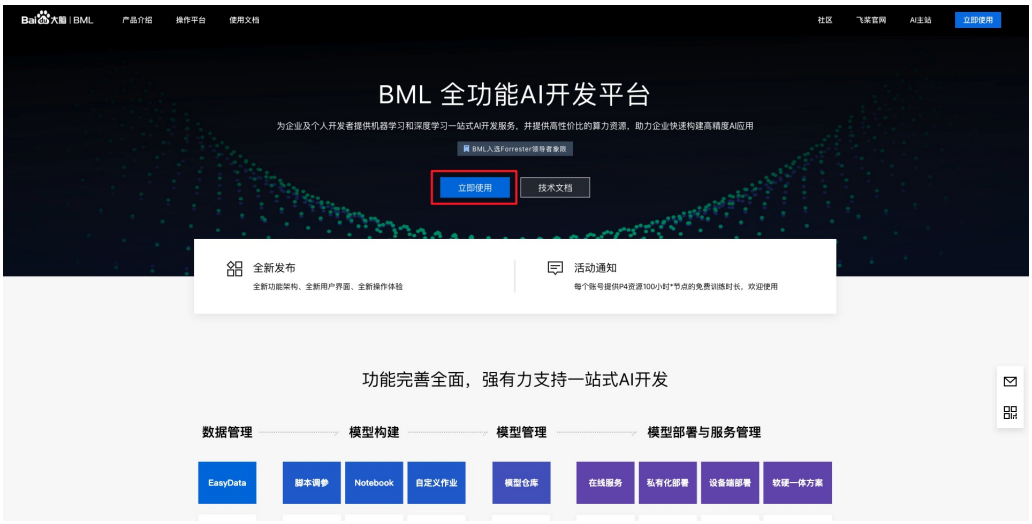
序列标注问题可以认为是分类问题的一个推广，或者是更复杂的结构预测（structure prediction）问题的简单形式。序列标注问题的输入是一个观测序列，输出是一个标记序列或状态序列。问题的目标在于学习一个模型，使它能够对观测序列给出标记序列作为预测。

下文中将以快递单信息处理为例，分步骤向您详细介绍如何使用百度BML全功能AI开发平台开发您自己的序列标注模型。

快递单信息处理任务简介：
快递单信息处理主要是研究如何从用户提供的快递单中，抽取姓名、电话、省、市、区、详细地址等内容，形成结构化信息。辅助物流行业从业者进行有效信息的提取，从而降低客户填单的成本，在物流场景有着广泛的应用。

平台入口

BML全功能AI开发平台为企业及个人开发者提供机器学习和深度学习一站式AI开发服务，并提供高性价比的算力资源，助力企业快速构建高精度AI应用，进入官方网站点击【立即使用】。



准备数据

准备数据是AI模型开发的关键一环，训练数据的质量决定了训练所得模型效果可达到的上限。
本文采用快递单数据作为示例，数据链接：序列标注任务，如何从快递单中抽取关键信息。
下面来介绍数据规范与相关操作步骤。

数据规范

本地上传数据规范：

- 序列标注上传的是压缩包，压缩包内需包含txt或tsv格式的文本文件和标注为JSON格式的标签映射文件，编码仅支持UTF-8，单个文件最大可支持40MB。
- 标注数据格式要求为“文本内容\t标注结果\n”文本内容和标注结果都需要按字切分，每一行表示一组数据，每组数据的字符数建议不超过512个。
- 标签映射文件名须为”label_map.json“,标签映射文件的格式为：{"key":"value"}。
- 平台限免阶段每个账户最多支持100万条样本数据，超出后会被平台忽略。

创建及导入数据集

1、在官网界面点击【数据总览】，进入数据集操作界面。



2、进入创建数据集界面，选择好数据类型和标注类型等信息，点击完成。



3、数据集创建完成后，可以在数据总览界面看到刚才创建好的数据集ID。



4、点击【导入】，将自己要训练的数据集导入，如这里选择本地导入Excel文件方式导入数据集，点击添加文件，然后确认并返回，完成数据集的导入。



5、回到数据总览界面，可实时查看导入状态信息和标注状态，最终成功则显示已完成。



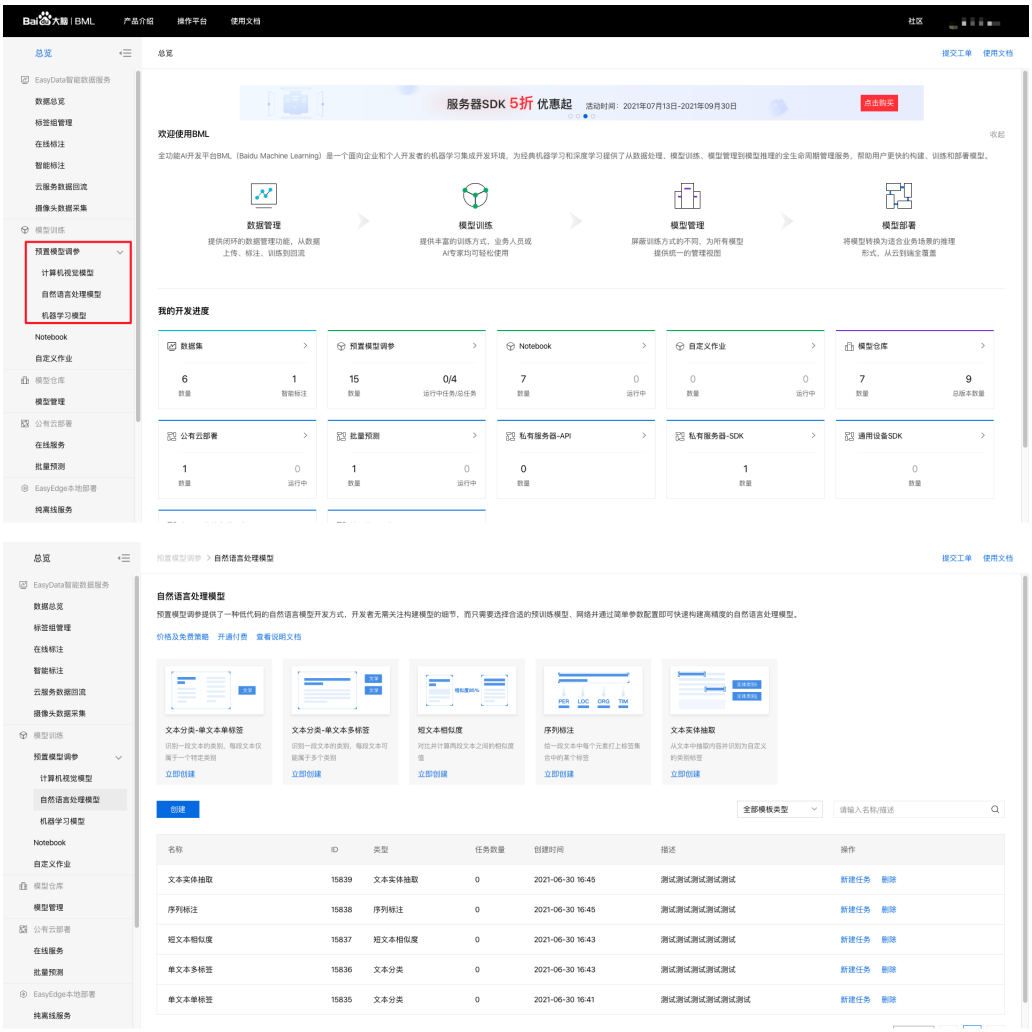
训练模型

BML上提供了预置模型调参、NoteBook建模、自定义作业三种开发模式，开发难度和开发的灵活性程度不一，分别满足不同水平和需求的开发者。

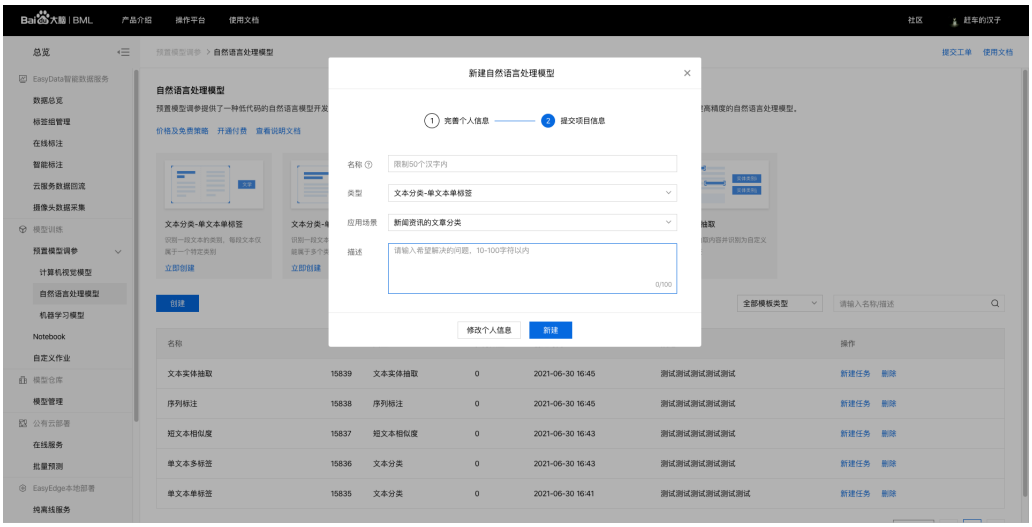
当前NLP方向仅支持使用者最多的预置模型调参，后续将陆续支持NoteBook建模、自定义作业开发模式。

本文将采用预置模型调参开发模式示意训练模型的基本步骤。

1、进入bml官方平台点击立即使用预置模型调参，点击【预置模型调参】-【自然语言处理模型】，进入操作台。

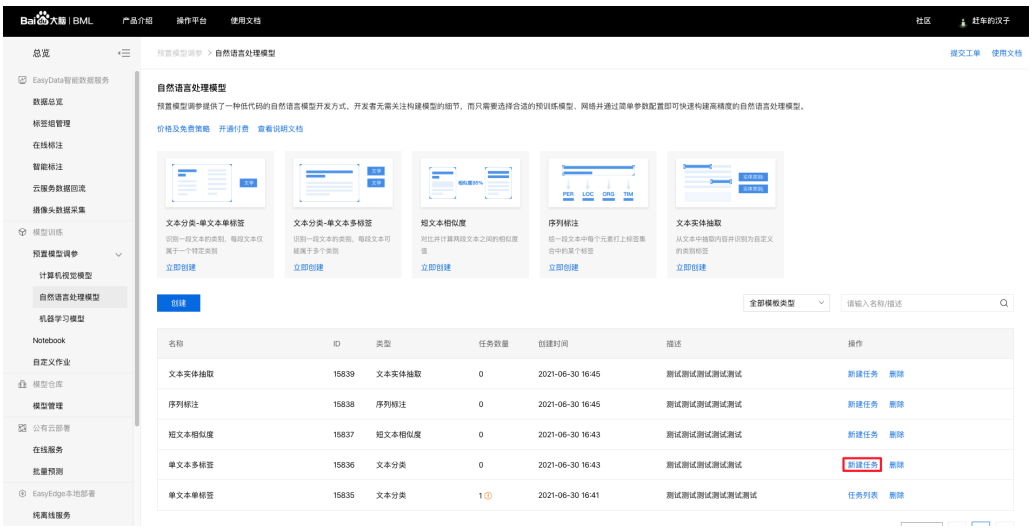


2、在模型列表下点击创建模型。



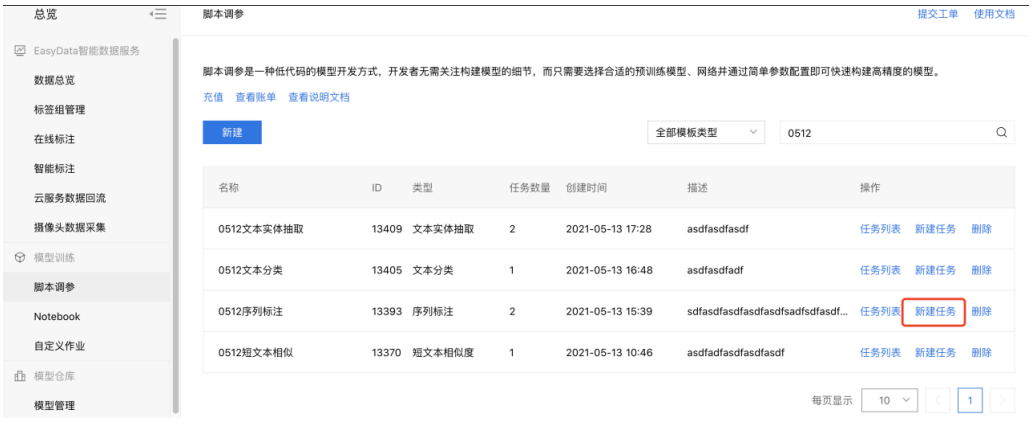
选择训练类型为序列标注，填写模型信息（名称、描述信息等）后，点击【创建】。

3、在模型列表下可看到所建模型信息，若平台已上传好数据集，则点击新建任务跳转至创建训练任务。



4、配置NLP分类训练任务。

之前已经建立好训练模型，现在开始配置NLP训练训练，点击【新建任务】。



在新建任务面板中，可查看相关项目“基本信息”、“配置任务类型”、“添加数据”、“配置网络”等操作，在添加任务时添加刚才数据集确定添加。



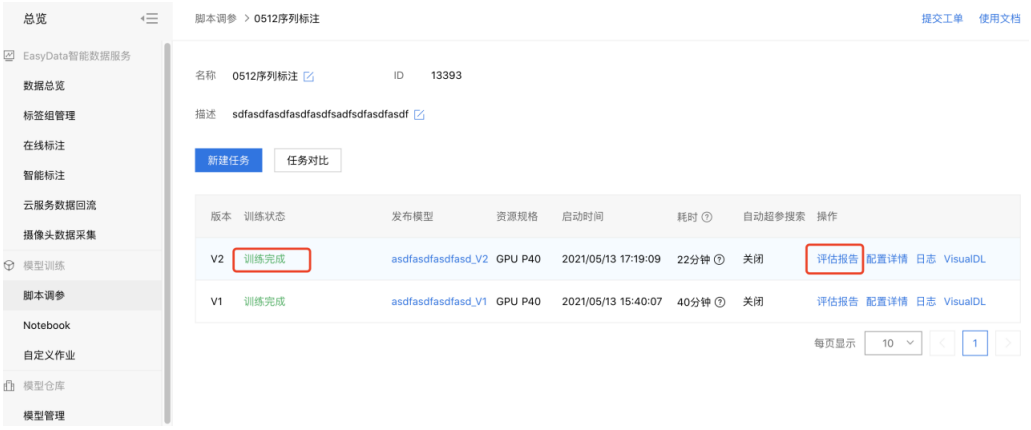
配置模型网络，选择显卡类型和是否启用分布式训练等。



配置好后，可提交训练任务开始训练。

5、训练完成。

等待训练过程，完成后显示训练完成，用户可查看训练时长，训练结果的评估报告等信息。

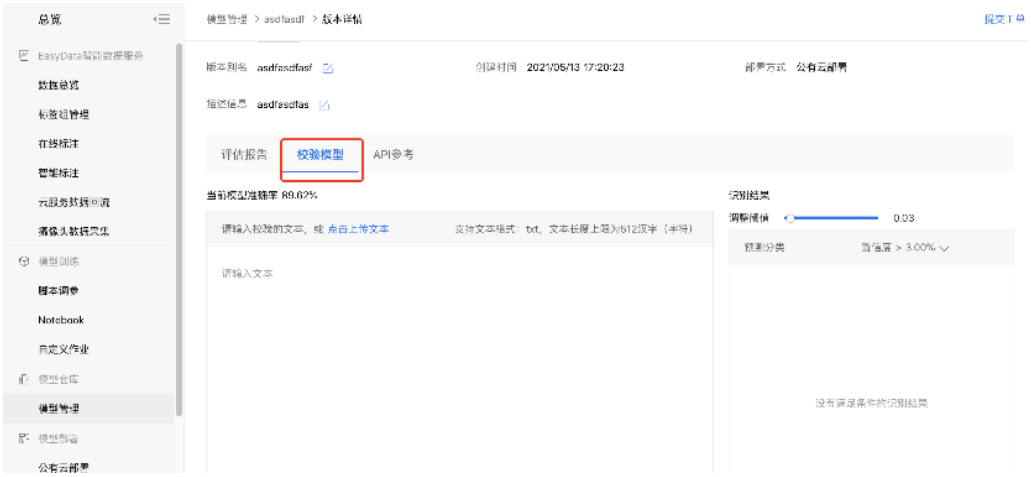


在评估报告中可查看本次训练过程的准确率，精确率等指标报告信息。



校验模型

启动模型校验。



用训练好的模型对输入的文本进行相似度校验。



部署模型

1、在模型管理中，可选择公有云和本地部署两种方式发布模型。



2、在模型部署中，用户按照自己情况填写信息完成模型部署。



用BML实现表格预测

目录

1.表格预测简介
2.平台入口
3.准备数据
3.1 数据要求
3.2 创建及导入数据集
4.训练模型
4.1AutoML模式
4.2专家模式
5.模型分析和调优
6.部署模型
6.1公有云部署

用BML实现表格预测：以空气质量预测为例

表格预测简介

亲爱的开发者您好，欢迎使用百度BML全功能AI开发平台开启您的AI开发之旅！

定制表格数据预测模型，旨在帮助用户通过机器学习技术从表格化数据中发现潜在规律，从而创建机器学习模型，并基于机器学习模型处理新的数据，为业务应用生成预测结果。根据预测数据的不同，可以分为如下几种类型：

- 回归：目标列是连续的实数范围，或者属于某一段连续的实数区间。如在销量预测场景中，销量值可能是某个取值范围内的任意值，解决该问题的模型属于回归模型。
- 二分类：目标列是离散值，且只有两种可能的取值。如在精准营销场景中预测一个用户是否为潜在购买用户，其目标列仅存在“True”和“False”两种取值，解决该问题的模型属于二分类模型。
- 多分类：目标列是离散值，并具有有限的可能取值。如在用户分类场景中，根据用户的历史消费数据，将用户划分到不同消费偏好的类别中，解决该问题的模型属于多分类模型。

其应用场景包括：

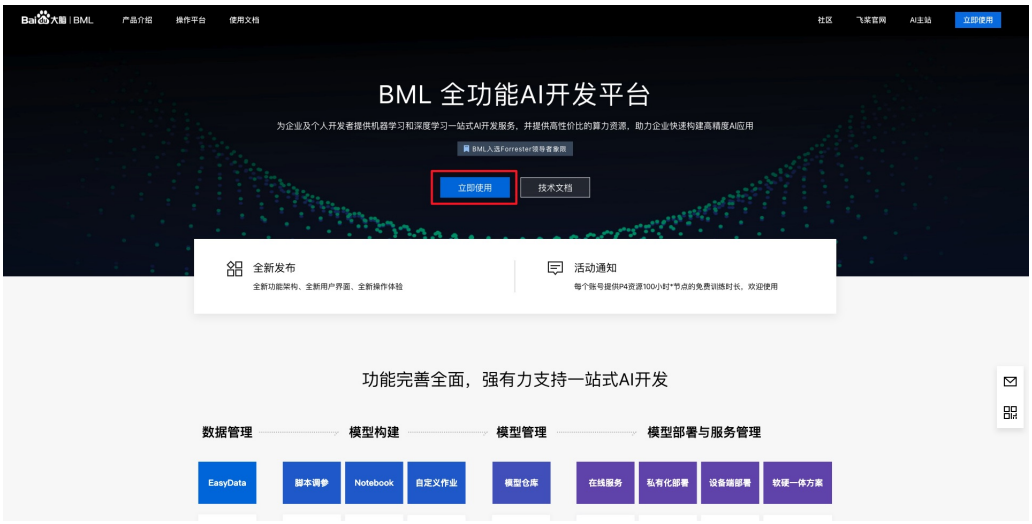
- 精准营销：从客户消费记录中挖掘客户群的共有特征，分析出客户的购物偏好，从而实现广告的精准投放。
- 信用评分：金融公司分析客户的历史行为数据，建立用户信用模型，从而确定贷款额度等。
- 价格预测：从历史数据中发现商品的变化规律以及影响价格的因素，从而为未来的商业行为提供支持。
- 客户流失预测：根据客户历史数据获得数据挖掘模型，从而生成客户流失预测列表，为市场营销策略提供有价值的业务洞察力。

下文中将以空气质量预测为例，分步骤向您详细介绍如何使用百度BML全功能AI开发平台开发您自己的表格预测模型。

空气质量预测任务：
现代社会科技进步，人们的生活质量逐步提高，但伴随着各类工业和科技的发展，环境问题凸显，最初人们粗放式的经济发展方式在一定程度上对环境造成不可逆转的破坏。在各种环境污染问题，空气污染问题又是如今人们关注的重中之重。北京是我国首都，同时也是我国空气污染较为严重的几个北方城市之一，因此关注北京市空气污染情况是我国观测空气污染情况的重要关注对象之一。

平台入口

BML全功能AI开发平台为企业及个人开发者提供机器学习和深度学习一站式AI开发服务，并提供高性价比的算力资源，助力企业快速构建高精度AI应用，进入官方网站点击【立即使用】。



准备数据

准备数据是AI模型开发的关键一环，训练数据的质量决定了训练所得模型效果可达到的上限。
本文采用北京2010.1.1至2014.12.31之间的空气污染数据作为示例，数据集链接：空气污染数据。
下面来介绍数据规范与相关操作步骤。

数据要求

数据文件格式要求如下：

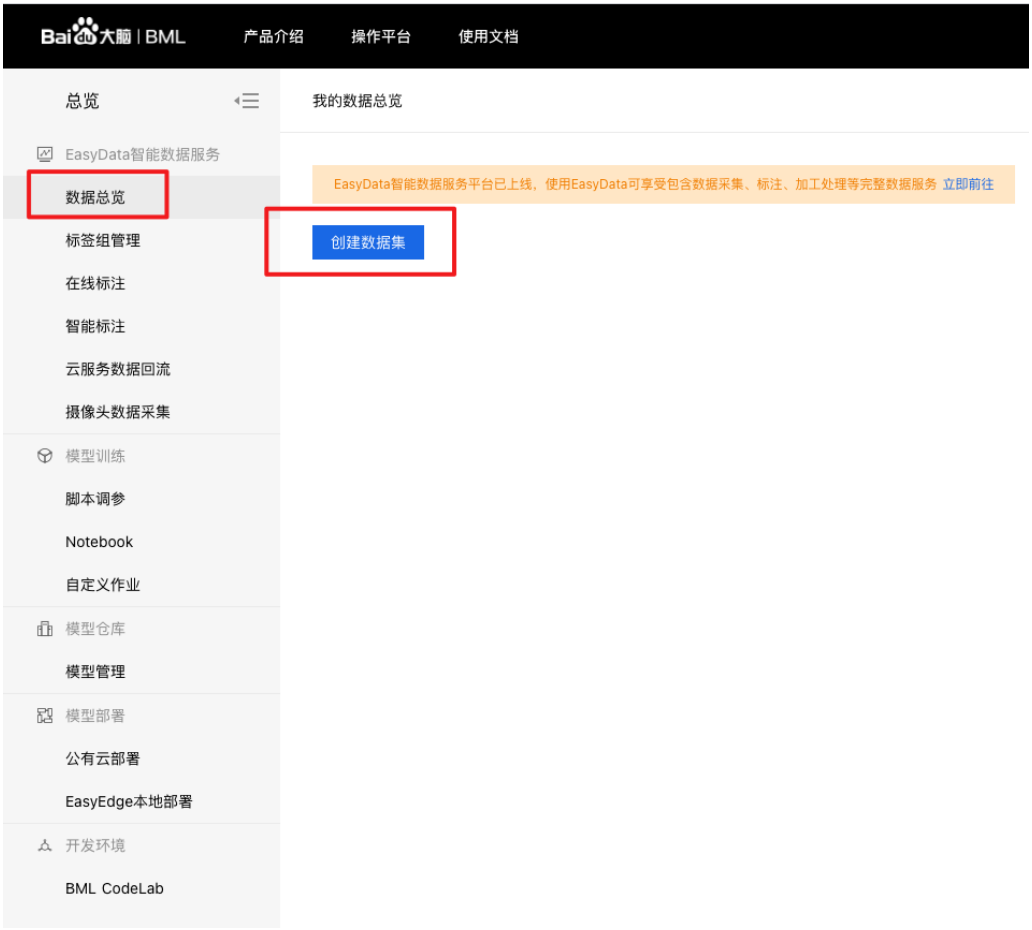
- 目前仅支持CSV格式的数据文件。
- 一次仅能上传一个文件，可以是一个CSV文件或由多个CSV文件压缩成的zip包。
- 单个上传文件大小不能超过5GB。
- 一个数据集包含的总文件大小不能超过20GB。

数据内容要求如下：

- 当数据文件包含列名时，列名称可以包含字母、数字和下划线（_），但不能以下划线开头。
- 文件内容以换行符（即字符“\n”，或称为LF）分隔各行，行内容以英文逗号（即字符“,”）分隔各列必须包含要预测的值即目标列，且目标列的数据类型会决定模型的类型。
- 文件中文本列取值长度不能超过4096个字符。
- 必须至少包含两列，且不得超过1000列。
- 数据集的总行数不能超过1000万行。
- zip包中的多个CSV文件必须使用相同的编码格式，都包含列名或都不包含列名；且列的顺序必须保持一致。
- 在扩充数据集时，新导入数据文件的首行与数据集的列名相同时，将被视为列名，否则将被视作数据。

创建及导入数据集

1、在官网界面点击【数据总览】，进入数据集操作界面，点击【创建数据集】。



2、点击数据类型中的【表格】，输入数据集名称，点击【完成】结束创建。



完成创建后，可以在数据集列表中查看新建的数据集。系统默认生成V1版本，当前数据内容为空，可以通过导入的方式向其中添加数据。

3、数据集创建完成后，可以在数据总览界面看到刚才创建好的数据集ID，点击【导入】，将自己要训练的数据集导入。



在导入方式的下拉菜单中选择【上传CSV文件】或【上传压缩包】，前者支持数据预览，后者由多个CSV文件组成，不支持预览。



若导入方式为上传CSV文件，则点击【上传CSV文件】，上传压缩包步骤与其一致。



上传文件界面有上传数据的格式要求，数据符合要求，则点击【已阅读并上传】。



根据数据文件的实际情况进行列名设置：

- 设置首行为列名：将导入的数据文件中的首行作为列名。
- 设置首行非列名：此时系统会自动生成列名，而将首行作为数据。

点击【确认并返回】后自动开始导入：



在数据集列表下可以看到所建数据集，数据上传后，需要等待一段时间，刷新后，当最近导入状态由【正在导入】转为【已完成】即为上传成功。



训练模型

BML上提供了脚本调参、NoteBook建模、自定义作业三种开发模式，开发难度和开发的灵活性程度不一，分别满足不同水平和需求的开发者。

本文以使用者最多的脚本调参开发模式为例，示意训练模型的基本步骤。

1、进入BML官方平台点击立即使用脚本调参，点击【脚本调参】，进入操作台。点击【新建】，进入模型创建信息页面。



点击【结构化数据】，填写模型创建信息。点击【创建】，完成模型创建。



注：可点击【修改个人信息】更新正确账号信息，方便后续人工审核和官方支持。



2、点击【新建任务】，进入模型训练配置阶段。



表格预测任务支持AutoML和专家两种运行方式：

- AutoML模式：全流程自动建模，用户只需设置数据集、目标列以及制定任务类型即可，而无需关注数据处理以及算法配置等过程，系统会自动完成建模过程，并从中挑选最优的模型作为训练任务的运行结果。
- 专家模式：高度开放的建模方式，用户可以进行特征工程、算法、超参搜索等配置，具备相关技能的开发者可以在方式下获得更多的开发自由度。

AutoML模式

配置参数如下所示：

1、基本信息：选择【AutoML模式】，并根据实际情况填写任务备注。



2、添加数据：选择之前创建的数据集，以及目标列名称，并设置算法类型，如不确定，可选择“自动”。



3、发布模型：

- 自动发布-开：即完成训练后，系统会自动将当前任务得到的模型发布到模型仓库中。
- 自动发布-关：完成训练后，用户可以根据模型精度等再决定是否将模型发布到模型仓库。



4、配置资源：

- 运行环境：请根据数据量以及期望的运行速度进行设置。根据经验值，在建模过程中，数据会在内存中膨胀为原始大小的10倍，为保证任务顺利完成，请尽量确保配置的资源的内存不小于原始数据集的10倍。
- 选择节点数：如果设置为1，则系统以单机算法进行建模，如果设置为大于1的值，则系统将使用分布式算法。单机算法比分布式算法更丰富。在单机资源满足要求的情况下，可优先使用单节点进行建模。
- 最长训练时间：该时长指算法求解阶段的最长时长，若超过该时长，算法仍未得到结果，系统会强制结束训练任务。



专家模式

配置参数如下所示：

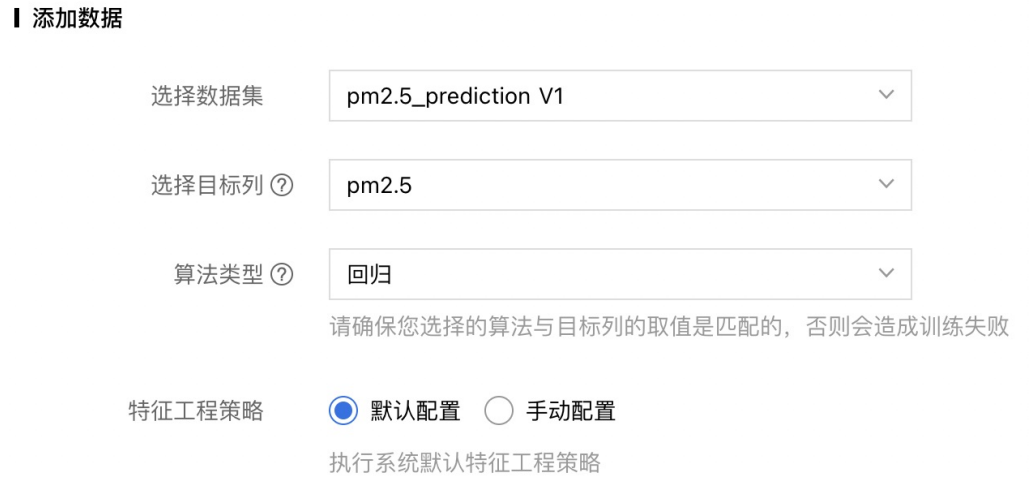
1、基本信息：选择【专家模式】，训练方法支持单机和分布式两种模式，单机模式支持更多的算法，任务备注根据实际情况填写。



2、添加数据：选择之前创建的数据集，以及目标列名称，并设置算法类型，确保选择的算法与目标列的取值是匹配的，否则会造成训练失败。

特征工程策略：

- 默认配置：执行系统默认的特征工程策略。



- 手动配置：用户可以手工配置各个特征工程算子的执行参数从而控制其执行方式。



3、配置任务：系统会根据用户选择的训练方式以及算法类型自动生成任务脚本，在不需要修改的情况下可直接启动训练。

如需自定义脚本内容，点击【立即编辑】。



自定义脚本内容过程中有如下注意事项：

- 可以自定义的部分为超参配置字典conf部分,包括是否开启超参搜索，训练模型类型和模型配置，具体见脚本中的注释。
- 训练默认开启自动超参搜索，如需关闭请手动将"hyperparameter_tune"的参数值由"True"更改为"False"。
- BML当前表格数据预测支持模型为CAT(CatBoost), LGBM(LightGBM),RF(RandomForest), LR(Logistic Regression), XGB(Xgboost), KNN(k-NearestNeighbor)。
- 在"hp_space"中已经预置可搜索的超参数，用户无需修改参数名称，随意设置可能会导致训练失败。
- 在超参搜索范围内的取值方式支持：平均采样(uniform),非平均采样(quniform), 离散值(choice), 对数平均采样(loguniform),随机整数(randint)五种，超参搜索范围设置过大可能会导致训练时间过长。
- CAT模型支持搜索的超参数，已经预置默认搜索范围，详见脚本内容。

4、发布模型：

- 自动发布-开：即完成训练后，系统会自动将当前任务得到的模型发布到模型仓库中。
- 自动发布-关：完成训练后，用户可以根据模型精度等再决定是否将模型发布到模型仓库。



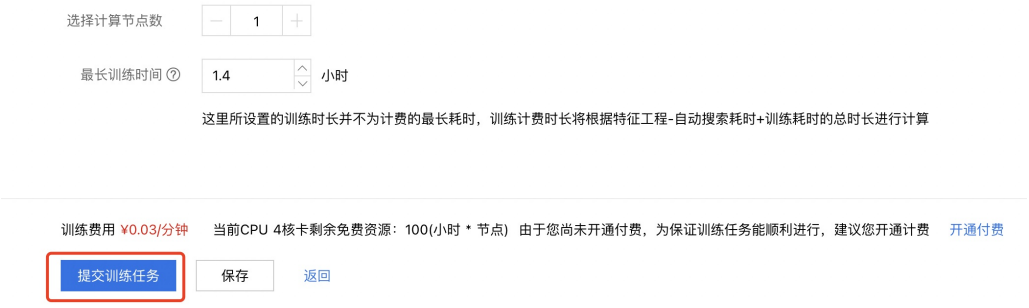
5、配置资源：

- 运行环境：请根据数据量以及期望的运行速度进行设置。根据经验值，在建模过程中，数据会在内存中膨胀为原始大小的10倍，为保证任务顺利完成，请尽量确保配置的资源的内存不小于原始数据集的10倍。

- 选择节点数：只有选择的“分布式”的训练方式才可以设置为大于1的值。
- 最长训练时间：该时长指算法求解阶段的最长时长，若超过该时长，算法仍未得到结果，系统会强制结束训练任务。



6、任务配置完成后，点击【提交训练任务】。



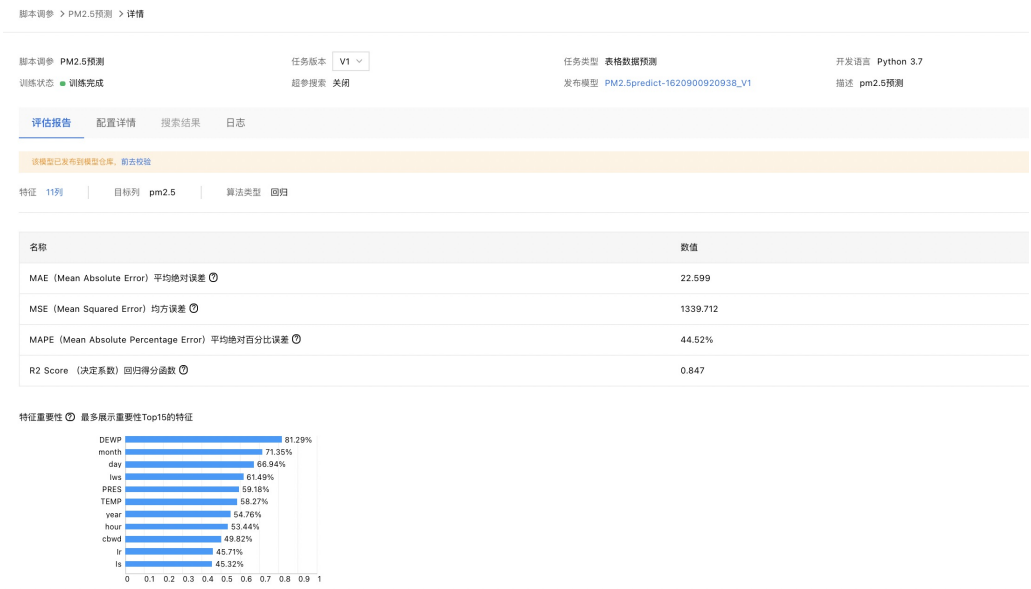
7、提交训练任务后，会进入任务列表页面，待训练状态由【训练中】变为【训练完成】时，训练结束。



8、发布模型选择自动发布时，训练结束，模型就会发布到模型仓库中。点击【评估报告】，可以查看任务结果。



任务生成的模型的报告信息，如下所示:



模型发布与校验

1、未开启自动发布模型时，需要手动将模型发布到模型仓库中。点击【发布】,进入模型发布页面。



2、设置版本别名，点击【确认】。



3、等待几分钟，待发布模型的状态由【发布中】变为模型名时，表明模型已导入到模型仓库中。



版本	训练状态	发布模型	训练方式	算法类型	资源规格	启动时间	耗时	自动超参搜索	操作
V2	训练完成	发布中	--	回归	CPU 16 核	2021/05/13 18:27:36	3分钟	关闭	评估报告 配置详情 日志
V1	训练完成	PM2.5predict-1620900920938_V1	--	回归	CPU 16 核	2021/05/13 18:10:17	4分钟	关闭	评估报告 配置详情 日志

4、点击模型名称进入校验界面，或者点击导航栏模型仓库的【模型管理】，在右侧找到对应的模型，并点击【版本列表】，点击【校验模型】，进入模型校验界面。





5、点击【启动模型校验】。校验模型需要在后端完成模型的部署，需要一定时间，请耐心等待。



6、设置完特征的取值后，点击【预测】，即可得到预测结果。





部署模型

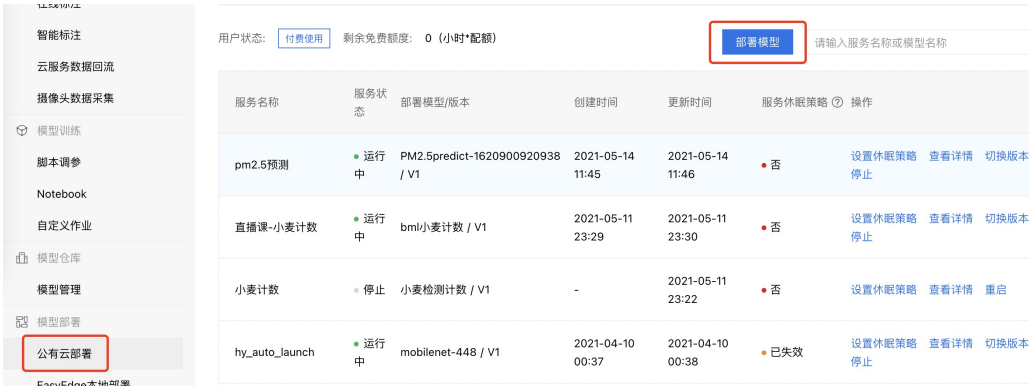
训练完成后，可将模型部署在公有云服务器上，通过API进行调用。当前表格预测模型仅支持公有云部署，即将模型部署在BML提供的云端机器资源中，您可以通过访问在线API实时获取模型结果。

公有云部署

1、将预测模型发布到模型仓库后，在模型管理界面，点击对应模型的【版本列表】，进入后点击对应版本的【部署】，进入部署页面。



或者点击左侧导航栏的【公有云部署】，进入公有云部署界面后点击【部署模型】，进入部署页面。



2、部署配置参数如下所示：

服务配置：选择对应的模型名称和模型版本，如果配置版本为模型的第一个版本，则需自定义服务名称以及接口地址后缀。



资源配置：对每个部署模型还需要进行资源配置，配置模型预测服务所需机器资源。目前仅支持CPU机型，支持按配额与实例进行灵活配置。

- 系统将运行公有云部署的机器资源1核cpu和4G内存折算为1配额，提高配额数量能提升QPS和降低推理时延。
- 实例为cpu资源运行的实例数，为保证服务的稳定性，实例数至少需要配置为2。实例数与QPS成正比关系，提高实例数时，QPS能正比增加。

3、配置完成后，点击【开始部署】，在完成部署后可以通过配置的API地址调用预测服务。



用BML实现开源大模型的预训练（Post-pretrain）

目录

1.准备工作（通过notebook或者本地IDE实现）
1.1.环境创建
1.2.代码下载
1.3.目录创建
1.4.准备预训练模型权重
1.5.准备训练数据
2.训练Pipeline说明
2.1.Step1:数据预处理
2.2.Step2:模型格式转换(HF->Megatron)
2.3.Step3:预训练(使用作业建模大模型案例实现)
2.4.Step4:模型格式转换(Megatron->HF)
2.5.Step5:快速导入并部署第三方模型

一、准备工作（通过notebook或者本地IDE实现）

预置的大模型案例，是针对处理后的数据以及转换成megatron格式的模型进行的。因此，如果需要使用自定义的模型，以及自定义的数据，可参考以下流程。

环境创建

数据处理以及模型转换均可使用作业建模+大模型案例提供的训练镜像实现，也可在任意环境实现，再将转换后的结果上传到存储中使用，以下提供了训练镜像构建时使用的基础镜像以及额外安装的第三方库。

```
docker pull nvcr.io/nvidia/pytorch:23.10-py3  # 训练镜像
pip3 install transformers==4.39.3 datasets==2.18.0 sentencepiece==0.2.0 nltk==3.8.1 accelerate==0.29.3# 第三方库
```

代码下载

目录创建

首先在个人环境或者bos存储中(使用作业建模的方式)创建自己的工作目录作为WORK_DIR，然后在WORK_DIR下分别创建code、data、output、pretrained_models目录，并上传代码bml-megatron-llm（提供的代码压缩包解压）至${WORK_DIR}/code。

准备预训练模型权重

下载好模型权重，保存至 ${WORK_DIR}/pretrained_models 目录下。

准备训练数据

下载开源数据集并保存到${WORK_DIR}/data/目录下，本案例的预训练数据集以Huggine Face上的suolyer/wudao为例，预置的大模型案例也是基于该数据集转化得到。

下载命令行：

```
#### 下载用于预训练的数据git clone
https://huggingface.co/datasets/suolyer/wudao
```

二、训练Pipeline说明

以llama2-7B模型的处理流程为例进行说明，以下每一个步骤的脚本可以在个人环境中运行，也可打包成一个sh脚本，使用作业建模实现，或者使用notebook功能进行实现。

Step1:数据预处理

预训练数据须为 MMAP 格式文件数据（含 .bin 和 .idx 文件），以 suolyer/wudao 数据集的validation.json文件为例，介绍预处理过程。

执行以下命令将预训练数据转换成 MMAP 格式（MMAP数据是一种预先执行tokenize的数据格式，可以减少训练微调过程中等待数据读入的时间，尤其在处理大规模数据时优势更为突出）。

命令执行成功后，在 output-prefix 的 fullpath 目录下生成 .bin 和 .idx 文件。

```
export WORK_DIR=/home/bml/storage/mnt/v-x2zt874luaxgmxbf/org/megatron_workspace/ #替换为个人工作目录
export MEGATRON_PATH=${WORK_DIR}/code/bml-megatron-llm/Megatron-LM/
export INPUT=${WORK_DIR}/data/wudao/validation.json
export TOKENIZER_MODEL_PATH=${WORK_DIR}/pretrained_models/tokenizer.model # 替换为huggingface模型的对应文件路径
export OUTPUT_PREFIX=${WORK_DIR}/data/wudao/validation
export WORKERS=64
export JSON_KEYS="content"
bash ${MEGATRON_PATH}/scripts/data_preprocess/preprocess_train_data.sh
```

脚本执行完成后，得到的数据如下。

```
${WORK_DIR}/data/wudao/
├──── README.md
├──── test.json
├──── validation.json
├──── validation_content_document.bin # 新增 ${output-prefix}_${json-keys}_document.bin
└──── validation_content_document.idx # 新增 ${output-prefix}_${json-keys}_document.idx
```

参数	描述
WORK_DIR	工作空间路径
MEGATRON_PATH	Megatron代码的基础路径（***/Megatron-LM）
INPUT	输入数据的json/jsonl文件路径，要求jsonl格式
OUTPUT_PREFIX	输出二进制训练文件前缀
WORKERS	处理数据的工作进程数，根据机器配置选择，多进程可以加快处理速度
JSON_KEYS	json文件中训练文本的字段名，若有多个，按空格拼接，比如“key0 key1 key2”，默认为“text”。本案例中训练文本在“content”下，因此设置：--json-keys content
TOKENIZER_MODEL_PATH	tokenizer.model文件路径

Step2:模型格式转换(HF->Megatron)

将 Huggingface 格式的模型文件转换为 Megatron 格式。

```
export WORK_DIR=/home/bml/storage/mnt/v-x2zt874luaxgmxbf/org/megatron_workspace/
export MEGATRON_PATH=${WORK_DIR}/code/bml-megatron-llm/Megatron-LM/
export HF_FORMAT_DIR=${WORK_DIR}/pretrained_models/
export TOKENIZER_MODEL=${WORK_DIR}/pretrained_models/
export MEGATRON_FORMAT_DIR=${WORK_DIR}/output/chinese-llama-2-7b_megatron_core_pp1_tp1/
export PIPELINE_MODEL_PARALLEL=1
export TENSOR_MODEL_PARALLEL=1
bash ${MEGATRON_PATH}/scripts/checkpoint_convert/convert_llama_hf2megatron.sh
```

参数	描述
HF_FORMAT_DIR	加载的HuggingFace模型的路径
TOKENIZER_MODEL	tokenizer.model文件所在的目录
MEGATRON_FORMAT_DIR	保存的Megatron模型checkpoint输出的路径
PIPELINE_MODEL_PARALLEL	流水切片数量，与训练保持一致。不同参数量下的切片数量不同，在转换模型时需进行针对性修改
TENSOR_MODEL_PARALLEL	张量切片数量，与训练保持一致。不同参数量下的切片数量不同，在转换模型时需进行针对性修改

导出的Megatron格式的checkpoint目录的文件结构如下：

```
${MEGATRON_FORMAT_DIR}
├──── iter_0000001
│     └──── mp_rank_00
│           └──── model_optim_rng.pt
└──── latest_checkpointed_iteration.txt
```

Step3:预训练(使用作业建模大模型案例实现)

大模型案例中，我们提供了预置的模型以及数据，可直接实现训练。用户可根据需要替换数据，经过Step1处理后，替换案例的数据集路径即可。对于模型文件，三个案例均已预置Megatron格式的权重，为了流程介绍的完整，因此我们提供了Step2的模型转换，建议用户直接使用预置案例里转换好的大模型即可。

预训练模型训练完后，保存为Megatron格式，需要使用step4将其转换为Huggingface格式，便于后续使用。

Step4:模型格式转换(Megatron->HF)

您可以将训练获得的 Megatron 格式的模型转换为 Huggingface 格式，具体操作步骤如下。后续您可以使用转换后的 Huggingface 格式的模型进行服务在线部署。

执行以下命令，可将训练生成的 Megatron 格式的模型转换为 Huggingface 格式的模型。

```
export WORK_DIR=/home/bml/storage/mnt/v-x2zt874luaxgmxbf/org/megatron_workspace/
export MEGATRON_PATH=${WORK_DIR}/code/bml-megatron-llm/Megatron-LM/
export MEGATRON_FORMAT_DIR=${WORK_DIR}/output/chinese-llama-2-7b_megatron_core_pp1_tp1/iter_0000001/
export HF_FORMAT_DIR=${WORK_DIR}/output/chinese-llama-2-7b_megatron_core_pp1_tp1_post_pretrain_hf/
export REF_HF_FORMAT_DIR=${WORK_DIR}/pretrained_models
export HF_PIPELINE_MODEL_PARALLEL=2
bash ${MEGATRON_PATH}/scripts/checkpoint_convert/convert_llama_megatron2hf.sh
```

部分参数介绍如下（其他参数设置参考“Step2:模型格式转换(HF->Megatron)”部分介绍）：

参数	描述
MEGATRON_FORMAT_DIR	加载的Megatron模型checkpoint的路径
HF_FORMAT_DIR	保存的HuggingFace模型checkpoint的路径
REF_HF_FORMAT_DIR	参考的Huggingface模型目录，用于复制原始config、tokenizer文件
HF_PIPELINE_MODEL_PARALLEL	保存的HuggingFace模型checkpoint的流水切片数量（因为Huggingface官方的7B权重被划分为2片，所以这里也设为2，保持一致）

转换后的Huggingface的checkpoint目录的文件结如下：

```
${HF_FORMAT_DIR}
├─── config.json
├─── generation_config.json
├─── pytorch_model-00001-of-00002.bin
├─── pytorch_model-00002-of-00002.bin
├─── pytorch_model.bin.index.json
├─── README.md
├─── special_tokens_map.json
├─── tokenizer_config.json
└─── tokenizer.model
```

Step5:快速导入并部署第三方模型

您可以按照以下步骤进行相关操作，可查看快速导入并部署第三方模型文档。

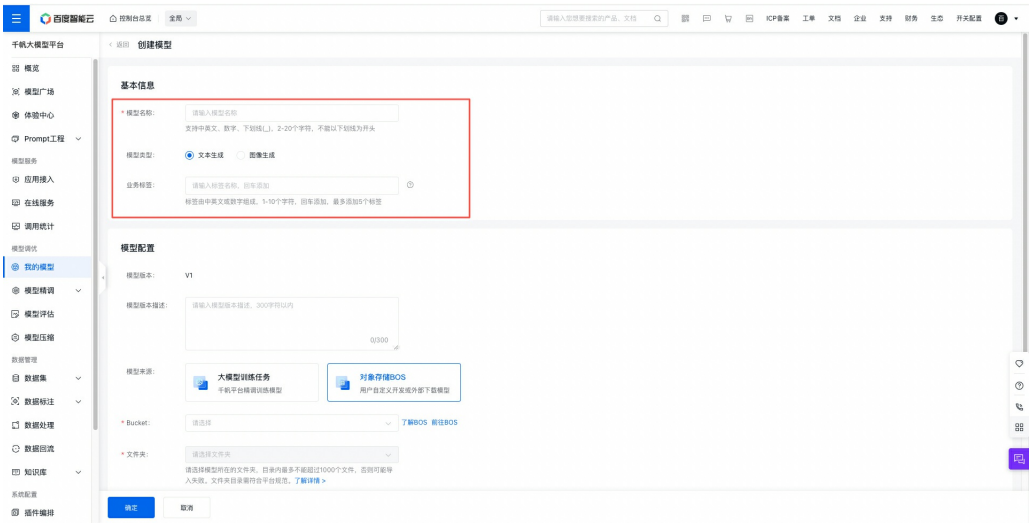
前提准备

需要您提前开通对象存储BOS服务。

导入模型

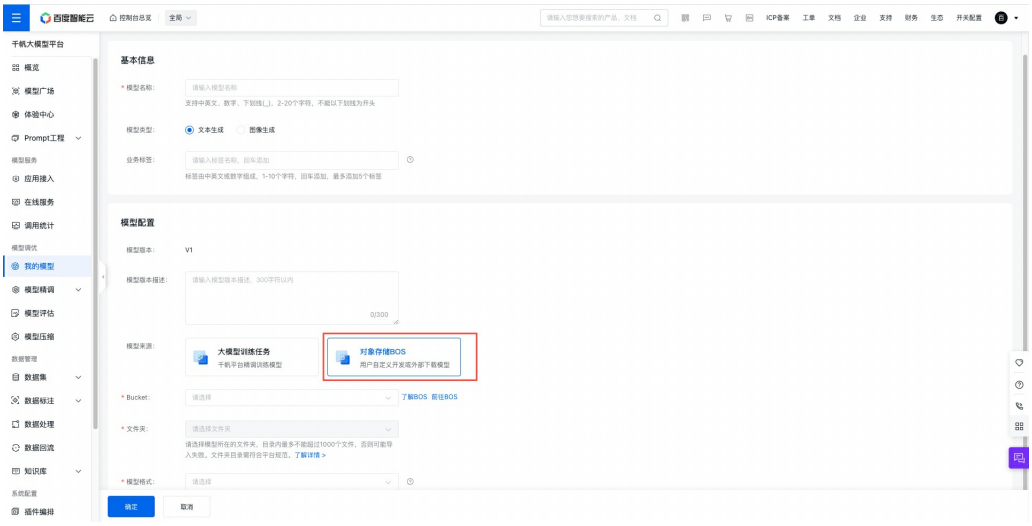
使用千帆大模型控制台，在左侧功能列模型管理中选择我的模型，进入创建模型的主任务界面。

1. 点击“创建模型”按钮，进行模型新建，填写以下基本信息：



2. 选择已完成的训练任务及运行，创建一个新的模型版本，具体字段包括：

- 模型版本：平台自动生成，版本号从V1起递增。
- 模型版本描述：非必填项，自定义的版本描述，记录模型的业务场景、使用方式等信息。
- 模型来源选择BOS，填写相应的Bucket和模型地址。模型需要提前上传到BOS平台，可以登录BOS控制台或者使用BOS工具上传。



导入HF系列模型

- 输入输出模式多轮对话chat模型选择【对话模式】，单轮问答模型选择【续写模式】
- 如果您选择对话模式还需配置历史对话和当前问题的prompt：平台将根据此配置自动拼接当前用户问题和历史轮次问答，方便用户在线测试或接入对话类应用。
- 两种模式都支持【高级配置】，默认示例如下：

```
{
  "load_model_class": "AutoModelForCausalLM",
  "load_tokenizer_class": "AutoTokenizer",
  "enable_auto_batch": true,
  "custom_end_str": "",
  "token_decode_return_blank": true,
  "tokenizer_special_tokens": {}
}
```

高级参数	说明
load_model_class	模型加载类，用于加载transformers模型；默认值 AutoModelForCausalLM
load_tokenizer_class	tokenizer加载类，用于加载模型；默认值 AutoTokenizer
enable_auto_batch	处理请求时，是否支持auto batch；是否开启auto batch推理，增加服务吞吐；若模型batch推理不兼容，开启后可能导致效果有误；默认值：true
custom_end_str	自定义生成结束字符串；防止模型不断生成；默认为空值，表示不设置
token_decode_return_blank	指定英文token解码后是否带有空格，设置为false时，平台会在英文单词间加入空格；默认值为true
tokenizer_special_tokens	选填，需要添加到tokenizer的特殊token；例如 {"pad_token": "[PAD]"}

部署服务 进入【我的模型】点击刚导入的模型进入【详情】页，模型【版本状态】变为“就绪”说明模型导入成功； 点击【部署】按钮进入【创建服务】。



按下图流程完善服务部署，将服务部署至私有资源池。





数据服务

数据服务简介

数据服务简介

在 BML 中，集成了百度大脑推出的智能数据服务平台 EasyData 作为其数据管理组件。EasyData 面向AI开发的一站式数据服务工具，其围绕 AI 开发过程中所需要数据采集、数据清洗、数据标注等任务需求提供完成的数据服务。

数据管理功能简介

在 BML 中，基于 EasyData 提供了丰富的数据管理能力，满足了不同类型数据在不同训练场景下的管理需求：

数据类型	场景	数据纳管	多人标注	智能标注
图像	图像分类（单标签、多标签）、物体检测、实例分割	支持	支持	部分支持
文本	文本分类（单标签、多标签）、短文本相似度、情感倾向分析、文本实体抽取	支持	支持	部分支持
表格	表格预测	支持	-	-

数据在线标注功能简介

在 BML 中，既支持用户直接上传已标注的数据并用于训练，也支持用户上传未标注数据，并在 BML 中完成数据标注操作。BML 支持的标注类型如下所示：

标注类型	说明	适用任务类型
单图多标签	一张图片可定义为多个标签	图像分类
矩形框标注	图片用矩形框标注需要识别的目标位置	物体检测
单文本单标签	一个文本定义为唯一标签	文本分类
短文本匹配	两个文本标注为相似或不相似	短文本匹配

数据智能标注功能简介

智能标注即自动标注，通过该功能可以大幅降低标注成本。在 BML 中支持图像和文本类数据的智能标注：

- 图像类：支持物体检测和实例分割两种类型。用户只需要标注少量数据（约占数据集的 30% 左右），即可通过智能标注任务自动完成剩余数据的标注。系统会自动判断自动标注时各图像标签的置信度，对于置信度较低的会作为难例，并由用户手工确认，从而保证整体标注的质量。
- 文本类：支持单标签文本分类。用户只需要提供少量已标注数据，即可通过智能标注任务完成剩余未标注数据的自动标注。自动标注的数据需要人工确认后，才能完成整个标注流程。

数据回流功能简介

BML 支持将模型部署为在线服务，通过数据回流功能可以将在线服务接收的请求数据进行保存。用户可以查看被保存数据的预测结果，以及将保存的数据添加到训练模型的数据集中并用于模型的迭代训练。

数据回流功能，完成了 AI 应用生命周期的闭环，通过初试数据完成模型训练和服务部署，利用新采集的线上数据不断更新模型。通过该方式可以有效的解决线上数据特征漂移的问题。

当前在 BML 中，可以在视觉类应用中使用数据回流功能。

数据集状态说明

- **导入中**：当线下大量数据批量导入至平台中时，需要等待几分钟-十几分钟时间，请耐心等待
- **导入失败**：最近一次数据导入失败可能有以下几种原因：
  -压缩包格式有误：如果导入格式以压缩包格式上传，请检查压缩包内的数据组织的层级结构是否符合要求，请修正后重新上传。
  -图片名称与对应标注 json 文件名称无法对应：请检查图片名称与 json 文件是否存在文件名称不一致的情况。
  -数据量超过上限：单个用户限制10万数据量，请检查是否超限，如需扩容请提交工单或加入官方QQ群(群号:868826008)联系群管进行扩容。
- **智能标注中**：状态为智能标注中的数据集无法删除。

管理视觉数据

图像分类数据

导入已标注数据

目录

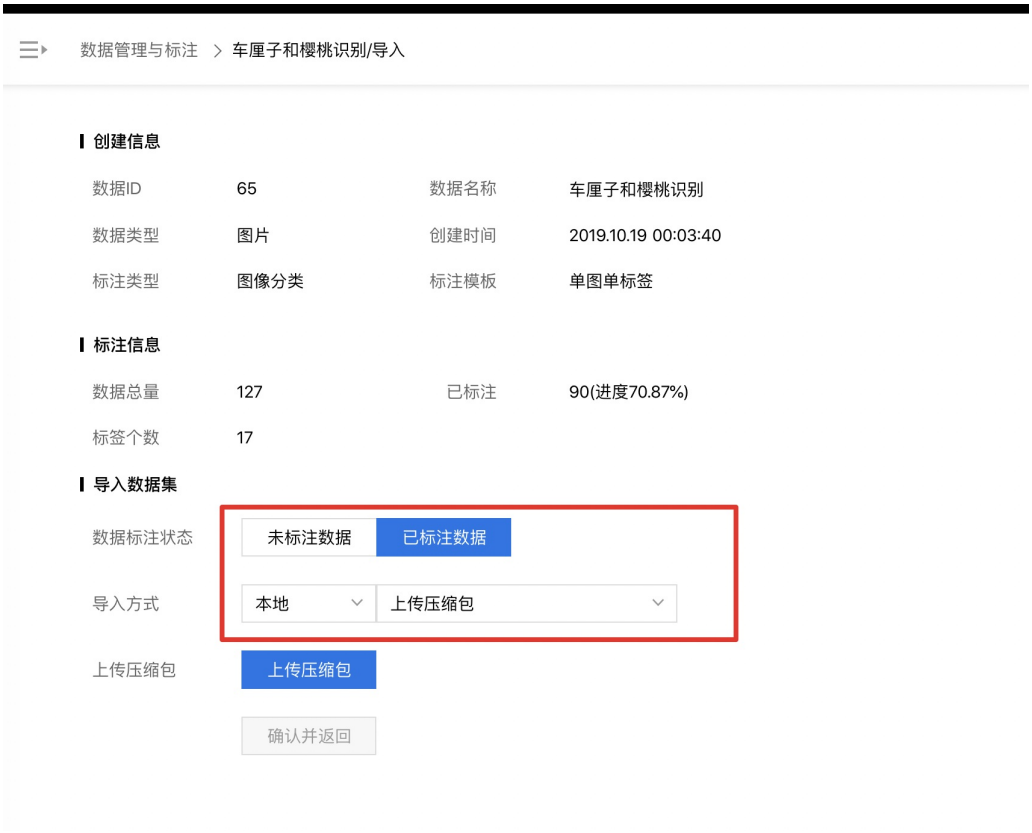
1. 导入已标注数据方式说明
2. 从本地导入已标注数据
3. 从已有数据集导入已标注数据
4. 查看已标注数据

导入已标注数据方式说明

如果您已有标注好的数据，可以在BML平台直接导入，方便直接进入后续训练环节。
向选定的数据集导入已标注好的数据有两种方式:
1、将单张图片和对应图片的标注信息以json格式上传
2、将BML其他数据集已经标注好的数据导入

从本地导入已标注数据

图像分类数据从本地导入已标注好的数据，支持整体以一个.zip格式的压缩包上传。**单次上传压缩包整体大小不超过5G**
注意，为了使平台能正确解析标注图片，请按照以下方式整理标注文件再打包上传

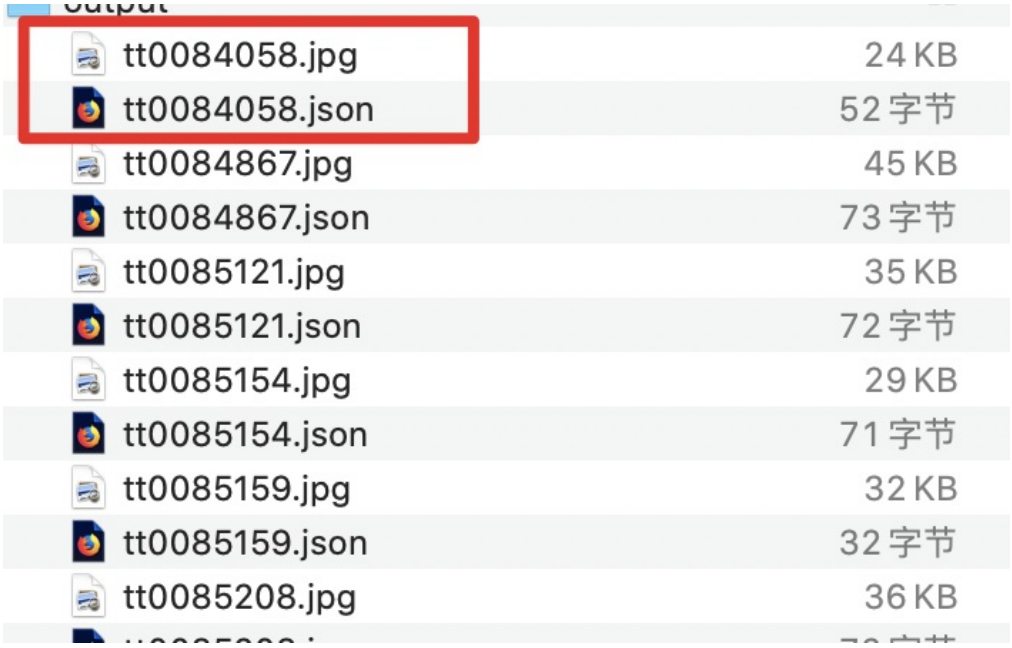


1、**图片及对应的标注文件一一对应，是文件名需要完全相同，否则会导入失败。**

- **图片格式要求**

1. 图片类型为jpg/png/ bmp/jpeg 。
2. 图片大小限制在4M内，长宽比在3：1以内，其中最长边需要小于4096px，最短边需要大于30px。
3. 每个账户图片数据量限制为20w张，超出后可被忽略。如需扩容，请提交工单申请

- **特别注意：**
- 对同一数据集存在多个内容完全一致的图片，将会做去重处理。
- 为保证模型训练效果，**所上传的图片应与实际业务场景的图片（光线、角度、采集设备）尽可能一致**



2、**json文件内容格式示例如下：**

单图单标签

```
{
  "labels":
  [
    {"name": "Drama"}
  ]
}
```

单图多标签

```
{
  "labels":
  [
    {"name": "Drama"},
    {"name": "Romance"}
  ]
}
```

从已有数据集导入已标注数据

从已有数据集导入已标注数据，支持选择数据集及导入的具体标签进行导入。



查看已标注数据

导入数据完成后，可以在数据集查看-已标注查看到分标签的图片列表

导入未标注数据

目录

1. 导入未标注数据方式说明
2. 从本地上传图片导入
3. 从本地上传压缩包导入
4. 导入线上已有数据集

导入未标注数据方式说明

为选定的图像分类数据集导入未标注数据（即原始图片）的方式有两种。一种为从本地选择图片文件；一种为其余同为图片类型数据集中导入。

从本地上传图片导入

选择**未标注数据-本地-上传图片**，点击上传图片后选择本地图片文件进行导入。

数据格式要求

1、图片类型为jpg/png/bmp/jpeg ，单次上传限制100个文件。

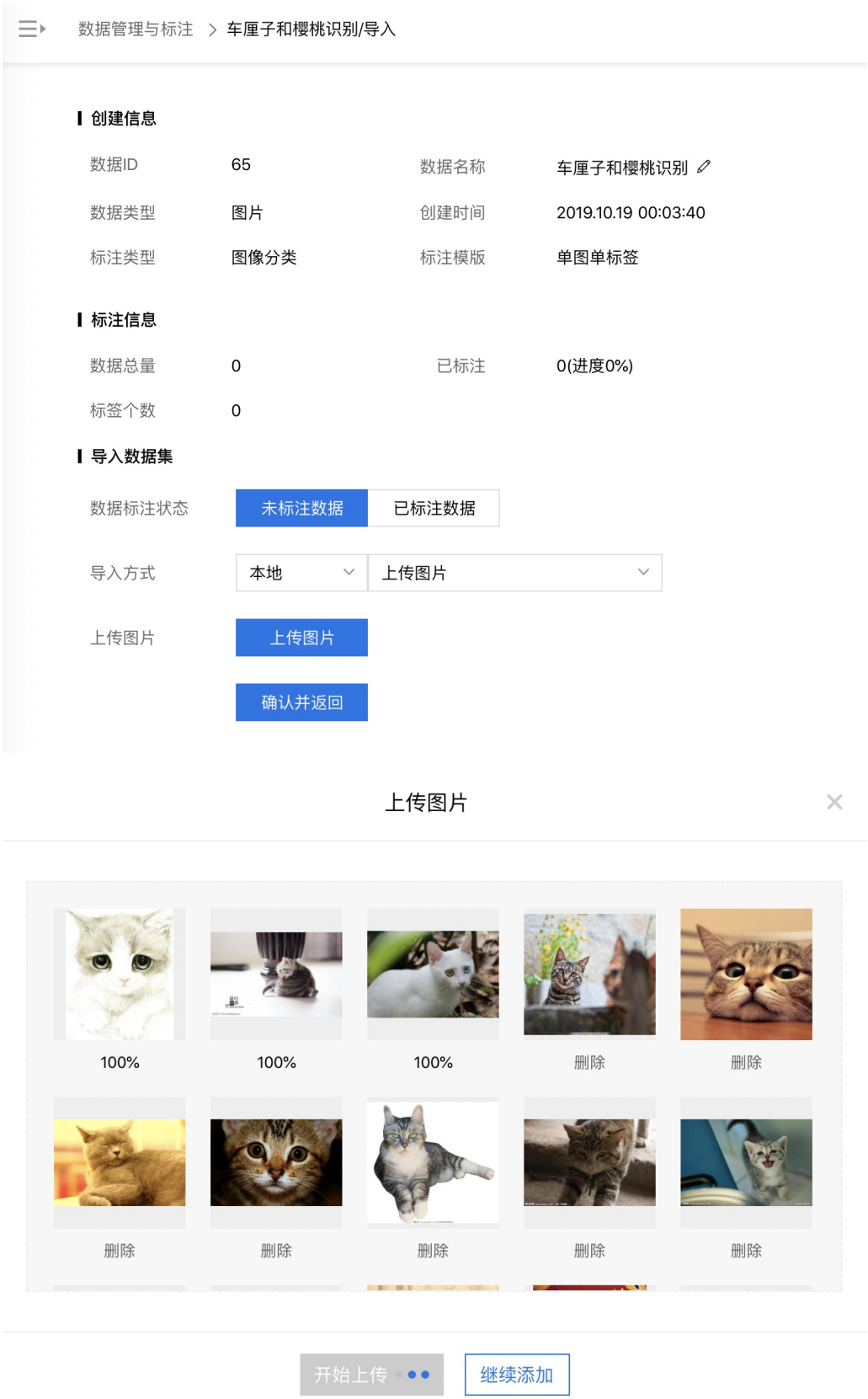
2、图片大小限制在4M内，长宽比在3：1以内，其中最长边需要小于4096px，最短边需要大于30px。

3、单个数据集大小限制为10万图片，超出后可被忽略。

特别注意：

—对同一数据集存在多个**内容完全一致的图片，将会做去重处理**。

—为保证模型训练效果，**所上传的图片应与实际业务场景的图片（光线、角度、采集设备）尽可能一致**



从本地上传压缩包导入

选择**未标注数据-本地-上传压缩包**，点击上传压缩包后选择本地图片压缩包完成数据的批量导入。

数据格式要求

1、压缩包仅支持**zip格式**，大小限制5GB以内

2、压缩包内图片格式要求为：图片类型为jpg/png/bmp/jpeg，图片大小限制在4M内，长宽比在3：1以内，其中最长边需要小于4096px，最短边需要大于30px。

3、每个账户图片数据量限制为20w张，超出后可被忽略。如需扩容，请提交工单申请

特别注意：

—对同一数据集存在多个**内容完全一致的图片，将会做去重处理**。

—为保证模型训练效果，**所上传的图片应与实际业务场景的图片（光线、角度、采集设备）尽可能一致**



导入线上已有数据

选择**未标注数据-已有数据集-导入已有数据集**，支持将已经上传的其它图片类数据集进行导入。在视觉任务中往往会有存在更换模型类型的可能性，比如某个业务场景问题既可以用图像分类解决又可以用物体检测解决，通过这个功能可以有效地快速切换数据类型，分别测试效果。

由于这里选择上传的是未标注数据，那么在导入线上已有数据的情况下，可以选择导入同为图片类型数据集的全部数据但去掉标注；或者选择未被标注的部分。



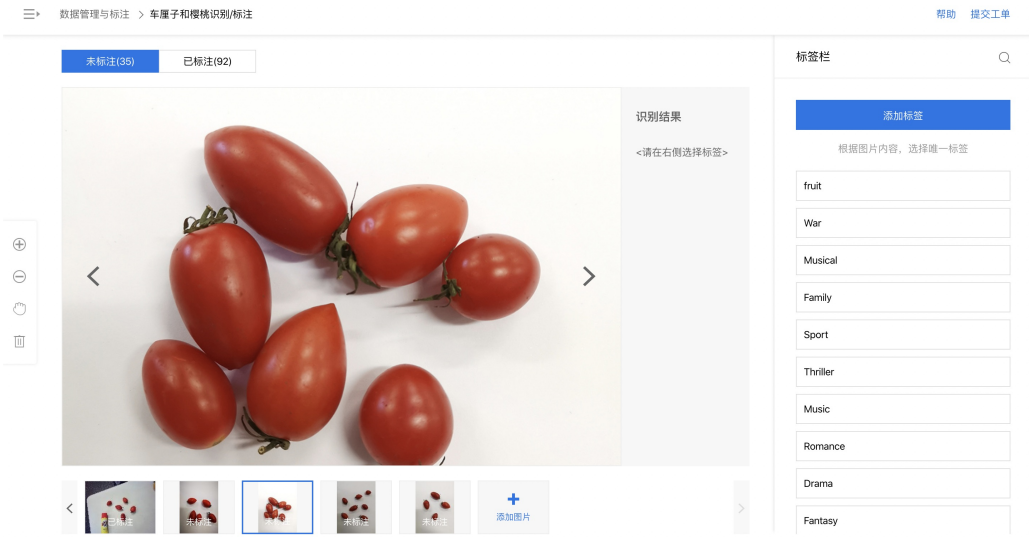
数据标注说明

目录

1. 标注操作说明
2. 标注技巧说明
3. 上传图片

标注操作说明

在数据集查看详情页面可以点击开始标注，进入到标注页面。



添加标签

点击右侧添加标签，输入不同分类名称，完成标签添加。

标签格式说明：目前平台标签名称只支持数据/字母/下划线，暂时不支持中文



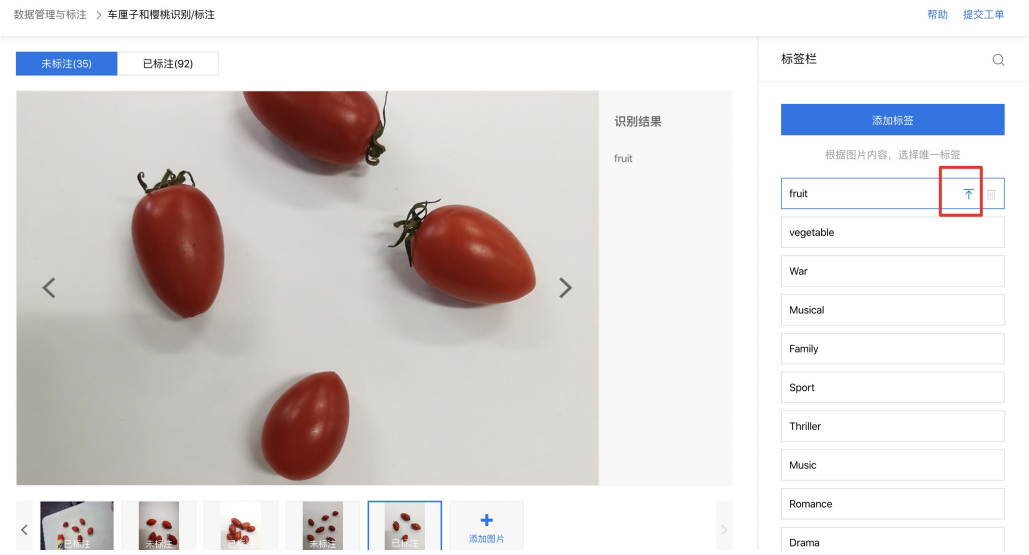
标注方式

根据左侧选定的当前图片，在右侧标签列表选择标签即可完成标注。**点击下一张图片/切换图片即可完成保存**

标注技巧说明

快速标注相同标签下的多张图片

如果一批图片的标签相同，建议使用右侧**标签置顶**的功能，把当前希望快速标注的标签置顶，方便快速选择标签。



上传更多图片

图像分类任务要提高模型效果，尽可能覆盖某一种标签可能有的各种情况，比如**光照不同、拍摄设备不同、背景环境不同、角度不同等等**，同时要注意**不同类别下的已标注数据量尽可能均衡**。
上传更多图片的操作说明，详见图像分类导入图片操作介绍。

物体检测数据

导入已标注数据

目录

1. 导入已标注数据方式说明
2. 从本地导入已标注数据
3. 从已有数据集导入已标注数据
4. 查看已标注数据

导入已标注数据方式说明

如果您已有标注好的数据，支持快速导入到BML，方便直接进入后续训练环节。
向选定的数据集导入已标注好的数据有两种方式:

1、将单张图片和对应图片的标注信息以json/xml格式上传

2、将BML中其他数据集已经标注好的数据导入

从本地导入已标注数据

物体检测数据从本地导入已标注好的数据，支持整体以一个.zip格式的压缩包上传。**单次上传压缩包整体大小不超过5G**

注意，为了使平台能正确解析标注图片，请按照以下方式整理标注文件再打包上传



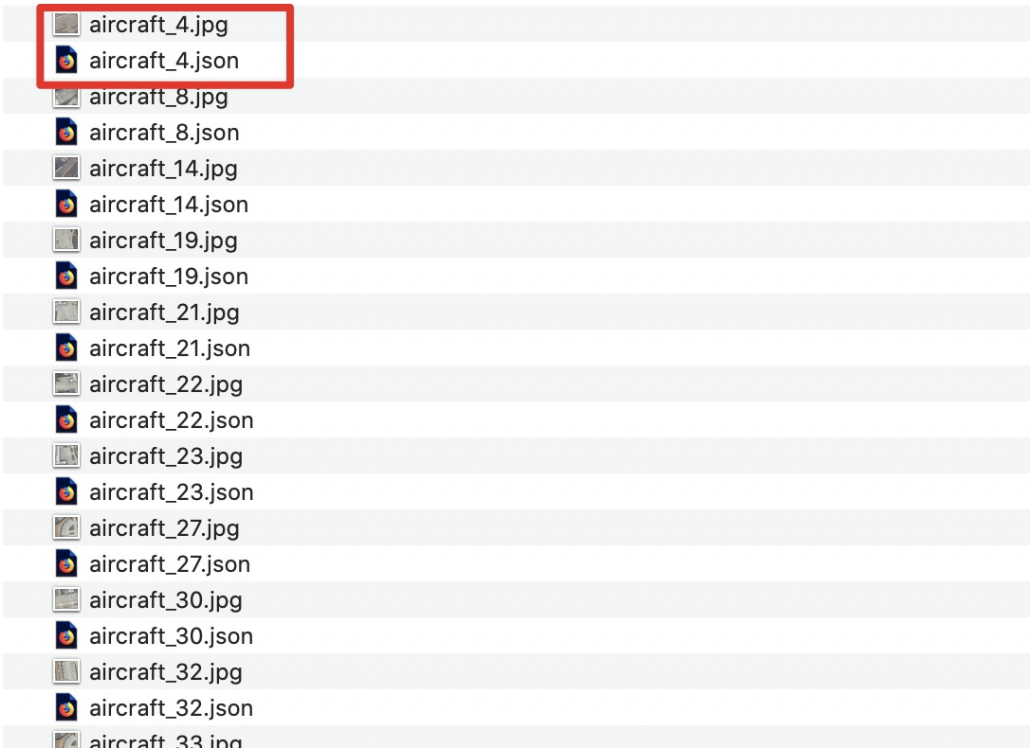
1、**图片及对应的标注文件一一对应，是文件名需要完全相同，否则会导入失败**。

- **图片格式要求**

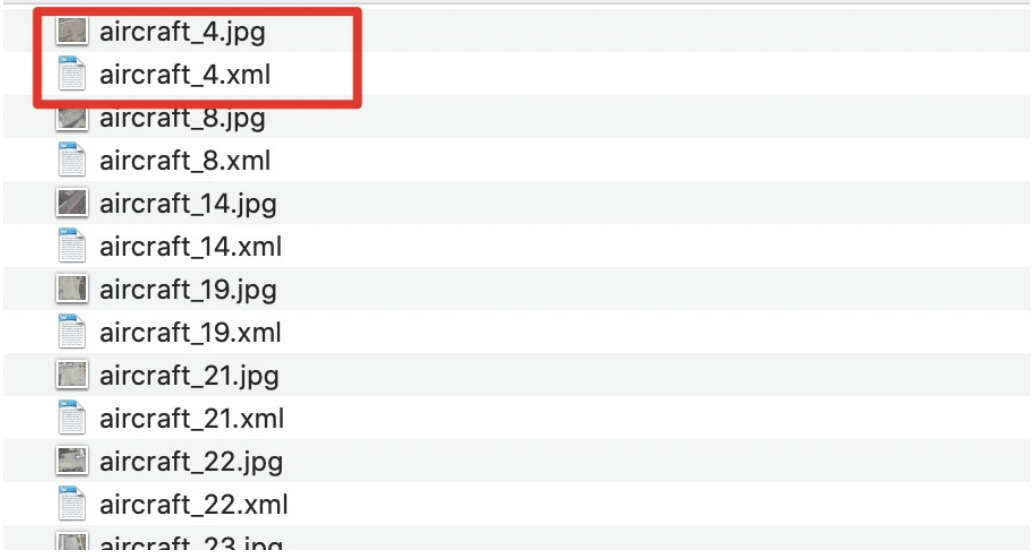
1. 图片类型为jpg/png/ bmp/jpeg 。
2. 图片大小限制在4M内，长宽比在3：1以内，其中最长边需要小于4096px，最短边需要大于30px。
3. 每个账户图片数据量限制为20w张，超出后可被忽略。如需扩容，请提交工单申请

- **特别注意：**
- 对同一数据集存在多个内容完全一致的图片，将会做去重处理。
- 为保证模型训练效果，**所上传的图片应与实际业务场景的图片（光线、角度、采集设备）尽可能一致。**
- **单张图片中的标注框须限制在500个之内。**如有特殊需求，请通过提交工单咨询。

json标注文件



xml标注文件



2、json文件内容格式示例如下：

矩形框标注

```
{

"labels":[
{
    "name":"car",
    "x1":278.7673,
    "y1":38.76027,
    "x2":328.51887,
    "y2":101.77642
},
    {
 "name":"car",
 "x1":333.0601,
 "y1":56.9776,
 "x2":378.78652,
 "y2":126.21715
}
]
}
```

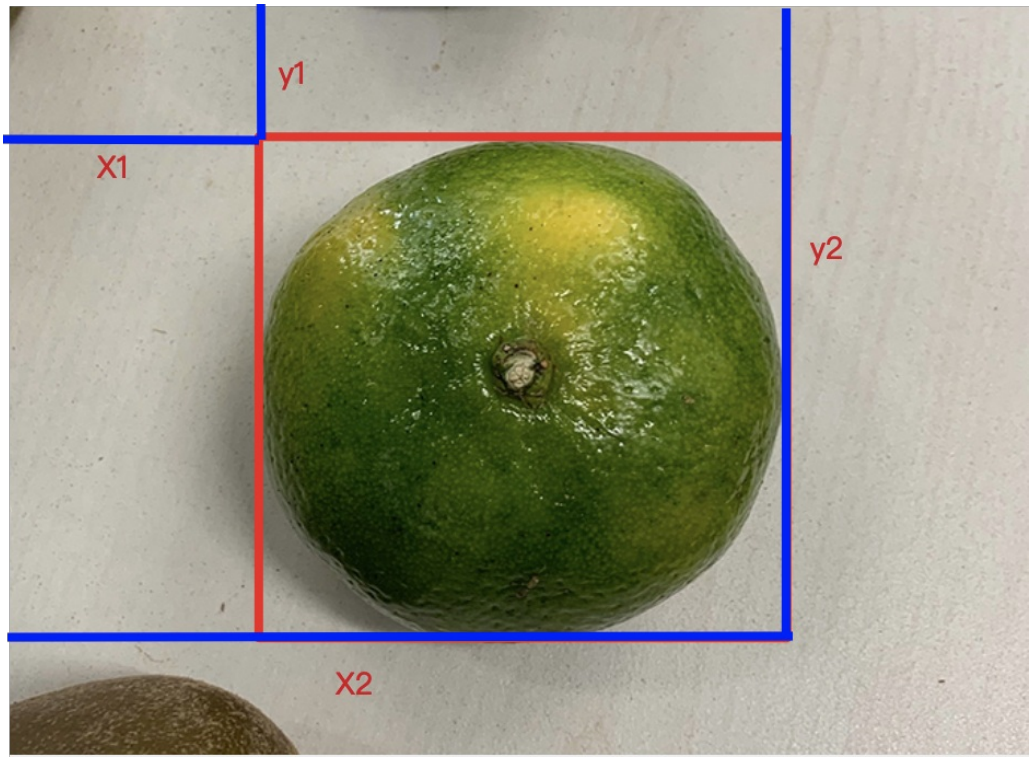
其中x1、y1、x2、y2含义如下

x1：目标定位位置的长方形左上顶点的水平坐标

y1：目标定位位置的长方形左上顶点的垂直坐标

x2：目标定位位置的长方形右下顶点的水平坐标

y2：目标定位位置的长方形右下顶点的垂直坐标



3、xml文件内容格式示例如下：

矩形框标注

```
<object>
    <bndbox>
        <xmin>937</xmin>
        <ymin>85</ymin>
        <ymax>160</ymax>
        <xmax>1033</xmax>
    </bndbox>
    <difficult>0</difficult>
    <pose>Left</pose>
    <name>apple</name>
    <truncated>1</truncated>
</object>
```

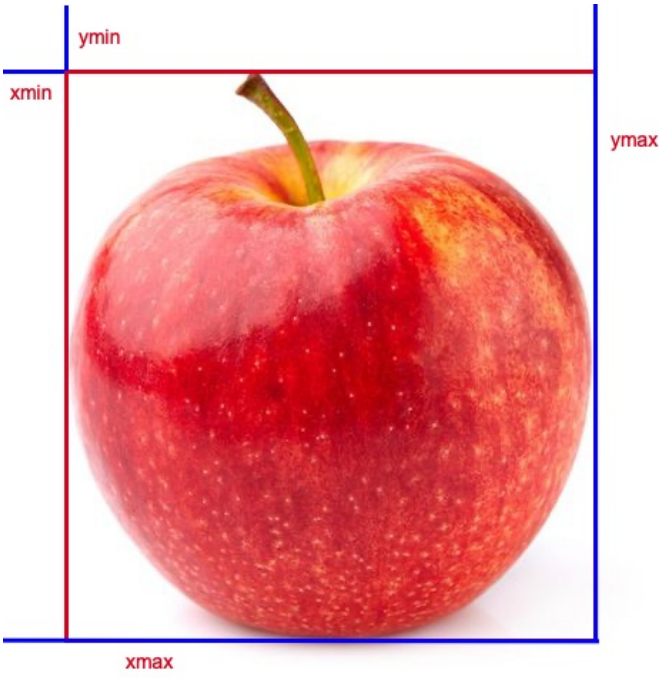
其中xmin、ymin、xmax、ymax含义如下

xmin：目标定位位置的长方形左上顶点的水平坐标

ymin：目标定位位置的长方形左上顶点的垂直坐标

xmax：目标定位位置的长方形右下顶点的水平坐标

ymax：目标定位位置的长方形右下顶点的垂直坐标



从已有数据集导入已标注数据

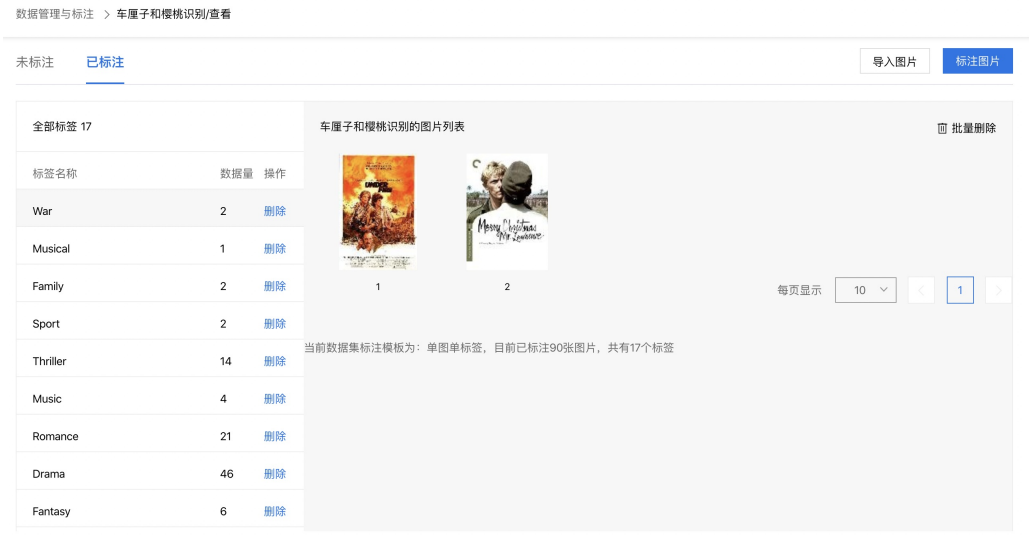
从已有数据集导入已标注数据，支持选择数据集及导入的具体标签进行导入。

导入数据集



查看已标注数据

导入数据完成后，可以在数据集查看-已标注重查看到分标签的图片列表



导入未标注数据

目录

1. 导入未标注数据方式说明
2. 从本地上传图片导入
3. 从本地上传压缩包导入
4. 导入线上已有数据集

导入未标注数据方式说明

为选定的图像分类数据集导入未标注数据（即原始图片）的方式有两种。一种为从本地选择图片文件；一种为从线上【专业版-数据服务】中其余同为图片类型数据集中导入。

从本地上传图片导入

选择**未标注数据-本地-上传图片**，点击上传图片后选择本地图片文件进行导入。

数据格式要求

1、图片类型为jpg/png/ bmp/jpeg ，单次上传限制100个文件。

2、图片大小限制在4M内，长宽比在3：1以内，其中最长边需要小于4096px，最短边需要大于30px。

3、每个账户图片数据量限制为20w张，超出后可被忽略。如需扩容，请提交工单申请

特别注意：

—对同一数据集存在多个**内容完全一致的图片，将会做去重处理**。

—为保证模型训练效果，**所上传的图片应与实际业务场景的图片（光线、角度、采集设备）尽可能一致**

—**单张图片中的标注框须限制在500个之内**。如有特殊需求，请通过提交工单咨询。





从本地上传压缩包导入

选择**未标注数据-本地-上传压缩包**，点击上传压缩包后选择本地图片压缩包完成数据的批量导入。

数据格式要求

1、压缩包仅支持**zip格式**，大小限制5GB以内

2、压缩包内图片格式要求为：图片类型为jpg/png/bmp/jpeg，图片大小限制在4M内，长宽比在3：1以内，其中最长边需要小于4096px，最短边需要大于30px。

3、单个数据集大小限制为10万图片，超出后可被忽略。

特别注意：

—对同一数据集存在多个**内容完全一致的图片，将会做去重处理**。

—为保证模型训练效果，**所上传的图片应与实际业务场景的图片（光线、角度、采集设备）尽可能一致。**

—**单张图片中的标注框须限制在500个之内。**如有特殊需求，请通过提交工单咨询。



导入线上已有数据

选择**未标注数据-已有数据集-导入已有数据集**，支持将已经上传在专业版同为图片类的其他数据集进行导入。在视觉任务中往往会有存在更换模型类型的可能性，比如某个业务场景问题既可以用图像分类解决又可以用物体检测解决，通过这个功能可以有效的快速切换数据类型，分别测试效果。

由于这里选择上传的是未标注数据，那么在导入线上已有数据的情况下，可以选择导入同为图片类型数据集的全部数据但去掉标注；或者选择未被标注的部分。



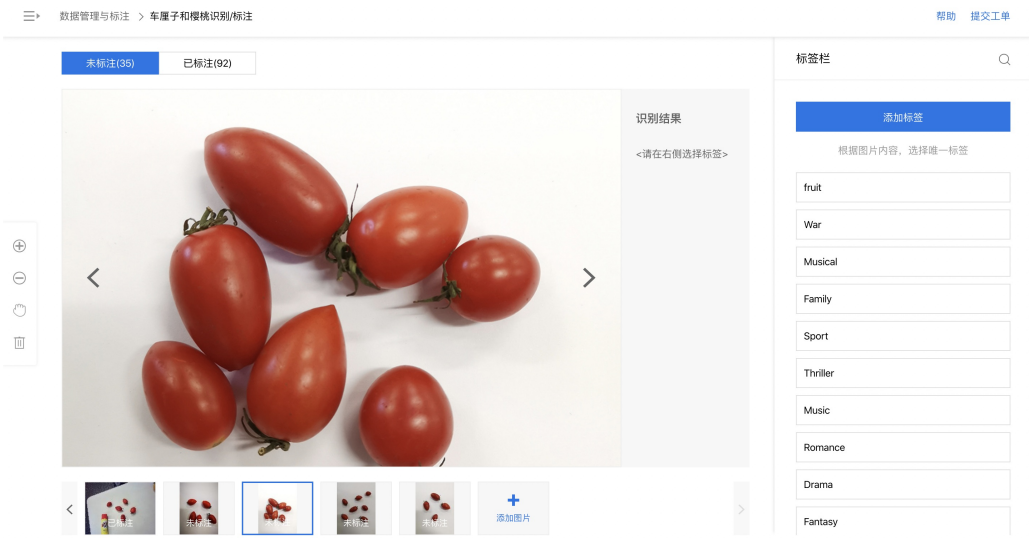
数据标注说明

目录

1. 标注操作说明
2. 标注技巧说明
3. 上传图片

标注操作说明

在数据集查看详情页面可以点击开始标注，进入到标注页面。



添加标签

点击右侧添加标签，输入不同分类名称，完成标签添加。

标签格式说明：目前平台标签名称只支持数字/字母/下划线，暂时不支持中文



标注方式

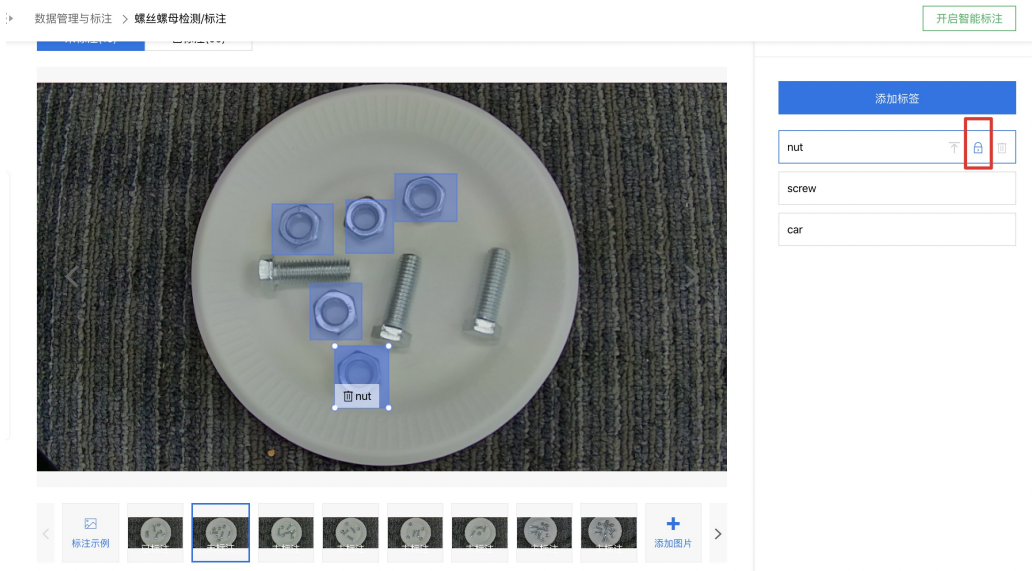
根据左侧选定的当前图片，在左侧选择标注工具针对目标区域构建矩形框。**点击下一张图片/切换图片即可完成保存**

注意，在标注过程中请不要标注太多背景信息，同时需要将图中的每个目标都进行标注

标注技巧说明

快速标注相同标签下的多张图片

如果一批图片的标签相同，建议使用右侧**标签置顶**的功能，把当前希望快速标注的标签置顶，同时点击锁定icon，这样在标注过程中只需在左侧画框即可。



上传更多图片

物体检测模型要提高模型效果，尽可能覆盖某一种标签可能有的各种情况，比如**光照不同、拍摄设备不同、背景环境不同、角度不同等等**，同时在注意要**不同类别下的已标注数据量尽可能均衡**。
上传更多图片的操作说明，详见图像分类导入图片操作介绍。

实例分割数据

导入已标注数据

目录

1. 导入已标注数据方式说明
2. 从已有数据集导入已标注数据
3. 查看已标注数据

导入已标注数据方式说明

如果您已有标注好的数据，支持快速导入到BML，方便直接进入后续训练环节。
实例分割任务向选定的数据集导入已标注好的数据目前支持一种方式:

将其他数据集已经标注好的数据导入

从已有数据集导入已标注数据

从已有数据集导入已标注数据，支持选择数据集及导入的具体标签进行导入。



查看已标注数据

导入数据完成后，可以在数据集查看-已标注查看到分标签的图片列表



导入未标注数据

目录

1. 导入未标注数据方式说明
2. 从本地上传图片导入
3. 从本地上传压缩包导入
4. 导入线上已有数据集

导入未标注数据方式说明

为选定的实例分割数据集导入未标注数据（即原始图片）的方式有两种。一种为从本地选择图片文件；一种为从平台中其余同为图片类型数据集中导入。

从本地上传图片导入

选择**无标注数据-本地-上传图片**，点击上传图片后选择本地图片文件进行导入。

数据格式要求

1、图片类型为jpg/png/ bmp/jpeg ，单次上传限制100个文件。

2、图片大小限制在4M内，长宽比在3：1以内，其中最长边需要小于4096px，最短边需要大于30px。

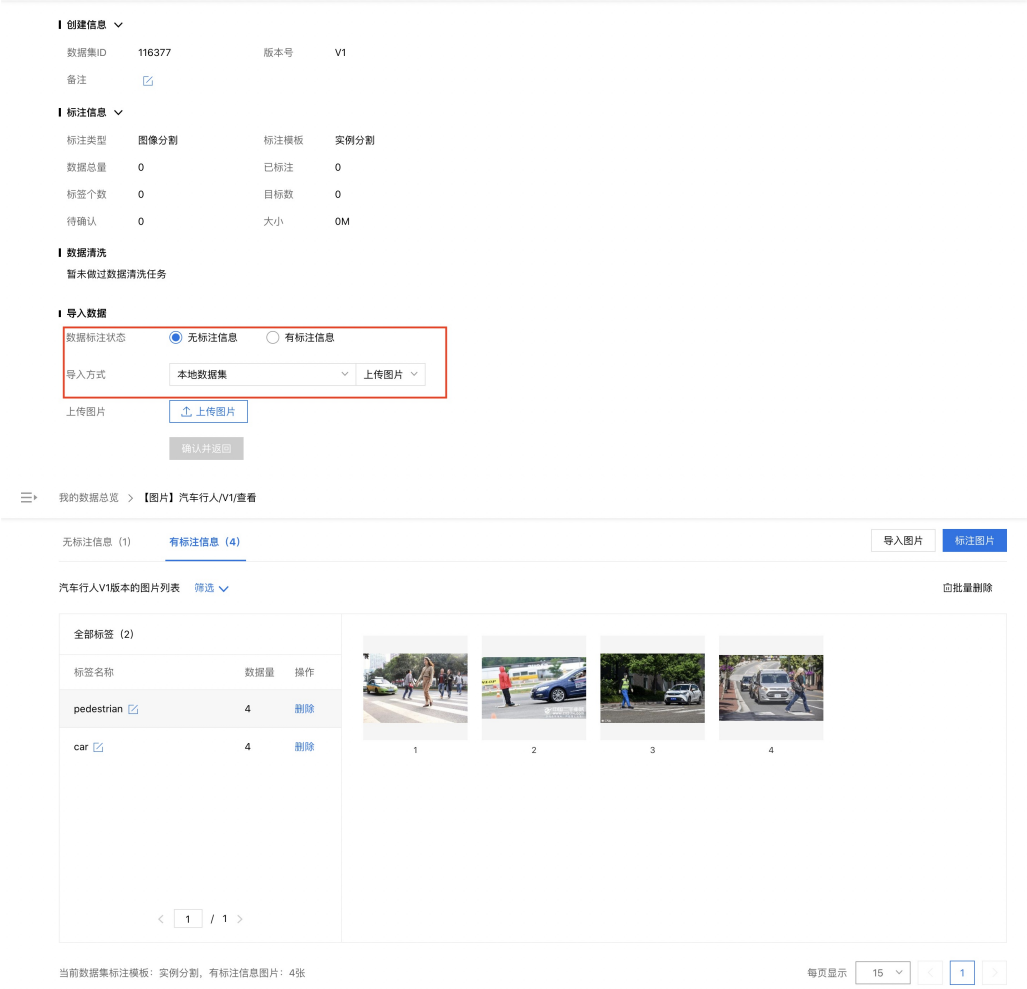
3、每个账户图片数据量限制为20w张，超出后可被忽略。如需扩容，请提交工单申请

特别注意：

—对同一数据集存在多个**内容完全一致的图片，将会做去重处理**。

—为保证模型训练效果，**所上传的图片应与实际业务场景的图片（光线、角度、采集设备）尽可能一致**

—**单张图片中的标注框须限制在500个之内。**如有特殊需求，请通过提交工单咨询。



从本地上传压缩包导入

选择**无标注数据-本地-上传压缩包**，点击上传压缩包后选择本地图片压缩包完成数据的批量导入。

数据格式要求

1、压缩包仅支持**zip格式**，大小限制5GB以内

2、压缩包内图片格式要求为：图片类型为jpg/png/bmp/jpeg，图片大小限制在4M内，长宽比在3：1以内，其中最长边需要小于4096px，最短边需要大于30px。

3、单个数据集大小限制为10万图片，超出后可被忽略。

特别注意：

—对同一数据集存在多个**内容完全一致的图片，将会做去重处理**。

—为保证模型训练效果，**所上传的图片应与实际业务场景的图片（光线、角度、采集设备）尽可能一致。**

—**单张图片中的标注框须限制在500个之内。**如有特殊需求，请通过提交工单咨询。

导入线上已有数据

选择**无标注数据-线上已有数据集-导入已有数据集**，支持从已经上传的其他同类型数据集进行导入。在视觉任务中往往会有存在更换模型类型的可能性，比如某个业务场景问题既可以用物体检测解决又可以用实例分割解决，通过这个功能可以有效的快速切换数据类型，分别测试效果。

由于这里选择上传的是未标注数据，那么在导入线上已有数据的情况下，可以选择导入同为图片类型数据集的全部数据但去掉标注；或者选择未被标注的部分。



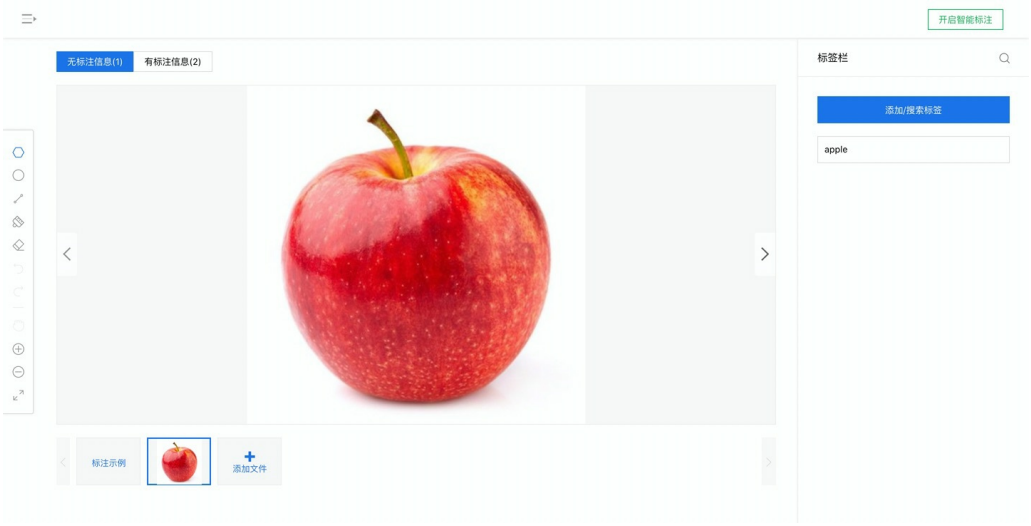
数据标注说明

目录

1. 标注操作说明
2. 标注技巧说明
3. 上传图片

标注操作说明

在数据集查看详情页面可以点击开始标注，进入到标注页面。



添加标签

点击右侧添加标签，输入不同分类名称，完成标签添加。

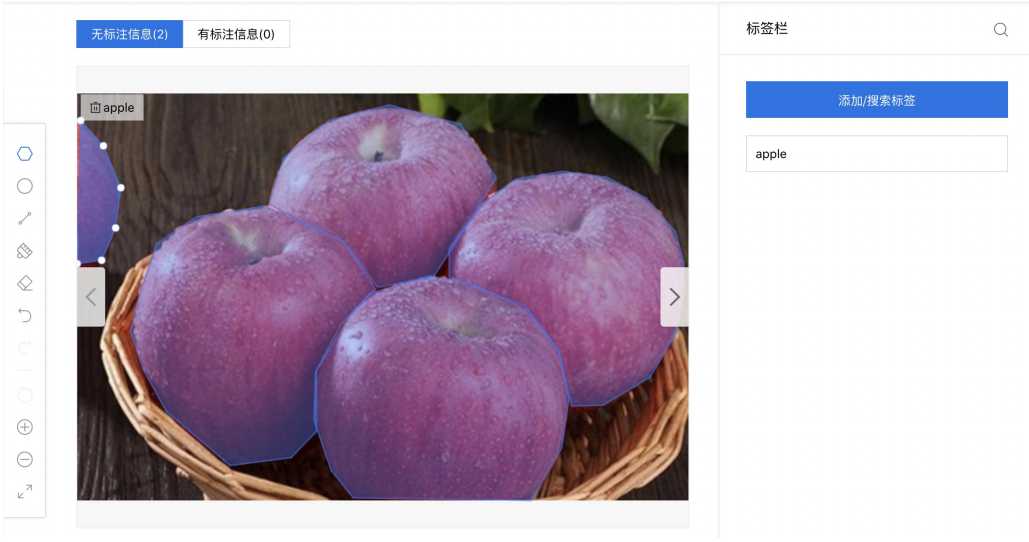
标签格式说明：目前平台标签名称只支持数字/字母/下划线，暂时不支持中文



标注方式

根据下侧选定的当前图片，在左侧选择合适的标注工具标注目标物体。**点击下一张图片/切换图片即可完成保存**

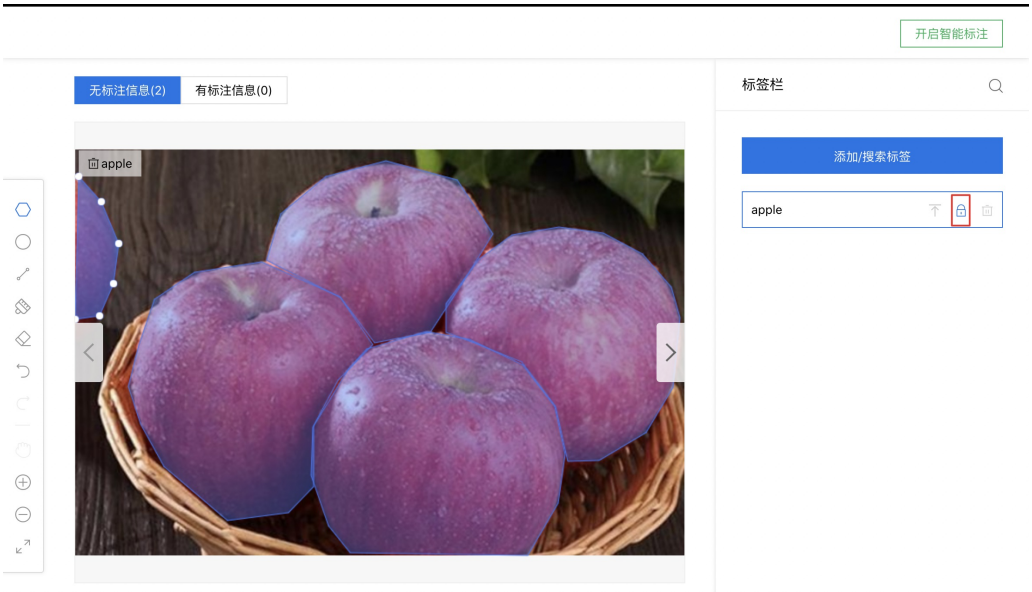
注意，在标注过程中请不要标注太多背景信息，同时需要将图中的每个目标都进行标注



标注技巧说明

快速标注相同标签下的多张图片

如果一批图片的标签相同，建议使用右侧**标签置顶**的功能，把当前希望快速标注的标签置顶，同时点击锁定icon，这样在标注过程中只需在左侧画框即可。



上传更多图片

物体检测模型要提高模型效果，尽可能覆盖某一种标签可能有的各种情况，比如**光照不同、拍摄设备不同、背景环境不同、角度不同等等**，同时在注意要**不同类别下的已标注数据量尽可能均衡**。
上传更多图片的操作说明，详见实例分割导入图片操作介绍。

管理文本数据

文本分类数据

导入与标注

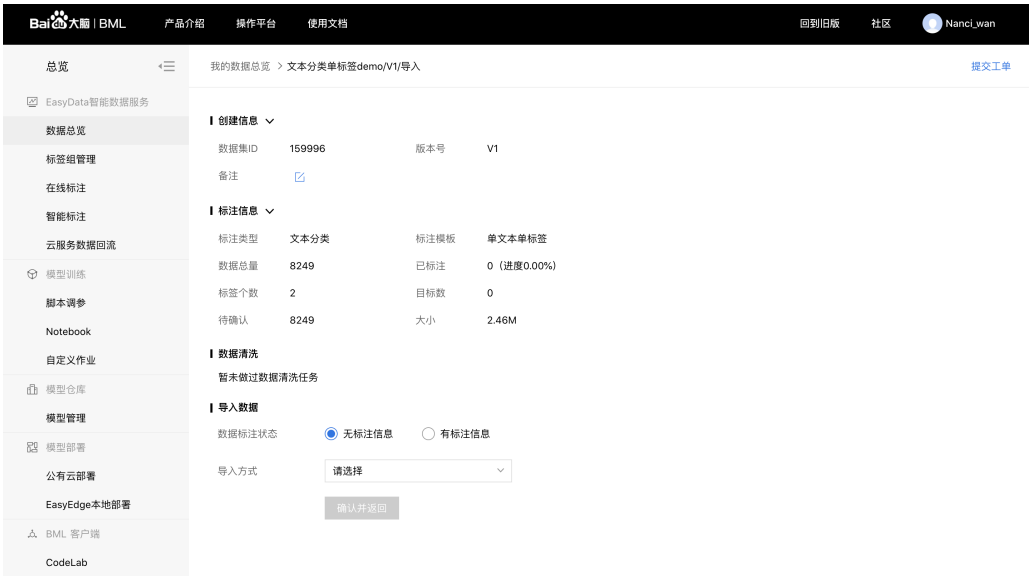
1. 创建数据集

您可以在左侧目录导航栏中选择【数据总览】并点击按钮“创建数据集”，对话框中选择数据类型为“文本”，标注类型选择“文本分类”，当前的文本分类的数据标注模板默认为单文本单标签格式。



2.导入未标注文本数据

进入到新创建的文本分类数据集中，如果您手中的数据是未标注数据，可以导入“未标注”中。平台提供两种方式导入数据，一种是从本地上传，一种是已有数据集导入



通过已有数据集导入时，平台将为您筛选出类型为文本分类的数据集。

本地上传，包括三种上传方式，分别为「上传文本」、「上传压缩包」、「API上传」

通过文本上传时，需注意：

1. 文本文件内数据格式要求为"文本内容\n"（即每行一个未标注样本，使用回车换行），每一行表示一组数据，每组数据的字符数建议不超过512个，超出将被截断
2. 文本文件类型支持txt，编码仅支持UTF-8，单次上传限制100个文本文件

相关的数据样例可在上传窗口内下载数据样例查看

通过压缩包上传时，需注意：

1. 压缩包内的一个文本文件将作为一个样本上传
2. 压缩包格式为.zip格式，压缩包内文件类型支持txt，编码仅支持UTF-8

相关的数据样例可在上传窗口内下载数据样例查看

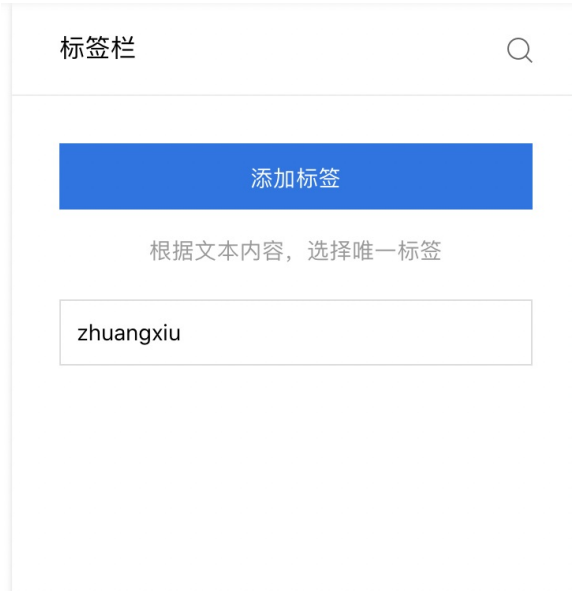
API导入时，可参考文档

文本标注

1. 上传未标注文本后，进入到标注页面，您可以逐一查看每一篇未标注的文本，如下图：



2. 设定分类对应的标签



3. 创建标签后，即可以进行对文本的标注，在文本的上方标签字段处，会显示对应的分类标签，完成标注



3.导入已标注文本数据

如果您手中的数据为已经标注好的数据，可以导入到“已标注”中。平台提供两种方式导入数据，一种是从本地上传，一种是从线上已有数据集导入。

通过已有数据集导入时，平台将为您筛选出类型为文本分类的数据集。

本地上传，包括三种上传方式，分别为「上传文本」、「上传压缩包」、「API上传」

通过文本上传时，需注意：

1. 文本分类的标注数据格式要求为"文本内容\t标注标签\n"（即每行一个未标注样本与一个标注标签，中间使用tab键间隔，每组数据使用回车换行），详见数据样例每一行表示一组数据，每组数据字符数不超过512个，超出将被截断
2. 支持文本文件类型为txt，编码仅支持UTF-8，单次上传限制100个文本文件

相关的数据样例可在上传窗口内下载数据样例查看

通过压缩包上传时，需注意：

1. 上传已标注文件要求格式为zip格式压缩包，同时压缩包大小在5GB以内，文件编码仅支持UTF-8

2. 压缩包内仅支持单个文本文件（txt）及同名的json格式标注文件的上传，可传多组样本，详细请见示例压缩包

相关的数据样例可在上传窗口内下载数据样例查看



API导入时，可参考文档进行上传

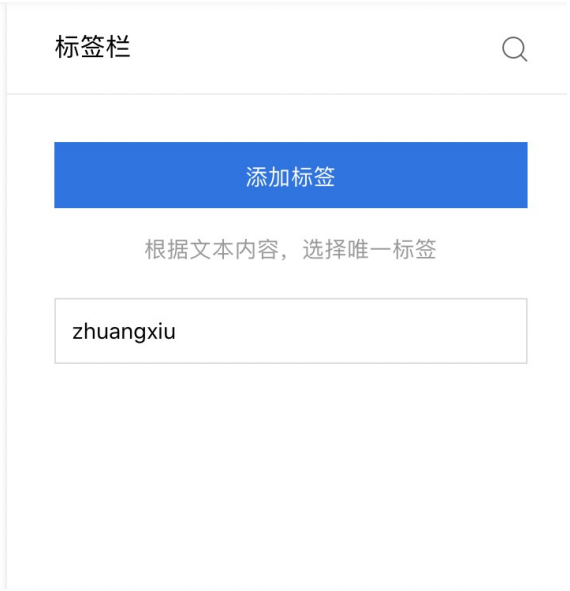
数据标注说明

文本标注

1. 上传未标注文本后，进入到标注页面，您可以逐一查看每一篇未标注的文本，如下图：



2. 设定分类对应的标签



3. 创建标签后，即可以进行对文本的标注，在文本的上方标签字段处，会显示对应的分类标签，完成标注



数据去重策略

重复样本的定义

一个样本包括文本内容和标签。重复样本的定义，是指您上传的数据中，存在两个样本的文本内容完全一致。则被判定为两个样本是重复样本。例如：

文本分类-单标签数据重复样本示例：

文本内容	标签
今天北京的空气不错	weather
今天北京的空气不错	weather
今天北京的空气不错	local

文本分类-多标签数据重复样本示例：

文本内容	标签
未来的学和教正在改变，学生将会在家里学习，机器人将走上讲台。	education/science
未来的学和教正在改变，学生将会在家里学习，机器人将走上讲台。	education/science
未来的学和教正在改变，学生将会在家里学习，机器人将走上讲台。	AI/robot

上述两个表中，都代表三个样本均为重复样本，后两个样本虽然标签不一，但文本内容一致，也为重复样本。

Tips：“如何利用好重复样本”，如果您在模型训练过程中，需要通过增加某个类别标签的预测权重，可以通过增加此标签的重复样本来达到此目标。

平台去重策略

平台提供了可去重的数据集，即对您上传的数据进行重复样本的去重。注意：当您确定了数据集为去重或非去重的属性后，便不可修改。

当您创建了一个去重的数据集时，在后续上传数据的过程中，平台可通过检验您当前上传的样本与已上传到此数据集下的样本是否相同，如果相同，则会使用新的样本替代旧的样本。此时分为几种情况，如下：

1. 数据集中有未标注样本，上传重复的已标注样本，此时未标注样本将被覆盖
2. 数据集中有已标注样本，上传重复的未标注样本，此时已标注样本将被覆盖
3. 数据集中有已标注样本，上传不同标注的已标注样本，此时已有的标注样本将被覆盖

短文本匹配数据

导入与标注

1. 创建数据集

您可以在控制面板中，选择“数据总览”并点击按钮“创建数据集”，即可完成一个数据集的创建。

2.导入短文本相似度数据

进入到新创建的短文本数据集中，如果您手中的数据是未标注数据，可以选择数据标注状态为“无标注信息”，如果您手中的数据是完成标注的数据，可以选择数据标注状态为“有标注信息”。



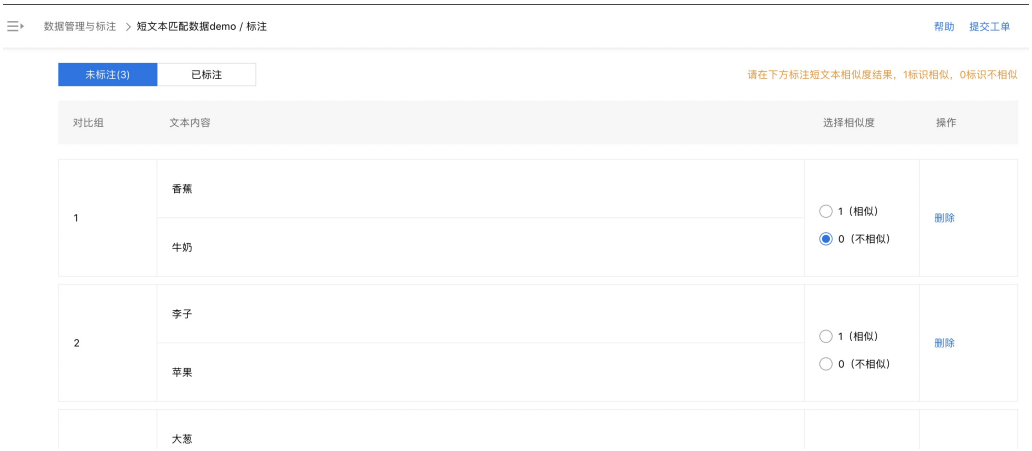
txt文件的未标注数据的样本的样例，如下图：

```
喜欢打篮球的男生喜欢什么样的女生	爱打篮球的男生喜欢什么样的女生
我手机丢了，我想换个手机	我想买个新手机，求推荐
大家觉得她好看吗	大家觉得跑男好看吗?
求秋色之空漫画全集	求秋色之空全集漫画
晚上睡觉带着耳机听音乐有什么害处吗?	孕妇可以戴耳机听音乐吗?
学日语软件手机上的	手机学日语的软件
打印机和电脑怎样连接，该如何设置	如何把带无线的电脑连接到打印机上
侠盗飞车罪恶都市怎样改车	侠盗飞车罪恶都市怎么改车
什么花一年四季都开	什么花一年四季都是开的
看图猜一电影名	看图猜电影！
这上面写的是什么?	胃上面是什么
建议您重新注册，辛苦您了。	可以的，您注销成功后，可以重新注册的，辛苦您了。
小草有什么的特点,可以象征什么?	小草有什么特点可以象征什么
校验失败了，	您好，您还是访客状态呢
尼玛什么意思	尼玛啊是什么意思?
自找苦吃的地方是哪儿?	自找苦吃的地方是哪儿
尾号4位多少	尾号是多少后4位
谢文东能在哪里看	谢文东在哪里看
新概念英语第二册练习册41课答案	新概念英语第二册练习册21练习答案
过年送礼送什么好	过年前什么时候送礼?
壮丁是什么生肖	欲钱买壮丁指的是什么生肖?
这是什么蔬菜啊	这是什么蔬菜呀?
最近有什么好电视剧看?	现在看电影都是有哪些号网站啊
性与爱有什么区别?	性与爱有什么区别
网上卖的烟有真的吗?	网上帮你刷钻是真的吗
```

短文本匹配数据标注

1. 打开短文本匹配数据集的标注页面，您可以看到全部无标注信息的数据

2. 在页面右侧选择两个文本是相似还是不相似，便可完成标注

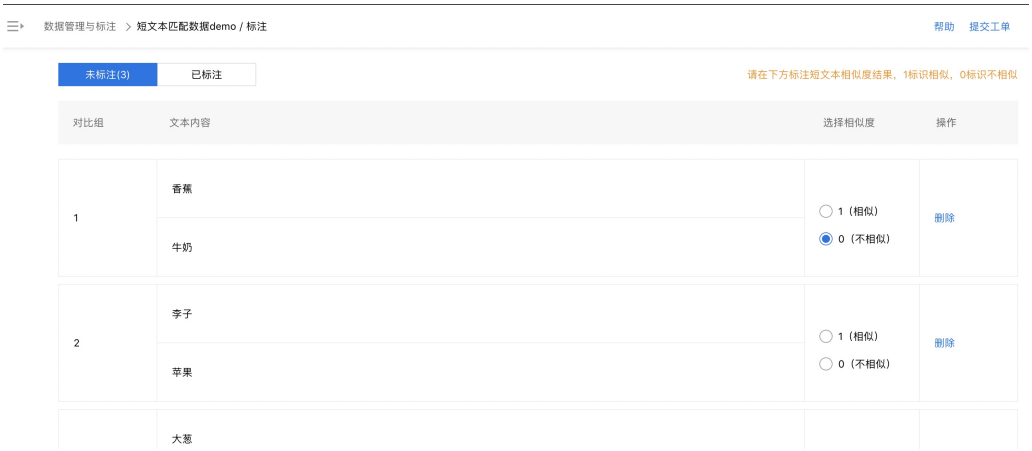


数据标注

短文本匹配数据标注

1. 打开短文本匹配数据集的标注页面，您可以看到全部未标注的数据

2. 在页面右侧选择两个文本是相似还是不相似，便可完成标注



数据去重策略说明

重复样本的定义

一个样本包括文本内容和标签。重复样本的定义，是指您上传的数据中，存在两个样本的文本内容完全一致。则被判定为两个样本是重复样本。例如：

文本内容1	文本内容2	是否相似
今天北京的空气不错	北京今天万里无云	1
今天北京的空气不错	北京今天万里无云	0
今天北京的空气不错	北京今天万里无云	1

上表三个样本均为重复样本，前两个样本虽然标签不一，但文本内容一致，也为重复样本。

Tips：“如何利用好重复样本”，如果您在模型训练过程中，需要通过增加某个类别标签的预测权重，可以通过增加此标签的重复样本来达到此目标。

平台去重策略

平台提供了可去重的数据集，即对您上传的数据进行重复样本的去重。注意：当您确定了数据集为去重或非去重的属性后，便不可修改。

当您创建了一个去重的数据集时，在后续上传数据的过程中，平台可通过检验您当前上传的样本与已上传到此数据集下的样本是否相同，如果相同，则会使用新的样本替代旧的样本。此时分为几种情况，如下：

1. 数据集中有未标注样本，上传重复的已标注样本，此时未标注样本将被覆盖
2. 数据集中有已标注样本，上传重复的未标注样本，此时已标注样本将被覆盖
3. 数据集中有已标注样本，上传不同标注的已标注样本，此时已有的标注样本将被覆盖

序列标注数据

数据导入

创建数据集

您可以在BML中，选择“数据管理/标注”并点击按钮“创建数据集”，对话框中选择数据类型为“文本”，标注类型选择“序列标注”，同时您需要根据您的标注数据，选择此序列标注数据集的标注体系（详见文档《序列标注标注说明》）。注意：当前序列标注数据集不支持在线标注功能，您在序列标注数据集中仅能上传标注数据。当您为此数据集选择标注体系后，标注体系不可修改，也不能导入其他标注体系的数据。



导入已标注数据

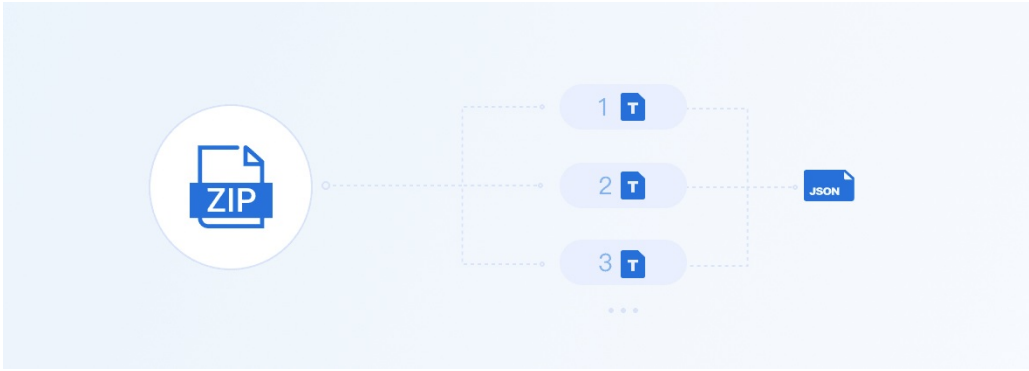
1. 通过下图的"导入"进入到新创建的序列标注数据集的导入页面



2. 在数据集的导入数据部分，您可以选择导入数据的方式，目前平台支持本地上传数据和在线导入已有数据集



3. 本地上传数据，需要以压缩包的形式上传，压缩包内需要包括标注文本文件（utf-8，txt或tsv格式），标注标签的映射文件（utf-8，JSON格式；文件名必须为"label_map.json"），一个压缩包里可以有多个标注文本文件，但都需要对应一个JSON映射文件。如下图示意：



4. 对于上传的数据文件的要求，请注意：

    1. 请上传对应标注体系的标注数据，避免数据上传失败
    2. 标注数据格式要求为“文本内容\t标注结果\n”文本内容和标注结果都需要按字切分，每一行表示一组数据，每组数据的字符数建议不超过512个
    3. 标签映射文件名须为”label_map.json“,标签映射文件的格式为：{"key":"value"}
    4. 上传压缩包文件支持zip格式，单个压缩包限制5G以内。压缩包内需包含txt或tsv格式的文本文件和标注为json格式的标签映射文件，编码仅支持UTF-8，单个文件最大可支持40MB。样本数据详见Demo数据
    5. 平台限免阶段每个账户最多支持100万条样本数据，超出后会被平台忽略

二次导入已标注数据

平台支持用户对于已有的数据集，进行二次导入数据。请注意，再次导入的已标注数据，标注标签需要完全一致。注意，二次导入过程**不校验json文件**。只对标注文本进行校验，与第一次上传的json文件里不一致的标签和标注数据，平台将过滤掉。

标注说明

1.标注体系说明

在序列标注任务中，一般会定义一个标签集合，来表示所有可能取到的预测结果。

标签是对字符串的token序列进行的表示：

对于英文字符串而言，token可以是一个单词（e.g. baidu），也可以是一个字符（e.g. b）；

对于中文字符串而言，token可以是一个分词后的词语，也可以是单个汉字字符；

当前平台支持主流的IOB、IO、IOE、IOBES四种标注体系：

- IOB: 标签B用于文本块的开始，标签I用于文本块中的字符，标签O用于文本块之外的字符
- IOE: 标签I用于文本块中的字符，每个文本块都以标签E结尾，标签O用于文本块之外的字符
- IOBES: 包含了全部的5种标签，使用S标签表示文本块由单个字符组成；由一个以上的字符组成时，首字符总是使用B标签，尾字符总是使用E标签，中间的字符使用I标签
- IO: 只使用I和O标签，如果文本中有连续的同种类型实体的文本块，使用该标签方案不能够区分这种情况

可参考如下案例了解四种标注体系的区别和使用：

	中	韩	对	抗	赛	中	，	于	大	宝	的	进	球	帮	中	国
IOB	CN-B	KR-B	O	O	O	O	O	PER-B	PER-I	PER-I	O	O	O	O	ORG-B	O
IO	CN-I	KR-I	O	O	O	O	O	PER-I	PER-I	PER-I	O	O	O	O	ORG-I	O
IOE	CN-E	KR-E	O	O	O	O	O	PER-I	PER-I	PER-E	O	O	O	O	ORG-I	O
IOBES	CN-S	KR-S	O	O	O	O	O	PER-B	PER-I	PER-E	O	O	O	O	ORG-B	O

英文的标注案例，请参考如下实例，注意由于ERNIE是由超大规模的中文知识语料进行的预训练模型，如果您需要在英文数据集上做序列标注任务，建议使用非ERNIE的任务进行训练。

```
====== ====== ====== ===== == ============ ===== ===== ===== == =========
    Li    Ming   works  at  Agricultural   Bank   of   China  in  Beijing.
====== ====== ====== ===== == ============ ===== ===== ===== == =========
IO    I-PER  I-PER   O     O   I-ORG        I-ORG  I-ORG I-ORG  O   I-LOC
IOB   B-PER  I-PER   O     O   B-ORG        I-ORG  I-ORG I-ORG  O   B-LOC
IOE   I-PER  E-PER   O     O   I-ORG        I-ORG  I-ORG E-ORG  O   E-LOC
IOBES B-PER  E-PER   O     O   B-ORG        I-ORG  I-ORG E-ORG  O   S-LOC
====== ====== ====== ===== == ============ ===== ===== ===== == =========
```

2.标注数据格式说明

本文以平台推荐的IOB标注体系进行详细讲解。在平台上传数据集过程中，提供了快递关键信息抽取场景的示例数据以供参考，Demo数据下载，详见下方截图位置：



2.1 定义标签集合

在本示例场景中，针对需要被抽取的"姓名、电话、省、市、区、详细地址"等实体，标签集合可以定义为：

label = {P-B, P-I, T-B, T-I, A1-B, A1-I, A2-B, A2-I, A3-B, A3-I, A4-B, A4-I, O}

标签集合将会在后续上传平台时放在json文件中用到，详见下方"平台数据准备模块"

每个标签的定义分别为：

标签	说明
P-B	姓名起始位置
P-I	姓名中间位置或结束位置
T-B	电话起始位置
T-I	电话中间位置或结束位置
A1-B	省份起始位置
A1-I	省份中间位置或结束位置
A2-B	城市起始位置
A2-I	城市中间位置或结束位置
A3-B	县区起始位置
A3-I	县区中间位置或结束位置
A4-B	详细地址起始位置
A4-I	详细地址中间位置或结束位置
O	不关注的字

注意每个标签的结果只有 B、I、O 三种。其中 B 表示一个标签类别的开头，比如 P-B 指的是姓名的开头；相应的，I 表示一个标签的延续。此时四种标注体系中，B、I、O、E、S对应的解释也可总结为：

- B，即Begin，表示开始
- I，即Intermediate，表示中间
- E，即End，表示结尾
- S，即Single，表示单个字符
- O，即Other，表示其他，用于标记无关字符

2.2对文本进行标注

对于句子“张三18625584663广东省深圳市南山区百度国际大厦”，每个汉字及对应标签为：

张	三	1	8	6	2	5	5	8	4	6	6	3	广	东	省	深
P-B	P-I	T-B	T-I	T-I	T-I	T-I	T-I	T-I	T-I	T-I	T-I	T-I	A1-B	A1-I	A1-I	A

注意到“张“，”三”在这里表示成了“P-B” 和 “P-I”，反过来讲，得到“P-B”和“P-I”这样的序列，也可以合并成“P” 这个标签。这样重新组合后可以得到以下信息抽取结果：

张三	18625584663	广东省	深圳市	南山区	百度国际大厦
P	T	A1	A2	A3	A4

在您任务配置过程中需要注意，如果您使用了ERNIE预训练模型，您需要对每一条样本进行字粒度的分词处理；如果您使用了非ERNIE的预置网络进行模型训练，则您需要对每一条样本进行词粒度的分词处理。

对于IO、IOBES、IOE三种标注体系，标注过程都类似，您可以根据您手中的训练集，在平台选择对应的标注体系进行数据集的上传。

3. 对应平台的数据集准备

将数据集上传平台，需要将上述2.1中的标签集合和标注文本以压缩包的形式上传。

在2.1中的标签集合为{P-B, P-I, T-B, T-I, A1-B, A1-I, A2-B, A2-I, A3-B, A3-I, A4-B, A4-I, O}，对应的标签映射文件见下方（注意，平台对标签映射文件的文件名有要求，必须为”label_map.json“）：

```
 {
  "P-B":0,
  "P-I":1,
  "T-B":2,
  "T-I":3,
  "A1-B":4,
  "A1-I":5,
  "A2-B":6,
  "A2-I":7,
  "A3-B":8,
  "A3-I":9,
  "A4-B":10,
  "A4-I":11,
  "O":12
}
```

对2.2中的标注数据逐行保存，第一列为切词后的文本token，第二列为标注标签label。两列以制表符tab分开，即 token\tlabel\n，一行标注数据样例如下：

```
张 三 1 8 6 2 5 5 8 4 6 6 3 广 东 省 深 圳 市 南 山 区 百 度 国 际 大 厦  P-B P-I T-B T-I T-I T-I T-I T-I T-I T-I T-I T-I T-I T-I A1-B A1-I A1-I A2-B A2-I A2-I A3-B A3-I A3-I A4-B A4-I A4-I A4-I A4-I A4-I
```

数据去重策略

重复样本的定义

一个样本包括文本内容和标签。重复样本的定义，是指您上传的数据中，存在两个样本的文本内容完全一致。则被判定为两个样本是重复样本。例如：

文本内容	标签
中 韩 对 抗 赛 中 ， 于 大 宝 的 进 球 帮 中 国 队 获 胜	CN-B KR-B O O O O O PER-B PER-I PER-I O O O O ORG-B ORG-I ORG-I O O
中 韩 对 抗 赛 中 ， 于 大 宝 的 进 球 帮 中 国 队 获 胜	CN-B KR-B O O O O O PER-B PER-I PER-I O O O O ORG-B ORG-I ORG-I O O
中 韩 对 抗 赛 中 ， 于 大 宝 的 进 球 帮 中 国 队 获 胜	O O O O O O O O O O O O O O O O O O O

上表三个样本均为重复样本，后两个样本虽然标签不一，但文本内容一致，也为重复样本。

Tips： “如何利用好重复样本”，如果您在模型训练过程中，需要通过增加某个类别标签的预测权重，可以通过增加此标签的重复样本来达到此目标。

平台去重策略

平台提供了可去重的数据集，即对您上传的数据进行重复样本的去重。注意：当您确定了数据集为去重或非去重的属性后，便不可修改。

当您创建了一个去重的数据集时，在后续上传数据的过程中，平台可通过检验您当前上传的样本与已上传到此数据集下的样本是否相同，如果相同，则会使用新的样本替代旧的样本。此时分为几种情况，如下：

1. 数据集中有未标注样本，上传重复的已标注样本，此时未标注样本将被覆盖

2. 数据集中有已标注样本，上传重复的未标注样本，此时已标注样本将被覆盖

3. 数据集中有已标注样本，上传不同标注的已标注样本，此时已有的标注样本将被覆盖

文本实体抽取数据

数据导入

1. 创建数据集

您可以在左侧导航栏中中，选择“数据总览”并点击主内容区域的按钮「创建数据集」，选择数据类型为“文本”，标注类型选择“文本实体抽取”。标注模板中使用默认选项”文本实体抽取“。

img

2.导入未标注文本数据 进入到新创建的文本实体抽取数据集中。如果您手中的数据是未标注数据，可以选择数据标注状态为“无标注信息”。平台暂不支持上传有标注信息的数据。

img

本地上传，包括三种上传方式，分别为「上传TXT文本」、「上传压缩包」、「上传Excel」；

通过文本上传时，需注意：

文本文件内数据格式要求为"文本内容\n"（即每行一个样本，使用回车换行），每一行表示一组数据，每组数据的数建议不超过10000个字符（约5000个字），超出将被截断

文本文件类型支持txt，编码仅支持UTF-8，单次上传限制100个文本文件

通过压缩包上传时，需注意：

压缩包内的一个文本文件将作为一个样本上传。压缩包格式为.zip格式，压缩包内文件类型支持txt，编码仅支持UTF-8。

每组数据的数建议不超过10000个字符（约5000个字），超出将被截断

通过Excel文件上传时，需注意：

Excel文件内首行为表头，每行为一个样本，每个样本字符数不得超过10000个字符（约5000个字），超出将被截断

文件格式支持xlsx格式，单次可上传100个文件

数据标注

通过平台导入「无标注信息」的数据集后，可对无标注数据进行标注操作。

1. 创建标签

进入到待标注的数据集，您需要在右侧的标签栏中创建标签，点击「添加/搜索标签」后，即可输入标签名称，注意平台仅支持数字和字母的标签名

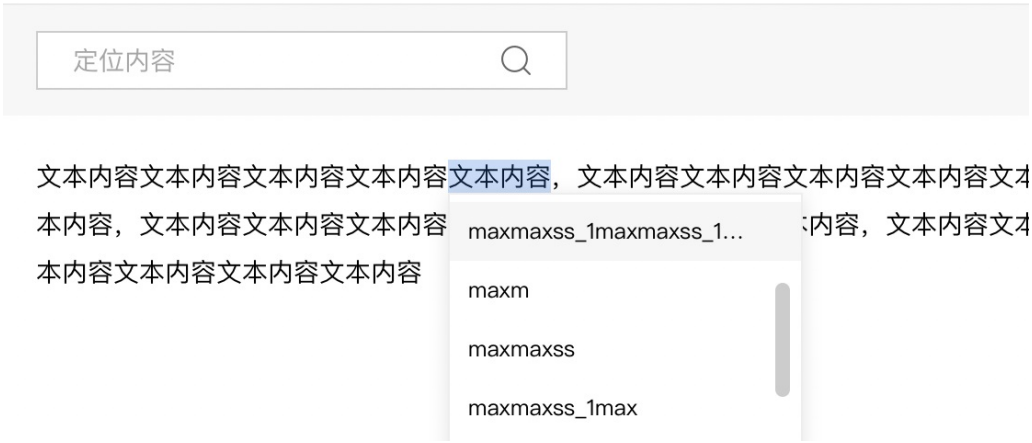


添加完标签后，可以添加标签的备注信息，如下图：



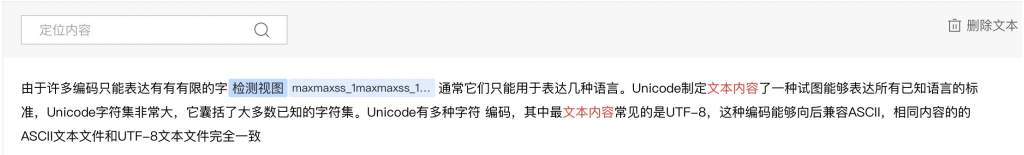
2.标注实体

您可以在文中划选需要标注的文本，然后在弹出的下落标签中选择需要标注的标签，如下图。也可以在划选文本后，在右侧的标签区域点击选择标签



您也可以通过点击某个标签的「连续标注」功能，在文中通过连续划选文本内容来进行连续标注。如下图：

在您需要进行在文中进行关键词检索时，可以在文本上方的「定位内容」区域，输入文本内容，便可在文本区域高亮出检索的关键词，以方便您定位内容和标注。



3.查看标注信息

您可以在标签中，点击「显示/隐藏」图标按钮，来达到让某个图标在文本区域显示标注信息，也可让某个标签隐藏其标注信息。以方便您的查看标注情况。「显示/隐藏」图标按钮如下图所示：



数据去重策略

重复样本的定义

一个文本实体抽取的样本包括文本内容和实体类别。重复样本的定义，是指您上传的数据中，存在两个样本的文本内容完全一致。则被判定为两个样本是重复样本。例如：

以下3条都是文本实体抽取任务的重复样本，样本示例如下：

文本内容	实体类别
今天北京的空气不错	北京：loc
今天北京的空气不错	今天：date
今天北京的空气不错	北京：local

上述两个表中，都代表三个样本均为重复样本，后两个样本虽然标签不一，但文本内容一致，也为重复样本。

Tips：“如何利用好重复样本”，如果您在模型训练过程中，需要通过增加某个类别标签的预测权重，可以通过增加此标签的重复样本来达到此目标。

平台去重策略

平台提供了可去重的数据集，即对您上传的数据进行重复样本的去重。注意：当您确定了数据集为去重或非去重的属性后，便不可修改。

当您创建了一个去重的数据集时，在后续上传数据的过程中，平台可通过检验您当前上传的样本与已上传到此数据集下的样本是否相同，如果相同，则会使用新的样本替代旧的样本。此时分为几种情况，如下：

1. 数据集中有未标注样本，上传重复的已标注样本，此时未标注样本将被覆盖
2. 数据集中有已标注样本，上传重复的未标注样本，此时已标注样本将被覆盖
3. 数据集中有已标注样本，上传不同标注的已标注样本，此时已有的标注样本将被覆盖

智能标注

图像智能标注简介

使用智能标注功能可降低数据的标注成本。启动后，系统会从数据集所有图片中筛选出最关键的图片并提示需要优先标注。通常情况下，只需标注数据集30%左右的数据即可训练模型。与标注所有数据后训练相比，模型效果几乎等同

整体流程以物体检测的智能标注流程为例:

创建智能标注任务

启动物体检测数据集的智能标注前，请先检查一下是否已满足以下条件：

- 所有需要识别的标签都已创建
- 每个标签的标注框数不少于10个
- 所有需要标注的图片都已加入数据集，且所有不相关的图片都已删除

若已满足，即可从导航栏进入「数据服务」-「智能标注」，创建智能标注任务，系统会基于您选择数据类型及数据量级，自动预估任务运行时长



系统筛选难例

系统会分批筛选出最关键需标注的图片，即难例图片。

Tips：难例筛选需要一定时间，您可以正常进行其他未标注图片的标注

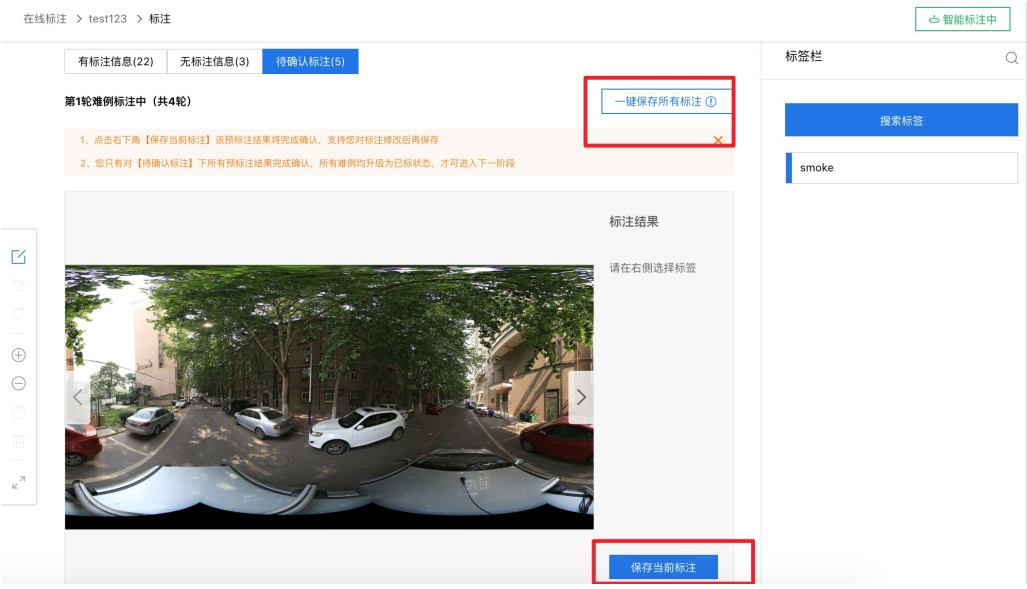


用户确认难例

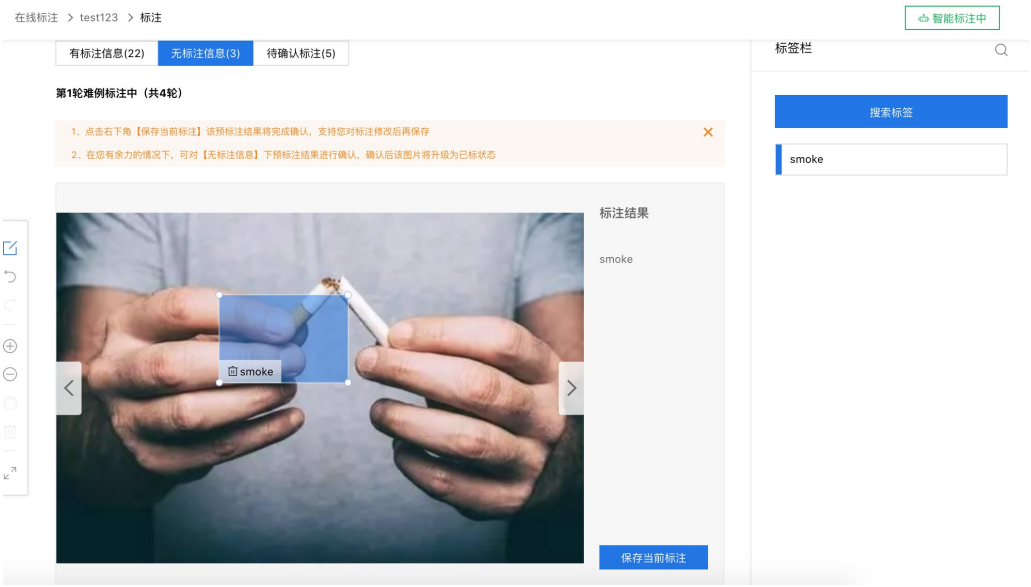
智能标注任务启动后，系统为您自动筛选难例，您可以通过总览页查看进度按钮查看当前难例筛选进度，同时，进度图中也会全局展示您处于难例筛选的具体哪一环节，以便您的操作后续。 筛选难例完成后，绿色进度条会进展到确认难例阶段，您可以点击【确认难例】完成对预标注结果的人工确认。



我们为您的人工确认提供两种模式：单张确认，在该模式下支持您对预标注结果进行修正后点击保存；一键保存所有标注，为提升您的确认效率，默认您对难例的预标注结果全部满意，即可进入下一阶段。



标注难例的预训练模型，也会对您无标注信息下的图片进行预标注结果的展示，您有余力的情况下，可以完成标注确认，确认后该张图片将升级为已标状态，该环节并非是您进入智能标注下一阶段的必备要求。



评估难例效果，完成任务

当您对难例完成确认后，您可以根据本轮次预标注的结果是否满意，判断您是否还需要进入下一轮难例筛选阶段，如果满意本轮难例的预标注效果，系统将自动为您系统其他的未标图片打标签。

第1轮难例标注中（共4轮）

1、点击右下角【保存当前标注】该预标注结果将完成确认，支持您对标注修改后再保存

2、您只有对【待确认标注】下所有预标注结果完成确认，所有难例均升级为已标状态，才可进入下一阶段



其他操作提示

- 在智能标注任务中，有任务上限吗？

支持五条智能标注任务同时运行，超过该上限您需要中止其他任务

- 智能标注中可以增删标签吗？

暂不支持。为了保证系统智能标注的效果，建议在启动功能前就创建好所有需要识别的标签 如果确实需要增删标签，可以先结束智能标注

- 智能标注中可以增删图片吗？

暂不支持。为了保证系统智能标注的效果，建议在启动功能前上传需要标注的所有图片，并删除不相关的图片。如果确实需要增删图片，可以先结束智能标注

- 智能标注中可以修改已标注图片的标注框吗？

可以。但为了保证智能标注的效果，建议不要大量改动。如果确实需要修改大量标注，建议先结束智能标注

- 为什么我已经人工标注了很多图片，但系统预标注依然不准？

系统预标注的结果会受以下因素影响：智能标注期间，对“已标注”图片的标签进行大量改动； 曾结束智能标注，并对标签、图片进行增删

- 多个数据集是否可以同时启动智能标注？

目前每个账号同一时间仅支持对一个数据集启动智能标注

- 共享中的数据集是否可以启动智能标注？

暂不支持。智能标注中的数据集也暂不支持共享

- 智能标注失败了怎么办？

可以先尝试稍后重新启动

问题反馈

如有疑问可提交工单或加入官方QQ群679517246咨询

文本智能标注简介

欢迎您使用「文本智能标注」。您可以通过提供少量人工标注数据和大量无标注数据，通过文本智能标注能力进行自动标注，并将需要人工优先复审的样本筛选出来，辅助您快速完成数据标注工作。您可以获得大规模的智能标注数据，并将数据用于模型的训练。本文将介绍说明如何利用「文本智能标注」获得智能标注数据，以及智能标注与后续如何使用智能标注数据训练模型。

详细使用「文本智能标注」的流程如下：

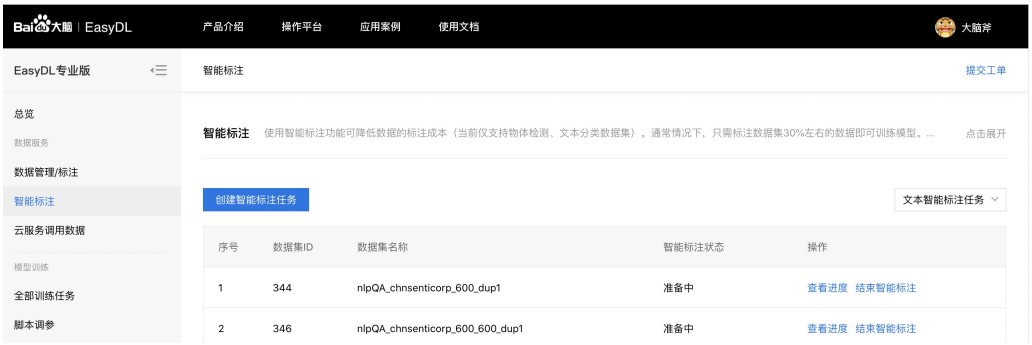
一、启动智能标注获得智能标注数据

Step1 启动智能标注任务前的准备工作

智能标注任务，是对一个数据集中的未标注数据进行智能标注。请您先在「数据管理/标注」模块，创建数据集，并上传一定量的已标注数据和未标注数据。具体数据量的要求，请见下文「创建文本智能标注任务」。

Step2 创建文本智能标注任务

您可以在「智能标注」功能页面下，选中「自然语言」查看文本智能标注的流程概述和说明。



在页面上，通过点击「创建智能标注任务」来选择对应操作的数据集。选择「文本分类」任务，数据集选择您想智能标注的数据集。



注意，请先检验所选择的数据集是否满足以下条件：

1. 数据集中已标注数据量超过600条

2. 每个标注标签的数据量超过50条

3. 未标注数据的数据量超过600条

选择好数据集后，点击「启动智能标注」，平台将对您提交的数据集进行启动前的校验，并可在下方的任务栏中，查看到新建的智能标注任务，任务状态处于「数据校验中」。

Step3 智能标注过程

当数据校验无误后，系统将自动启动智能标注，页面将显示当前智能标注所处的流程状态。 如页面图中所示，智能标注过程分为三个部分，第四个部分为「前往模型训练」的说明，后文将详细说明。

下面，将对智能标注所涉及的三个阶段进行说明：

阶段一：智能标注预学习阶段

在本阶段，系统将使用您提交的已标注数据进行机器学习。预学习阶段预计耗时约40-60分钟不等，根据您的数据集中已标注数据的量而定。如果您需要中止预学习阶段，您可以点击「关闭智能标注」，则系统将重置已完成的智能学习状态。此时您的数据集中「已标注」和「未标注」的数据不会产生变化。

阶段二：智能标注阶段

完成智能标注预学习阶段，系统将自动进入智能标注阶段。此时系统将对您提交数据集下的「未标注」数据进行智能标注，智能标注过程的耗时跟您提交的未标注数据量正相关，即数据量越大，耗时将越长。您可以根据需要，也可「关闭智能标注」。此时注意，关闭智能标注后，已经完成的智能标注的数据，将被恢复为未标注数据。

Tips：按照我们的估算，预计5w条未标注数据，将消耗2小时进行智能标注。

阶段三：完成智能标注，查看智能标注数据

当完成智能标注后，您可前往您的数据集查看。此时数据集将被分为「未标注」、「已标注（人工）」和「已标注（智能）」三个标签。完成智能标注的数据，将放置在「已标注（智能）」中，您可以对智能标注数据进行人工校验，完成校验的数据，将被转移到「已标注（人工）」，代表此样本确认无误，后文将详细说明智能标注的原理。智能标注前的「已标注」数据将放入「已标注（人工）」中，且数据不发生变化。

您也可以在「已标注（智能）」中，查看本次智能标注的准确率，以及智能标注数据的总样本数。 如果您对智能标注的准确率不满意，可以点击「优化智能标注效果」，来对全部的智能标注数据进行优化。



二、使用智能标注数据训练模型

如果您对智能标注的数据质量满意，您可以直接使用「已标注（人工）」和「已标注（智能）」下的数据，对模型进行训练。后文有详细介绍此步骤的原理。

您在配置任务时，在选择数据集的过程中，可以在「选择数据集-可选项」下，勾选「使用智能标注数据」，如果此数据集下有智能标注数据，训练任务时将会使用「已标注（智能）」的数据。

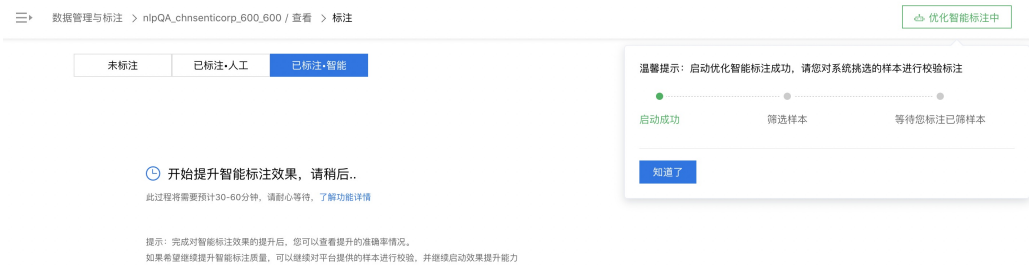


三、启动智能标注数据优化流程

如果您对数据质量不满意，可以在「已标注（智能）」页面中，通过点击「优化智能标注」进入优化流程。

进入后，平台将为您提供至多300条的优先校验样本，优先校验样本是平台挑选的机器较难确认的样本，同时也是对提升智能标注准确率最有帮助的样本。此部分样本将按照优先级从高到底排序，建议您按顺序逐一校验。

在优化流程的页面中，您将发现已完成校验的数量，当前的智能标注准确率情况。当您完成100条以上的样本校验后，「启动效果提升」按钮将置亮，您可以点击并启动优化智能标注的能力。



此过程将对您完成校验的样本重新学习，并对剩余智能标注数据进行重新标注。**此过程不可关闭，请您耐心等待。**完成优化后，将为您展示智能标注准确率的提升状况，并同时为您再次筛选出优先标注样本进行再次的校验和提升。通常情况下，此过程将持续3次，即您完成3次的样本校验。您也可以根据智能标注优化的效果，选择退出优化流程。

Tips：您也可以在启动优化智能标注前增加「未标注」数据，在优化智能标注过程中，也将把新提交的「未标注」数据进行智能标注。

如果您点击「退出优化效果」或您点击进入到了其他页面中，您在优化流程中的进度将会保存。下次返回后，将可以继续进行。

对样本的校验过程，将和文本的标注过程类似。当您发现样本的智能标注有误时，您可以点击选择正确的标签，点击「确认」后，样本将被放入到「已标注（人工）」中。

如果您对当前页的所有样本都判断为正确，则可以点击「本页全部确认」将全部样本添加到「已标注（人工）」中。

四、再次启动智能标注

如果您新建了一个文本智能标注任务，并且完成过一次智能标注。此时，您可以继续在此数据集下，增加「未标注」数据，并对新增的「未标注」数据进行智能标注。

7	343	nlpQA_chnsenticorp_600_600	已完成	再次启动 查看结果
8	342	nlpQA_chnsenticorp_tmp	数据校验失败	再次启动
9	340	nlpQA_chnsenticorp_600	已完成	再次启动 查看结果

Tips：您在再次启动智能标注前，建议先完成至少一次「优化智能标注」的过程。此时，智能标注的数据将比「优化智能标注」前质量更高。

五、文本智能标注原理说明

训练模型过程中，通常需要经历数据集准备（标注）、任务网络配置开发、模型的训练和部署等重要过程。很多时候，模型训练在数据准备阶段遇到数据量不足的问题，使模型开发过程迟迟不能启动。

平台推出的文本智能标注功能，目标是通过少量的已标注数据样本，来获得大规模的智能标注数据，通过减少人工逐一校验的工作，使用智能标注数据来训练小型网络模型，以获得效果和性能更优的模型预测服务。使用文本智能标注数据来训练模型的原理，即当前较为流行的「模型蒸馏」。

下面，将为您简要描述智能标注和模型蒸馏的原理。

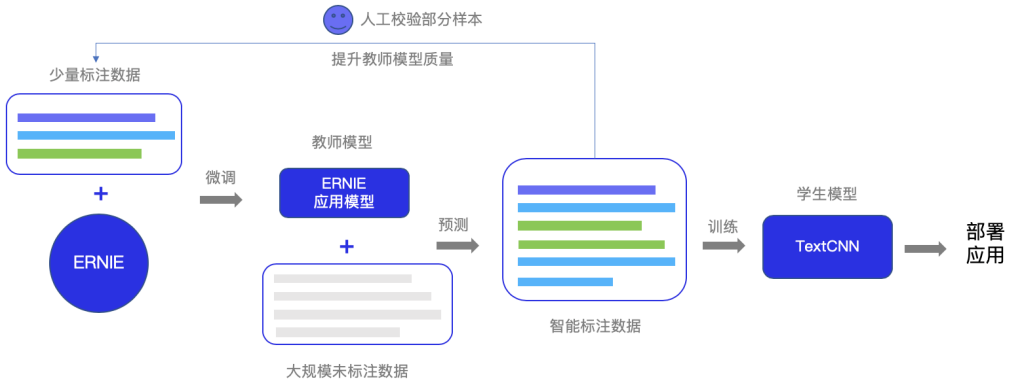
平台智能标注原理

平台提供的文本智能标注，是使用了当前中文模型效果最好的预训练模型ERNIE2.0。完成对少量人工标注数据学习后，可以对未标注数据进行预测，从而获得智能标注数据。由于预测的数据质量，与您提供的人工标注数据的质量强相关。建议您完成智能标注后，查看数据标注的质量效果，然后根据数据情况来判断是否进入到优化智能标注的流程中。

优化智能标注的过程中，系统根据算法挑选出优先标注样本。您完成对优先校验样本的人工校验后，系统将使用此部分样本重新训练模型，从而获得更精准的智能标注数据。

智能标注数据训练模型

ERNIE完成对人工标注数据学习后，生成的模型称之为「教师模型」，通过预测的大规模无监督语料，把「教师模型」的泛化能力通过模型训练教给「学生模型」。如下图所示：



所以，您可以使用智能标注数据，在训练任务配置中，不使用ERNIE预训练模型。通过选择小型的网络，即可训练出效果逼近教师模型ERNIE Large的模型。且此模型的也保留了小网络的预测性能，可以在您的业务中达到具体业务使用的目标。

六、文本智能标注常见问题

问：智能标注可以支持哪几种类型的数据？

答：目前平台提供的文本智能标注，仅支持「单文本-单标签」的数据类型。

问：我是否能够对智能标注数据进行删除操作？

答：可以，您可以处理所有智能标注数据。通常情况下，您可以根据后续模型训练所需，对某些样本进行删除，以减少模型训练过程中的干扰。

问：为什么效果没有提升到一定程度，就提示已经完成优化智能标注？

答：因为您的智能标注数据过少，通过人工标注即可更好的完成校验；

问：为什么我使用智能标注数据训练模型效果不佳。不如只使用人工已标注的数据训练的效果？

答：您可以先进入「优化智能标注」过程，平台将为您挑选提升效果最佳的样本，将您的智能标注数据进行优化。同时您也需要查看数据集的数据分布是否与验证集的分布保持一致。

问：我将优化流程中的样本，都删除了，没法启动优化智能标注了，改怎么办？

答：您可以对此数据集增加「未标注」数据，然后返回到「智能标注」任务页面下，找到此数据集对应的智能标注任务，点击「再次启动」。系统将重新启动智能标注过程，并重置「优化流程」

问：完成智能标注后，我是否能在数据集新增带有新标签的标注数据？

答：您可以上传标注数据。但请注意，如果您上传的已标注数据带有新的标签（或您在已经完成智能标注任务的数据集上新增标签），将会影响智能标注质量。如果您需要使用带有新标签的数据，建议您新建数据集，并将全部人工标注数据导入后，再根据智能标注启动条件来启动新的智能标注任务。

问：为什么启动智能标注任务，或启动优化任务时，有时会不成功？

答：目前平台集中公测阶段，因算力限制，仅提供给一种任务的使用。即如果您已经启动了智能标注任务，则无法在另一个智能标注任务中启动优化任务。如果您有业务中的需求，可加入技术交流群，联系工作人员；

问：非中文语种的数据，可以使用智能标注吗？

答：目前平台仅支持中文的智能标注。不支持其他语种的智能标注任务。

问：为什么我使用智能标注数据训练模型，效果无明显提升？

答：不同场景和行业的数据集，使用智能标注数据提升效果不同。建议您针对智能标注数据进行优化流程，在优化流程中将有优先校验样本进行人工校验。完成至少一次优化流程，然后再使用智能标注数据进行模型训练。

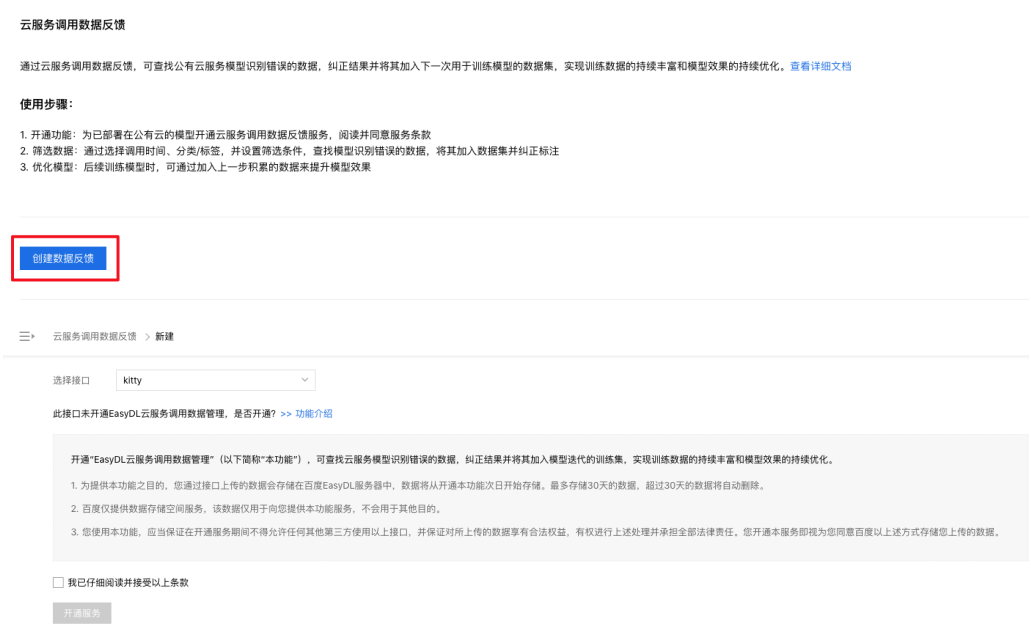
公有云服务调用数据反馈

开通云服务调用数据反馈功能后，可查找云服务模型识别错误的数据，纠正结果并将其加入模型迭代的训练集，实现训练数据的持续丰富和模型效果的持续优化。

具体使用流程如下：

Step 1 为已部署在公有云的模型开通服务

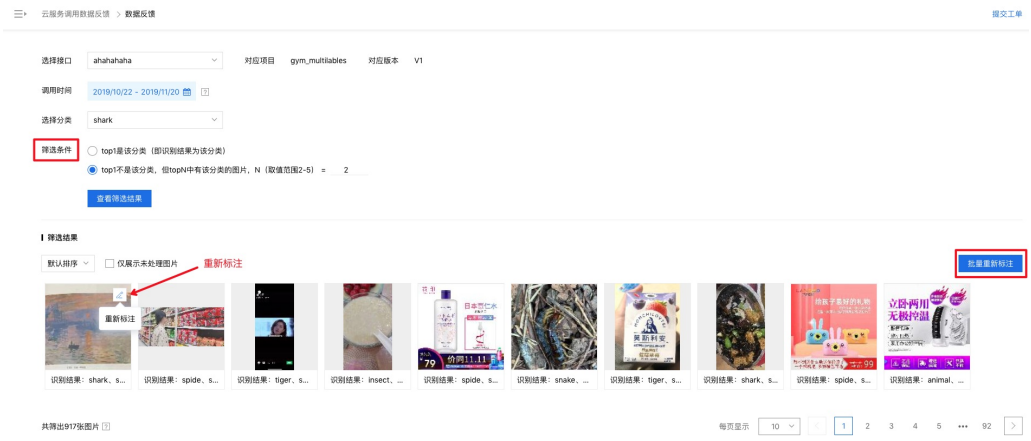
在数据服务-云服务调用数据中创建新的数据反馈，阅读并同意服务条款：



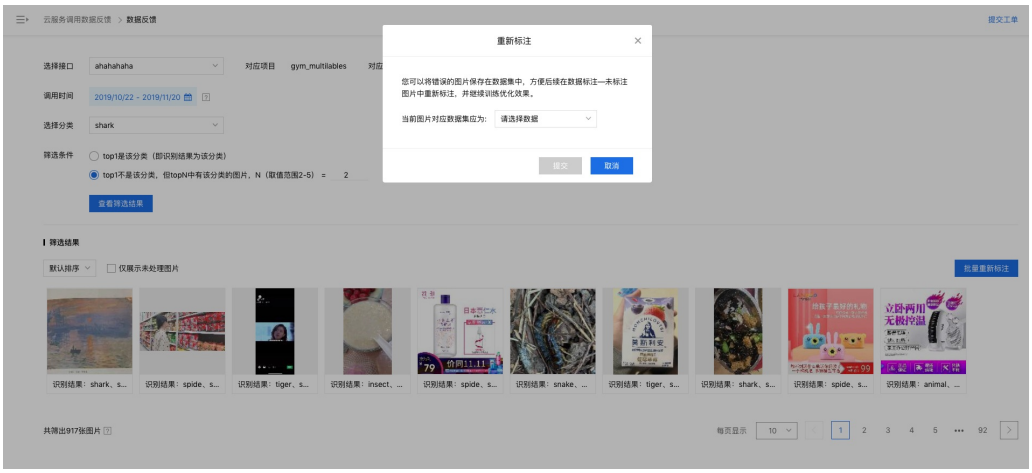
Step 2 查找模型识别错误的数据，将其加入数据集并纠正标注

通过选择调用时间、分类/标签，并设置筛选条件，查找模型识别错误的数据：

注意：数据将从开通功能后开始存储，最多存储30天的数据。当天调用的数据暂不支持即时查看，可在第二天查看



将模型识别错误的数据加入指定数据集（建议新建数据集）并重新标注：



Step 3 在新的训练任务中，通过加入上一步积累的数据来提升模型效果



智能数据API

智能数据API

目前BML对数据集进行创建、删除、导入、导出、新增版本等操作有两种操作方式。 一种是通过在平台页面直接进行操作； 一种为通过调用相关API对数据集进行相应操作。本文档主要介绍如何用API对数据集进行相关操作。

目录

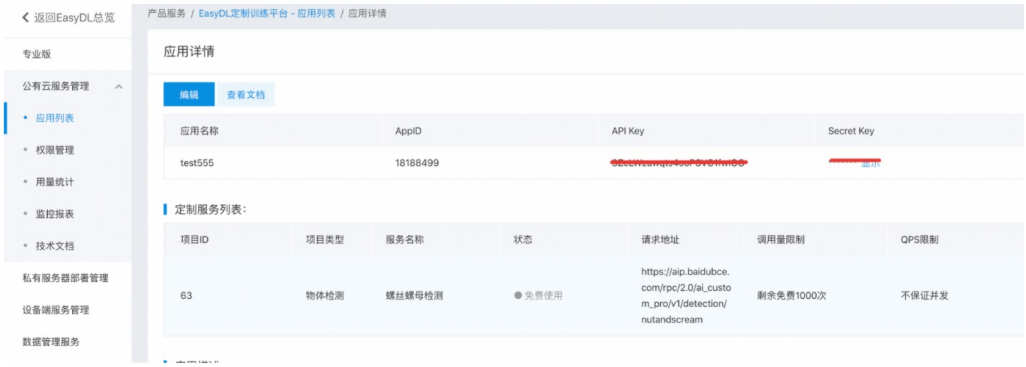
1. 接口鉴权
2. 创建数据集API接口文档
3. 数据集上传API接口文档
4. 删除数据集API接口文档
5. 新增数据集版本API接口文档
6. 获取数据集版本详情API接口文档
7. 删除数据集版本API接口文档
8. 发起数据导入任务API接口文档
9. 发起数据导入任务详情API接口文档
10. 发起数据导出任务API接口文档
11. 获取数据集版本导出详情API接口文档
12. 获取数据集列表API接口文档
13. 获取数据集版本列表API接口文档
14. 获取数据表schema结构API接口文档
15. 错误码

接口鉴权

1、进入BML控制台的公有云服务管理—应用列表——创建应用

2、根据页面提示输入应用名称、应用场景等说明完成应用创建

3、应用详情页获取API Key和Secret Key



创建数据集API接口文档

接口功能描述

本接口用于创建新的数据集，数据集创建完成后自带版本V1。目前支持图像分类、物体检测、文本分类、短文本匹配、表格预测、时序预测六种数据类型创建。

请求说明

请求示例

HTTP 方法：`POST`

请求URL：https://aip.baidubce.com/rpc/2.0/easydl/pro/dataset/create

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body中放置请求参数，参数详情如下：

请求参数

名称	类型	必填	说明
dataset_name	string	是	数据集名称，示例："helloDataset"
type	string	是	数据集类型图像分类：“IMAGE_CLASSIFICATION”物体检测：“OBJECT_DETECTION”短文本相似度：“TEXT_MATCHING”文本分类：“TEXT_CLASSIFICATION”表格预测：“DATA_TABLE_FORECASTING”时序预测：“TIME_SERIES_FORECASTING”
template_type	string	是	标注模板类型图像分类-单图单标签：“IMAGE_CLASSIFICATION_ONE_LABEL”图像分类-单图多标签：“IMAGE_CLASSIFICATION_MUL_LABEL”矩形框标注：“OBJECT_DETECTION_RECTANGLE”短文本单标签：“TEXT_CLASSIFICATION_ONE_LABEL”短文本相似度：“TEXT_SIMILARITY”表格预测：“DATA_TABLE_FORECASTING”时序预测：“ TIME_SERIES_FORECASTING”
storage_type	string	否	存储类型对象存储BOS："usrBos"平台共享存储："sysBos"（默认）
storage_path	string	否	存储类型为对象存储BOS时，对应的完整存储路径即：bos:/bucket/dir/示例：将数据集存储到 bucket：yourBucket，dir：yourDir/text需填写的bosPath为 bos:/yourBucket/yourDir/text/注意：以左斜杠结尾

响应说明

响应参数

名称	类型	说明
log_id	int	操作记录ID
dataset_id	int	数据集版本id
dataset_group_id	int	数据集id
error_code	int	错误码，失败才有（见错误码表）
error_msg	string	错误消息，失败才有

数据集上传API接口文档

接口功能描述

该接口可用于对已创建的数据集上传数据内容，目前支持图像分类、物体检测、文本分类、短文本匹配四类数据创建。

请求说明

请求示例

HTTP 方法：`POST`

请求URL：https://aip.baidubce.com/rpc/2.0/easydl/pro/dataset/addentity

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body中放置请求参数，参数详情如下：

请求参数

字段	必选	类型	说明
dataset_id	是	number	数据集ID
type	是	string	数据集类型，可包括： IMAGE_CLASSIFICATION, OBJECT_DETECTION, TEXT_MATCHING, TEXT_CLASSIFICATION 分别对应：图像分类、物体检测、短文本匹配、文本分类
template_type	是	string	标注模板类型，图像分类-单图单标签传值IMAGE_CLASSIFICATION_ONE_LABEL ；图像分类-单图多标签传值IMAGE_CLASSIFICATION_MUL_LABEL ；矩形框标注传值OBJECT_DETECTION_RECTANGLE 单文本单分类传值：TEXT_CLASSIFICATION_ONE_LABEL 短文本相似度传值：TEXT_SIMILARITY
entity_content	是	string	type为IMAGE_CLASSIFY/OBJECT_DETECT时，填入图片的base64编码；type为TEXT_CLASSIFY/TEXT_MATCHING时，填入utf-8编码的文本。**注意：当type为TEXT_MATCHING（即短文本匹配）时，短文本之间用\t分隔。**
labels	否	array(object)	标签/分类数据
+label_name	否	string	标签/分类名称（由数字、字母、中划线、下划线组成），长度限制20B
+left	否	number	物体检测时需给出，标注框左上角到图片左边界的距离(像素)
+top	否	number	物体检测时需给出，标注框左上角到图片上边界的距离(像素)
+width	否	number	物体检测时需给出，标注框的宽度(像素)
+height	否	number	物体检测时需给出，标注框的高度(像素)
append_label	否	boolean	确定添加标签/分类的行为：追加(true)、替换(false)。默认为追加(true)。

返回说明

返回参数

字段	必选	类型	说明
log_id	是	int	唯一的log id，用于问题定位

删除数据集API接口文档

接口功能描述

本接口用于删除数据集（包含全部版本数据）。目前支持对表格预测、时序预测两种数据集进行操作。

请求说明

请求示例

HTTP 方法：`POST`

请求URL：https://aip.baidubce.com/rpc/2.0/easydl/pro/dataset/deleteDataset

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body中放置请求参数，参数详情如下：

请求参数

名称	类型	必填	说明
dataset_group_id	int	是	数据集ID

响应说明

响应参数

名称	类型	说明
log_id	int	操作记录ID
error_code	int	错误码，失败才有（见错误码表）
error_msg	string	错误消息，失败才有
result	bool	返回结果，成功才有

新增数据集版本API接口文档

接口功能描述

本接口用于为数据集新增版本（不继承历史版本）。目前支持对表格预测、时序预测两种数据集进行操作。

请求说明

请求示例

HTTP 方法：`POST`

请求URL：https://aip.baidubce.com/rpc/2.0/easydl/pro/dataset/createDatasetVersion

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body中放置请求参数，参数详情如下：

请求参数

名称	类型	必填	说明
dataset_group_id	int	是	需要新增版本的数据集ID
description	string	否	版本描述
type	string	是	数据集类型图像分类：“IMAGE_CLASSIFICATION”物体检测：“OBJECT_DETECTION”短文本相似度：“TEXT_MATCHING”文本分类：“TEXT_CLASSIFICATION”表格预测：“DATA_TABLE_FORECASTING”时序预测：“TIME_SERIES_FORECASTING”
template_type	string	是	标注模板类型图像分类-单图单标签：“IMAGE_CLASSIFICATION_ONE_LABEL”图像分类-单图多标签：“IMAGE_CLASSIFICATION_MUL_LABEL”矩形框标注：“OBJECT_DETECTION_RECTANGLE”短文本单标签：“TEXT_CLASSIFICATION_ONE_LABEL”短文本相似度：“TEXT_SIMILARITY”表格预测：“DATA_TABLE_FORECASTING”时序预测：“ TIME_SERIES_FORECASTING”

响应说明

响应参数

名称	类型	说明
log_id	int	操作记录ID
error_code	int	错误码，失败才有（见错误码表）
error_msg	string	错误消息，失败才有
dataset_id	int	版本id，成功才有

获取数据集版本详情API接口文档

接口功能描述

本接口用于获取特定数据集版本的详情信息，包括基本信息和统计信息。目前支持对表格预测、时序预测两种数据集进行操作。

请求说明

请求示例

HTTP 方法：`POST`

请求URL：https://aip.baidubce.com/rpc/2.0/easydl/pro/dataset/describeDatasetVersion

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body中放置请求参数，参数详情如下：

请求参数

名称	类型	必填	说明
dataset_id	int	是	数据集版本ID

响应说明

响应参数

名称	类型	说明
log_id	int	操作记录ID
error_code	int	错误码，失败才有（见错误码表）
error_msg	string	错误消息，失败才有
result	object	返回结果，成功才有

返回说明

返回参数

名称	类型	说明
dataset_group_id	int	数据集ID
dataset_id	int	数据集版本ID
version_number	int	数据集版本序号
dataset_name	string	数据集名称
description	string	版本描述
storage_type	string	存储类型对象存储BOS："usrBos"平台共享存储："sysBos"
storage_path	string	存储类型为对象存储BOS时，对应的完整存储路径
size_MB	float	数据集大小，单位为MB，例：0.01
columns	int	数据集列数
rows	int	数据集行数
import_status	string	导入状态正在导入：Importing已完成：ImportFinished导入失败：ImportFailed
type	string	数据集类型图像分类："IMAGE_CLASSIFICATION"物体检测："OBJECT_DETECTION"短文本相似度："TEXT_MATCHING"文本分类："TEXT_CLASSIFICATION"表格预测："DATA_TABLE_FORECASTING"时序预测：" TIME_SERIES_FORECASTING"
template_type	string	标注模板类型图像分类-单图单标签："IMAGE_CLASSIFICATION_ONE_LABEL"图像分类-单图多标签："IMAGE_CLASSIFICATION_MUL_LABEL"矩形框标注："OBJECT_DETECTION_RECTANGLE"短文本单标签："TEXT_CLASSIFICATION_ONE_LABEL"短文本相似度："TEXT_SIMILARITY"表格预测："DATA_TABLE_FORECASTING"时序预测：" TIME_SERIES_FORECASTING"
import_progress	float	导入进度例：进度21%，表示为"0.21"
create_time	string	创建时间
modify_time	string	更新时间

删除数据集版本API接口文档

接口功能描述

本接口用于删除数据集特定版本。目前支持对表格预测、时序预测两种数据集进行操作。

请求说明

请求示例

HTTP 方法：`POST`

请求URL：https://aip.baidubce.com/rpc/2.0/easydl/pro/dataset/deleteDatasetVersion

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body中放置请求参数，参数详情如下：

请求参数

名称	类型	必填	说明
dataset_id	int	是	数据集版本ID

响应说明

响应参数

名称	类型	说明
log_id	int	操作记录ID
error_code	int	错误码，失败才有（见错误码表）
error_msg	string	错误消息，失败才有
result	bool	返回结果，成功才有

发起数据导入任务API接口文档

接口功能描述

本接口用于导入Esaydap数据源。目前支持对表格预测、时序预测两种数据集进行操作。

请求说明

请求示例

HTTP 方法：`POST`

请求URL：https://aip.baidubce.com/rpc/2.0/easydl/pro/dataset/createImportTask

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body中放置请求参数，参数详情如下：

请求参数

名称	类型	必填	说明
dataset_id	int	是	数据集版本ID
data_source_type	string	是	固定值cloud_edap
database_name	string	是	edap数据库名
table_name	string	是	edap数据表名
column_error	string	否	处理升级auto自动处理（默认）skip放弃错误行fail快速失败
schema	object	否	表格式，如果不传会使用默认的推测格式，全部列导入
schema.columns	list	是	数据列序号，即可挑选需要的列导入，0表示第一列
schema.raw_names	list	是	数据表原来的列名
schema.names	list	是	数据表新的列名
schema.types	list	是	数量列的类型，只支持基础类型，不支持复合类型 smallint/int/bigint/float/double/date/timestamp/uuid/string/decimal(x,y)，如decimal(10,2)表示总长度10，小数精度为2

例如

```
{
  "dataset_id":19313,
  "data_source_type":"cloud_edap",
  "database_name":"dae_test",
  "table_name":"dae_table",
  "upload_type":"overwrite",
  "column_error":"auto",
  "schema":{
      "columns":[0,1,2,3,4,5,6],
      "types":["int","string","decimal(10,0)","double","date","string","smallint"],
      "names":["id","name","c_decimal","c_double","c_date","c_bool","part_id"],
      "raw_names":["id","name","c_decimal","c_double","c_date","c_bool","part_id"]
  }
}
```

响应说明

响应参数

名称	类型	说明
log_id	int	操作记录ID
error_code	int	错误码，失败才有（见错误码表）
error_msg	string	错误消息，失败才有
task_id	int	导入任务id，成功才有

例如

```
{
  "task_id": 48459,
  "log_id": 1525440098
}
```

获取数据导入任务详情API接口文档

接口功能描述

本接口用于获取数据集版本导入详情。目前支持对表格预测、时序预测两种数据集进行操作。

请求说明

请求示例

HTTP 方法：`POST`

请求URL：https://aip.baidubce.com/rpc/2.0/easydl/pro/dataset/describeImportTask

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body中放置请求参数，参数详情如下：

请求参数

名称	类型	必填	说明
task_id	int	是	导入任务ID

响应说明

响应参数

名称	类型	说明
log_id	int	操作记录ID
error_code	int	错误码，失败才有（见错误码表）
error_msg	string	错误消息，失败才有
result	object	导入任务信息，成功才有

返回说明

返回参数

名称	类型	说明
dataset_id	int	数据集版本ID
size_MB	float	文件大小，单位为MB，例：0.01（导入成功后）
columns	int	数据集列数
rows	int	数据集行数
import_status	string	导入状态正在导入：Importing已完成：ImportFinished导入失败：ImportFailed
import_progress	float	导入进度例：进度21%，表示为“0.21”
start_time	string	导入开始时间
finish_time	string	导入完成时间

发起数据导出任务API接口文档

接口功能描述

本接口用于创建数据导出任务。目前支持对表格预测、时序预测两种数据集进行操作。

请求说明

请求示例

HTTP 方法：`POST`

请求URL：https://aip.baidubce.com/rpc/2.0/easydl/pro/dataset/createExportTask

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body中放置请求参数，参数详情如下：

请求参数

名称	类型	必填	说明
dataset_id	int	int	数据集版本ID
storage_type	string	是	导出位置的存储类型：对象存储BOS："usrBos"平台共享存储："sysBos"
storage_path	string	否	存储类型为对象存储BOS时，对应的导出完整路径

示例

```
{
  "dataset_id": 19297,
  "storage_type":"sysBos"
}
```

响应说明

响应参数

名称	类型	说明
log_id	int	操作记录ID
error_code	int	错误码，失败才有（见错误码表）
error_msg	string	错误消息，失败才有
task_id	int	导出任务ID，成功才有

返回

```
{
  "task_id": 1470,
  "log_id": 2475571425
}
```

获取数据集版本导出详情API接口文档

接口功能描述

本接口用于获取数据集版本导出详情。目前支持对表格预测、时序预测两种数据集进行操作。

请求说明

请求示例

HTTP 方法：`POST`

请求URL：https://aip.baidubce.com/rpc/2.0/easydl/pro/dataset/describeExportTask

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body中放置请求参数，参数详情如下：

请求参数

名称	类型	必填	说明
task_id	int	是	导出任务ID

示例

```
{
  "task_id": 1470
}
```

响应说明

响应参数

名称	类型	说明
log_id	int	操作记录ID
error_code	int	错误码，失败才有（见错误码表）
error_msg	string	错误消息，失败才有
result	object	导出任务，成功才有

返回说明

返回参数

名称	类型	说明
storage_type	string	导出位置的存储类型：对象存储BOS："usrBos"平台共享存储："sysBos"
storage_path	string	存储类型为对象存储BOS时，对应的导出完整路径
size_MB	float	导出文件大小，单位为MB，例：0.01 （成功后）
sample_count	int	数据量
export_status	string	导出状态，状态说明：导出中：Exporting已完成完成：ExportFinished导出失败：ExportFailed
start_time	string	导出开始时间
finish_time	string	导出完成时间
download_url	string	下载地址

返回

```
{
  "storage_type": "sysBos",
  "storage_path": "bos:/sdc-default/_system_/dataset/ds-7a7cbwemm1m2f6fz/download/DatasetId_19297_export_1719560476.zip",
  "size_MB": 0.01,
  "sample_count": "16",
  "export_status": "ExportFinished",
  "start_time": "2024-06-28 15:41:16",
  "finish_time": "2024-06-28 15:42:12",
  "download_url": "https://bj.bcebos.com/sdc-default/_system_/dataset/ds-7a7cbwemm1m2f6fz/download/DatasetId_19297_export_1719560476.zip?authorization=bce-auth-v1%2F50c8bb753dcb4e1d8646bb1ffefd3503%2F2024-06-28T07%3A42%3A41Z%2F3600%2F%2Fbf302fbb453bc1ce28a7d46c7b1336c8db10f6d8375849dd725bdc0213ce1366",
  "log_id": 2560913930
}
```


获取数据集列表API接口文档

接口功能描述

本接口用于获取符合特定查询条件的数据集列表。目前支持对表格预测、时序预测两种数据集进行操作。

请求说明

请求示例

HTTP 方法：`POST`

请求URL：https://aip.baidubce.com/rpc/2.0/easydl/pro/dataset/describeDatasets

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body中放置请求参数，参数详情如下：

请求参数

名称	类型	必填	说明
filter	map	否	筛选条件，此处支持对于数据类型进行筛选，对应Key为type
filter.type	List	否	根据数据集数据类型进行筛选，所有可选值如下：图像分类：“IMAGE_CLASSIFICATION”物体检测：“OBJECT_DETECTION”短文本相似度：“TEXT_MATCHING”文本分类：“TEXT_CLASSIFICATION”表格预测：“DATA_TABLE_FORECASTING”时序预测：“ TIME_SERIES_FORECASTING”
filter.template_type	List	否	标注模板类型图像分类-单图单标签：“IMAGE_CLASSIFICATION_ONE_LABEL”图像分类-单图多标签：“IMAGE_CLASSIFICATION_MUL_LABEL”矩形框标注：“OBJECT_DETECTION_RECTANGLE”短文本单标签：“TEXT_CLASSIFICATION_ONE_LABEL”短文本相似度：“TEXT_SIMILARITY”表格预测：“DATA_TABLE_FORECASTING”时序预测：“ TIME_SERIES_FORECASTING”
word	string	否	搜索关键词
limit	int	否	分页大小，默认10
offset	int	否	分页偏移，默认0

请求

```
{
  "limit": 20,
  "offset": 0,
  "filter":{"type":["IMAGE_CLASSIFICATION"]}
}
```

响应说明

响应参数

名称	类型	说明
log_id	int	操作记录ID
error_code	int	错误码，失败才有（见错误码表）
error_msg	string	错误消息，失败才有
result	object	返回结果，成功才有

返回说明

返回参数

名称	类型	说明
page_info	object	分页信息
datasets	List	数据集列表

返回

```
{
  "page_info": {
     "limit": 20,
     "offset": 0,
     "total": 1040
  },
  "datasets": [
     {
        "dataset_group_id": 13814,
        "dataset_name": "pdf_quality_good"
     },
     {
        "dataset_group_id": 13813,
        "dataset_name": "pdf_quality_bad"
     },
     {
        "dataset_group_id": 13800,
        "dataset_name": "pdf_quality3"
     },
     {
        "dataset_group_id": 13798,
        "dataset_name": "pdf_quality2"
     }
     ......
  ],
  "log_id": 517361526
}
```

pageInfo说明

名称	类型	说明
limit	int	分页大小
offset	int	数据偏移
total	int	总数

dataset说明

名称	类型	说明
dataset_group_id	int	数据集ID
dataset_name	string	数据集名称

获取数据集版本列表API接口文档

接口功能描述

本接口用于查询某数据集的版本列表。目前支持对表格预测、时序预测两种数据集进行操作。

请求说明

请求示例

HTTP 方法：`POST`

请求URL：https://aip.baidubce.com/rpc/2.0/easydl/pro/dataset/describeDatasetVersions

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body中放置请求参数，参数详情如下：

请求参数

名称	类型	必填	说明
dataset_group_id	int	是	数据集ID
limit	int	否	分页大小，默认10
offset	int	否	分页便宜，默认0

示例

```
{
  "dataset_group_id":13805,
  "limit":10,
  "offset":0
}
```

响应说明

响应参数

名称	类型	说明
log_id	int	操作记录ID
error_code	int	错误码，失败才有（见错误码表）
error_msg	string	错误消息，失败才有
result	object	返回结果，成功才有

返回说明

返回参数

名称	类型	说明
page_info	object	分页信息
dataset_versions	list	数据集版本列表

返回


```
{
  "page_info": {
     "limit": 10,
     "offset": 0,
     "total": 1
  },
  "dataset_versions": [
     {
        "dataset_id": 19297,
        "dataset_name": "数据集测试",
        "version_number": 1,
        "import_status": 0,
        "columns": 2,
        "rows": 16,
        "type": "DATA_TABLE_FORECASTING",
        "template_type": "DATA_TABLE_FORECASTING",
        "create_time": "2024-06-24 14:52:26",
        "modify_time": "2024-06-24 15:54:29"
     }
  ],
  "log_id": 2268245848
}
```


pageInfo说明

名称	类型	说明
limit	int	分页大小
offset	int	数据便宜
total	int	总数

datasetVersion说明

名称	类型	说明
dataset_id	int	数据集版本ID
dataset_name	string	数据集名称
version_number	int	数据集版本序号
import_status	string	导入状态正在导入：Importing已完成：ImportFinished导入失败：ImportFailed
columns	int	数据集列数
rows	int	数据集行数
type	string	标注类型图像分类：“IMAGE_CLASSIFICATION”物体检测：“OBJECT_DETECTION”短文本相似度：“TEXT_MATCHING”文本分类：“TEXT_CLASSIFICATION”表格预测：“DATA_TABLE_FORECASTING”时序预测：“ TIME_SERIES_FORECASTING”
template_type	string	标注模版图像分类-单图单标签：“IMAGE_CLASSIFICATION_ONE_LABEL”图像分类-单图多标签：“IMAGE_CLASSIFICATION_MUL_LABEL”矩形框标注：“OBJECT_DETECTION_RECTANGLE”短文本单标签：“TEXT_CLASSIFICATION_ONE_LABEL”短文本相似度：“TEXT_SIMILARITY”表格预测：“DATA_TABLE_FORECASTING”时序预测：“ TIME_SERIES_FORECASTING”
create_time	string	创建时间
modify_time	string	更新时间

获取数据表schema结构API接口文档

接口功能描述

本接口用于获取数据表的schema结构。目前支持对表格预测、时序预测两种数据集进行操作。

请求说明

请求示例

HTTP 方法：`POST`

请求URL：https://aip.baidubce.com/rpc/2.0/easydl/pro/dataset/guessSchema

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body中放置请求参数，参数详情如下：

请求参数

名称	类型	必填	说明
data_source_type	string	是	固定值cloud_edap
database_name	string	是	数据库ID
table_name	string	是	数据表ID

示例

```
{
  "data_source_type":"cloud_edap",
  "database_name":"dae_test",
  "table_name":"dae_table"
}
```

响应说明

响应参数

名称	类型	说明
log_id	int	操作记录ID
error_code	int	错误码，失败才有（见错误码表）
error_msg	string	错误消息，失败才有
result	object	返回结果，成功才有

返回说明

返回参数

名称	类型	说明
schema	object	数据表结构
schema.names	list	数据表列名
schema.types	list	数据表中字段的数据类型
schema.columns	list	数据表中列的序号
values	list	数据表中每个字段的真值，根据types中描述的数据类型决定
total_size	int	总行数
has_preview	bool	是否有预览数据

返回

```
{
  "result": {
    "schema": {
      "names": [
        "id",
        "name",
        "c_decimal",
        "c_double",
        "c_date",
        "c_bool",
        "part_id"
      ],
      "types": [
        "int",
        "string",
        "decimal(10,0)",
        "double",
        "date",
        "string",
        "smallint"
      ],
      "columns": [
        0,
        1,
        2,
        3,
        4,
        5,
        6
      ]
    },
    "values": [],
    "total_size": 7,
    "has_preview": false
  },
  "log_id": 1293726887
}
```


错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如Access Token失效返回：


```
{
 "error_code": 110,
 "error_msg": "Access token invalid or no longer valid"
}
```


需要重新获取新的Access Token再次请求即可。

错误码	错误信息	描述
1	Unknown error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（679517246）或工单联系技术支持团队。
2	Service temporarily unavailable	服务暂不可用，请再次请求， 如果持续出现此类错误，请通过QQ群（679517246）或工单联系技术支持团队。
3	Unsupported openapi method	调用的API不存在，请检查后重新尝试
4	Open api request limit reached	集群超限额
6	No permission to access data	无权限访问该用户数据
13	Get service token failed	获取token失败
14	IAM Certification failed	IAM鉴权失败
15	app not exsits or create failed	应用不存在或者创建失败
17	Open api daily request limit reached	每天请求量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（679517246）联系群管手动提额
18	Open api qps request limit reached	QPS超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（679517246）联系群管手动提额
19	Open api total request limit reached	请求总量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（679517246）联系群管手动提额
100	Invalid parameter	无效的access_token参数，请检查后重新尝试
110	Access token invalid or no longer valid	access_token无效
111	Access token expired	access token过期
406000	internal server error	服务错误
406001	param[xx] invalid	参数xx不合法，请检查相关参数
406002	dataset not exist	数据集不存在
406003	dataset already exists	数据集已存在
406004	dataset can not be modified temporarily	数据集暂不可修改
406005	label not exist	标签/分类不存在
406006	no permission to modify the dataset	没有修改数据集的权限
406007	dataset cannot be modified while smart annotation is running	智能标注期间不可修改数据集
406008	quota exceeded	配额超限

模型训练

预置模型调参建模

预置模型调参简介

在BML中，面向初级的AI开发者提供了预置模型调参建模方式。在该方式下，开发者无需关注构建模型的代码细节，而只需要选择合适的预训练模型以及网络即可。对于系统预置的可配置网络参数，可以适用于大部分场景，开发者也可以根据自己的经验进行调整，以获得更适合特定场景的模型。

预置模型调参支持视觉和自然语言处理两大技术方向：

技术方向	类型
视觉	图像分类
	物体检测
	实例分割
自然语言处理	文本分类
	短文本相似度
	序列标注
	文本实体抽取
表格数据预测	二分类
	多分类
	回归

开发视觉模型

视觉任务简介

预置模型调参支持多种视觉任务，训练得到的模型可以应用到广泛实际场景中。

视觉模型类型

预置模型调参支持图像分类、物体检测及实例分割三类模型类型。

图像分类

识别一张图中是否是某类物体/状态/场景，适用于图片内容单一、需要给整张图片分类的场景。

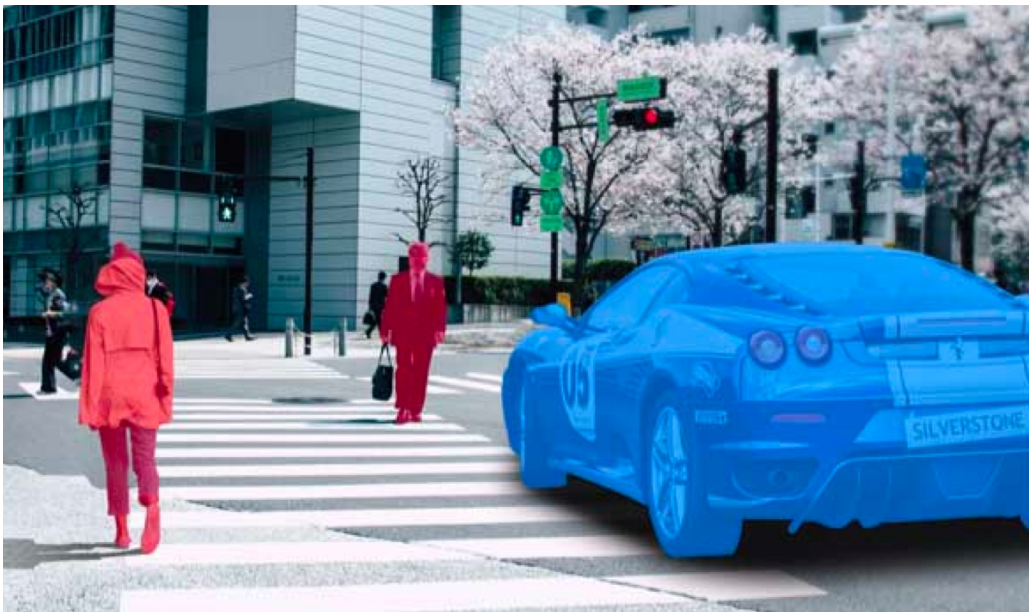


物体检测

检测图中每个物体的位置、名称。适合图中有多个主体要识别、或要识别主体位置及数量的场景。



实例分割 对比物体检测，支持用多边形标注训练数据，模型可像素级识别目标。适合图中有多个主体、需识别其位置或轮廓的场景



视觉模型应用场景

生产环境安全监控

对生产环境现场做安全性监控，如是否出现挖掘机等危险物品、工人是否佩戴安全帽、是否穿工作服等进行检查，辅助人工判断安全隐患并及时预警，保证生产环境安全运行

工业质检

在流水线作业中针对组合型产品可能存在的不合格情况进行列举，并投入示例图片进行训练，从而训练出自动判断合格或不合格的模型，辅助人工判断产品质量

图像定制分类标签

将图片进行定制分类标签并进行训练，实现对海量图片自动打标签，实现将图片更好的在前端面向C端用户展示，以及判断用户点击图片内容从而进行相关图片推荐等功能

专业领域研究

针对医疗检验场景中，可能存在的正常或异常的结果判断，基于图片关键特征进行标注进行物体检测/图像分类训练，协助医生高效完成结果判断

图像审核

根据业务需求，制定图片审核标准，用EasyDL判断是否合规。常用于视频、新闻等内容平台定制内容审核策略，过滤不良信息。或用于线上活动，判断C端用户提交图片的合规性。

创建视觉任务

在任务列表点击【创建项目】，在弹窗中提交以下信息完成项目创建：

1. **完善个人信息**：填写项目归属、行业、联系方式完成项目创建。
   注意：有效的联系方式将有助于后续模型上线的人工快速审核，以及更快的百度官方支持



2. **提交项目信息**:提交模型名称、技术方向、任务类型、应用场景及功能描述，即可完成项目创建。其中**任务类型与配置任务可选的数据类型一一对应**。



配置视觉任务

配置训练集及评测集

配置训练集

在添加任务备注后，在添加数据的位置选择添加训练集。

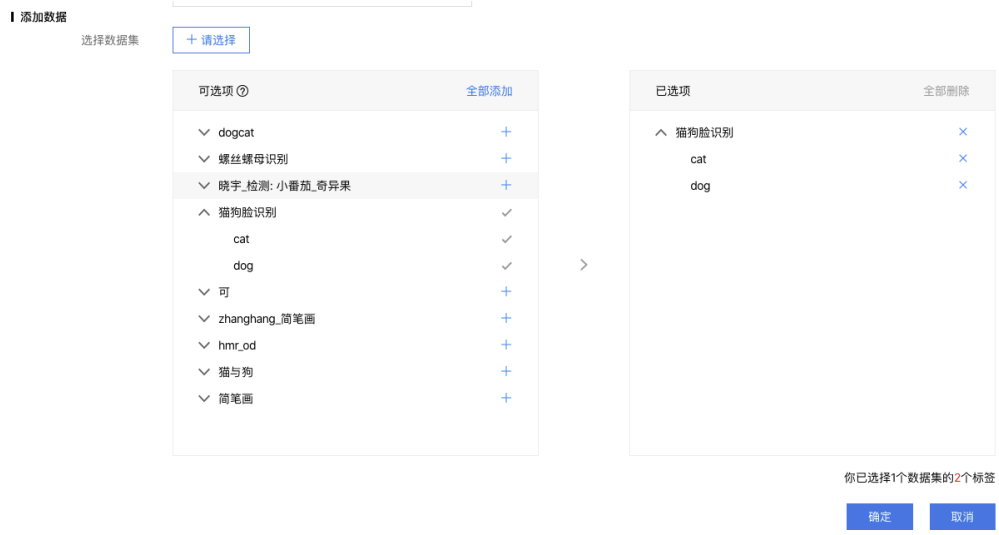
可选的数据集要求如下：

1. 数据集标注类型与项目类型一致，如同为图像分类-单图单标签/物体检测-矩形框标注等

2. 数据集数据量不为0；

3. 数据集状态非智能标注中，非导入中

数据集状态支持导入多个数据集的多个标签，**如果选择标签名称一致，则相应数据内容会被合并**。 当一个项目新建了多个任务类型进行迭代训练时，训练数据会默认为最近一次训练版本所选的数据，**如果标签不变仅每个标签下的标注数据发生变化，或者配置脚本内容发生变化，则数据的部分可以不做任何操作。**



配置数据增强策略

在配置完训练集之后，紧接着就可以配置数据增强策略。

深度学习模型的成功很大程度上要归功于大量的标注数据集。通常来说，通过增加数据的数量和多样性往往能提升模型的效果。当在实践中无法收集到数目庞大的高质量数据时，可以通过配置数据增强策略，对数据本身进行一定程度的扰动从而产生"新"数据。模型会通过学习大量的"新"数据，提高泛化能力。

你可以在「默认配置」、「手动配置」、「自动搜索」三个选项进行选择，完成数据增强策略的配置。



配置评测集

评测集为非必选项，如果额外上传一批非在训练集的数据作为评测集，可在模型训练完毕后单独获得额外评测集的测试效果。**其中评测集的标签需要与训练集所选标签完全一致，否则将无法启动训练任务。**

配置脚本注意事项

说明：BML目前支持飞桨（Paddle Paddle）深度学习框架 了解飞桨

目前BML为每一种预训练模型都预置了脚本代码，在不需要修改的情况下可直接启动训练。
自定义脚本内容过程中有如下注意事项：

1. 可以自定义的部分包括超参配置字典conf和模型头部网络定义函数create_model。主体网络（含预训练参数）在页面可选。

2. 超参配置conf包括可调的超参，比如训练轮数epoch，其类型和范围见注释。不要随意设置否则可能会训练失败，比如batch size太大，可能导

致显存不够而失败。

3. 超参配置conf不需要设置类别数，这是通过数据集推断出来的。
4. 函数create_model完成头部网络定义，输出要求是一个字典，其中包括'loss'和'infer_out'两个key，'loss'是网络损失，'infer_out'是预测输出。如果修改有误可能会导致训练失败。
5. 在分类任务中，请保证infer_out的Tensor形状是BatchSize x ClassNumber
6. 在检测任务中，请保证infer_out的LodTensor形状是 BoxNumber x 6，其中每行是(label, confidence, xmin, ymin, xmax, ymax)
7. conf中如果有可供配置的模型结构参数，请阅读相关注释或链接后再行修改，随意设置可能会导致训练失败或训练精度异常。

查看训练结果

在脚本调参任务训练完成后，可以查看任务结果。

1. 在BML左侧导航栏中单击“脚本调参”，进入脚本调参列表页面。
2. 在脚本调参表单中单击“任务列表”，可以进入其任务列表页面，如下所示：



3. 在任务列表中，单击评估报告，可以查看该任务生成的模型的报告信息，如下所示：



开发参考

视觉预训练模型

图像分类模型

图像分类网络包含了基于飞桨深度学习平台模型库的十四种预训练网络，利用海量数据进行预训练，并且在ImageNet-2012数据集上进行了top-1准确率和cpu测试时间的测试。详细测试数据

预训练模型	预训练网络	top-1准确率	cpu测试时间(ms)	模型特点
公开数据集常规预训练模型	EffcientNetB0_small	0.751	129	去掉SE模块的EffcientNetB0，精度略有下降，速度大幅上升
	EfficientNetB4	0.8285	415	基于EffcientNetB0增加了网络的复杂度，预测准确率和时间也相应提高
	MobileNetV2	0.7215	--	轻量级网络，预测时间很短，但是精度也相应降低
	MobileNetV3_large_x1_0	0.753	96	MobileNet的最新版，同样为轻量化网络，但性能得到了进一步提升
	NASNet	0.827	--	如果数据量不大，由于模型复杂度较大，一般情况不建议使用
	ResNet18_vd	0.7226	142	轻量级ResNet，适用于要求小体积模型的情况，预测时间短，但准确率也相应较低
	ResNet50	0.765	--	模型效果稳定，并且预测时间较短，适用于大多数情况
	ResNet101	0.7756	--	在处理较大数据量时有更高的准确率，预测时间相比ResNet50也会更长
	ResNeXt101_32x16d_wsl	0.8424	1838	基于超大量图片的弱监督预训练模型，准确率高但是预测时间较长
	Res2Net101_vd_26w_4s	0.8064	575	在单个残差块内进一步构造了分层的残差类连接，比ResNet101准确度更高
	SE_ResNet18_vd	0.7333	152	比ResNet18_vd增加了SE模块，提高了准确率，但是预测时间也稍有增加
	SE-ResNeXt50	0.7844	--	与ResNet50相仿，准确率和预测时间稍有增加，但同样适用于大多数情况
	SE-ResNeXt101	0.7912	--	比ResNet101准确度更高，同时预测时间也更长
	Xception71	0.8111	747	采用了深度可分离卷积，使模型预测的准确率得到了很大的提升
百度超大规模数据集通用分类预训练模型	MobileNetV3_large_x1_0	-	-	使用百度自有大规模数据集的预训练网络，对比公开数据集的预训练模型具有更高的精度
	ResNet50_vd	-	-	使用百度自有大规模数据集的预训练网络，对比公开数据集的预训练模型具有更高的精度
	ResNet101_vd	-	-	使用百度自有大规模数据集的预训练网络，对比公开数据集的预训练模型具有更高的精度

物体检测模型 物体检测网络包含了基于飞桨深度学习平台模型库的七种预训练网络。使用COCO17数据集进行box AP和cpu测试时间的测试。详细测试数据

预训练模型	预训练网络	box AP	cpu测试时间(ms)	模型特点
公开数据集常规预训练模型	SSD-MobileNetV1	25.1	54	轻量级网络，重视预测速度时优先使用
	YOLOv3-MobileNetV1	23.9	171	轻量高速，重视预测速度时优先使用
	YOLOv3-DarkNet	38.9	514	模型在预测速度和模型精度上都有居中表现，适用于一般情况
	YOLOv3-ResNet50vd-DCN	41.4	456	以ResNet50_vd作为backbone的YOLOv3模型，预测精度高
	Faster_R-CNN-ResNet50-FPN	37.7	2842	预测精度高，训练时间长，适用于重视模型精度的使用情况
	RetinaNet-ResNet50-FPN	36	2308	经典的两阶段检测器，预测精度高，训练时间长，适用于重视模型精度的使用情况
	Cascade_RCNN-RestNet50-FPN	40.9	4454	经典的两阶段检测器，预测精度高，训练时间长，适用于重视模型精度的使用情况
百度超大规模数据集通用检测预训练模型	YOLOv3-DarkNet	-	-	使用百度自有大规模数据集的预训练网络，对比公开数据集的预训练模型具有更高的精度
	Faster_R-CNN-ResNet50-FPN	-	-	使用百度自有大规模数据集的预训练网络，对比公开数据集的预训练模型具有更高的精度

实例分割模型 实例分割网络包含了基于飞桨深度学习平台模型库的两种预训练网络。

预训练模型	预训练网络	模型特点
公开数据集常规预训练模型	Mask_RCNN-ResNet50-VD-FPN	预测精度较高，训练收敛快，适用于重视模型效果和速度均衡的使用情况
	Cascade_Mask_RCNN-RestNet50-FPN	预测精度高，适用于重视模型精度的使用情况

网络选型参考

图像分类任务网络选型参考

- 大多数情况建议使用 ResNet50和SE-ResNeXt50，模型效果稳定，并且预测时间较短。SE-ResNeXt50相比ResNet50增加了SE模块，能进一步提高准确率，但是预测时间稍有增加
- 当数据量较大并且要求更高的准确率时，建议从 ResNet101，ResNeXt101_32x16d_wsl，Xception71，Res2Net101_vd_26w_4s，SE-ResNeXt101，EffcientNetB4 中选择模型，预测时间相比ResNet50会更长
- 如果要求预测时间或模型体积尽量小时，建议使用 MobileNetV2，MobileNetV3_large_xi_0，ResNet18_vd，EffcientNetB0_small，SE_ResNet18_vd，
- 如果数据量不大，由于模型复杂度较大，一般情况不建议使用NASNet

物体检测任务网络选型参考

- 如果重视预测速度，优先使用 SSD-MobileNetV1，YOLOv3-MobileNetV1
- 如果重视模型精度，优先使用 YOLOv3-ResNet50vd_DCN，Faster_R-CNN-ResNet50-FPN，RetinaNet-ResNet50-FPN，Cascade Faster-ResNet50-FPN
- 如果在预测速度和模型精度上折中，可以使用 YOLOv3-DarkNet

实例分割任务网络选型参考

- 大多数情况建议使用 Mask_RCNN-ResNet50-VD-FPN，模型效果稳定，预测时间适中
- 当重视模型精度时，优先使用 Cascade_Mask_RCNN-RestNet50-FPN

数据增强算子参考

适用于图像分类的数据增强算子

算子名	功能
ShearX	剪切图像的水平边
ShearY	剪切图像的垂直边
TranslateX	按指定距离（像素点个数）水平移动图像
TranslateY	按指定距离（像素点个数）垂直移动图像
Rotate	按指定角度旋转图像
AutoContrast	自动优化图像对比度
Contrast	调整图像对比度
Invert	将图像转换为反色图像
Equalize	将图像转换为灰色值均匀分布的图像
Solarize	为图像中指定阈值之上的所有像素值取反
Posterize	减少每个颜色通道的bits至指定位数
Color	调整图像颜色平衡
Brightness	调整图像亮度
Sharpness	调整图像清晰度
Cutout	通过随机遮挡增加模型鲁棒性，可设定遮挡区域的长宽比例

适用于物体检测的数据增强算子

算子名	功能
ShearX_BBox	剪切图像的水平边
ShearX_Only_BBoxes	剪切标注框内图像的水平边
ShearY_BBox	剪切图像的垂直边
ShearY_Only_BBoxes	剪切标注框内图像的垂直边
TranslateX_BBox	按指定距离（像素点个数）水平移动图像及标注框
TranslateX_Only_BBoxes	按指定距离（像素点个数）水平移动标注框内的图像
TranslateY_BBox	按指定距离（像素点个数）垂直移动图像及标注框
TranslateY_Only_BBoxes	按指定距离（像素点个数）垂直移动标注框内的图像
Rotate_BBox	按指定角度旋转图像及标注框
Rotate_Only_BBoxes	按指定角度旋转标注框内的图像
AutoContrast	自动优化图像对比度
Contrast	调整图像对比度
Equalize	将图像转换为灰色值均匀分布的图像
Equalize_Only_BBoxes	将标注框内的图像转换为灰色值均匀分布的图像
Solarize	为图像中指定阈值之上的所有像素值取反
Solarize_Only_BBoxes	为标注框内的图像中指定阈值之上的所有像素值取反
Solarize_add	为图像中指定阈值之下的所有像素值加上像素偏移值
Posterize	减少每个颜色通道的bits至指定位数
Color	调整图像颜色平衡
Brightness	调整图像亮度
Sharpness	调整图像清晰度
Cutout	通过随机遮挡增加模型鲁棒性，可设定遮挡区域的长宽比例
BBox_Cutout	通过在标注框附近进行随机遮挡增加模型鲁棒性，可设定遮挡区域的长宽比例
Cutout_Only_BBoxes	只在标注框内通过随机遮挡增加模型鲁棒性，可设定遮挡区域的长宽比例
Flip_Only_BBoxes	对标注框内的图像进行左右翻转

超参数配置参考

超参来源 目前BML脚本调参任务类型支持三种方法配置任务中网络的超参数，你可以沿用「脚本编辑」中设定的超参数，为获取更高的模型精度，也可以选择「自动超参搜索」对网络的超参数进行搜索。选择「已有超参搜索结果」时，可以在预训练模型和预训练网络相同的任务中，复用自动超参搜索结果，高效地训练出高精度模型。 **自动超参搜索** 在超参搜索的过程中，由于模型复杂、计算成本很高，且每个超参数都有较大的取值范围，导致搜索空间十分庞大，因此需要利用自动超参搜索。自动超参搜索使用超参搜索策略，消耗相对较长的时间对网络模型进行自动调参。目前BML共支持三种超参搜索策略，详细配置方式请参考技术文档

超参搜索策略 | 策略名称 | 说明 | | ---------- | ---------- | | 随机搜索 | 在参数变量的变化区间中随机采样参数组合成候选集，应用候选集进行训练和效果对比。 | | 贝叶斯搜索 |贝叶斯搜索在搜索空间中随机选取初始超参数点，然后根据已有超参对应的指标结果拟合概率模型，通过概率模型推测最佳超参点，然后再试验得到这些超参点的结果。如此反复优化，在有限试验次数中搜索出合适的超参数。基于模型的序贯优化方法(SMBO, Sequential Model-Based Optimization)是贝叶斯搜索的一种范式，包括两个部分: 代理模型(surrogate model) 和 采集函数(acquisition function)。根据代理模型和采集函数的不同，贝叶斯搜索方法也有许多实现形式，其中TPE(Tree-structured Parzen Estimator)是一种全局探索能力较佳的方法，采用核密度估计方法(KDE, Kernel Density Estimation)生成代理模型，采用EI(Expected Improvement)作为其采集函数生成新采样点。 | |进化算法 |进化算法是基于种群概念的超参搜索策略，把超参配置视为一个种群，并行优化多个种群并在种群内部进行优胜劣汰的筛选，最终输出最佳模型。这个过程是从遗传算法获得的灵感，种群的初始化采用随机的方式生成，个体的优胜劣汰具体指利用(exploit)和探索(explore)两个步骤，不仅可能会从表现较好的个体中复制参数，它还能通过随机扰动修正当前的值而探索新的超参数组合。百度创新提出了随机微分方程无梯度优化算法PSHE2，采用哈密尔顿动力系统搜索参数空间中“势能”最低的点以替代随机扰动，加速迭代收敛。|

搜索策略比较

搜索策略	优点	缺点
随机搜索	多参数联合搜索时，增大了每个参数的搜索范围，提高搜索效率。	效果不稳定
贝叶斯搜索	搜索迭代次数少，效率高，借助已有采样点通过贝叶斯估计最佳超参。	可能陷入局部最优
进化算法	降低每轮迭代训练时间，将计算资源集中于参数搜索上，通过复用历史表现好的个体的预训练参数来加快收敛	如果超参导致模型结构变化等因素导致不能加载预训练参数时则不适用

如何选择超参搜索策略:

随机搜索：普适的效率较高的搜索方法，通常作为基线标准。适用于对**效率要求较高**的情况
贝叶斯搜索：利用已有参数信息推测最优点，是一种效率较高的黑盒优化算法。效果优于随机搜索，**适用于一般情况**。
进化算法：**效果较好**的算法，适合直接在自动超参搜索中过程中输出最佳模型，它的搜索结果应用到新的训练时可能导致效果下降

自动超参搜索配置参考

在深度学习建模过程中，除了大规模的数据集，超参数调节也显著影响模型效果。即使对于有经验的算法工程师，有时候也很难把握调节超参数的规律，而对于初学者来说，要花更多的时间和精力。
自动超参搜索应用超参搜索策略，对模型的超参数进行自动调优，在配置训练任务的页面，完成数据集的配置后，可以选择使用自动超参搜索进行配置。
您可以选择「随机搜索」，「贝叶斯搜索」，「进化算法」作为搜索策略进行超参搜索，所有默认值都是基于试验得出的效果较好的取值。



随机搜索 作为基线标准，不需要更多额外的设置，就能高效地进行超参数搜索。 **贝叶斯搜索** 贝叶斯搜索中，您需要设置初始点数量和最大并发量.



「初始点数量」:代表贝叶斯搜索中，初始化时参数点的数量，该算法基于这些参数信息推测最优点 ，填写范围1-20;
「最大并发量」:贝叶斯搜索中，同时进行试验的数量，并发量越大，搜索效率越高，填写范围1-20。

进化算法

进化算法是一种效果较好的算法，应用此算法时也需要进行较多的选项设置



「迭代轮数」:进化算法运行中迭代的轮数，范围5-50；
「扰动间隔」:进化算法每隔几个epoch就会进行随机扰动，利用随机因素防止算法结果收敛于局部最优解；
「扰动比例」：类似于染色体交叉的形式，迭代中一个种群内最好与最坏的个体依据扰动比例进行交叉；
「随机初始化概率」：在扰动中，有一定概率对个体的超参数进行初始化；
「种群个体数量」：一个个体代表一种超参数设置，一个种群中包含多个个体。

超参范围设置 超参范围设置中，可以设置需要搜索的超参数和搜索范围。您可以在「默认配置」，「手动配置」中进行选择。 **默认配置** 如果您不行进行超参数选择和范围设置，可以选用默认配置，后台会根据所选网络默认配置必要的超参数进行搜索。 **手动配置** 针对不同的预训练网络，我们提供了丰富的超参数供开发者手动配置，每一个超参数都能选择数据类型和设置搜索范围。



详细超参数列表请见下面超参数列表 **搜索超参数列表** |超参数|参数类型|说明| |--|--|--| |batch_size|离散值、随机整数|每一个批次处理的数据数量，需根据网络不同机型和网络调整最大值，否则可能因显存不足导致失败| |learning_rate|离散值、平均采样、对数平均采样|控制深度学习网络的学习速度，学习率越低，损失函数的变化速度就越慢，反之亦然| |weight_decay|离散值、平均采样、对数平均采样|对深度学习网络进行权重衰减，防止网络出现过拟合情况| |mix_up|布尔值|选择是否使用mix_up策略，mix_up是一种数据增广方式| |label_smoothing|布尔值|选择是否使用label_smoothing策略，label_smoothing是一种正则化方法| |anchor_scale|离散值，随机整数|anchor大小的缩放尺度，anchor是一组预设的参考框| |anchor_ratio|离散值，随机整数|anchor的长宽比，anchor是一组预设的参考框|

通用配置项 『最大搜索次数』：是指最多组合出多少组超参并跑试验，当然有可能会因为提前达到目标而停止，节约费用。
『数据采样比例』：使用超参搜索时，会对原始数据集进行采样后再训练，加快搜索速度。当数据集并不大时，不推荐采样哟，可能会影响最终效果，只有大数据量时才有使用采样的必要。
『最高mAP/最高准确率』：是指大家期望模型效果可以达到的mAP（物体检测）或准确率（图像分类）的值，**当试验中达到这个值了搜索就会停止，避免后续浪费搜索时间。**

训练时长设置参考

运行环境说明

目前 BML 支持选择GPU P4、GPU P40、GPU V100三种运行环境，性能从高到低为V100＞P40＞P4。具体规格说明如下：

机型	规格说明
GPU V100	TeslaGPU_V100_16G显存单卡_12核CPU_56G内存
GPU P40	TeslaGPU_P40_24G显存单卡_12核CPU_40G内存
GPU P4	TeslaGPU_P4_8G显存单卡_12核CPU_40G内存

温馨提示：

- 未开通付费的情况下，可选的运行环境为GPU P4,我们为每位用户提供了GPU P4运行环境下100（小时*节点）免费算力支持，超出后请您付费购买。详见页面提示的价格说明。
- 图像分类NASNet网络暂时不支持在 GPU P4环境上运行，若您未开通付费，请选择其他网络。

如何设置节点数及运行时间

1、选择计算节点数：节点数越多运行效率越高，可结合训练数据量及运行效率灵活选择。目前可设置的最高计算节点数为6.

2、最长训练时间：超过最长训练时间后模型会自动停止运行，注意这里设置的最长训练时间与最长计费时长有一定差异，训练计费时长将根据数据增强-自动搜索耗时+训练耗时的总时长进行计算。

注意：实际训练时长与所选机型、节点数、网络、数据量均有关系，其中性能上V100>P40>P4，在同一网络前提下，训练机型新能越高、节点数越多、数据量越少训练速度会越快。

根据历史经验，不同数据量范围、网络的大致耗时范围如下，供辅助参考

图像分类

网络及设置	数据量范围	训练耗时预估	
ResNet50（batch size=16）	1-1.5w	2节点预估12-14min左右；4节点11min左右	
	4-6k	5min左右，10min以内,多节点或单节点不太影响训练时长	
	1k 以下	2-3min左右,多节点或单节点不太影响训练时长	

物体检测

网络及设置	数据量范围	训练耗时预估	
Faster_R-CNN-ResNet50-FPN	200-400之间	2节点预估十几-二十分钟左右；4节点预估在十分钟以内	
	1800-2000	2节点预估在60min左右，4节点预估在40min左右	

以下为典型网络在不同数据量、不同机型、不同节点数的实际测试训练耗时，供辅助参考。

图像分类

网络及数据量	机型	节点数	训练耗时
ResNet50（数据量6000+）	V100	1	400s
	V100	2	326s
	V100	6	198s
	P40	1	867s
	P40	2	562s
	P40	6	540s
	P4	1	1385s
	P4	2	751s
	P4	6	302s
MobileNet（数据量6000+）	V100	1	259s
	V100	2	197s
	V100	6	161s
	P40	1	396s
	P40	2	285s
	P40	6	333s
	P4	1	630s
	P4	2	370s
	P4	6	178s

物体检测

网络及数据量	机型	节点数	训练耗时
Faster_R-CNN-ResNet50-FPN（数据量400+）20epoch	V100	1	29.1min
	V100	2	16.1min
	V100	6	6min
	P40	1	40min
	P40	2	21.8min
	P40	6	11.8min
	P4	1	66min
	P4	2	34.4min
	P4	6	13.28min
SSD-MobileNetV1（数据量400+）20epoch	V100	1	19min
	V100	2	10.3min
	V100	6	3.9min
	P40	1	28.1min
	P40	2	14.6min
	P40	6	4.8min
	P4	1	25.4min
	P4	2	13.4min
	P4	6	5.1min

评估报告参考

图像分类模型评估报告

模型评估报告内容说明

模型训练完成后我们可以在模型列表中看到模型效果及查看模型评估报告的入口。进入模型评估报告页面，我们可以看到整体报告内容中包含以

下几个区域内容：

模型基本信息

在这个部分可以选择应用类型、训练版本

整体评估

在这个部分可以看到模型训练整体的情况说明，包括基本结论、准确率、F1-score、精确率、召回率。这部分模型效果的结果内容是基于训练数据集，随机抽出部分数据不参与训练，仅参与模型效果评估计算得来。所以**当数据量较少时（如图片数量低于100个），参与评估的数据可能不超过30个图片，这样得出的模型评估报告效果仅供参考，无法完全准确体现模型效果。**

注意：若想要更充分了解模型效果情况，建议发布模型为API后，通过调用接口测试批量图片数据获取更准确的模型效果。

详细评估

在这个部分可以看到上述训练效果背后的原始评估数据。以及不同top结果的准确率效果情况，下面为相关名词解释。

图像分类名词解释

准确率

准确率含义为正确分类的样本数与总样本数之比，这里指的总样本是指**从总训练数据中随机抽取部分数据参与模型评估的总样本**，在上文截图中，参与训练的图片数200个，实际参与评估的数据为下面详细评估预测表现表格的数据总和，即50个。那么准确率为参与评估的正确数量46/50，结果为92.0%

F1-Score

F1-score是指对某类别而言为精确率和召回率的调和平均数，此处为各类别F1-score的平均数。 就某类而言，精确率和召回率体现了该分类的精确率及召回率的平衡情况：

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

精确率

精确率是对某类别而言为正确预测为该类别的样本数与预测为该类别的总样本数之比，此处为各类别精确率的平均数。如果精确率比较低，有一定可能原因为不同类别效果存在差异所致，请检查下不同类型样本量是否均衡。

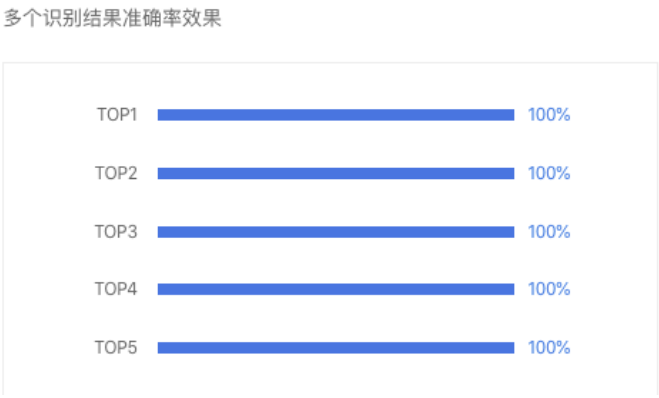
召回率

召回率是指对某类别而言为正确预测为该类别的样本数与该类别的总样本数之比，此处为各类别召回率的平均数。



top1、top2、top…5

是指对于每一个评估的图片文件，模型会给根据置信度高低，依次给出top1-top5的识别结果，其中top1置信度最高，top5的置信度最低。那么top1的准确率值是指对于评估标准为“top1结果识别为正确时，判定为正确”给出准确率。top2准确率值是指对于评估标准为“top1或者top2只要有一个命中正确的结果，即判定为正确”给出的准确率。……以此类推。

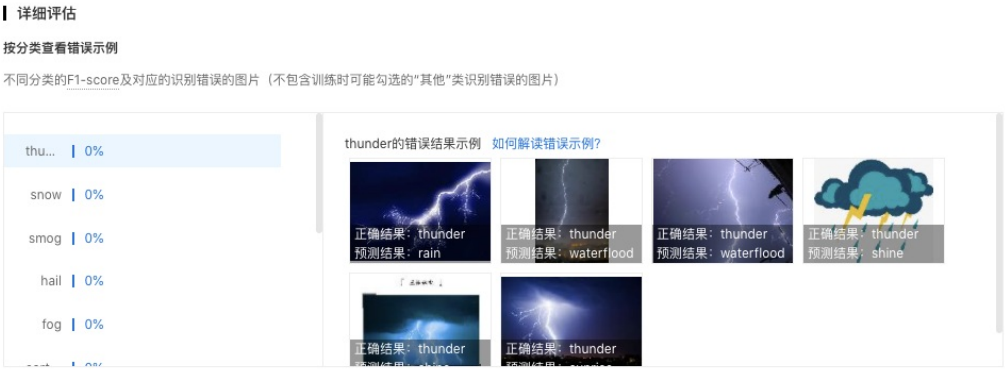


不同分类的F1-score 这个部分很重要，如果不同分类的F1-score存在差异很大的情况下，整体模型效果有被低精度的分类效果拉低的可能。建议检查训练数据中不同标签的目标数情况，**建议不同分类的目标数尽可能均衡。**



识别错误图片示例

通过分标签查看模型识别错误的图片，寻找其中的共性，进而有针对性的扩充训练数据。



例如，你训练了一个将小番茄和樱桃分类的模型。在查看小番茄分类的错误示例时，发现错误示例中有好几张图片都是带着绿色根茎的小番茄（与樱桃比较相似）。这种情况下，就需要在小番茄分类的训练集中，多增加一些带绿色根茎的图片，让模型有足够的数据能够学习到带根茎的小番茄和樱桃的区别。

这个例子中，我们找到的是识别错误的图片中，目标特征上的共性。除此之外，还可以观察识别错误的图片在以下维度是否有共性，比如：图片的拍摄设备、拍摄角度，图片的亮度、背景等等。



物体检测模型评估报告

模型评估报告内容说明

模型训练完成后我们可以在模型列表中看到模型效果及查看模型评估报告的入口。进入模型评估报告页面，我们可以看到整体报告内容中包含以下几个区域内容：

模型基本信息

在这个部分可以选择应用类型、训练版本

整体评估

在这个部分可以看到模型训练整体的情况说明，包括基本结论、准确率、F1-score、精确率、召回率。这部分模型效果的结果内容是基于训练数据集，随机抽出部分数据不参与训练，仅参与模型效果评估计算得来。所以**当数据量较少时（如图片数量低于100个），参与评估的数据可能不超过30个图片，这样得出的模型评估报告效果仅供参考，无法完全准确体现模型效果。**

注意：若想要更充分了解模型效果情况，建议发布模型为API后，通过调用接口测试批量图片数据获取更准确的模型效果。

物体检测名词解释

mAP

mAP(mean average precision)是物体检测(Object Detection)算法中衡量算法效果的指标。对于物体检测任务，每一类object都可以计算出其精确率(Precision)和召回率(Recall)，在不同阈值下多次计算/试验，每个类都可以得到一条P-R曲线，曲线下的面积就是average

F1-Score

F1-score是指对某类别而言为精确率和召回率的调和平均数，此处为各类别F1-score的平均数。 就某类而言，精确率和召回率体现了该分类的精确率及召回率的平衡情况：

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

精确率

经比较F1-score最高的阈值0.3下的结果。精确率为该阈值下正确预测的物体数与预测物体总数之比

召回率

经比较F1-score最高的阈值0.3下的结果。召回率为该阈值下正确预测的物体数与真实物体数之比



阈值

阈值，是正确结果的判定标准，例如阈值是0.6，置信度大于0.6的识别结果会被当作正确结果返回。每个物体检测模型训练完毕后会给出建议阈值，建议后续在模型校验及正式调用服务过程中，阈值参数按照模型评估报告中的建议阈值进行配置。



不同标签的平均精度

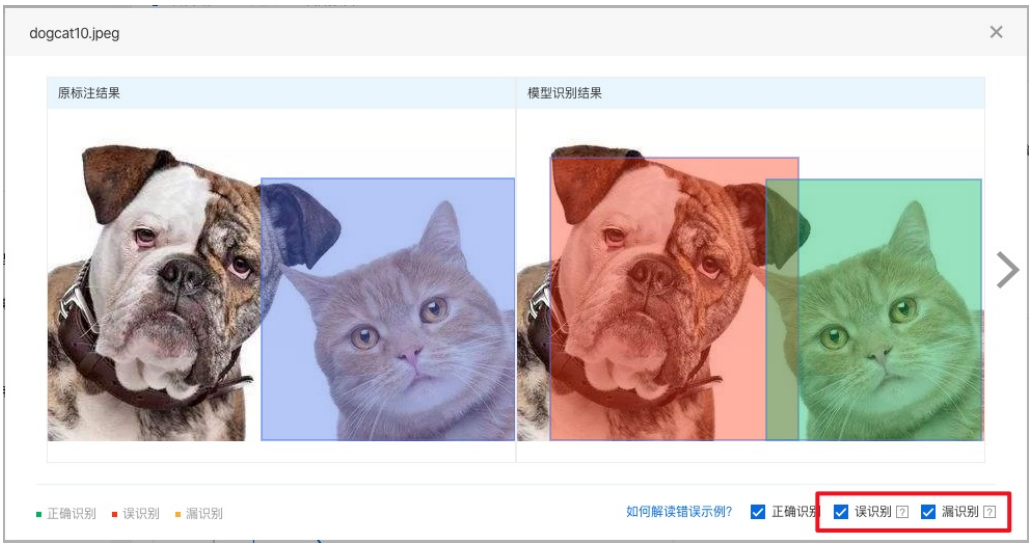
这个部分很重要，通过查看不同标签的平均精度可以有效观测不同标签的精度对比。如果存在精度差异很大的情况下，整体模型效果有被低精度的标签效果拉低的可能。建议检查训练数据中不同标签的目标数情况，**建议不同标签的目标数尽可能均衡。**



识别错误图片示例

通过分标签查看模型识别错误的图片，寻找其中的共性，进而有针对性的扩充训练数据。

如下图所示，可以通过勾选「误识别」、「漏识别」来分别查看两种错误识别的情况：



- **误识别：红框内没有目标物体（准备训练数据时没有标注），但模型识别到了目标物体**

观察误识别的目标有什么共性：例如，一个检测电动车的模型，把很多自行车误识别成了电动车（因为电动车和自行车外观上比较相似）。这时，就需要在训练集中为自行车特别建立一个标签，并且在所有训练集图片中，将自行车标注出来。

可以把模型想象成一个在认识世界的孩童，当你告诉他电动车和自行车分别是什么样时，他就能认出来；当你没有告诉他的时候，他就有可能把自行车认成电动车。

- **漏识别：橙框内应该有目标物体（准备训练数据时标注了），但模型没能识别出目标物体**

观察漏识别的目标有什么共性：例如，一个检测会议室参会人数的模型，会漏识别图片中出现的白色人种。这大概率是因为训练集中缺少白色人种的标注数据造成的。因此，需要在训练集中添加包含白色人种的图片，并将白色人种标注出来。

黄色人种和白色人种在外貌的差别上是比较明显的，由于几乎所有的训练数据都标注的是黄色人种，所以模型很可能认不出白色人种。需要增加白色人种的标注数据，让模型学习到黄色人种和白色人种都属于「参会人员」这个标签。

以上例子中，我们找到的是识别错误的图片中，目标特征上的共性。除此之外，还可以观察识别错误的图片在以下维度是否有共性，比如：图片的拍摄设备、拍摄角度，图片的亮度、背景等等。

实例分割模型评估报告

模型评估报告内容说明

模型训练完成后我们可以在模型列表中看到模型效果及查看模型评估报告的入口。进入模型评估报告页面，我们可以看到整体报告内容中包含以下几个区域内容：

模型基本信息

在这个部分可以选择应用类型、训练版本

整体评估

在这个部分可以看到模型训练整体的情况说明，包括基本结论、准确率、F1-score、精确率、召回率。这部分模型效果的结果内容是基于训练数据集，随机抽出部分数据不参与训练，仅参与模型效果评估计算得来。所以**当数据量较少时（如图片数量低于100个），参与评估的数据可能不超过30个图片，这样得出的模型评估报告效果仅供参考，无法完全准确体现模型效果。**

注意：若想要更充分了解模型效果情况，建议发布模型为API后，通过调用接口测试批量图片数据获取更准确的模型效果。

实例分割名词解释

mAP

mAP(mean average precision)是实例分割(Instance Segmentation)算法中衡量算法效果的指标。对于实例分割任务，每一类object都可以计算出其精确率(Precision)和召回率(Recall)，在不同阈值下多次计算/试验，每个类都可以得到一条P-R曲线，曲线下的面积就是average

F1-Score

F1-score是指对某类别而言为精确率和召回率的调和平均数，此处为各类别F1-score的平均数。 就某类而言，精确率和召回率体现了该分类的精确率及召回率的平衡情况：

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

精确率

经比较F1-score最高的阈值0.3下的结果。精确率为该阈值下正确预测的物体数与预测物体总数之比

召回率

经比较F1-score最高的阈值0.3下的结果。召回率为该阈值下正确预测的物体数与真实物体数之比



阈值

阈值，是正确结果的判定标准，例如阈值是0.6，置信度大于0.6的识别结果会被当作正确结果返回。每个实例分割模型训练完毕后会给出建议阈值，建议后续在模型校验及正式调用服务过程中，阈值参数按照模型评估报告中的建议阈值进行配置。



不同标签的平均精度

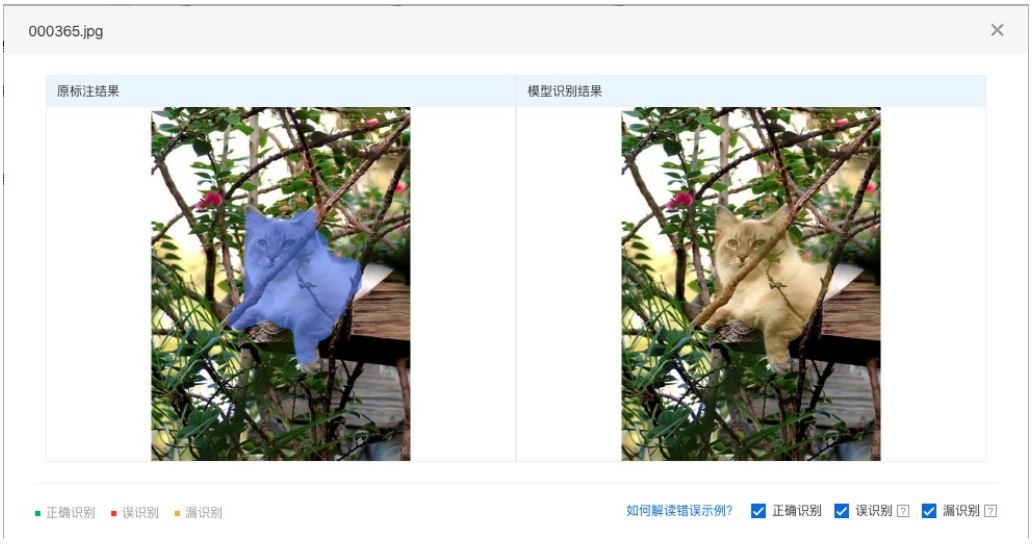
这个部分很重要，通过查看不同标签的平均精度可以有效观测不同标签的精度对比。如果存在精度差异很大的情况下，整体模型效果有被低精度的标签效果拉低的可能。建议检查训练数据中不同标签的目标数情况，**建议不同标签的目标数尽可能均衡。**



识别错误图片示例

通过分标签查看模型识别错误的图片，寻找其中的共性，进而有针对性的扩充训练数据。

如下图所示，可以通过勾选「误识别」、「漏识别」来分别查看两种错误识别的情况：



- **误识别：红色遮盖内没有目标物体（准备训练数据时没有标注），但模型识别到了目标物体**

观察误识别的目标有什么共性：例如，一个检测电动车的模型，把很多自行车误识别成了电动车（因为电动车和自行车外观上比较相似）。这时，就需要在训练集中为自行车特别建立一个标签，并且在所有训练集图片中，将自行车标注出来。

可以把模型想象成一个在认识世界的孩童，当你告诉他电动车和自行车分别是什么样时，他就能认出来；当你没有告诉他的时候，他就有可能把自行车认成电动车。

- **漏识别：橙色遮盖内应该有目标物体（准备训练数据时标注了），但模型没能识别出目标物体**

观察漏识别的目标有什么共性：例如，一个检测会议室参会人数的模型，会漏识别图片中出现的白色人种。这大概率是因为训练集中缺少白色人种的标注数据造成的。因此，需要在训练集中添加包含白色人种的图片，并将白色人种标注出来。

黄色人种和白色人种在外貌的差别上是比较明显的，由于几乎所有的训练数据都标注的是黄色人种，所以模型很可能认不出白色人种。需要增加白色人种的标注数据，让模型学习到黄色人种和白色人种都属于「参会人员」这个标签。

以上例子中，我们找到的是识别错误的图片中，目标特征上的共性。除此之外，还可以观察识别错误的图片在以下维度是否有共性，比如：图片的拍摄设备、拍摄角度，图片的亮度、背景等等。

开发自然语言处理模型

自然语言处理任务简介

EasyDL专业版平台将支持代码级的NLP任务包括文本分类任务、短文本匹配任务、序列标注任务、文本实体抽取任务。同时平台集成了百度大脑文心的ERNIE2.0预训练模型，可以选择ERNIE-Base、ERNIE-Large、ERNIE-Tiny三种类型的预训练模型。

预训练模型ERNIE

ERNIE 是一个从海量无监督数据中学习大量知识的超大预训练模型，通过将 ERNIE 模型中的这些知识迁移到上层特定任务相关的网络从而显著提升任务效果；基于 ERNIE 来 Finetune 特定任务就好比是站在巨人的肩膀上看世界，起点会显著高于从 0 开始训练特定任务的深度学习模型；您可以通过ERNIE官网查看最新动态和相关信息。

不同版本的ERNIE特点

- Base 模型的网路结构主体由 12 层 transformer block 构成，此版本的预测精度、预测性能、训练速度相对均衡

- Large模型拥有更大的参数量，以及更好的 Finetune 效果，Large模型网络结构主体由 24 层Transformer Block构成，此版本的预测精度突出
- Tiny模型有更浅的模型，在实际真实数据中有超过4倍的预测提速，预测精度逼近Base版

文本分类

文本分类任务中，平台提供的网络有：BOW、CNN、GRU、TextCNN、LSTM、BiLSTM； 同时，平台提供了BOW、CNN、GRU、TextCNN、LSTM、BiLSTM的ERNIE预训练模型版本的网络；

网络特点描述

- BOW：词袋模型，不考虑语法和语序，用一组无序单词来表达一段文本
- CNN：卷积神经网络，通过卷积核的权重共享减少参数数量，能够处理变长的序列输入
- GRU：门控循环单元网络，能够较好地处理序列文本中长距离依赖的问题
- TextCNN：该模型可以自主设定多组不同窗口的CNN模型
- LSTM：长短期记忆网络，能够较好地处理序列文本中长距离依赖问题
- BiLSTM：双向长短期记忆网络，能够更好的捕捉句子中的语义特征

短文本语义匹配

短文本匹配任务中，平台提供的网络有：SimNet-BOW、SimNet-CNN、SimNet-GRU、SimNet-LSTM、FC；

同时，ERNIE预训练模型可选FC网络；

网络特点描述

- FC：基础句对匹配模型，可以任意搭配其他各种经典网络
- SimNet-CNN：卷积神经网络，能够处理变长的序列输入，提取一个局部区域之内的特征
- SimNet-BOW：词袋模型，不考虑语法和语序，用一组无序单词来表达一段文本
- SimNet-GRU：门控循环单元网络，能够较好地处理序列文本中长距离依赖的问题
- SimNet-LSTM：长短期记忆网络，能够较好地处理序列文本中长距离依赖问题

序列标注

序列标注任务中，平台提供的网络为CRF； ERNIE预训练模型可选CRF网络；

网络特点描述

- CRF：条件随机场，能够考虑上下文关联信息的进行模型学习

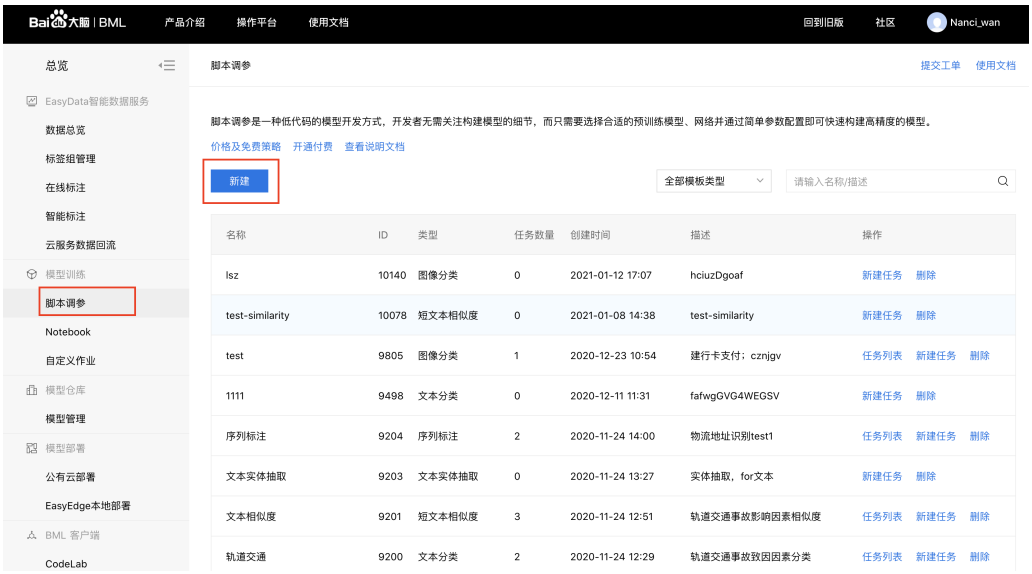
文本实体抽取

文本实体抽取基于序列标注任务，平台提供CRF网络，使用预训练模型ERNIE时，也可接入CRF网络；

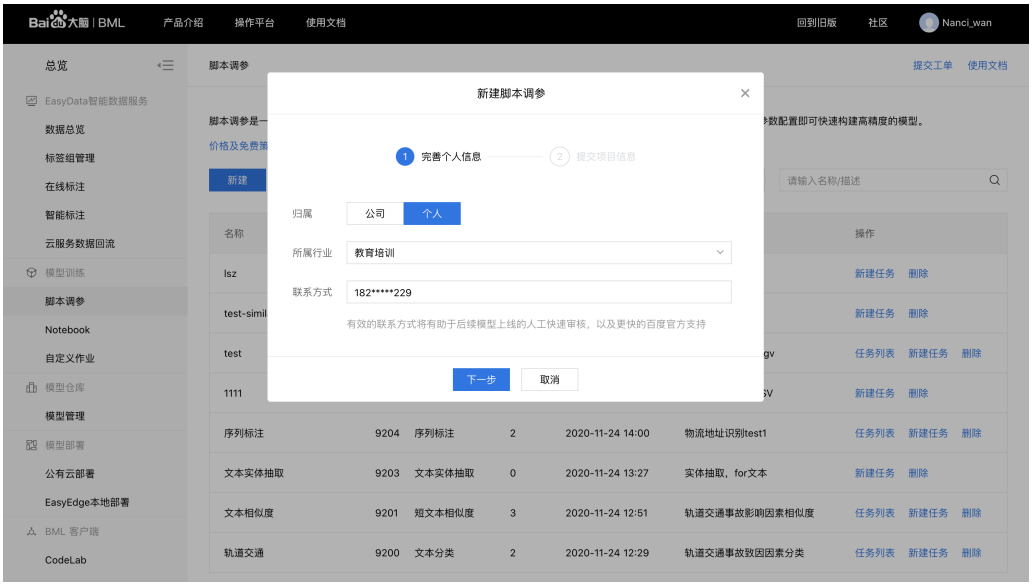
创建NLP任务

一个项目是指对应的一个场景或者领域的问题，例如智能客服场景、智能教育场景等。在BML中，提供了文本分类单标签、文本分类多标签、短文本相似度、序列标注、文本实体抽取任务的模型训练能力。本章节将通过演示，如何创建一个自然语言处理任务来完成问文本分类的任务。

1. 在BML的控制面板左侧，打开【预制模型调参】，并点击内容主板的“新建”按钮：



2. 在弹出的个人信息表中，填写对应的个人信息，有效的个人信息，将有助于百度对项目信息进行及时的人工审核，获得更有效的支持。完成信息填写后进入下一步：



3. 对项目的任务信息需要进行编辑，完成编辑后进入下一步：



4. 完成项目创建，便能够在项目列表中查看，并可进行“新建任务”操作



配置NLP任务

1. 在新建任务的面板中，可查看相关项目“基本信息”、“配置任务类型”、“添加数据”、“配置任务”等操作，如下图：



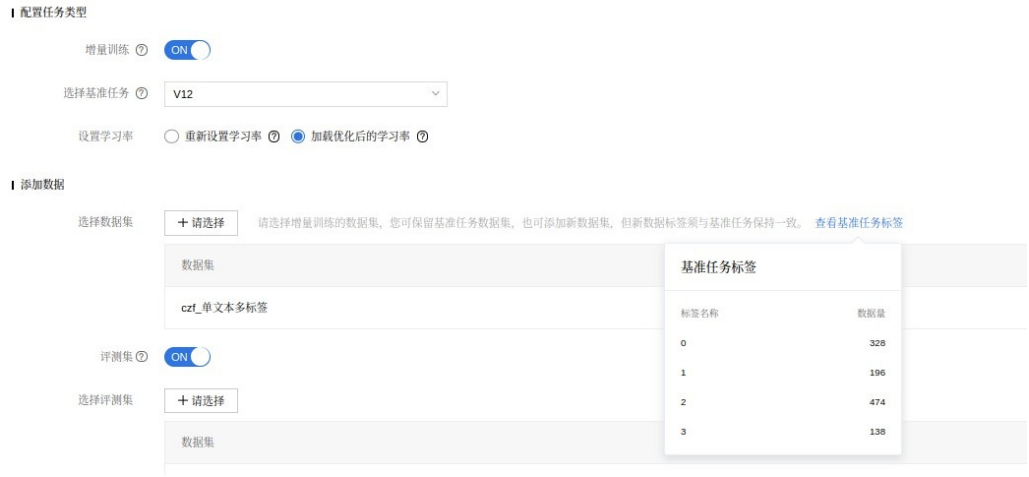
配置任务类型时，您可以看到增量训练有一个开关选项。增量训练也成为模型热启动，当您训练完模型但效果不理想想继续添加数据，或您的训练模型超过训练时长被中断是，您可以开启增量训练，使该新任务能在基准任务的基础上继续运行。在配置任务类型上您需要注意以下几点，如您是第一次新建任务，增量训练的开关将默认关闭，不允许打开。如您在满足已有训练完成或是训练中断的任务的情况下，您可以开启增量训练的开关，选中您想继续的基准任务版本。在设置学习率时，您可以重新设置学习率，也可以加载优化后的学习率。如您选择重新设置学习率，您可以在下方【配置网络】的脚本编辑中修改学习率的大小，如您选择加载优化后的学习率，您将直接继承您选中的基准任务的超参数，在脚本编辑中将不允许再修改。



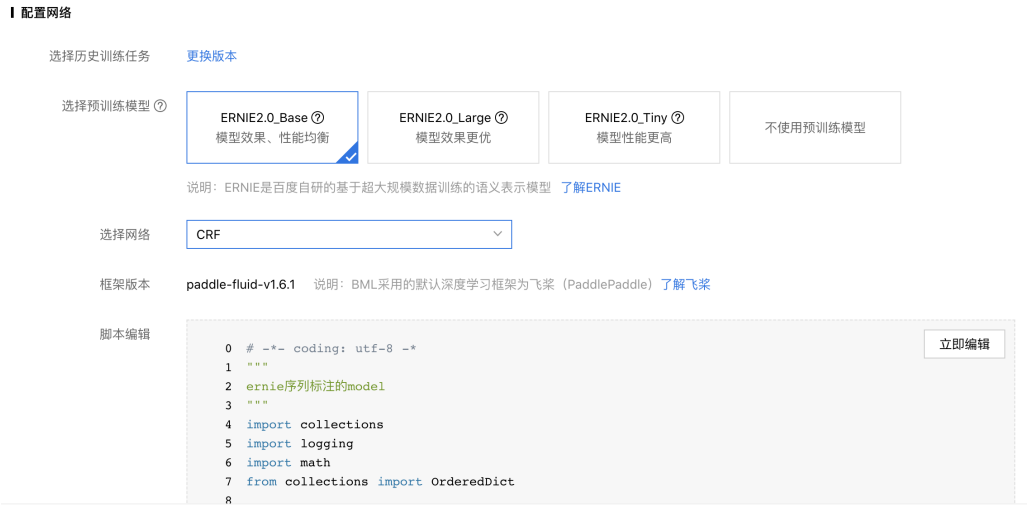
训练任务，我们需要添加模型训练使用的训练集和模型评测的评测集，如果您没有特别创建评测集，可以将评测集开关设置为“OFF”，系统将从训练集中取出一定比例的数据作为评测集。如果您有准备评测集，则可以将评测集开关设置为“ON”，并上传对应的数据集即可，如下图：



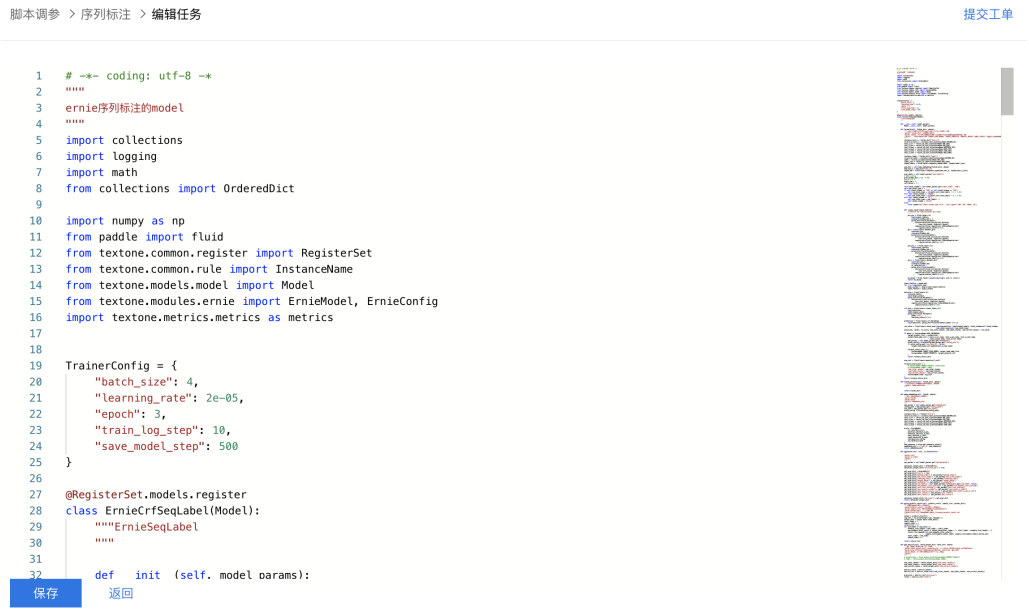
在上传数据的时候，您需要注意，如您打开了增量训练的开关，那么将直接选中您选中的基准任务的使用的数据集。点击查看基准任务标签时，您可以看到基准任务数据集的标签信息，此时您选择增量训练的数据集，您可保留基准任务数据集，也可添加新数据集，但新数据标签须与基准任务保持一致。



2. 配置网络过程中，您可以选择使用预训练模型ERNIE2.0对应的三个版本：ERNIE2.0_Base、ERNIE2.0_Large和ERNIE2.0_Tiny。本章节演示创建流程，先选择“ERNIE2.0_Tiny”，并选择TextCNN网络。 EasyDL专业版目前支持飞桨（Paddle Paddle）深度学习框架，自然语言训练任务支持的框架脚本为paddle-fluid-v1.6.1。了解飞桨



平台提供了脚本编辑的工具，您可以直接在平台点击“立即编辑”进行脚本编辑，如下图：

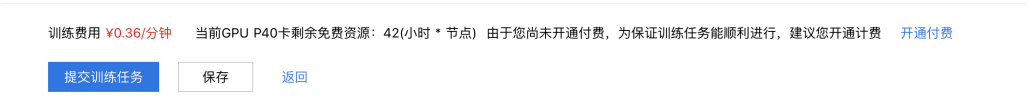


注意，当您打开增量开关后，如您选中重新设置学习率，那么您可以脚本编辑中设置学习率，此时，如您修改其他超参数，仍将默认基准任务的超参数。如您选中加载优化后的学习率将脚本将不允许编辑。目前平台为您提供了对应代码的注释，方便您了解如何进行代码编辑和修改。

3.在配置资源部分，目前平台已提供GPU V100、P40两种机型。选中您所需的机型，根据需要设置计算节点数，如您选择多节点进行分布式训练，可有效提升训练速度，节点数越多训练效率越高。在最长训练设置部分，目前文本任务方向没有时间上线，建议您尽量设置较长时长，注意如您使用ERNIE预训练模型，建议训练时长不低于4小时。



4. 点击提交训练任务，任务将在平台提供的GPU训练资源上进行训练。



提交成功的任务，将会在项目列表中展示状态。



在此，NLP任务的创建，已经完成。

错误码

如果您的任务出现失败情况，您可以通过查看平台反馈的错误码来查询，如下表：

错误码	训练失败原因说明
01-05、07、08、10、11	后端服务超时或训练失败，请提交工单或加入EasyDL官方QQ群(群号:86882600)联系群管解决
6	数据量过少导致训练失败，请补充数据重新提交模型训练
9	代码错误导致训练失败，请检查脚本代码内容修改后重新提交模型训练，有疑问请提交工单或加入EasyDL官方QQ群(群号:868826008)联系群管解决

查看训练结果

1. 当平台跑完NLP任务后，您可以点击“模型效果”进行查看



2. 您可以选择对应任务的不同版本，查看不同的效果数据，如图，当前是V1版本的效果数据：



效果校验

平台提供了针对模型的校验工具，您可以通过上传文本，了解如何模型的识别效果，从而对模型进行校验

文本分类模型校验

您可以上传文本文件（txt格式），也可以直接在输入框中输入文本即可校验，见下图：



短文本匹配模型校验

您可输入两个文本，点击”测试相似度“，即可获得两个文本的语义相似度的score值。



序列标注模型校验

您可输入一段文本，注意，如果您训练使用的是ERNIE模型，则需要注意输入文本需要进行切词（可以空格间隔）；如果您训练使用的非ERNIE模型，请参考您的训练数据集的分词词典，对文本分词后再输入文本矿中进行校验。



日志

您可以通过查看任务的日志内容，确认任务训练过程中的细节，



代码模板升级及迁移说明

一、代码模板升级整体说明

平台于2020年4月16日，对自然语言处理模型训练的代码模板进行了升级，建议您在5月1日前完成对旧版任务的迁移。本次升级，平台的自然语言处理任务将为您提供封装性和灵活性俱佳的开发体验，并支持了飞桨深度学习框架Paddle-Fluid 1.6.1的版本。

针对平台升级新版本模型代码框架，请您关注如下几个时间点：

1. 平台于4月16日，更新上线新版代码框架，暂时保留对旧版任务的克隆功能
2. 平台于5月1日，将不再提供旧版本代码模板的克隆。当您对旧版代码模板克隆任务时，平台仅提供新版代码框架，请您做好代码的迁移工作。

说明：您在5月1日前，基于旧版本代码模板发布的任务，可长期在线上进行云端调用或本地使用。

后续平台也将基于升级后框架，继续新增任务类型，您可以加入BML用户交流群，针对您在模型开发过程中的需要，提出您所需的功能需求。

二、如何操作迁移到新版任务下

将老版本任务迁移到新版本的任务，以分类任务为例可分为三步：

第一步：查看老版本的任务中，代码编辑部分，存在哪些字段的修改

在任务列表中，找到对应的任务详情：



在代码配置详情中，找到脚本内容部分，查看修改的代码内容：



第二步：在平台「自然语言处理」任务列表下，新建项目并新建任务，此任务将会是基于最新的代码模板创建



第三部：在下图中代码编辑部分，将第一步中对应修改过的代码部分，在新任务中进行逐一调整（关于新版代码模板的说明，详见下文）



三、新老版本的代码编辑模板区别说明

您可以通过了解旧版本的代码模板和新版本的代码模板的代码结构，来了解两版代码的编写逻辑，从而方便您将旧版本的代码迁移到新版本的代码模板中。我们将以文本分类任务中，使用ERNIE和BOW网络的设计进行讲解说明。

下面我们先看一下旧版本的代码编辑模板的内容：

3.1 旧版代码模板说明

1. 模型训练的配置信息

关于模型训练过程中，关于学习率、batch size等配置，代码如下：

```
def __init__(self):
    Config.__init__(self)
    self.batch_size = 8  # batch_size大小,尽量设置成2的整数倍
    self.learning_rate = 2e-5  # 学习率设置
    self.save_steps = 500  # 模型保存的间隔，即训练多少个batch之后保存一次模型，请设置成100的倍数
    self.epoch = 3  # 训练多少轮
    self.skip_steps = 10  # 间隔多少个batch时打印训练日志
```

组网部分，包括了ERNIE表示、BOW模型表示和分类任务几个部分，代码介绍如下：

2. 网络设计中ERNIE表示的代码部分

```
emb = fluid.layers.sequence_unpad(ernie_output["sequence_output"], length=ernie_input["seq_lens"])
```

3. 网络设计中BOW的模型表示代码部分

```
# bow layer
bow = fluid.layers.sequence_pool(input=emb, pool_type='sum')
bow_tanh = fluid.layers.tanh(bow)
# full connect layer
fc_1 = fluid.layers.fc(input=bow_tanh, size=hid_dim, act="tanh")
fc_2 = fluid.layers.fc(input=fc_1, size=hid_dim2, act="tanh")
```

4. 网络设计中分类器的代码部分

```
logits = fluid.layers.fc(input=fc_2, size=cfg.num_labels)
```

5. 网络设计中预测部分的内容

```
if is_inference:
    probs = fluid.layers.softmax(logits)
    feed_targets_name = [
        ernie_input["src_ids"].name, ernie_input["sent_ids"].name,
        ernie_input["pos_ids"].name, ernie_input["input_mask"].name,
        ernie_input["seq_lens"].name, ernie_input["task_ids"].name
    ]
    return feed_targets_name, probs
```

6. 网络设计中评估部分的代码

```
ce_loss, probs = fluid.layers.softmax_with_cross_entropy(
    logits=logits, label=label, return_softmax=True)
loss = fluid.layers.mean(x=ce_loss)
graph_vars = {
    "loss": loss,
    "classify_infer": probs,
    "label": label
}
```

3.2 新版代码模板说明

下文中，我们将介绍新升级的代码模板结构，您可以对比新旧版本对应的结构来进行代码的迁移：

1. 新版本代码结构：模型训练的配置信息

```
TrainerConfig = {
  "batch_size": 8,
  "learning_rate": 2e-05,
  "epoch": 3,
  "train_log_step": 10,
  "save_model_step": 500
}
```

2. 新版本代码结构：数据读入的部分

```
fields_dict = self.fields_process(fields_dict, phase)
    instance_text_a = fields_dict["text_a"]
    record_id_text_a = instance_text_a[InstanceName.RECORD_ID]
    text_a_src = record_id_text_a[InstanceName.SRC_IDS]
    text_a_pos = record_id_text_a[InstanceName.POS_IDS]
    text_a_sent = record_id_text_a[InstanceName.SENTENCE_IDS]
    text_a_mask = record_id_text_a[InstanceName.MASK_IDS]
    text_a_task = record_id_text_a[InstanceName.TASK_IDS]
    text_a_lens = record_id_text_a[InstanceName.SEQ_LENS]
    instance_label = fields_dict["label"]
    record_id_label = instance_label[InstanceName.RECORD_ID]
    label = record_id_label[InstanceName.SRC_IDS]
```

3. 新版本代码结构：读取ERNIE表示

```
    emb_dict = self.make_embedding(fields_dict, phase)
    emb = emb_dict["text_a"]
    text_a_emb = fluid.layers.sequence_unpad(emb, length=text_a_lens)
```

4. 新版本代码：模型超参数的设置

```
    hid_dim = 128
    hid_dim2 = 96
    num_labels = 2
```

5. 新版本代码：BOW模型表示部分

```
# bow layer
    bow = fluid.layers.sequence_pool(input=text_a_emb, pool_type='sum')
    bow_tanh = fluid.layers.tanh(bow)
```

6. 新版本代码：分类器部分的代码

```
    fc_1 = fluid.layers.fc(input=bow_tanh, size=hid_dim, act="tanh")
    fc_2 = fluid.layers.fc(input=fc_1, size=hid_dim2, act="tanh")
    logits = fluid.layers.fc(input=fc_2, size=num_labels)
```

7. 新版本代码：预测部分的内容

```
if phase == InstanceName.SAVE_INFERENCE:
        """保存模型时需要的入参：表示预测时最终输出的结果"""
        probs = fluid.layers.softmax(logits)
        target_predict_list = [probs]
        target_feed_name_list = [text_a_src.name, text_a_pos.name, text_a_sent.name,
                                 text_a_mask.name, text_a_lens.name]
        emb_params = self.model_params.get("embedding")
        ernie_config = ErnieConfig(emb_params.get("config_path"))
        if ernie_config.get('use_task_id', False):
            target_feed_name_list.append(text_a_task.name)

        forward_return_dict = {
            InstanceName.TARGET_FEED_NAMES: target_feed_name_list,
            InstanceName.TARGET_PREDICTS: target_predict_list
        }
        return forward_return_dict
```

8. 新版本代码：loss部分的设置

```
    cost, probs = fluid.layers.softmax_with_cross_entropy(
        logits=logits, label=label, return_softmax=True)
    avg_cost = fluid.layers.mean(x=cost)
    """PREDICT_RESULT,LABEL,LOSS 是关键字，必须要赋值并返回"""
```

开发表格预测模型

表格预测任务简介

脚本调参支持多种表格预测任务，训练得到的模型可以应用到广泛实际场景中。

表格预测模型类型

表格预测帮助用户通过机器学习技术发现已有数据中的规律，从而创建机器学习模型，并基于机器学习模型处理新的数据，为业务应用生成预测结果。脚本调参中提供的表格数据预测支持创建回归、二分类、多分类三种类型的机器学习模型。

- 回归：目标列是连续的实数范围，或者属于某一段连续的实数区间。如在销量预测场景中，销量值可能是某个取值范围内的任意值，解决该问题的模型属于回归模型。
- 二分类：目标列是离散值，且只有两种可能的取值。如在精准营销场景中预测一个用户是否为潜在购买用户，其目标列仅存在“True”和“False”两种取值，解决该问题的模型属于二分类模型。
- 多分类：目标列是离散值，并具有有限的可能取值。如在用户分类场景中，根据用户的历史消费数据，将用户划分到不同消费偏好的类别中，解决该问题的模型属于多分类模型。

表格预测模型应用场景

- 精准营销：从客户消费记录中挖掘客户群的共有特征，分析出客户的购物偏好，从而实现广告的精准投放
- 信用评分：金融公司分析客户的历史行为数据，建立用户信用模型，从而确定贷款额度等
- 价格预测：从历史数据中发现商品的变化规律以及影响价格的因素，从而为未来的商业行为提供支持
- 客户流失预测：根据客户历史数据获得数据挖掘模型，从而生成客户流失预测列表，为市场营销策略提供有价值的业务洞察

创建表格预测任务

在任务列表点击【创建项目】，在弹窗中提交以下信息完成项目创建：

1. **完善个人信息**：填写项目归属、行业、联系方式完成项目创建。
   注意：有效的联系方式将有助于后续模型上线的人工快速审核，以及更快的百度官方支持



2. **提交项目信息**:提交模型名称、技术方向、任务类型、应用场景及功能描述，即可完成项目创建。其中**任务类型与配置任务可选的数据类型一一对应**。



配置AutoML模式表格数据预测任务

表格预测任务支持AutoML和专家两种运行方式：

- AutoML模式：全流程自动建模，用户只需设置数据集、目标列以及制定任务类型即可，而无需关注数据处理以及算法配置等过程，系统会自动完成建模过程，并从中挑选最优的模型作为训练任务的运行结果。
- 专家模式：高度开放的建模方式，用户可以进行特征工程、算法、超参搜索等配置，具备相关技能的开发者可以在方式下获得更多的开发自由

度。

创建AutoML建模任务

操作场景

以iris数据集为例，创建多分类模型，iris数据集示例如下：

```
 sepal_length,sepal_width,petal_length,petal_width,species
5.1,3.5,1.4,0.2,setosa
4.9,3.0,1.4,0.2,setosa
4.7,3.2,1.3,0.2,setosa
4.6,3.1,1.5,0.2,setosa
```

前提条件

在创建表格预测任务前，需满足如下条件：

- 必须已成功创建“表格”类数据，数据集导入状态为“已完成”
- 数据集中行数必须大于0，即必须是非空数据集

操作步骤

1. 在BML操作台的左侧导航菜单上单击“脚本调参”，进入脚本调参列表页面。
2. 单击已创建的“iris分类”所在行的“新建任务”，进入“新建任务”页面。

   配置参数如下所示：

   - 基本信息
     - 开发方式：选择AutoML模式
     - 任务备注：请根据实际情况填写
   - 添加数据
     - 选择数据集：选择已创建的iris数据集
     - 选择目标列：设置为species
     - 算法类型：可以设置为“自动”或“多分类”
   - 发布模型
     - 自动发布-开：即完成训练后，系统会自动将当前任务得到的模型发布到模型仓库中
     - 自动发布-关：完成训练后，用户可以根据模型精度等再决定是否将模型发布到模型仓库
   - 配置资源
     - 运行环境：请根据数据量以及期望的运行速度进行设置。根据经验值，在建模过程中，数据会在内存中膨胀为原始大小的10倍，为保证任务顺利完成，请尽量确保配置的资源的内存不小于原始数据集的10倍。
     - 选择节点数：如果设置为1，则系统以单机算法进行建模，如果设置为大于1的值，则系统将使用分布式算法。单机算法比分布式算法更丰富。在单机资源满足要求的情况下，可优先使用单节点进行建模。
     - 最长训练时间：该时长指算法求解阶段的最长时长，若超过该时长，算法仍未得到结果，系统会强制结束训练任务。

   配置示例如下所示：

   - 基本信息部分：



- 添加数据部分：



3. 单击"提交训练任务"，开始执行模型训练。

   提交任务后可以在任务列表中查看任务的执行状态。

配置专家模式表格数据预测任务

表格预测任务支持AutoML和专家两种运行方式：

- AutoML模式：全流程自动建模，用户只需设置数据集、目标列以及制定任务类型即可，而无需关注数据处理以及算法配置等过程，系统会自动完成建模过程，并从中挑选最优的模型作为训练任务的运行结果。
- 专家模式：高度开放的建模方式，用户可以进行特征工程、算法、超参搜索等配置，具备相关技能的开发者可以在方式下获得更多的开发自由度。 **创建专家建模任务**

操作场景

以iris数据集为例，创建多分类模型，iris数据集示例如下：

```
 sepal_length,sepal_width,petal_length,petal_width,species
5.1,3.5,1.4,0.2,setosa
4.9,3.0,1.4,0.2,setosa
4.7,3.2,1.3,0.2,setosa
4.6,3.1,1.5,0.2,setosa
```

前提条件

在创建表格预测任务前，需满足如下条件：

- 必须已成功创建"表格"类数据，数据集导入状态为"已完成"
- 数据集中行数必须大于0，即必须是非空数据集

操作步骤

1. 在BML操作台的左侧导航菜单上单击"脚本调参"，进入脚本调参列表页面。
2. 单击已创建的"iris分类"所在行的"新建任务"，进入"新建任务"页面。

   配置参数如下所示：

- 基本信息
  - 开发方式：选择专家模式
  - 训练方式：支持单机和分布式两种模式，单击模式支持更多的算法。
  - 任务备注：请根据实际情况填写，详细的配置说明可以参考初始化脚本头部的注释内容。
- 添加数据
  - 选择数据集：选择已创建的iris数据集
  - 选择目标列：设置为species
  - 算法类型：设置为“多分类”
  - 特征工程策略：
    - 默认配置：执行系统默认的特征工程策略
    - 手动配置：用户可以手工配置各个特征工程算子的执行参数从而控制其执行方式
    - 自动搜索：用户可以选择要使用那些特征工程算子，系统会自动搜索并完成用户选择的特征工程算子的配置。
- 配置任务
  - 系统会根据用户选择的训练方式以及算法类型自动生成任务脚本，在不需要修改的情况下可直接启动训练。
  - **自定义脚本内容过程中有如下注意事项：**
    1. 可以自定义的部分为超参配置字典conf部分,包括是否开启超参搜索，训练模型类型和模型配置，具体见脚本中的注释
    2. 训练默认开启自动超参搜索，如需关闭请手动将"hyperparameter_tune"的参数值由"True"更改为"False"
    3. BML当前表格数据预测支持模型为CAT(CatBoost), LGBM(LightGBM),RF(RandomForest), LR(Logistic Regression), XGB(Xgboost), KNN(k-NearestNeighbor)
    4. 在"hp_space"中已经预置可搜索的超参数，用户无需修改参数名称，随意设置可能会导致训练失败
    5. 在超参搜索范围内的取值方式支持：平均采样(uniform),非平均采样(quniform), 离散值(choice), 对数平均采样(loguniform), 随机整数(randint)五种，超参搜索范围设置过大可能会导致训练时间过长。
    6. CAT模型支持搜索的超参数，已经预置默认搜索范围，详见脚本内容

超参数	说明
depth	决策树的深度
iterations	最大树数
learning_rate	学习率，控制机器学习网络的学习速度，学习率越低，损失函数的 变化速度就越慢，反之亦然
l2_leaf_reg	L2正则项，防止模型过拟合
border_count	numerical features的分割数

    7. LGBM模型支持搜索的超参数，已经预置默认搜索范围，详见脚本内容

超参数	说明
num_leaves	决策树的叶子数
boosting_type	boosting参数的类型，默认包含"rf", "gbdt", "dart", "goss"
n_estimators	对原始数据集进行有放回抽样生成的子数据集个数，即决策树的个 数，n_estimators太小，容易欠拟合，n_estimators太大，计算量会太大
max_depth	决策树最大深度
subsample	每棵树随机采样的比例
learning_rate	学习率
colsample_bytree	每棵随机采样的列数的占比(每一列是一个特征)
reg_alpha	L1正则项，防止模型过拟合
reg_lambda	L2正则项，防止模型过拟合

8. RF模型支持搜索的超参数，已经预置默认搜索范围，详见脚本内容

超参数	说明
n_estimators	对原始数据集进行有放回抽样生成的子数据集个数，即决策树的个 数，n_estimators太小，容易欠拟合，n_estimators太大，计算量会太大
max_features	随机森林允许单个决策树使用特征的最大数量
max_depth	决策树最大深度
min_samples_split	节点可分的最小样本数，少于该数值次叶子节点不再可分
min_samples_leaf	叶子节点上应有的最少样例数，样例数量不符合则不能构成一个叶子节点
bootstrap	是否对样本集进行有放回抽样来构建树

- 发布模型
  - 自动发布-开：即完成训练后，系统会自动将当前任务得到的模型发布到模型仓库中
  - 自动发布-关：完成训练后，用户可以根据模型精度等再决定是否将模型发布到模型仓库
- 配置资源
  - 运行环境：请根据数据量以及期望的运行速度进行设置。根据经验值，在建模过程中，数据会在内存中膨胀为原始大小的10倍，为保证任务顺利完成，请尽量确保配置的资源的内存不小于原始数据集的10倍。
  - 选择节点数：只有选择的"分布式"的训练方式才可以设置为大于1的值。
  - 最长训练时间：该时长指算法求解阶段的最长时长，若超过该时长，算法仍未得到结果，系统会强制结束训练任务。

配置示例如下所示：

- 基本信息部分：
- 

- 添加数据部分：



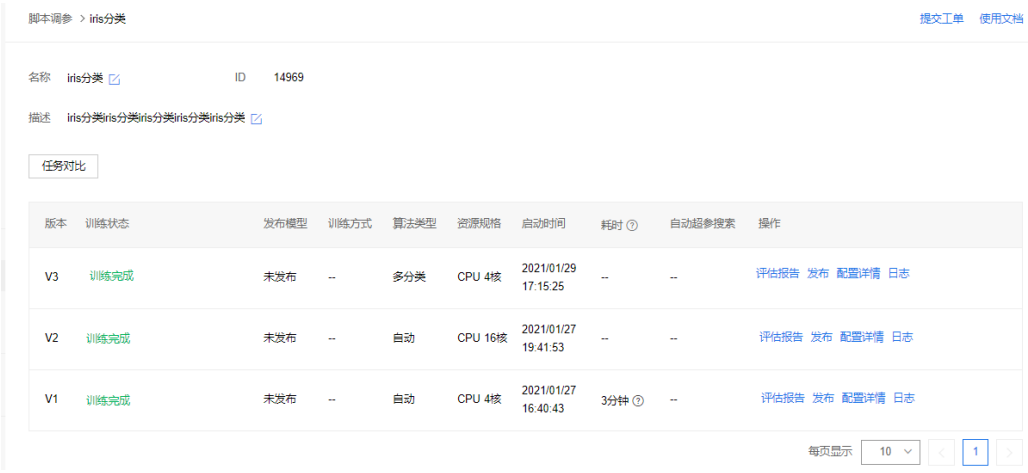
3. 单击“提交训练任务”，开始执行模型训练。

   提交任务后可以在任务列表中查看任务的执行状态。

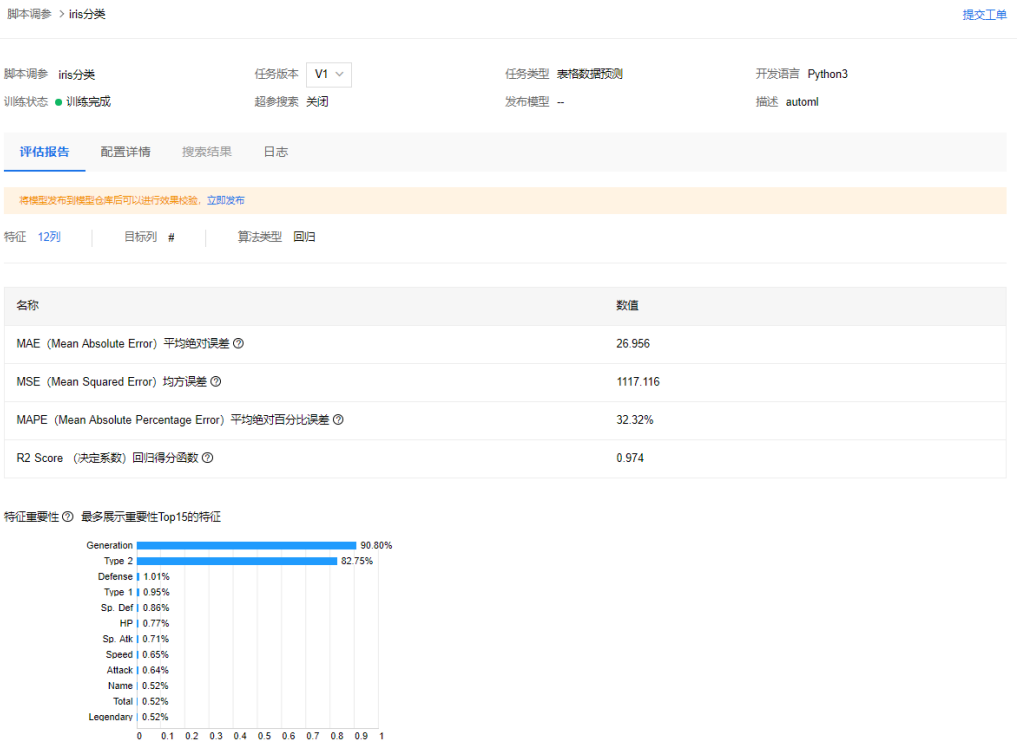
查看训练结果

在脚本调参任务训练完成后，可以查看任务结果。

1. 在BML左侧导航栏中单击“脚本调参”，进入脚本调参列表页面。
2. 在脚本调参表单中单击“任务列表”，可以进入其任务列表页面，如下所示：



3. 在任务列表中，单击评估报告，可以查看该任务生成的模型的报告信息，如下所示：



开发文字识别模型

文字识别任务简介

文字识别模型类型

文字识别模型即是常说的OCR模型，预置模型调参目前提供了通用的全文本识别场景，可以应对常规的文字识别任务，且支持多种文字。

文字识别模型应用场景

纸质文档电子化

通用文字识别模型支持针对多语种的纸质文档进行电子化，开发者可以采集文档图片并标注，对模型进行训练，从而实现纸质文档的自动电子化，提升工作效率。

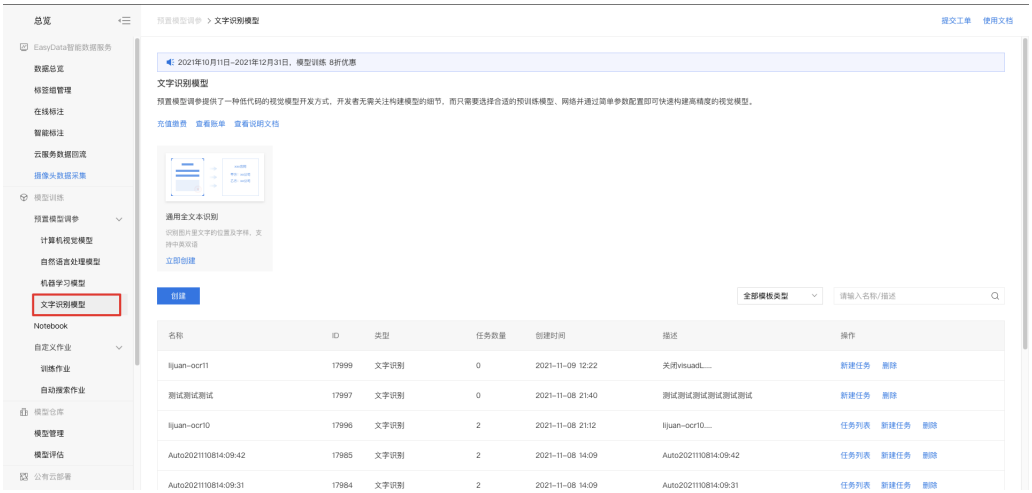
图像转文字

通用文字识别模型支持识别图像上的文字，开发者可以采集图片并标注，对模型进行训练，从而实现图像转文字的工作。

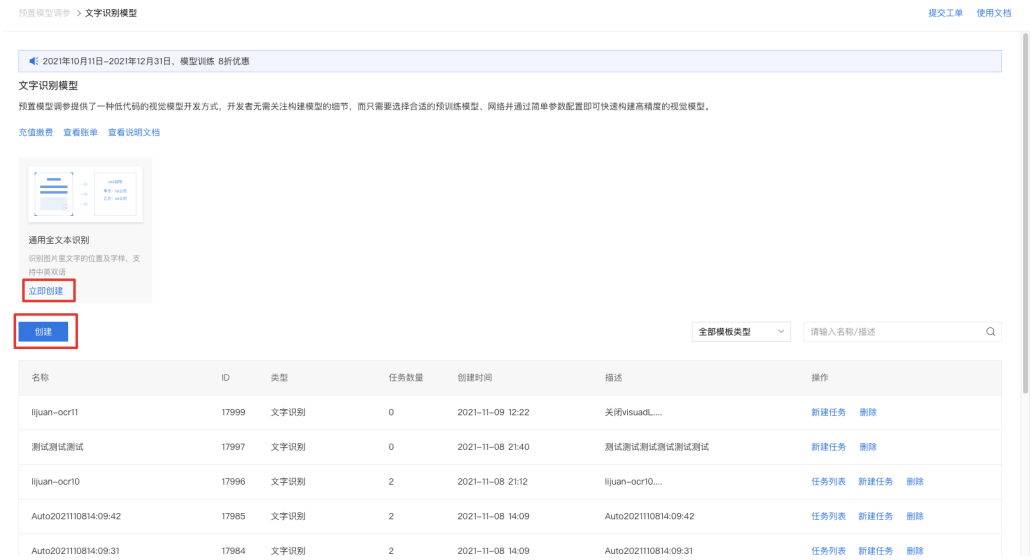
文字识别任务操作流程

文字识别任务操作流程

第一步：点击进入文字识别模型页面



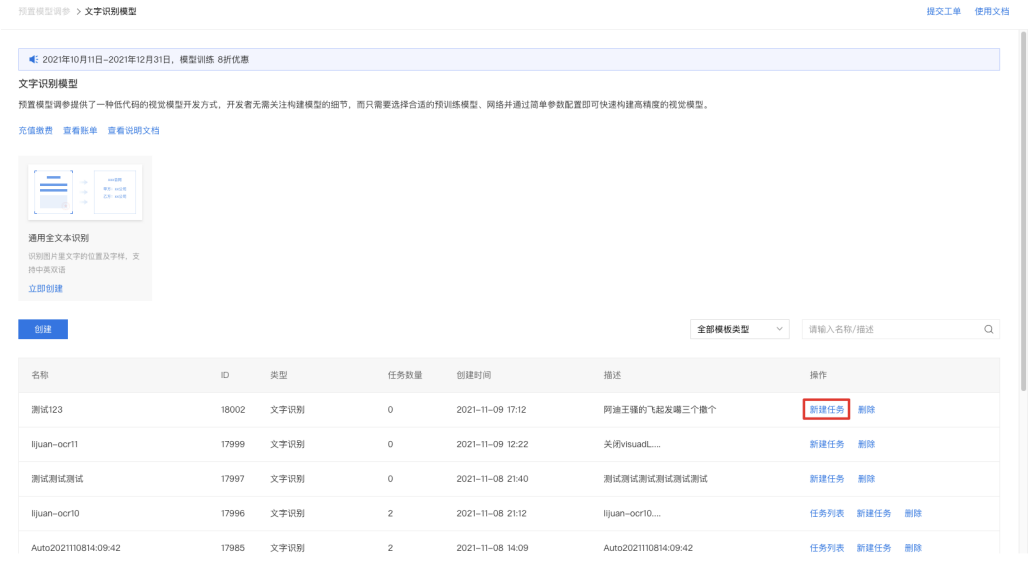
第二步：点击创建



第三步：填写项目基本信息

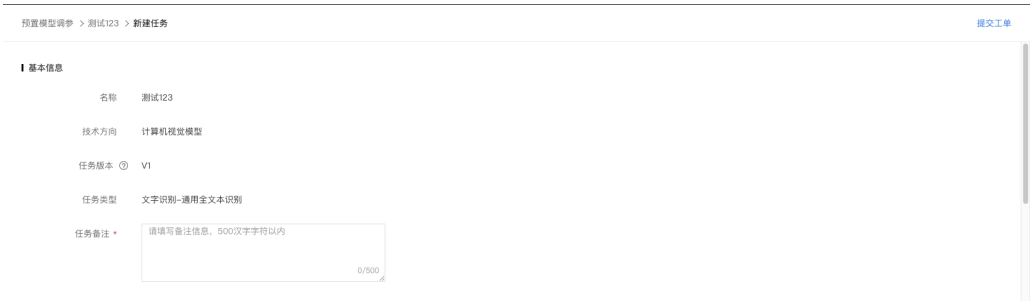


第四步：点击新建任务

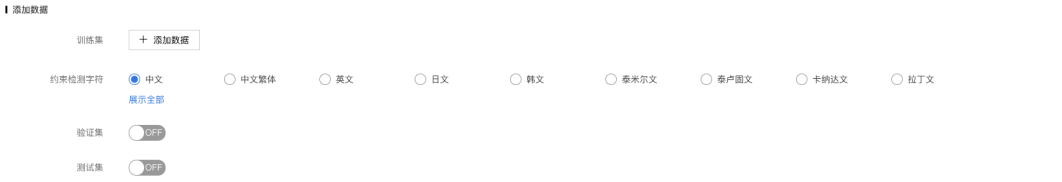


第五步：完善任务信息

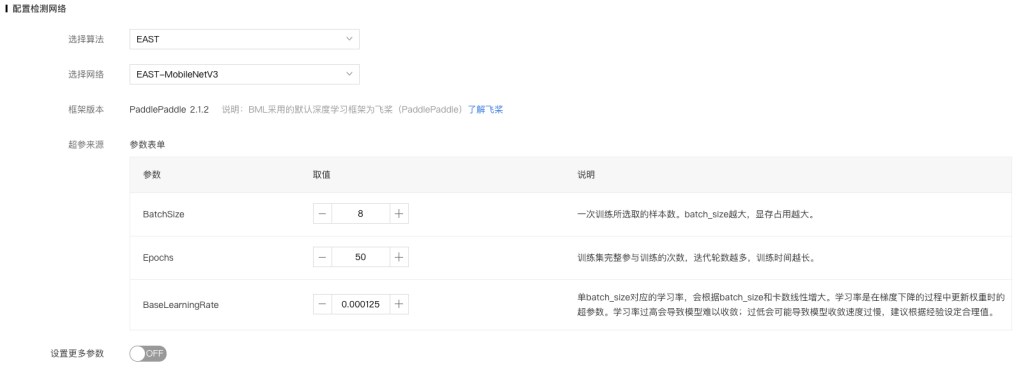
填写任务备注



添加数据



配置检测网络



配置识别网络

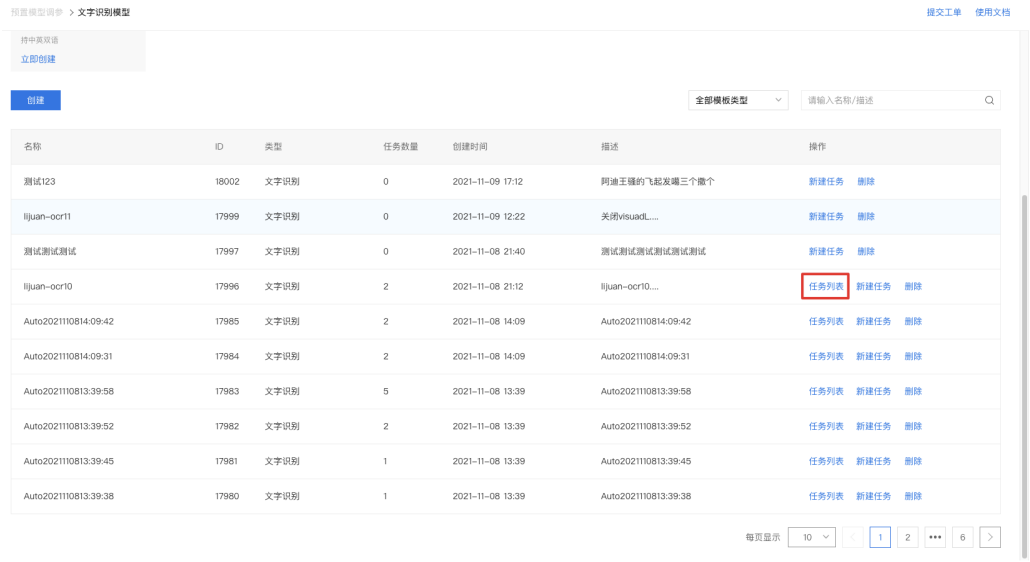


第六步：配置资源并提交训练

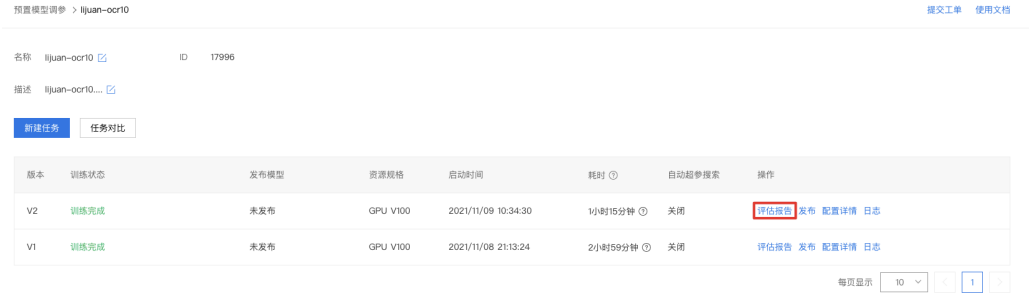


训练结果查看

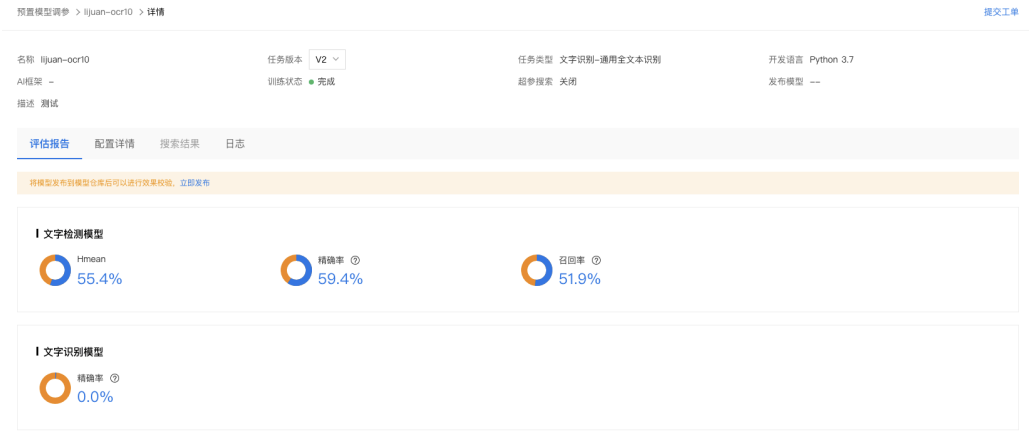
点击任务列表



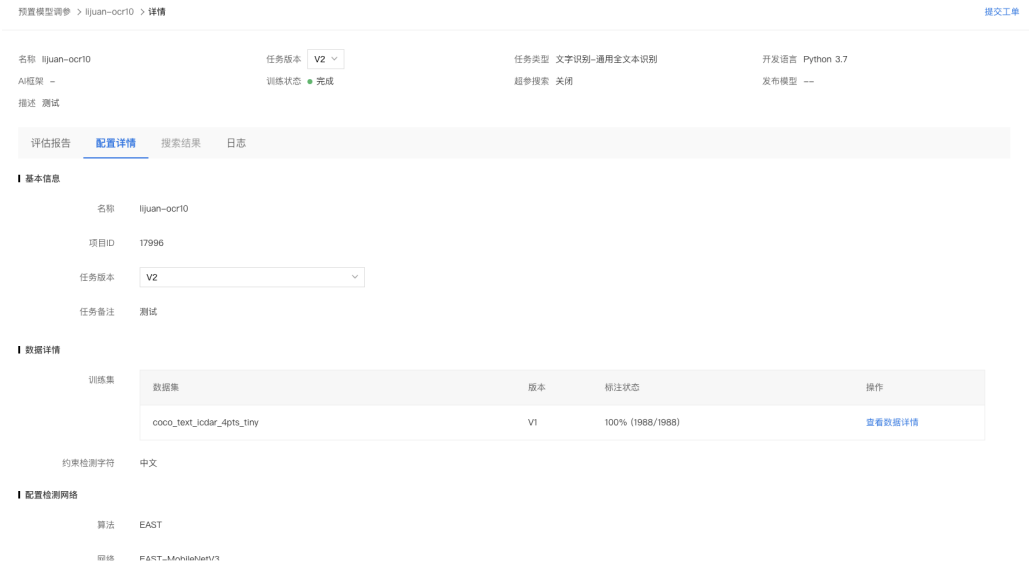
点击评估报告



查看评估结果



查看配置详情

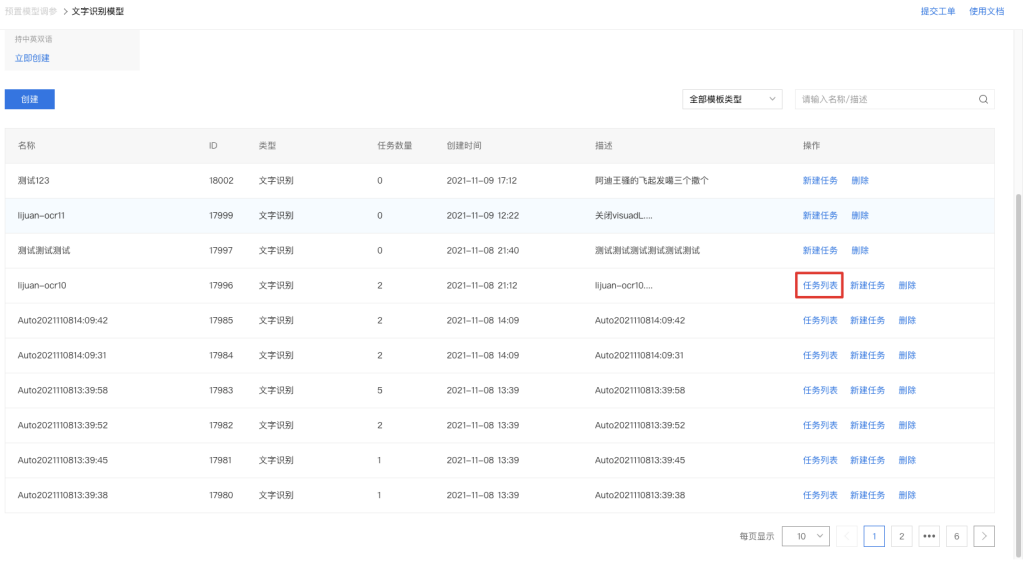


查看训练日志

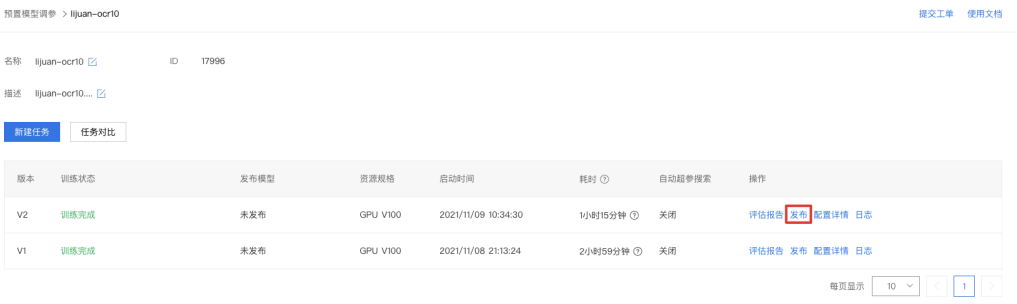


发布模型

点击任务列表



点击发布



神经网络训练搜索

了解神经网络训练搜索

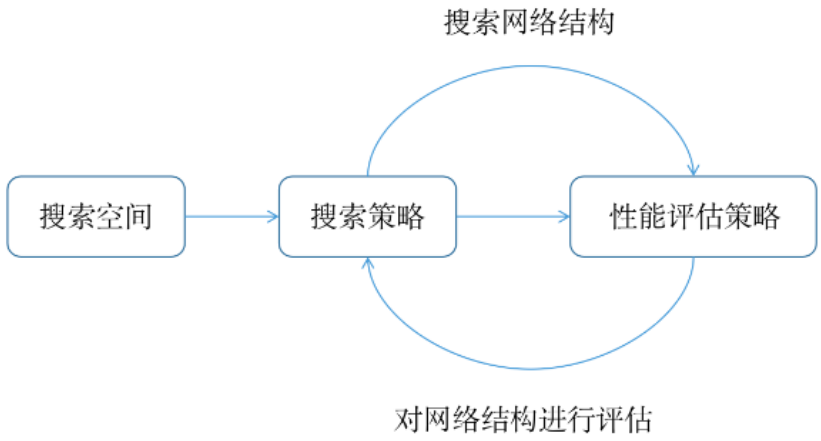
1 概述

神经网络训练搜索又名神经网络结构搜索 (Neural Architecture Search, NAS)，是自动机器学习 (Automated Machine Learning, AutoML) 的子领域。

神经网络训练搜索NAS本质上是优化问题，其目的是不通过人工，实现最优网络模型设计和选择自动化。使用者不需要具备丰富的专家经验，降低了网络模型设计难度和人力成本，有效保证业务应用效果的时延。

2 主要构成

NAS有三个基本要素：搜索空间、搜索策略和性能评估策略。

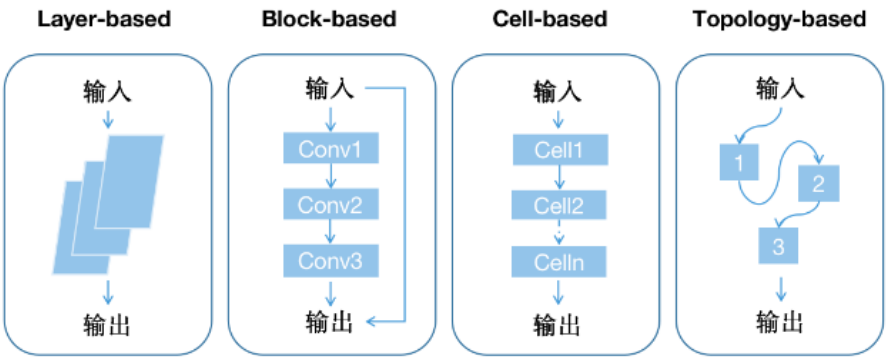


给定一个称为搜索空间的候选神经网络结构集合，用某种搜索策略从中搜索出最优网络结构。

神经网络结构的性能用某些指标如验证精度、计算开销等来度量，称为性能评估。

2.1 搜索空间 搜索空间可以看成是一个集合，集合中的每个元素代表一种候选网络结构。根据搜索空间的颗粒度不同，可以大概分成以下四类：

- Layer-based：关注的是每一层怎么设计。
- Block-based：关注的是每一块怎么设计，每个块中包含多个层，块里面每个层的设计是不一样的。
- Cell-based：关注的是每个cell怎么设计，每个cell中包含多个块，每个块的设计是不一样的。
- Topology-based：关注的是基本单元如何连接。



2.2 搜索策略 搜索策略主要包括强化学习（Reinforcement Learning）方法，梯度下降 (Gradient Descent) 方法，进化计算（Evolution Computation ）方法等。

- 强化学习：基于强化学习方法，在搜索空间中采样后组成网络，对性能进行评估。
- 梯度下降：基于梯度下降的方法，关注基本单元的连接方式，将搜索问题建模为分类问题并进行梯度求解。
- 进化计算：基于进化计算的方法，让神经网络结构的集合基于适应度不断地被训练和突变（mutate），从而达到更好的验证指标。 **2.3 性能评估策略** 性能评估策略和搜索策略循环迭代，搜索策略搜索一次之后，由性能评估策略评估结果反馈给搜索策略，决定下一步往哪个方向搜索。

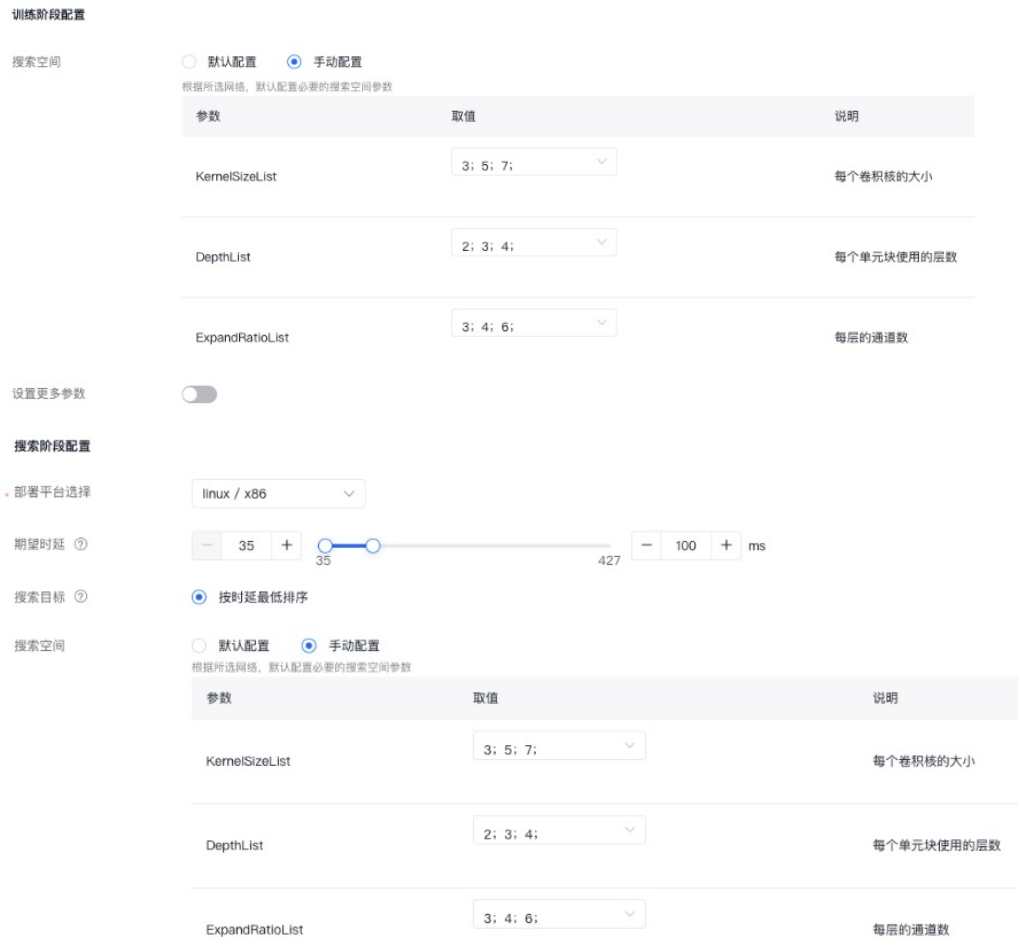
3 NAS的使用



平台中对NAS的使用分为四个步骤：

（1）创建神经网络搜索任务 确保已经准备好神经网络搜索训练数据，或前往“数据集管理”中标注数据完成数据准备，之后创建神经网络搜索任务。 （2）新建神经网络搜索运行 配置训练阶段和搜索阶段的参数，可以使用平台的默认参数，也可以手动配置，训练自己的神经网络子网模

型。 首次创建任务时仅支持共同选择训练和搜索阶段，已训练出超网后可支持选择仅搜索阶段。



（3）通用模型使用子网模型 在通用模型创建训练任务，网络配置方式选择神经网络搜索，并选择已经训练好的子网模型。

（4）发布模型 完成训练后，可以发布训练好的模型到模型仓库。

Notebook建模

Notebook从训练到部署快速入门

Notebook 模板使用指南概述

Notebook 模板使用指南概述

BML 基于 Jupyter 提供了在线的交互式开发环境 Notebook 建模，不但免去了用户准备开发环境的过程，而且可以帮助用户轻松的实现数据集、代码以及模型的管理。

模板概述

BML Notebook 建模为开发者提供了图像分类-单图单标签、图像分类-单图多标签、物体检测、NLP 通用和通用等多种模板。



其中需要注意的是：

1、图像分类-单图单标签、图像分类-单图多标签、物体检测等模板**仅支持 PaddlePaddle 框架训练产出的对应方向模型进行后续的部署**，如：图像分类-单图单标签仅支持 PaddlePaddle 框架训练的图像分类-单图单标签模型进行后续部署。

2、NLP 通用模板仅支持 PaddlePaddle 框架 wenxin 套件训练产出的**文本分类-单标签、文本分类-多标签、短文本相似度、序列标注模型**的后续部署。

3、通用 Notebook 产出的模型支持 **图像分类单标签、图像分类多标签、物体检测、表格预测** 四种应用方向的模型，各个方向支持的模型框架如下：

- 图像分类单标签：`PaddlePaddle、Tensorflow、Caffe、Pytorch、MXMet、Darknet、ONNX`
- 图像分类多标签：`PaddlePaddle、Tensorflow、Caffe、Pytorch、MXMet、Darknet、ONNX`
- 物体检测：`PaddlePaddle、caffe、Darknet`
- 表格预测：`sklearn、xgboost`

使用指南列表

本文针对各模板均提供了使用指南如下：

1、基于 Notebook 的 NLP 通用模板使用指南

2、基于 Notebook 的图像分类模板使用指南

3、基于 Notebook 的物体检测模板使用指南

4、基于 Notebook 的通用模板使用指南

基于 Notebook 的 NLP 通用模板使用指南

目录

1.创建并启动Notebook
2.训练文本分类-单文本单标签模型
3.配置并发布模型
4.校验模型
5.部署在线服务

基于 Notebook 的 NLP 通用模板使用指南

本文采用模板中预置的文心套件开发文本分类-单文本单标签模型的过程为例，介绍在使用 NLP 模板时，从创建 Notebook 任务到引入数据、训练模型，再到保存模型、部署模型的全流程。

创建并启动Notebook

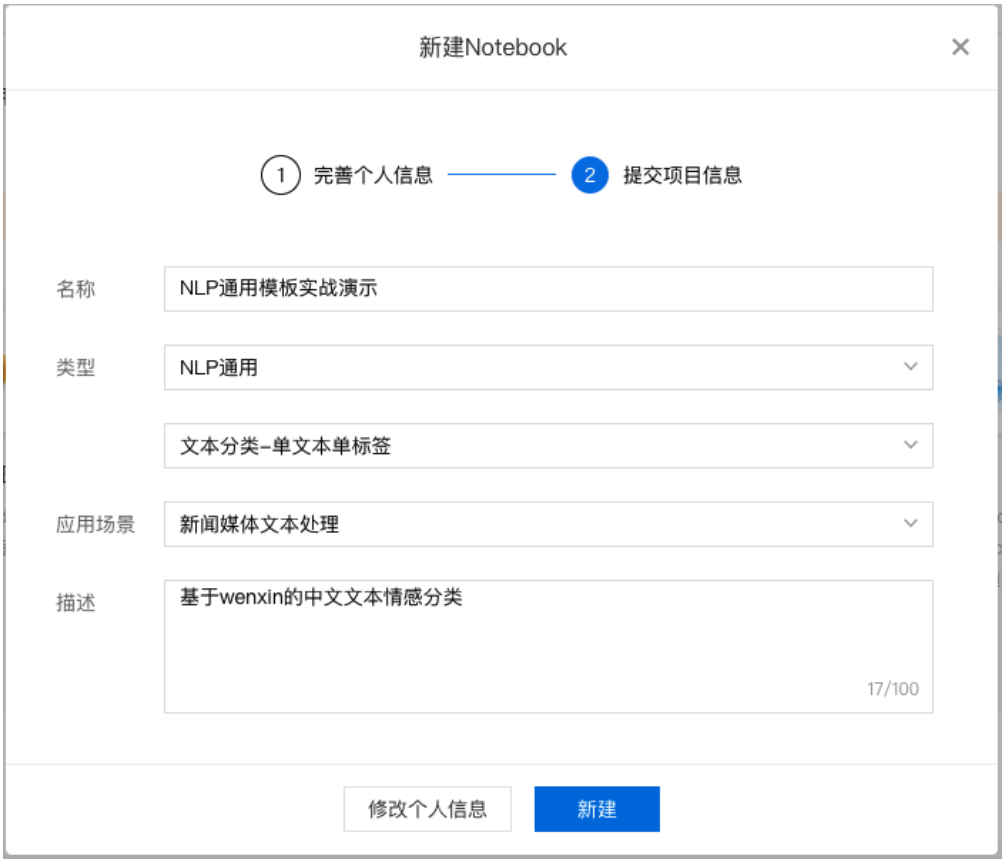
1、在 BML 左侧导航栏中点击『Notebook』

2、在 Notebook 页面点击『新建』，在弹出框中填写公司/个人信息以及项目信息，示例如下：

填写基础信息



填写项目信息



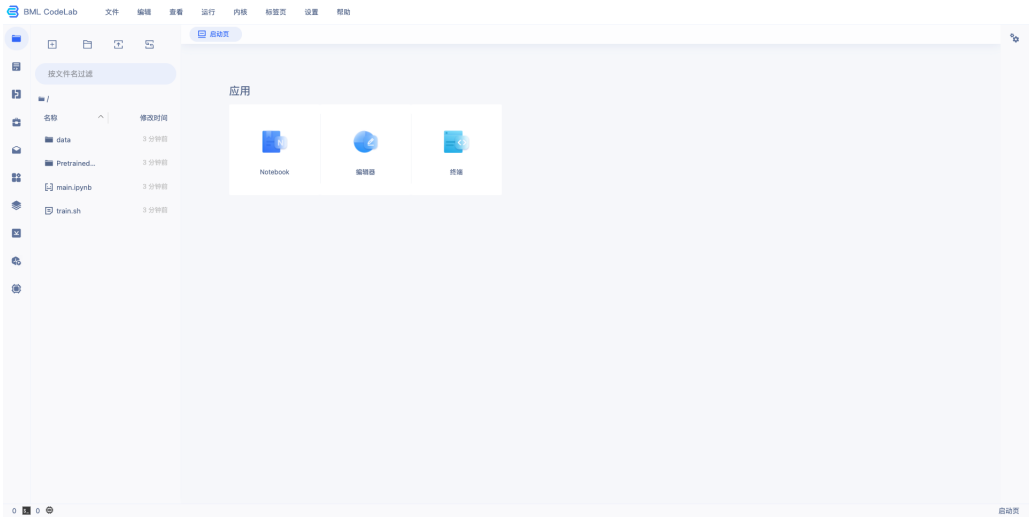
3、对 Notebook 任务操作入口中点击『配置』进行资源配置，示例如下：

选择开发语言、AI 框架，由于本次采用 wenxin 进行演示，所以需要选择 python3.7、PaddlePaddle2.0.0。选择资源规格，由于深度学习所需的训练资源一般较多，需要选择GPU V100的资源规格。



完成配置后点击『确认并启动』，即可启动 Notebook，启动过程中需要完成资源的申请以及实例创建，请耐心等待。

4、等待 Notebook 启动后，点击『打开』，页面跳转到 Notebook，即完成 Notebook 的创建与启动，示例如下：



训练文本分类-单文本单标签模型

下载文心套件

打开进入 Notebook，在左侧包管理插件找到 wenxin，点击安装最新版本。



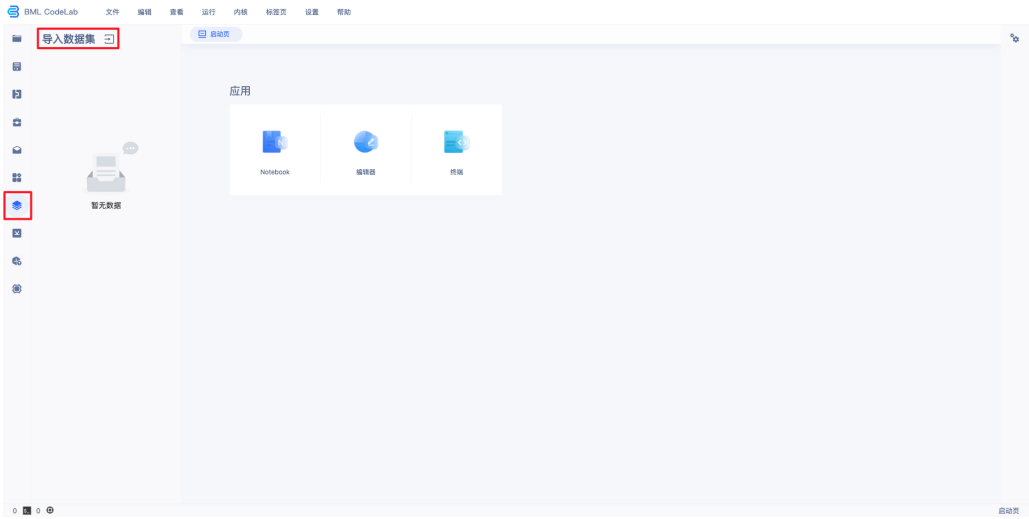
安装成功后，回到用户当前目录，即可看到 wenxin 文件夹。



准备训练数据

1、数据来源一：导入用户数据。

通过左边选择栏的导入数据集选项，进行数据集导入。导出的数据位于用户目录的 `data/` 文件夹。

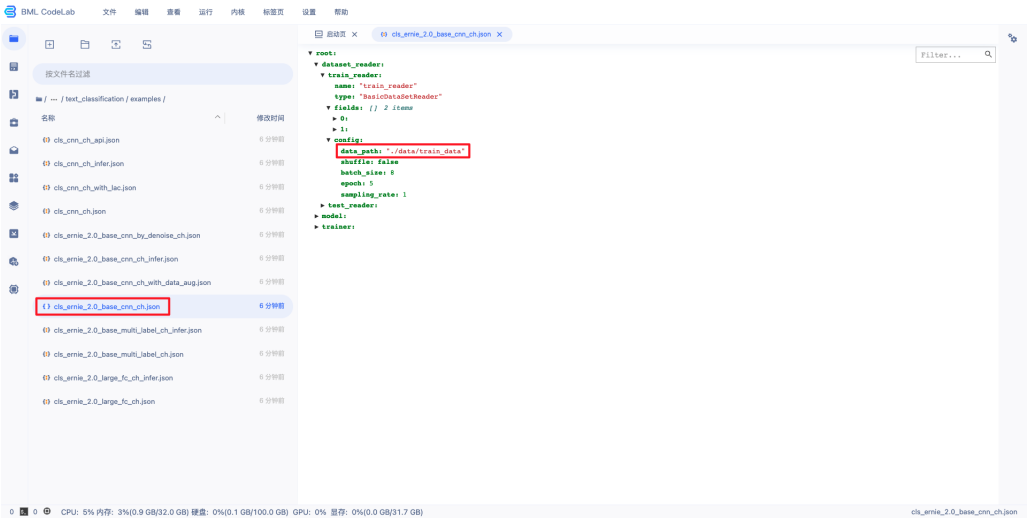


2、数据来源二：使用 wenxin 套件自带的数据集。

训练模型

1、若需要使用用户导入的数据集，则需要更改训练的配置文件中的数据集路径。

例如使用`wenxin/tasks/text_classification/examples/cls_ernie_2.0_base_cnn_ch.json`配置文件，则使用 `vim` 进行编辑，修改 `data_path`。



2、下载预训练模型，在终端中执行命令如下：

```
cd wenxin/tasks/model_files/
bash download_ernie_2.0_base_ch.sh
```

3、模型训练。

```
cd ../text_classification
python run_with_json.py --param_path examples/cls_ernie_2.0_base_cnn_ch.json
```

训练完成后会在该目录中生成 output 文件夹，里面存放了训练生成的模型。



4、模型拷贝，由于生成模型版本的组件只能读取 PretrainedModel 文件夹下的文件，所以需要将模型部署相关的文件都拷贝至该文件夹下。

在终端中执行命令，将模型拷贝至 PretrainedModel 文件夹下。

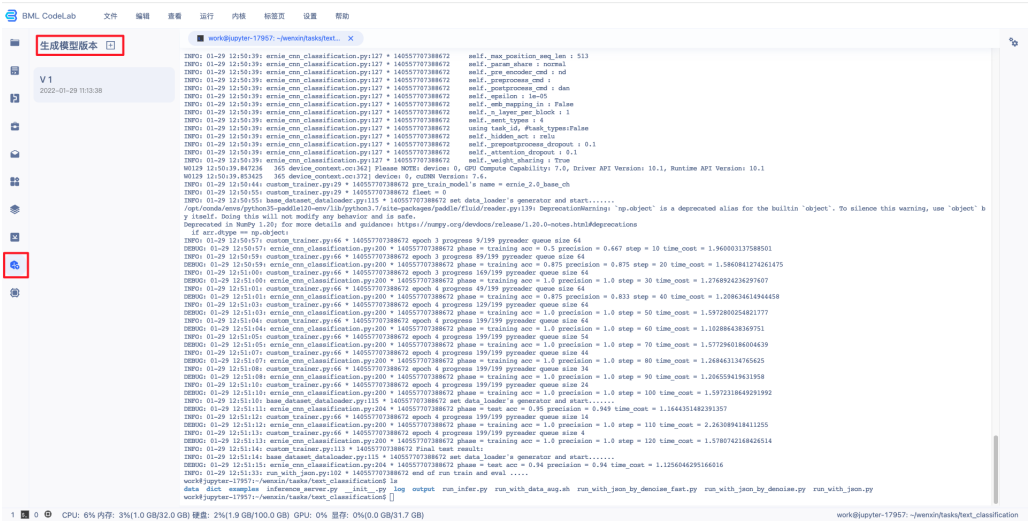
```
cp output /home/work/PretrainedModel/ -r
```

拷贝训练参数配置 cls_ernie_2.0_base_cnn_ch.json 中 reader 的 vocab 和 embedding.config_path 指定的文件，以便用于后续的公有云部署。

```
cp ../model_files/config/ernie_2.0_base_ch_config.json /home/work/PretrainedModel/ernie_2.0_base_ch_config.json
cp ../model_files/dict/vocab_ernie_2.0_base_ch.txt /home/work/PretrainedModel/vocab.txt
cp output/cls_ernie_2.0_base_cnn_ch/save_inference_model/inference_step_126_enc/infer_data_params.json
/home/work/PretrainedModel/infer_data_params.json
```

5、生成模型版本。

点击左侧导航栏中的生成模型版本组件，打开弹窗填写信息。







点击『生成』即可生成模型版本，生成模型版本一般需要数十秒，请耐心等待。

配置并发布模型

BML NoteBook 的 NLP 通用模板产出的模型支持基于百度自研的PaddlePaddle深度学习框架和文心套件进行开发的文本分类-单文本单标签、文本分类-单文本多标签、短文本相似度、序列标注四种应用方向的模型进行部署。下面以文本分类-单文本单标签的模型为例，详细介绍如何配置模型：

1、查看前置条件是否满足：需要训练完成，并生成了相应的模型生成版本（详见训练模型的第五步）。

2、回到 BML Notebook 列表页，点击『模型发布列表』即可进入配置页面。



3、点击配置，即可进入配置流程。



4、填写模型信息。



5、选择待发布的模型文件，点击确定按钮。



其中：vocab.txt 为词表文件，output为模型文件，ernie_2.0_base_ch_config.json 和 infer_data_params.json 为配置文件，非必须。

6、配置出入参及数据逻辑处理。

点击立即编辑，即可进入代码编辑页面。主要需要修改 vocab_path 和 inference_model_path 两个字段的路径为所选择的路径（注：『选择模型文件』后的模型文件将存放在系统制定的路径下，用户在代码中可直接采用相对路径，如 vocab.txt 文件的路径为 ./vocab.txt ,output 文件夹的路径为 ./output）。

在本次示例中， vocab_path 默认路径就是所选择的路径，无须修改，仅需要修改inference_model_path即可。



```
"inference": {
  "output_path": "./output/predict_result.txt",
  "inference_model_path": "./",
  "PADDLE_USE_GPU": 1,
  "PADDLE_IS_LOCAL": 1,
  "num_labels": 2,
  "extra_param": {
    "meta":{
      "job_type": "text_classification"
    }
  }
}
```

修改为：



```
"inference": {
  "output_path": "./output/predict_result.txt",
  "inference_model_path": "./output/cls_ernie_2.0_base_cnn_ch/save_inference_model/inference_step_126_enc",
  "PADDLE_USE_GPU": 1,
  "PADDLE_IS_LOCAL": 1,
  "num_labels": 2,
  "extra_param": {
    "meta":{
      "job_type": "text_classification"
    }
  }
}
```

最终的代码如下：

```
{
 "dataset_reader": {
  "predict_reader": {
   "name": "predict_reader",
   "type": "BasicDataSetReader",
   "fields": [
    {
     "name": "text_a",
     "data_type": "string",
     "reader": {
      "type": "ErnieTextFieldReader"
     },
     "tokenizer": {
      "type": "FullTokenizer",
      "split_char": " ",
      "unk_token": "[UNK]"
     },
     "need_convert": true,
     "vocab_path": "./vocab.txt",
     "max_seq_len": 512,
     "truncation_type": 0,
     "padding_id": 0,
     "embedding": {
      "type": "ErnieTokenEmbedding",
      "use_reader_emb": false,
      "emb_dim": 768,
      "config_path": "ernie_2.0_base_ch_config.json"
     }
    }
   ],
   "config": {
    "data_path": "./data/predict_data",
    "shuffle": false,
    "batch_size": 8,
    "epoch": 1,
    "sampling_rate": 1.0
   }
  }
 },

 "inference": {
  "output_path": "./output/predict_result.txt",
  "inference_model_path": "./output/cls_ernie_2.0_base_cnn_ch/save_inference_model/inference_step_126_enc",
  "PADDLE_USE_GPU": 1,
  "PADDLE_IS_LOCAL": 1,
  "num_labels": 2,
  "extra_param": {
   "meta":{
    "job_type": "text_classification"
   }

  }
 }
}
```

配置文件对于服务部署来说包括两个部分: `dataset_reader`（数据部分）和 `inference` 部分，相关字段的解释如下表所示：

`dataset_reader` 数据部分

```
 {
  "dataset_reader": {
    "predict_reader": { ## 预测推理，则必须配置predict_reader，其配置方式与train_reader、test_reader类似，需要注意的是predict_reader不需要label域，shuffle参数必须是false，epoch参数必须是1。
      "name": "predict_reader",
      "type": "BasicDataSetReader", ## 采用BasicDataSetReader，其封装了常见的读取tsv、txt文件、组batch等操作。
      "fields": [ ## 域（field）是文心的高阶封装，对于同一个样本存在不同域的时候，不同域有单独的数据类型（文本、数值、整型、浮点型）、单独的词表(vocabulary)等，可以根据不同域进行语义表示，如文本转id等操作，field_reader是实现这些操作的类。
        {
          "name": "text_a", ## 文本分类只有一个文本特征域，命名为"text_a"。
          "data_type": "string", ## data_type定义域的数据类型，文本域的类型为string，整型数值为int，浮点型数值为float。
          "reader": {
            "type": "ErnieTextFieldReader"  ## 文本域的通用reader "CustomTextFieldReader"，数值数组类型域为"ScalarArrayFieldReader"，数值标量类型域为"ScalarFieldReader"，这里的取值是对应FieldReader的类名
          },
          "tokenizer": {
            "type": "FullTokenizer", ## 指定该文本域的tokenizer为FullTokenizer，type的取值是对应Tokenizer的类名，
            "split_char": " ", ## 非Ernie任务需要自己切词，切词之后的明文使用的分隔符在这里设置，默认是通过空格区分不同的token。
            "unk_token": "[UNK]" ## unk标记为"[UNK]"， 即词表之外的token所对应的默认id，unk必须是词表文件中存在的token。
          },
          "need_convert": true, ## "need_convert"为true说明数据格式是明文字符串，需要通过词表转换为id。
          "vocab_path": "./vocab.txt", ## 指定该文本域的词表，"need_convert"为true时一定要设置
          "max_seq_len": 512, ## 设定当前域转为id之后的最大长度
          "truncation_type": 0, ## 选择文本超长截断的策略，0为从头开始到最大长度截断，1为从头开始到max_len-1的位置截断，末尾补上最后一个id（词或字），2为保留头和尾两个位置，然后按从头开始到最大长度方式截断。
          "padding_id": 0, ## 设定padding时对应的id值，文心内部会按batch中的最长文本大小对整个batch中的数据进行padding补齐。
          "embedding": {
            "type": "ErnieTokenEmbedding",
            "use_reader_emb": false,
            "emb_dim": 768,
            "config_path": "ernie_2.0_base_ch_config.json"
          }
        }
      ],
      "config": {
        "data_path": "./data/predict_data",
        "shuffle": false, ## 注意！这里的参数必须关掉，打乱顺序输出之后不方便比对数据看结果
        "batch_size": 8,
        "epoch": 1, ## 注意！这里的epoch要设置为1，重复多次预测没意义。
        "sampling_rate": 1.0
      }
    }
  }
```

inference 部分

```
    "inference": {
    "output_path": "./output/predict_result.txt", ## 预测结果的输出路径，封装成Web服务可忽略
    "inference_model_path": "./output/cls_ernie_2.0_base_cnn_ch/save_inference_model/inference_step_126_enc", ## 待预测模型的路径，相对路径
    "PADDLE_USE_GPU": 1, ## 是否使用GPU进行训练，1为使用GPU
    "PADDLE_IS_LOCAL": 1, ## 是否单机预测，必须设置为1，目前文心的python版本预测仅支持单机
    "num_labels": 2, ## 必填参数，表示分类模型的类别数目是多少，预测结果解析时会用到
    "extra_param": {  ## 同trainer，除核心必要信息之外，需要额外标明的参数信息，比如一些meta信息可以作为日志统计的关键字。
      "meta":{
        "job_type": "text_classification"
      }
    }
  }
}
```

7、点击提交即可进入模型验证阶段，验证时间一般需要数十秒，请耐心等待。



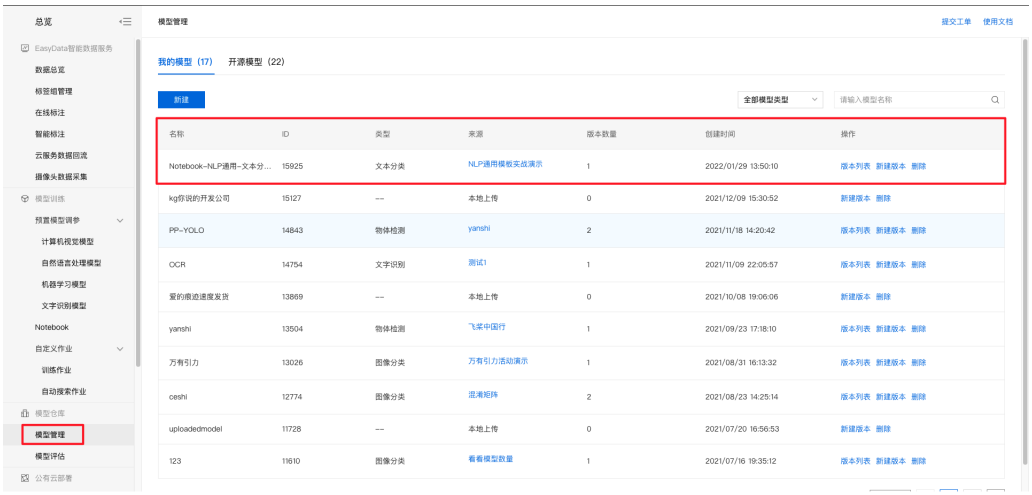
验证通过后，显示有效。



8、点击发布，填写相关信息，即可发布模型。



点击左侧导航栏的『模型管理』，即可进入模型管理列表页查看已发布的模型。



校验模型

1、点击『版本列表』。



2、点击『校验模型』。



3、点击『启动模型校验』，启动约需5分钟，请耐心等待。



4、输入文本或者上传文本即可开始校验，示例如下：



部署在线服务

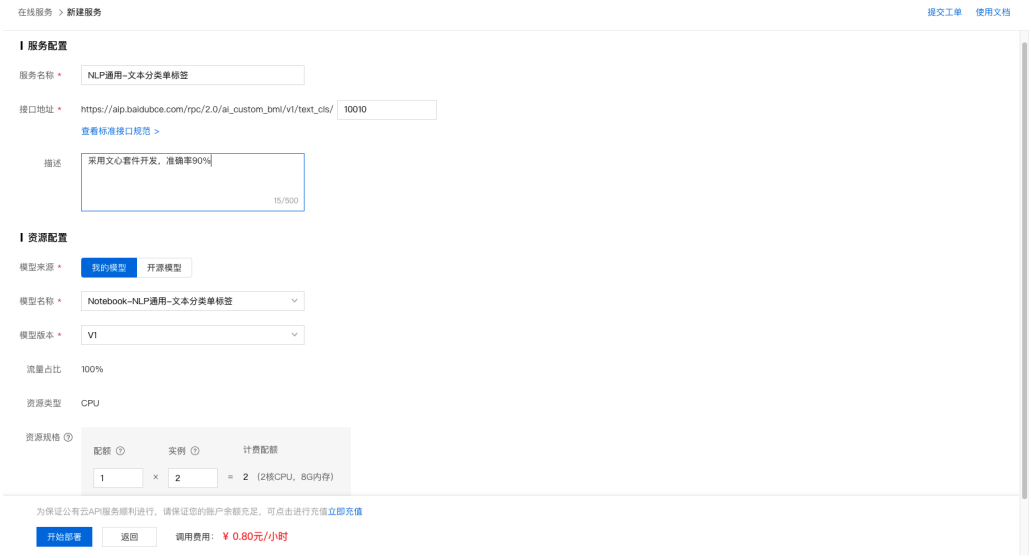
1、点击『版本列表』。



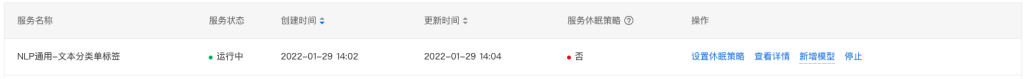
2、点击部署-在线服务。



3、完成信息填写及资源规格选择后，即可开始部署。



4、部署过程需要数十秒时间，请耐心等待。部署完成后，示例如下：



5、API调用方法请参考 文字识别模型部署文档。

基于Notebook的图像分类模板使用指南

目录

1.创建并启动Notebook
2.训练图像分类-单图单标签模型
3.配置并发布模型
4.校验模型
5.部署在线服务

基于Notebook的图像分类模板使用指南

本文采用图像分类-单图单标签模板开发模型的过程为例，介绍从创建 Notebook 任务到引入数据、训练模型，再到保存模型、部署模型的全流程。

创建并启动Notebook

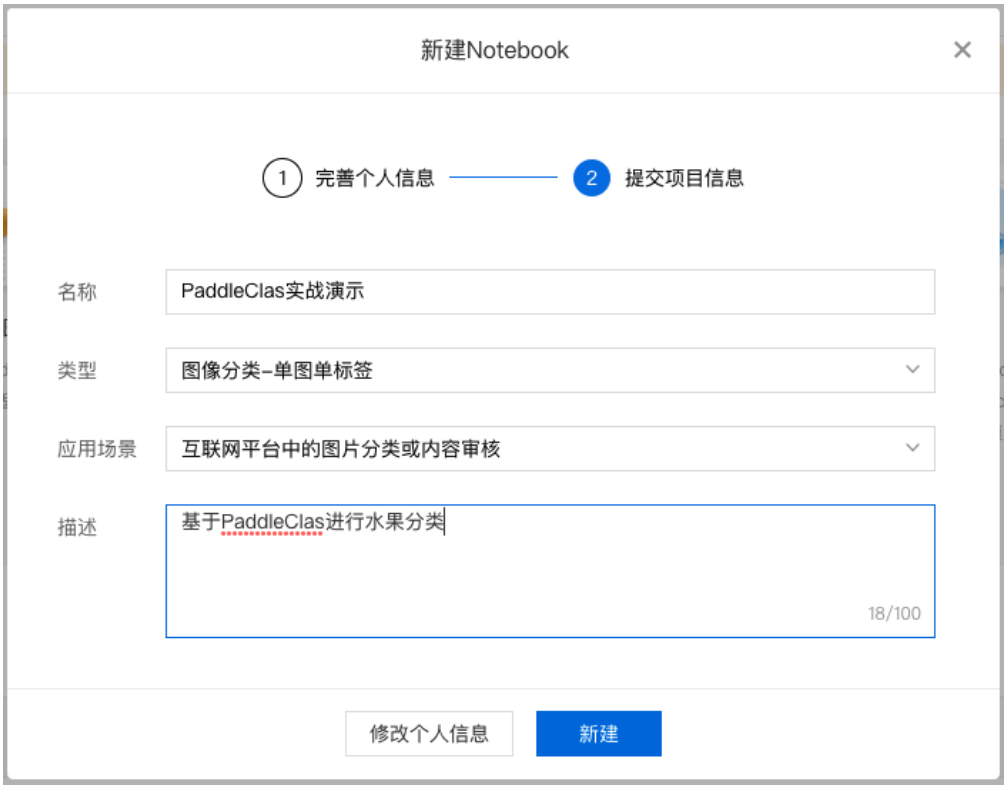
1、在 BML 左侧导航栏中点击『Notebook』

2、在 Notebook 页面点击『新建』，在弹出框中填写公司/个人信息以及项目信息，示例如下：

填写基础信息



填写项目信息



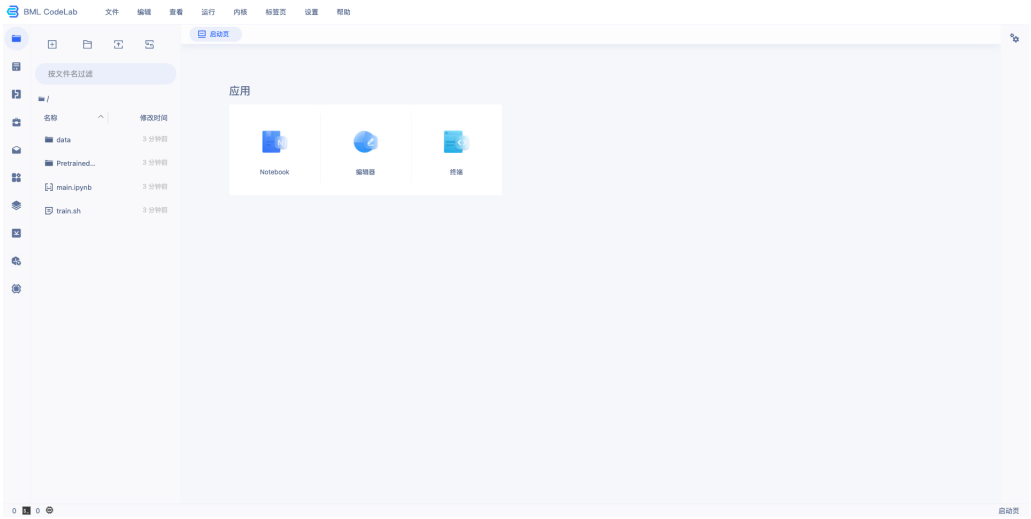
3、对 Notebook 任务操作入口中点击『配置』进行资源配置，示例如下：

选择开发语言、AI 框架，由于本次采用 PaddleClas 进行演示，所以需要选择 python3.7、PaddlePaddle2.0.0。选择资源规格，由于深度学习所需的训练资源一般较多，需要选择GPU V100的资源规格。



完成配置后点击『确认并启动』，即可启动 Notebook，启动过程中需要完成资源的申请以及实例创建，请耐心等待。

4、等待 Notebook 启动后，点击『打开』，页面跳转到 Notebook，即完成 Notebook 的创建与启动，示例如下：



训练图像分类-单图单标签模型

下载 PaddleClas 套件

打开进入 Notebook，点击进入终端，输入如下命令切换到 `/home/work/` 目录。

```
cd /home/work/
```

本文以 `PaddleClas` 代码库 `release/2.3` 分支为例，输入如下命令下载并解压代码包。整个过程需要数十秒，请耐心等待。

```
wget https://github.com/PaddlePaddle/PaddleClas/archive/refs/heads/release/2.3.zip && unzip 2.3.zip
```

安装环境

在终端环境中，安装该版本的 `PaddleClas` 代码包依赖的 `paddlepaddle-gpu`，执行如下命令：

```
python -m pip install paddlepaddle-gpu==2.1.3.post101 -f https://www.paddlepaddle.org.cn/whl/linux/mkl/avx/stable.html
```

安装完成后，使用 `python` 或 `python3` 进入python解释器，输入 `import paddle` ，再输入 `paddle.utils.run_check()`

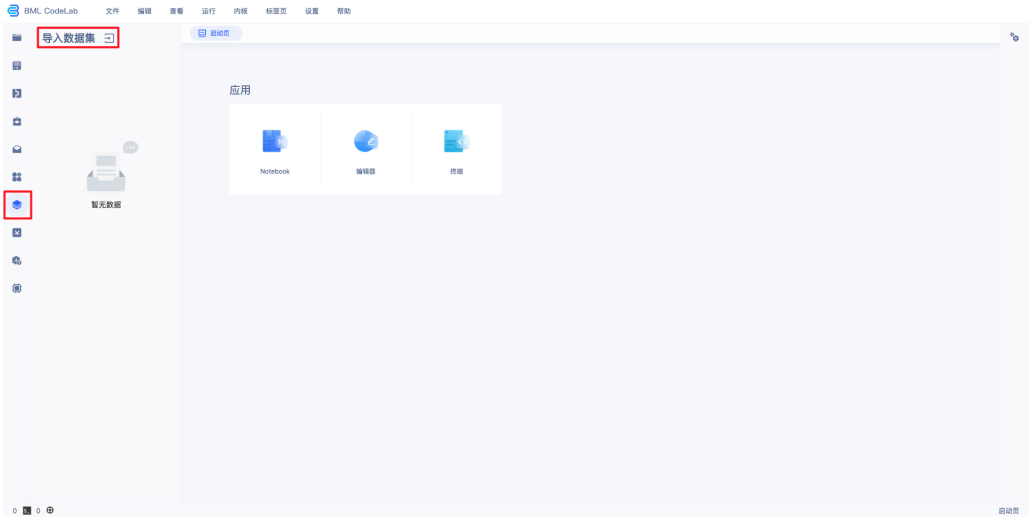
如果出现 `PaddlePaddle is installed successfully!`，说明成功安装。

准备训练数据

训练数据是模型生产的重要条件，优质的数据集可以很大程度上的提升模型训练效果，准备数据可以参考链接。本文所用的安全帽检测数据集可前往此链接进行下载：下载链接。

1、导入用户数据。

在 Notebook 中并不能直接访问您在 BML 中创建的数据集，需要通过左边选择栏的导入数据集选项，进行数据集导入。导入的数据位于用户目录的 `data/` 文件夹（当原始数据集有更新时，不会自动同步，需要手工进行同步）。



注：若在BML中未创建数据集，请先参考 数据服务 ，创建、上传、标注数据集。

2、数据转换。

`PaddleClas` 训练所需要的数据格式与 `BML` 默认的数据格式有所不同，所以需要利用脚本将导入的数据转为 `PaddleClas` 支持的数据格式，并进行3：7切分。

`PaddleClas` 默认支持的标注格式为 `txt`，文件中每行格式如下：

```
图像相对路径 图像的label_id（数字类别）（注意：中间有空格）
```

转换脚本如下：

```
 import os
import json
import glob
import codecs
import random

def parse_label_list(src_data_dir, save_dir):
    """
    遍历标注文件，获取label_list
    :param src_data_dir:
    :return:
    """
    label_list = []
    anno_files = glob.glob(src_data_dir + "*.json")
    for anno_f in anno_files:
        annos = json.loads(codecs.open(anno_f).read())
        for object in annos["labels"]:
            label_list.append(object["name"])
    label_list = list(set(label_list))
    with codecs.open(os.path.join(save_dir, "label_list.txt"), 'w', encoding="utf-8") as f:
        for id, label in enumerate(label_list):
            f.writelines("%s:%s\n" % (id, label))
    return len(label_list), label_list


def trans_split_data(src_data_dir, save_dir):
    """转换数据格式，并3/7分切分数据"""
    image_list = glob.glob(src_data_dir + "*.[jJPpBb][PpNnMm]*")
    image_label_list = []
    for image_file in image_list:
        json_file = image_file.split('.')[0]+".json"
        if os.path.isfile(json_file):
            annos = json.loads(codecs.open(json_file).read())
            label = annos["labels"][0]["name"]
            image_label_list.append("{} {}\n".format(os.path.basename(image_file), label_list.index(label)))

    random.shuffle(image_label_list)
    split_nums = int(len(image_label_list) * 0.3)
    val_list = image_label_list[:split_nums]
    train_list = image_label_list[split_nums:]
    with open(os.path.join(save_dir, "train.txt"), 'w') as f:
        f.writelines(train_list)
    with open(os.path.join(save_dir, "val.txt"), 'w') as f:
        f.writelines(val_list)

class_nums, label_list = parse_label_list("/home/work/data/${dataset_id}/", "/home/work/PretrainedModel/")
trans_split_data("/home/work/data/${dataset_id}/", "/home/work/PretrainedModel/")
```

将上述脚本存放为 `coversion.py` 代码脚本，并将脚本最后两行的 `${dataset_id}` 替换为所指定数据集的 ID（下图红框中的ID），在终端中运行即可。



运行代码。

```
python coversion.py
```

运行之后将在 `PretrainedModel/` 文件夹下生成对应的数据文件，包括 `label_list.txt`、`tran.txt`、`val.txt`。



训练模型

1、在终端中打开 `PaddleClas` 目录。

```
cd /path/to/PaddleClas
```

2、修改yaml配置文件。

本文以 ResNet50_vd 为例，配置文件路径为:

```
/home/work/PaddleClas-release-2.3/ppcls/configs/ImageNet/ResNet/ResNet50_vd.yaml
```

```yaml
  ###### global configs
Global:
  checkpoints: null
  pretrained_model: null
  output_dir: ./output/
  # 使用GPU训练
  device: gpu
  # 每几个轮次保存一次
  save_interval: 1
  eval_during_train: True
  # 每几个轮次验证一次
```

```yaml
  eval_interval: 1
  # 训练轮次
  epochs: 10
  print_batch_step: 10
  use_visualdl: False #是否开启可视化
  # used for static mode and model export
  # 图像大小
  image_shape: [3, 224, 224]
  save_inference_dir: ./inference

###### model architecture
Arch:
  # 采用的网络
  name: ResNet50_vd
  # 类别数
  class_num: 1000

###### loss function config for traing/eval process
Loss:
  Train:
    - CELoss:
        weight: 1.0
        epsilon: 0.1
  Eval:
    - CELoss:
        weight: 1.0


Optimizer:
  name: Momentum
  momentum: 0.9
  lr:
    name: Cosine
    learning_rate: 0.1
  regularizer:
    name: 'L2'
    coeff: 0.00007


###### data loader for train and eval
DataLoader:
  Train:
    dataset:
      name: ImageNetDataset
      # 数据集根路径
      image_root: /home/work/data/${dataset_id}
      # 前面生产得到的训练集列表文件路径
      cls_label_path: /home/work/PretrainedModel/train_list.txt
      # 数据预处理
      transform_ops:
        - DecodeImage:
            to_rgb: True
            channel_first: False
        - RandCropImage:
            size: 224
        - RandFlipImage:
            flip_code: 1
        - NormalizeImage:
            scale: 1.0/255.0
            mean: [0.485, 0.456, 0.406]
            std: [0.229, 0.224, 0.225]
            order: ''
      batch_transform_ops:
        - MixupOperator:
            alpha: 0.2

    sampler:
      name: DistributedBatchSampler
      batch_size: 64
      drop_last: False
```

```
    shuffle: True
  loader:
    num_workers: 0
    use_shared_memory: True

Eval:
  dataset:
    name: ImageNetDataset
    # 数据集根路径
    image_root: /home/work/data/${dataset_id}
    # 前面生产得到的训练集列表文件路径
    cls_label_path: /home/work/PretrainedModel/val_list.txt
    transform_ops:
      - DecodeImage:
          to_rgb: True
          channel_first: False
      - ResizeImage:
          resize_short: 256
      - CropImage:
          size: 224
      - NormalizeImage:
          scale: 1.0/255.0
          mean: [0.485, 0.456, 0.406]
          std: [0.229, 0.224, 0.225]
          order: ''
  sampler:
    name: DistributedBatchSampler
    batch_size: 64
    drop_last: False
    shuffle: False
  loader:
    num_workers: 0
    use_shared_memory: True

Infer:
  infer_imgs: docs/images/whl/demo.jpg
  batch_size: 10
  transforms:
    - DecodeImage:
        to_rgb: True
        channel_first: False
    - ResizeImage:
        resize_short: 256
    - CropImage:
        size: 224
    - NormalizeImage:
        scale: 1.0/255.0
        mean: [0.485, 0.456, 0.406]
        std: [0.229, 0.224, 0.225]
        order: ''
    - ToCHWImage:
  PostProcess:
    # 输出的可能性最高的前topk个
    name: Topk
    topk: 5
    # 前面得到标签文件
    class_id_map_file: /home/work/PretrainedModel/label_list.txt

Metric:
  Train:
  Eval:
    - TopkAcc:
        topk: [1, 5]
```

根据相关文件的地址对上述yaml文件进行修订，主要修改点：类别数、训练和验证集的路径、标签文件地址、训练和验证的 `num_workers` 需修改为 0。

注：Notebook 因为是单卡的，需要将 num_workers 改为0，在本地的话则需要根据实际情况进行更改

```
 修改类别数
20行：class_num: 5

修改训练集的路径(数据集id根据您自己的情况调整)
49行：image_root: /home/work/data/302273
50行：cls_label_path: /home/work/PretrainedModel/train.txt

修改训练GPU
74行：num_workers: 0

修改验证集的路径(数据集id根据您自己的情况调整)
80行：image_root: /home/work/data/302273
81行：cls_label_path: /home/work/PretrainedModel/val.txt

修改验证GPU
101行：num_workers: 0

修改标签文件地址
124行：class_id_map_file: /home/work/PretrainedModel/label_list.txt
```

3、训练模型。

在终端中执行以下命令，开始模型训练。

```
 cd PaddleClas-release-2.3/
python tools/train.py -c ./ppcls/configs/ImageNet/ResNet/ResNet50_vd.yaml
```

4、模型预测。

在终端中执行以下命令，开始模型预测。

```
python tools/infer.py \
  -c ./ppcls/configs/ImageNet/ResNet/ResNet50_vd.yaml \
  -o Infer.infer_imgs=/home/work/data/${dataset_id}/xxx.jpg \
  -o Global.pretrained_model=output/ResNet50_vd/best_model
```

预测结果如下：

```
[{'class_ids': [4, 0, 2, 1, 3], 'scores': [0.74596, 0.08715, 0.0704, 0.05722, 0.03927], 'file_name': '/home/work/data/302273/1.jpeg', 'label_names': []}]
```

5、导出模型。

在终端中执行以下命令，将最佳模型转为可以用于发布的 inference 模型。

```
python tools/export_model.py \
  -c ./ppcls/configs/ImageNet/ResNet/ResNet50_vd.yaml \
  -o Global.pretrained_model=output/ResNet50_vd/best_model \
  -o Global.save_inference_dir=/home/work/PretrainedModel/
```

在终端中执行以下命令，将导出模型重命名为以“model”为前缀的模型文件。

```
 mv /home/work/PretrainedModel/inference.pdiparams /home/work/PretrainedModel/model.pdiparams
mv /home/work/PretrainedModel/inference.pdmodel /home/work/PretrainedModel/model.pdmodel
mv /home/work/PretrainedModel/inference.pdiparams.info /home/work/PretrainedModel/model.pdiparams.info
```

分类模型部署时默认配置如下，在 `/home/work/PretrainedModel/` 路径下创建并保存 `infer_cfg.yaml`文件。若需自定义相关参数，可在此基础上进行修改。

```yaml
Global:
  inference_model_dir: "/home/work/PretrainedModel/"
  batch_size: 1
  use_gpu: True
  enable_mkldnn: True
  cpu_num_threads: 10
  enable_benchmark: True
  use_fp16: False
  ir_optim: True
  use_tensorrt: False
  gpu_mem: 8000
  enable_profile: False
PreProcess:
  transform_ops:
    - ResizeImage:
        resize_short: 256
    - CropImage:
        size: 224
    - NormalizeImage:
        scale: 0.00392157
        mean: [0.485, 0.456, 0.406]
        std: [0.229, 0.224, 0.225]
        order: ''
        channel_num: 3
    - ToCHWImage:
```

在此步骤后/home/work/PretrainedModel目录下将有如下文件：



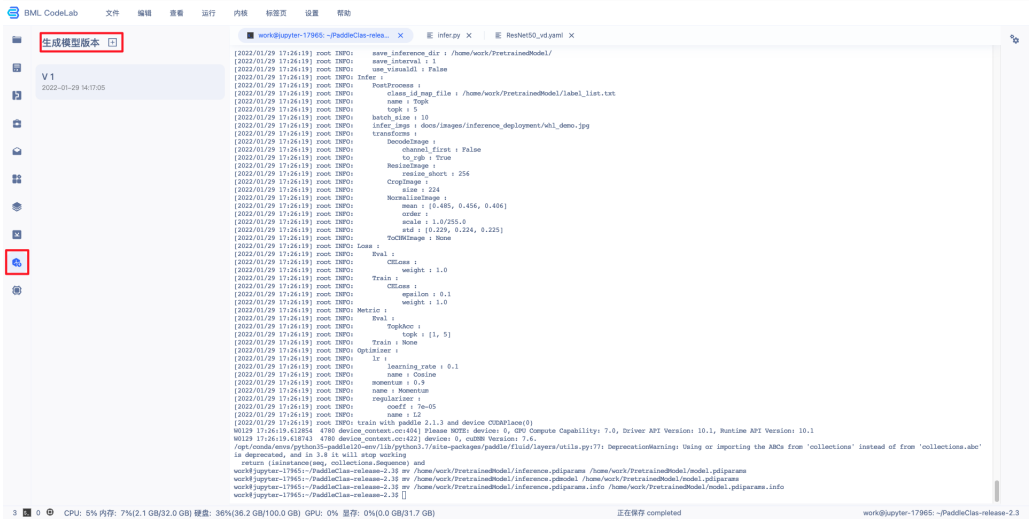
- 数据文件：label_list.txt、train.txt、val.txt
- 模型文件：model.pdiparams、model.pdmodel、model.pdiparams.info
- 配置文件：infer_cfg.yaml

6、生成模型版本。

Notebook中的模型文件只有生成模型版本后，才可以执行发布和部署功能：

- 请确保要保存的模型文件在/home/work/PretrainedModel目录下。模型支持版本管理功能，在保存时可以生成新版本也可以覆盖已有的且尚未部署的模型版本，每个版本的模型都可以独立部署。每个模型版本中保存的模型文件大小上限为1.5GB。
- 在保存模式时也可以将训练模型的代码一并保存。代码支持版本管理功能，用户再次启动Notebook时，可以使用指定的代码版本来初始化Notebook工作空间即/home/work目录下data以外的空间。每个代码版本中保存的文件大小上限为150M。

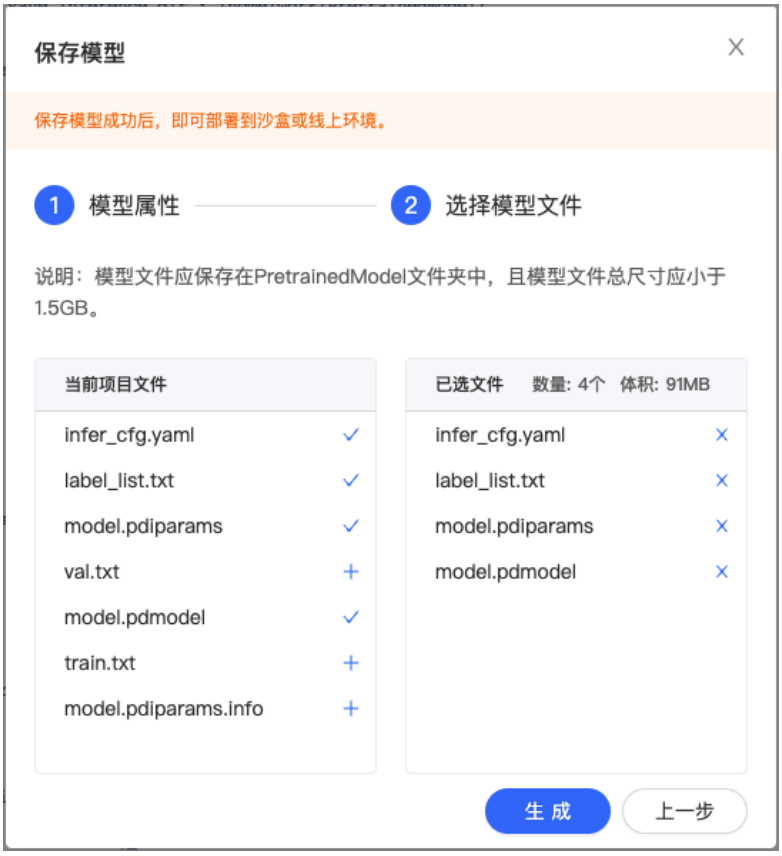
点击左侧导航栏中的生成模型版本组件，打开弹窗填写信息。



模型属性-选择 AI 框架选择 PaddlePaddle2.0.0，若上一次操作中进行了代码保存，可在“代码版本”选择对应的代码版本。



选择模型文件-选择 `label_list.txt`、`model.pdiparams`、`model.pdmodel` 文件，若有自定义 `infer_cfg.yaml` 文件，也一并选上（非必须）。



点击『生成』即可生成模型版本，生成模型版本一般需要数十秒，请耐心等待。

配置并发布模型

BML NoteBook 的图像分类单标签模板产出的模型支持进行部署，下面以 PaddleClas 的模型为例，详细介绍如何配置模型：

1、查看前置条件是否满足：需要训练完成，并生成了相应的模型生成版本（详见训练模型的第六步）。

2、回到 BML Notebook 列表页，点击『模型发布列表』即可进入配置页面。



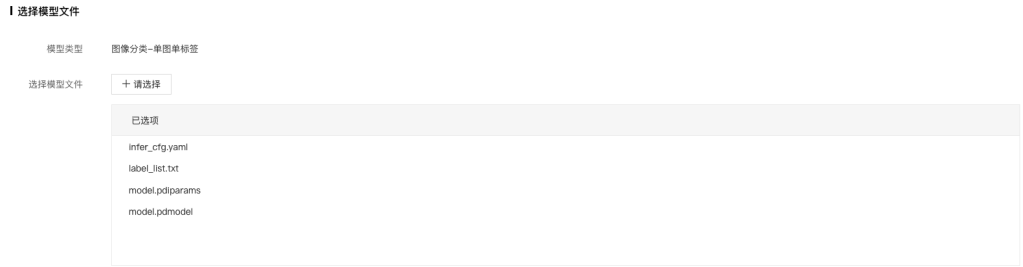
3、点击配置，即可进入配置流程。



4、填写模型信息。



5、选择待发布的模型文件，点击确定按钮。



其中：

- 网络结构文件 model.pdmodel：必需选择，且名字固定。
- 网络参数文件 model.pdiparams：必需选择，且名字固定。
- 模型标签文件：label_list.txt，非必须，主要看自定义逻辑代码是如何实现的。本文中在【配置出入参及数据处理逻辑脚本】的脚本代码里面会用到，所以需要选择。
- 预/后处理配置文件等其他文件：infer_cfg.yaml，非必须，主要看自定义逻辑代码是如何实现的。本文中在【配置出入参及数据处理逻辑脚本】的脚本代码里面会用到，所以需要选择。

6、配置出入参及数据逻辑处理。

这部分配置主要实现自定义的模型预处理和后处理逻辑。该脚本用于将选择的模型文件发布成模型服务。用户可以通过修改 PredictWrapper 的 preprocess 和 postprocess 方法来实现自定义的请求预处理和预测结果后处理。当提交该脚本时，系统会根据用户选择的模型文件和脚本内容，来验证是否可以启动模型服务，只有验证通过，才可以进行模型效果校验以及将模型发布到模型仓库。

实现脚本有一些建议和限制：

- CustomException 必需存在且是异常类；在自定义的逻辑中，建议当处理进入错误的分支时，抛出 CustomException 并指定 message，指定的 message 在请求回包中会作为 error_msg 返回。
- PredictWrapper 类必需存在，且必需包含 preprocess 和 postprocess 两个方法。
- PredictWrapper 的 preprocess 和 postprocess 方法，是用户自定义模型服务请求预处理和预测结果后处理的入口。
- preprocess 方法接收的第一个参数为用户请求的 json 字典，对于图像类服务，传入图像的参数 key 必须是"image"，且传入的是图片的 base64 编码。
- 系统会根据 postprocess 方法的返回结果 result 类型的不同，做以下处理：

```
 - dict: 不作修改
- list: 转换为 {"results": result}
```

  - 其他: 转换为 {"result": result}
  - 最终的处理结果可以转换为json字符串，以确保能够正常将结果作为请求回包返回

- 单击“提交”，完成模型配置。 提交后，系统会自动对当前版本模型进行代码验证，通过后模型会更新为“有效”状态。

配置出入参及数据处理逻辑脚本：实现图片的预处理和模型输出结果后处理的逻辑； 这一步是比较关键，但也比较复杂的一步。上面介绍了脚本实现时的限制和建议。这里针对PaddleClass套件，实现了一套对应的脚本代码（在后面），可以点击立即编辑，将脚本代码拷贝替换掉。



注：

```
 - 可以看下平台预置代码文件，以及各个类及函数的注释了解实现细节。这里贴了`PaddleClass套件`对应的脚本文件，整个代码比较长，但大部分内容都拷贝于[PaddleClas套件的推理示例](https://github.com/PaddlePaddle/PaddleClas/blob/develop/deploy/python/predict_cls.py)。如果自身训练模型比较特殊，当前脚本支持不了，需要自己去PaddleClas套件中寻找逻辑，并更新到该脚本中。
 - 对于处理脚本里面的预处理`preprocess`方法，第一个返回参数`input_info`为字典类型，其中字典的key 为模型的输入节点名称，需要根据模型修改。
 ![图片](http://bjhw-bce-online-public0.bjhw.baidu.com:8109/proxy-external/?url=http://agroup.baidu-int.com/file/stream/bj/bj-70c34318f1ccd776e0c1a1c755ad0a3e6b11420a)
 如果不知道训练模型的输入节点名称是什么，可以先用下面提供的脚本进行一次模型验证，验证日志里面查看对应的输入名称。
 ![图片](http://bjhw-bce-online-public0.bjhw.baidu.com:8109/proxy-external/?url=http://agroup.baidu-int.com/file/stream/bj/bj-0dd53385ff6d94e0397f063ac12ea3ba700253bf)
 ![图片](http://bjhw-bce-online-public0.bjhw.baidu.com:8109/proxy-external/?url=http://agroup.baidu-int.com/file/stream/bj/bj-2219a7252b9fda78281faa7bcb47955519b718e9)
```

PaddleClas脚本样例：

```
 ###### !/usr/bin/env python
###### -*- coding: utf-8 -*-
###### ********************************************************************************
#
###### Copyright (c) 2020 Baidu.com, Inc. All Rights Reserved
#
###### ********************************************************************************

###### 注意事项：
###### 该脚本用于将通过notebook训练产出的模型发布成模型服务
```

```
###### 用户可以通过修改PredictWrapper的preprocess和postprocess方法来实现自定义的请求预处理和预测结果后处理
###### 当在EasyDL提交该脚本时，系统会根据用户选择的模型文件和脚本内容，来验证是否可以启动模型服务，如果验证通过，即可进行模型效果校验以及部署模型服务
###### 下面是修改脚本的一些限制和建议：
###### 1. CustomException必须存在且是异常类；在自定义的逻辑中，建议当处理进入错误的分支时,抛出CustomException并指定message，指定的message在请求回包中会作为error_msg返回；
###### 2. PredictWrapper类必须存在，且必需包含preprocess和postprocess两个方法；
###### 3. PredictWrapper的preprocess和postprocess方法，是用户自定义模型服务请求预处理和预测结果后处理的入口；
###### 4. preprocess方法接收的第一个参数为用户请求的json字典，对于图像类服务，传入图像的参数key必须是"image"，且传入的是图片的base64编码
###### 5. 系统会根据postprocess方法的返回结果`result`类型的不同，做以下处理：
###### dict: 不作修改
###### list: 转换为 {"results": result}
###### 其他: 转换为 {"result": result}
###### 处理结果将转换为json字符串，以确保系统能正常将结果作为请求回包返回。

import base64
import math
import random

import cv2
import logging
import numpy as np
import six
import yaml

from functools import partial
from PIL import Image

DEFAULT_TOP_NUM = 6


class CustomException(RuntimeError):
    """
    进行模型验证和部署服务必需的异常类，缺少该类在代码验证时将会失败
    在处理异常数据或者请求时，推荐在`PredictWrapper`中的自定义预处理preprocess和后处理postprocess函数中抛出`CustomException`类，
    并为`message`指定准确可读的错误信息，以便在服务响应包中的`error_msg`参数中返回。
    """
    def __init__(self, message, orig_error=None):
        """ 根据`message`初始化 """
        super(CustomException, self).__init__(message)
        self.orig_error = orig_error


class PredictWrapper(object):
    """ 模型服务预测封装类，支持用户自定义对服务请求数据的预处理和模型预测结果的后处理函数 """

    def __init__(self, model_path):
        """
        根据`model_path`初始化`PredictWrapper`类，如解析label_list.txt，加载模型输出标签id和标签名称的映射关系
        :param model_path: 该目录下存放了用户选择的模型版本中包含的所有文件
        """
        # 加载推理配置文件，获取【预处理配置】及【标签id和名称的映射关系】
        ops_config = [
            {"ResizeImage": {"resize_short": 256}},
            {"CropImage": {"size": 224}},
            {"NormalizeImage": {"scale": 0.00392157, "mean": [0.485, 0.456, 0.406],
                                "std": [0.229, 0.224, 0.225], "order": '', "channel_num": 3}},
            {"ToCHWImage": {}},
        ]
        conf_path = '{model_path}/{conf_file}'.format(model_path=model_path, conf_file='infer_cfg.yml')
        try:
            with open(conf_path) as conf_fin:
                infer_conf = yaml.load(conf_fin, Loader=yaml.FullLoader)
                if "PreProcess" in infer_conf and "transform_ops" in infer_conf["PreProcess"]:
                    ops_config = infer_conf["PreProcess"]["transform_ops"]
        except:
            pass
```

```
        self._preprocess_ops = create_operators(ops_config)

        try:
            label_path = '{model_path}/{label_file}'.format(model_path=model_path, label_file='label_list.txt')
            self._label_list = {}
            with open(label_path) as label_list_f:
                for index, line in enumerate(label_list_f):
                    line = line.strip()
                    label_info = line.split(":", 1)
                    if len(label_info) == 2:
                        self._label_list[int(label_info[0])] = label_info[1]
                    else:
                        self._label_list[index] = line
        except:
            pass

    def preprocess(self, request_body, **preprocess_args):
        """
        自定义对请求体的预处理，针对图像类模型服务，包括对图片对图像的解析、转化等
        :param request_body: 请求体的json字典
        :param preprocess_args: 从`{model_path}/preprocess_args.json`中加载的预处理参数字典，json文件不存在时，传入为空字典
        :return:
            data: 用于模型预测的输入。注意：data结构为dict，key为模型输入节点的名称，value为对应需要喂入的值，batch只能为1
            infer_args: 用于模型预测的其他参数
            request_context: 透传给自定义后处理函数`postprocess`的参数，例如指定返回预测结果的top N，过滤低score的阈值threshold.
        """
        # decode image from base64 string in request
        try:
            image_b64 = request_body['image']
            img_bin = base64.b64decode(image_b64)
        except KeyError:
            raise CustomException('Missing required parameter')
        except Exception:
            raise CustomException('Invalid BASE64')

        data = np.frombuffer(img_bin, dtype='uint8')
        im = cv2.imdecode(data, 1)  # BGR mode, but need RGB mode
        im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)

        # paddle cls
        # code: https://github.com/PaddlePaddle/PaddleClas/blob/develop/deploy/python/predict_cls.py
        try:
            for preprocess_op in self._preprocess_ops:
                im = preprocess_op(im)

            input_info = {"x": im[np.newaxis, ...]}
        except Exception:
            raise CustomException('Failed decoding input')

        return input_info, {}, {"top_num": request_body.get("top_num", DEFAULT_TOP_NUM)}

    def postprocess(self, infer_result, request_context, **postprocess_args):
        """
        自定义对图像分类模型输出结果的后处理，例如根据score对label进行排序，获取top N分类结果等
        :param infer_result: fluid模型的预测结果
        :param request_context: 自定义预处理函数中返回的`request context`
        :param postprocess_args: 从`{model_path}/postprocess_args.json`中加载的后处理参数字典，json文件不存在时，传入为空字典
        :return: request results 请求的处理结果
        """
        top_num = request_context["top_num"]
        result = infer_result[0]
        result = get_result_list(result)

        indices = np.flip(np.argsort(result), 0)[:top_num]
        top_results = list()
        for item in indices:
            item = int(item)
            top_results.append({
                'score': float(result[item]),
                'name': self._label_list.get(item, str(item))
```

```
            })
        return top_results


def create_operators(params):
    """
    create operators based on the config
    Args:
        params(list): a dict list, used to create some operators
    """
    assert isinstance(params, list), ('operator config should be a list')
    op_mapping = {
        "UnifiedResize": UnifiedResize,
        "DecodeImage": DecodeImage,
        "ResizeImage": ResizeImage,
        "CropImage": CropImage,
        "RandCropImage": RandCropImage,
        "RandFlipImage": RandFlipImage,
        "NormalizeImage": NormalizeImage,
        "ToCHWImage": ToCHWImage,
    }
    ops = []
    for operator in params:
        assert isinstance(operator,
                          dict) and len(operator) == 1, "yaml format error"
        op_name = list(operator)[0]
        param = {} if operator[op_name] is None else operator[op_name]
        op = op_mapping[op_name](**param)
        ops.append(op)

    return ops


class UnifiedResize(object):
    def __init__(self, interpolation=None, backend="cv2"):
        _cv2_interp_from_str = {
            'nearest': cv2.INTER_NEAREST,
            'bilinear': cv2.INTER_LINEAR,
            'area': cv2.INTER_AREA,
            'bicubic': cv2.INTER_CUBIC,
            'lanczos': cv2.INTER_LANCZOS4
        }
        _pil_interp_from_str = {
            'nearest': Image.NEAREST,
            'bilinear': Image.BILINEAR,
            'bicubic': Image.BICUBIC,
            'box': Image.BOX,
            'lanczos': Image.LANCZOS,
            'hamming': Image.HAMMING
        }

        def _pil_resize(src, size, resample):
            pil_img = Image.fromarray(src)
            pil_img = pil_img.resize(size, resample)
            return np.asarray(pil_img)

        if backend.lower() == "cv2":
            if isinstance(interpolation, str):
                interpolation = _cv2_interp_from_str[interpolation.lower()]
            # compatible with opencv < version 4.4.0
            elif interpolation is None:
                interpolation = cv2.INTER_LINEAR
            self.resize_func = partial(cv2.resize, interpolation=interpolation)
        elif backend.lower() == "pil":
            if isinstance(interpolation, str):
                interpolation = _pil_interp_from_str[interpolation.lower()]
            self.resize_func = partial(_pil_resize, resample=interpolation)
        else:
            logging.warning(
                "The backend of Resize only support \"cv2\" or \"PIL\". \"%s\" is unavailable. Use \"cv2\" instead.",
```

```
            backend
        )
        self.resize_func = cv2.resize

    def __call__(self, src, size):
        return self.resize_func(src, size)


class OperatorParamError(ValueError):
    """ OperatorParamError
    """
    pass


class DecodeImage(object):
    """ decode image """

    def __init__(self, to_rgb=True, to_np=False, channel_first=False):
        self.to_rgb = to_rgb
        self.to_np = to_np  # to numpy
        self.channel_first = channel_first  # only enabled when to_np is True

    def __call__(self, img):
        if six.PY2:
            assert type(img) is str and len(
                img) > 0, "invalid input 'img' in DecodeImage"
        else:
            assert type(img) is bytes and len(
                img) > 0, "invalid input 'img' in DecodeImage"
        data = np.frombuffer(img, dtype='uint8')
        img = cv2.imdecode(data, 1)
        if self.to_rgb:
            assert img.shape[2] == 3, 'invalid shape of image[%s]' % (
                img.shape)
            img = img[:, :, ::-1]

        if self.channel_first:
            img = img.transpose((2, 0, 1))

        return img


class ResizeImage(object):
    """ resize image """

    def __init__(self,
                 size=None,
                 resize_short=None,
                 interpolation=None,
                 backend="cv2"):
        if resize_short is not None and resize_short > 0:
            self.resize_short = resize_short
            self.w = None
            self.h = None
        elif size is not None:
            self.resize_short = None
            self.w = size if type(size) is int else size[0]
            self.h = size if type(size) is int else size[1]
        else:
            raise OperatorParamError("invalid params for ReisizeImage for '\
                'both 'size' and 'resize_short' are None")

        self._resize_func = UnifiedResize(
            interpolation=interpolation, backend=backend)

    def __call__(self, img):
        img_h, img_w = img.shape[:2]
        if self.resize_short is not None:
            percent = float(self.resize_short) / min(img_w, img_h)
            w = int(round(img_w * percent))
```

```
            h = int(round(img_h * percent))
        else:
            w = self.w
            h = self.h
        return self._resize_func(img, (w, h))


class CropImage(object):
    """ crop image """

    def __init__(self, size):
        if type(size) is int:
            self.size = (size, size)
        else:
            self.size = size  # (h, w)

    def __call__(self, img):
        w, h = self.size
        img_h, img_w = img.shape[:2]

        if img_h < h or img_w < w:
            raise Exception(
                "The size({h}, {w}) of CropImage must be greater than size({img_h}, {img_w}) of image. "
                "Please check image original size and size of ResizeImage if used.".format(h=h, w=w, img_w=img_w,
                                                                                           img_h=img_h)
            )

        w_start = (img_w - w) // 2
        h_start = (img_h - h) // 2

        w_end = w_start + w
        h_end = h_start + h
        return img[h_start:h_end, w_start:w_end, :]


class RandCropImage(object):
    """ random crop image """

    def __init__(self,
                 size,
                 scale=None,
                 ratio=None,
                 interpolation=None,
                 backend="cv2"):
        if type(size) is int:
            self.size = (size, size)  # (h, w)
        else:
            self.size = size

        self.scale = [0.08, 1.0] if scale is None else scale
        self.ratio = [3. / 4., 4. / 3.] if ratio is None else ratio

        self._resize_func = UnifiedResize(
            interpolation=interpolation, backend=backend)

    def __call__(self, img):
        size = self.size
        scale = self.scale
        ratio = self.ratio

        aspect_ratio = math.sqrt(random.uniform(*ratio))
        w = 1. * aspect_ratio
        h = 1. / aspect_ratio

        img_h, img_w = img.shape[:2]

        bound = min((float(img_w) / img_h) / (w**2),
                    (float(img_h) / img_w) / (h**2))
        scale_max = min(scale[1], bound)
        scale_min = min(scale[0], bound)
```

```
        target_area = img_w * img_h * random.uniform(scale_min, scale_max)
        target_size = math.sqrt(target_area)
        w = int(target_size * w)
        h = int(target_size * h)

        i = random.randint(0, img_w - w)
        j = random.randint(0, img_h - h)

        img = img[j:j + h, i:i + w, :]

        return self._resize_func(img, size)


class RandFlipImage(object):
    """ random flip image
        flip_code:
            1: Flipped Horizontally
            0: Flipped Vertically
            -1: Flipped Horizontally & Vertically
    """

    def __init__(self, flip_code=1):
        assert flip_code in [-1, 0, 1
                             ], "flip_code should be a value in [-1, 0, 1]"
        self.flip_code = flip_code

    def __call__(self, img):
        if random.randint(0, 1) == 1:
            return cv2.flip(img, self.flip_code)
        else:
            return img


class NormalizeImage(object):
    """ normalize image such as substract mean, divide std
    """

    def __init__(self,
                 scale=None,
                 mean=None,
                 std=None,
                 order='chw',
                 output_fp16=False,
                 channel_num=3):
        if isinstance(scale, str):
            scale = eval(scale)
        assert channel_num in [
            3, 4
        ], "channel number of input image should be set to 3 or 4."
        self.channel_num = channel_num
        self.output_dtype = 'float16' if output_fp16 else 'float32'
        self.scale = np.float32(scale if scale is not None else 1.0 / 255.0)
        self.order = order
        mean = mean if mean is not None else [0.485, 0.456, 0.406]
        std = std if std is not None else [0.229, 0.224, 0.225]

        shape = (3, 1, 1) if self.order == 'chw' else (1, 1, 3)
        self.mean = np.array(mean).reshape(shape).astype('float32')
        self.std = np.array(std).reshape(shape).astype('float32')

    def __call__(self, img):
        from PIL import Image
        if isinstance(img, Image.Image):
            img = np.array(img)

        assert isinstance(img,
                          np.ndarray), "invalid input 'img' in NormalizeImage"

        img = (img.astype('float32') * self.scale - self.mean) / self.std
```

```
        if self.channel_num == 4:
            img_h = img.shape[1] if self.order == 'chw' else img.shape[0]
            img_w = img.shape[2] if self.order == 'chw' else img.shape[1]
            pad_zeros = np.zeros(
                (1, img_h, img_w)) if self.order == 'chw' else np.zeros(
                    (img_h, img_w, 1))
            img = (np.concatenate(
                (img, pad_zeros), axis=0)
                   if self.order == 'chw' else np.concatenate(
                       (img, pad_zeros), axis=2))
        return img.astype(self.output_dtype)


class ToCHWImage(object):
    """ convert hwc image to chw image
    """

    def __init__(self):
        pass

    def __call__(self, img):
        from PIL import Image
        if isinstance(img, Image.Image):
            img = np.array(img)

        return img.transpose((2, 0, 1))


def get_result_list(results):
    """ 判断模型输出结果的shape，并返回一维的soft-maxed 数组"""
    shape = np.array(results).shape
    max_val = max(shape)
    max_dim = shape.index(max_val)
    ret = results
    for i in range(max_dim):
        ret = ret[0]
    result_arr = []
    for item in ret:
        real_item = item
        for i in range(max_dim + 1, len(shape)):
            real_item = real_item[0]
        result_arr.append(real_item)
    return np.array(result_arr)
```

7、点击提交即可进入模型验证阶段，验证时间一般需要数十秒，请耐心等待。

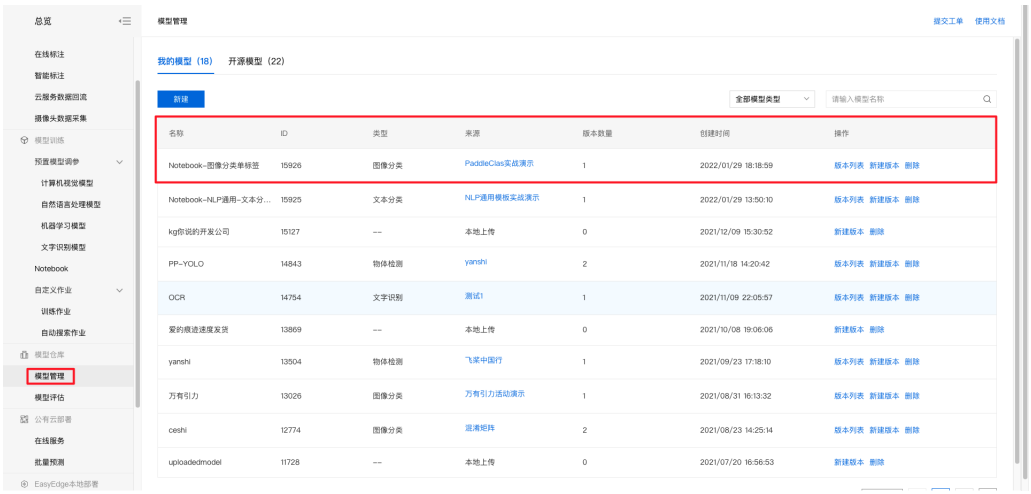


验证通过后，显示有效。



8、点击发布，填写相关信息后，即可发布成功。

9、点击左侧导航栏模型管理，即可查看发布成功的模型。



校验模型

1、点击『版本列表』。



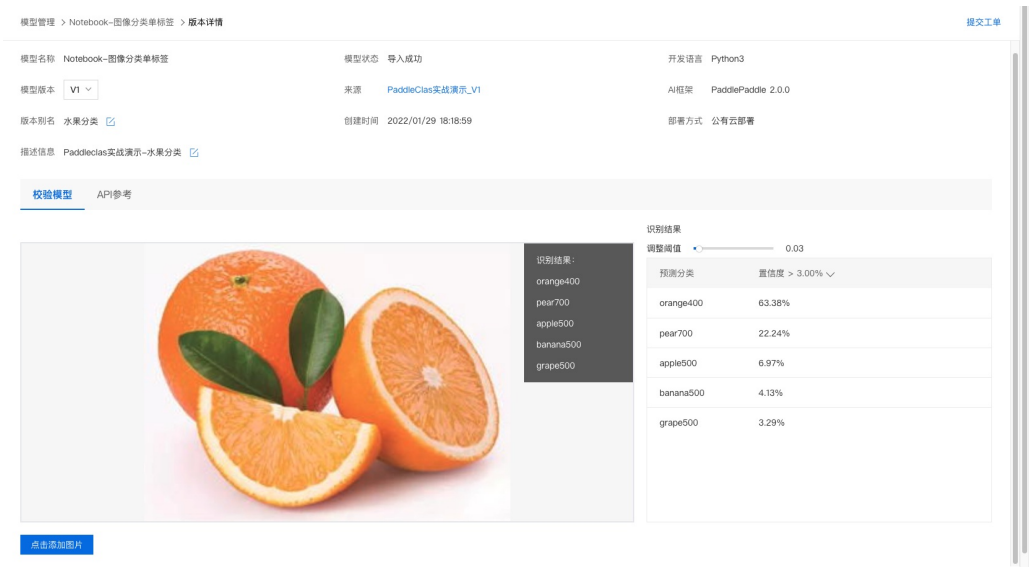
2、点击『校验模型』。



3、点击『启动模型校验』，启动约需5分钟，请耐心等待。



4、上传图像即可开始校验，示例如下：



部署在线服务

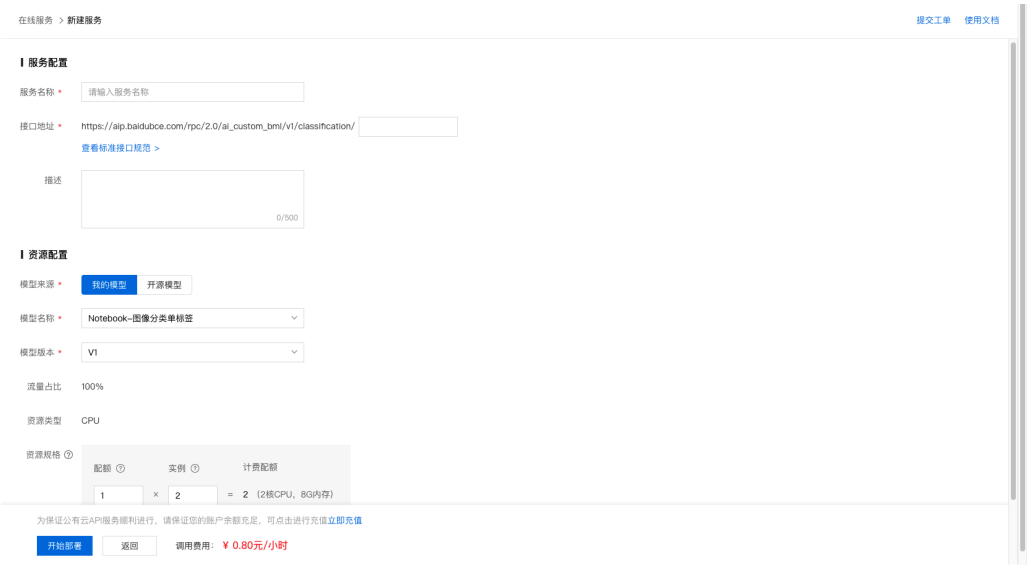
1、点击『版本列表』。



2、点击部署-在线服务。



3、完成信息填写及资源规格选择后，即可开始部署。



4、部署过程需要数十秒时间，请耐心等待。部署完成后，示例如下：



5、API调用方法请参考 公有云部署管理。

基于 Notebook 的物体检测模板使用指南

目录

1.创建并启动Notebook
2.训练物体检测模型
3.配置并发布模型
4.校验模型
5.部署在线服务

基于 Notebook 的物体检测模板使用指南

本文介绍了物体检测模板下，从创建 Notebook 任务到引入数据、训练模型，再到保存模型、部署模型的全流程。

创建并启动Notebook

1、在 BML 左侧导航栏中点击『Notebook』

2、在 Notebook 页面点击『新建』，在弹出框中填写公司/个人信息以及项目信息，示例如下：

填写基础信息



填写项目信息



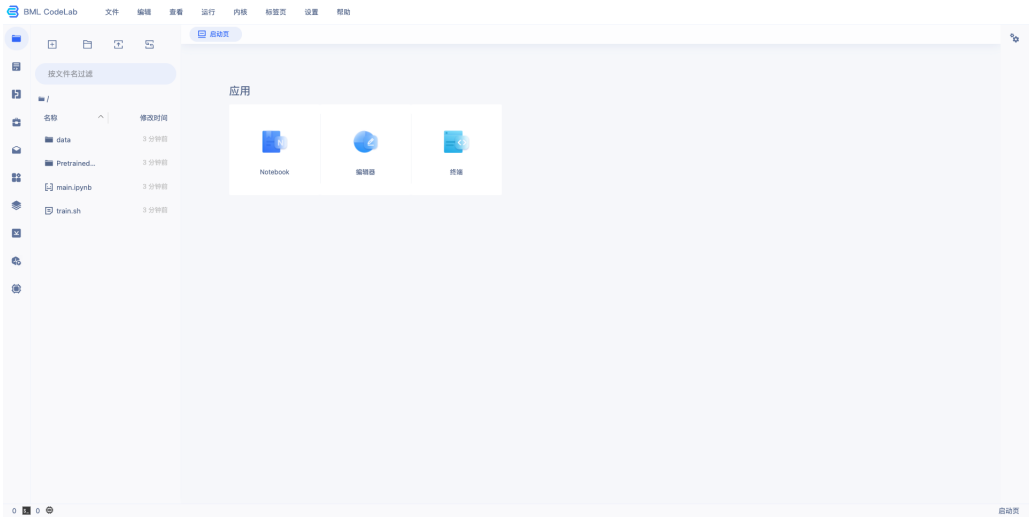
3、对 Notebook 任务操作入口中点击『配置』进行资源配置，示例如下：

选择开发语言、AI 框架，由于本次采用 PaddleDetection 进行演示，所以需要选择 python3.7、PaddlePaddle2.0.0。选择资源规格，由于深度学习所需的训练资源一般较多，需要选择GPU V100的资源规格。



完成配置后点击『确认并启动』，即可启动 Notebook，启动过程中需要完成资源的申请以及实例创建，请耐心等待。

4、等待 Notebook 启动后，点击『打开』，页面跳转到 Notebook，即完成 Notebook 的创建与启动，示例如下：



训练物体检测模型

下载 PaddleDetection 套件

打开进入 Notebook，点击进入终端，输入如下命令切换到 `/home/work/` 目录。

```
cd /home/work/
```

本文以 `PaddleDetection` 代码库 `release/2.3` 分支为例，输入如下命令克隆PaddleDetection代码库并切换至release/2.3分支。整个过程需要数十秒，请耐心等待。

```
###### gitee 国内下载比较快
git clone https://gitee.com/paddlepaddle/PaddleDetection.git -b release/2.3
###### github
###### git clone https://github.com/PaddlePaddle/PaddleDetection.git -b release/2.3
```

安装环境

在终端环境中，安装该版本的 PaddleDetection 代码包依赖的 paddlepaddle-gpu，执行如下命令：

```
python -m pip install paddlepaddle-gpu==2.1.3.post101 -f https://www.paddlepaddle.org.cn/whl/linux/mkl/avx/stable.html
```

安装完成后，使用 python 或 python3 进入python解释器，输入 import paddle ，再输入 paddle.utils.run_check()

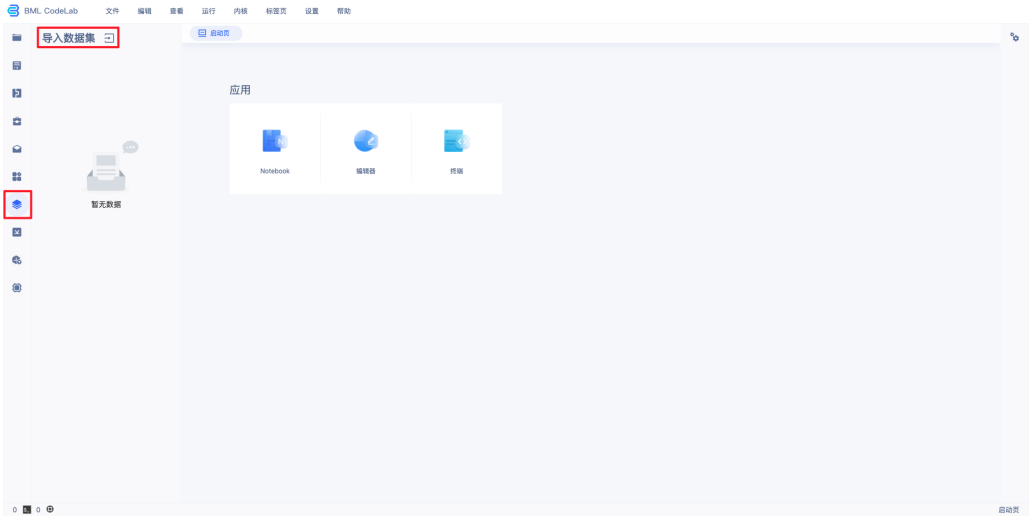
如果出现 PaddlePaddle is installed successfully!，说明成功安装。

准备训练数据

训练数据是模型生产的重要条件，优质的数据集可以很大程度上的提升模型训练效果，准备数据可以参考链接。本文所用的安全帽检测数据集可前往此链接进行下载：下载链接。

1、导入用户数据。

在 Notebook 中并不能直接访问您在 BML 中创建的数据集，需要通过左边选择栏的导入数据集选项，进行数据集导入。导入的数据位于用户目录的 data/ 文件夹（当原始数据集有更新时，不会自动同步，需要手工进行同步）。



注：若在BML中未创建数据集，请先参考 数据服务 ，创建、上传、标注数据集。

2、数据转换。

PaddleDetection 训练所需要的数据格式与 BML 默认的数据格式有所不同，所以需要利用脚本将导入的数据转为 PaddleDetection 支持的数据格式，并进行3：7切分。

PaddleDetection 默认支持的标注格式为 COCO格式，转换脚本如下：

```
 import os
import cv2
import json
import glob
import codecs
import random
from pycocotools.coco import COCO

def parse_bml_json(json_file):
    """
    解析BML标注文件
    :return:
    """
    annos = json.loads(codecs.open(json_file).read())
    labels = annos['labels']
    bboxes = []
    for label in labels:
        x1 = label["x1"]
        y1 = label["y1"]
        x2 = label["x2"]
        y2 = label["y2"]
        id = label["name"]
```

```
        bboxes.append([x1, y1, x2, y2, id])
    return bboxes


def bbox_transform(box):
    """
    x1, y1, x2, y2 转为 x1, y1, width, height
    :return
    """
    box = list(map(lambda x: float(x), box))
    box[2] = box[2] - box[0]
    box[3] = box[3] - box[1]
    return box


def parse_label_list(src_data_dir, save_dir):
    """
    遍历标注文件，获取label_list
    :return:
    """
    label_list = []
    anno_files = glob.glob(src_data_dir + "*.json")
    for anno_f in anno_files:
        annos = json.loads(codecs.open(anno_f).read())
        for lb in annos["labels"]:
            label_list.append(lb["name"])
    label_list = list(set(label_list))
    with codecs.open(os.path.join(save_dir, "label_list.txt"), 'w', encoding="utf-8") as f:
        for id, label in enumerate(label_list):
            f.writelines("%s:%s\n" % (id, label))
    return len(label_list), label_list


def bml2coco(src_dir, coco_json_file):
    """
    BML标注格式转为COCO标注格式
    :return:
    """
    coco_images = []
    coco_annotations = []

    image_id = 0
    anno_id = 0
    image_list = glob.glob(src_dir + "*.[jJPpBb][PpNnMm]*")
    for image_file in image_list:
        anno_f = image_file.split(".")[0] + ".json"
        if not os.path.isfile(anno_f):
            continue
        bboxes = parse_bml_json(anno_f)
        im = cv2.imread(image_file)
        h, w, _ = im.shape
        image_i = {"file_name": os.path.basename(image_file), "id": image_id, "width": w, "height": h}
        coco_images.append(image_i)
        for id, bbox in enumerate(bboxes):
            # bbox : [x1, y1, x2, y2, label_name]
            anno_i = {"image_id": image_id, "bbox": bbox_transform(bbox[:4]), 'category_id': label_list.index(bbox[4]),
                      'id': anno_id, 'area': 1.1, 'iscrowd': 0, "segmentation": None}
            anno_id += 1
            coco_annotations.append(anno_i)

        image_id += 1

    coco_categories = [{"id": id, "name": label_name} for id, label_name in enumerate(label_list)]
    coco_dict = {"info": "info", "licenses": "BMLCloud", "images": coco_images, "annotations": coco_annotations,
                 "categories": coco_categories}
    with open(coco_json_file, 'w', encoding="utf-8") as fin:
        json.dump(coco_dict, fin, ensure_ascii=False)


def split_det_origin_dataset(
```

```
        origin_file_path,
        train_file_path,
        eval_file_path,
        ratio=0.7):
    """
    按比例切分物体检测原始数据集
    :return:
    """
    coco = COCO(origin_file_path)
    img_ids = coco.getImgIds()
    items_num = len(img_ids)
    train_indexes, eval_indexes = random_split_indexes(items_num, ratio)
    train_items = [img_ids[i] for i in train_indexes]
    eval_items = [img_ids[i] for i in eval_indexes]

    dump_det_dataset(coco, train_items, train_file_path)
    dump_det_dataset(coco, eval_items, eval_file_path)

    return items_num, len(train_items), len(eval_items)


def random_split_indexes(items_num, ratio=0.7):
    """
    按比例分割整个list的index
    :return:分割后的两个index子列表
    """
    offset = round(items_num * ratio)
    full_indexes = list(range(items_num))
    random.shuffle(full_indexes)
    sub_indexes_1 = full_indexes[:offset]
    sub_indexes_2 = full_indexes[offset:]

    return sub_indexes_1, sub_indexes_2


def dump_det_dataset(coco, img_id_list, save_file_path):
    """
    物体检测数据集保存
    :return:
    """
    imgs = coco.loadImgs(img_id_list)
    img_anno_ids = coco.getAnnIds(imgIds=img_id_list, iscrowd=0)
    instances = coco.loadAnns(img_anno_ids)
    cat_ids = coco.getCatIds()
    categories = coco.loadCats(cat_ids)
    common_dict = {
        "info": coco.dataset["info"],
        "licenses": coco.dataset["licenses"],
        "categories": categories
    }
    img_dict = {
        "image_nums": len(imgs),
        "images": imgs,
        "annotations": instances
    }
    img_dict.update(common_dict)

    json_file = open(save_file_path, 'w', encoding='UTF-8')
    json.dump(img_dict, json_file)


class_nums, label_list = parse_label_list("/home/work/data/${dataset_id}/", "/home/work/PretrainedModel/")
bml2coco("/home/work/data/${dataset_id}/", "/home/work/PretrainedModel/org_data_list.json")
split_det_origin_dataset("/home/work/PretrainedModel/org_data_list.json", "/home/work/PretrainedModel/train_data_list.json",
"/home/work/PretrainedModel/eval_data_list.json")
```

将上述脚本存放为 `coversion.py` 代码脚本，并将脚本最后两行的 `${dataset_id}` 替换为所指定数据集的 ID（下图红框中的ID），在终端中运行即可。



运行代码。

```
python coversion.py
```

注意：如果报错 `No module named 'pycocotools'`，需要通过如下命令安装相关依赖包，再运行 `coversion.py` 代码。

```
pip install pycocotools
```

运行 `coversion.py` 代码成功之后将在 `PretrainedModel/` 文件夹下生成对应的数据文件，包括 `label_list.txt`、`train_data_list.json`、`eval_data_list.json`、`org_data_list.json`。



训练模型

开发者准备好训练数据和安装环境之后即可开始训练物体检测模型。

1、在终端中打开 `PaddleDetection` 目录。

```
cd /home/work/PaddleDetection
```

2、修改yaml配置文件。

在PaddleDetection 2.0后续版本，采用了模块解耦设计，用户可以组合配置模块实现检测器，并可自由修改覆盖各模块配置，本文以 `configs/yolov3/yolov3_darknet53_270e_coco.yml` 为例：

```
 yolov3_darknet53_270e_coco.yml 主配置入口文件
coco_detection.yml 主要说明了训练数据和验证数据的路径
runtime.yml 主要说明了公共的运行参数，比如说是否使用GPU、每多少个epoch存储checkpoint等
optimizer_270e.yml 主要说明了学习率和优化器的配置。
yolov3_darknet53.yml 主要说明模型、和主干网络的情况。
yolov3_reader.yml 主要说明数据读取器配置，如batch size，并发加载子进程数等，同时包含读取后预处理操作，如resize、数据增强等等
```

需要修改/覆盖的参数均可写在主配置入口文件中，主要修改点为训练、验证数据集路径、运行epoch数、学习率等，修改后的主配置文件如下（注释行即为需要修改的点）：

```
_BASE_: [
 '../datasets/coco_detection.yml',
 '../runtime.yml',
 '_base_/optimizer_270e.yml',
 '_base_/yolov3_darknet53.yml',
 '_base_/yolov3_reader.yml',
]

snapshot_epoch: 5
weights: output/yolov3_darknet53_270e_coco/model_final

###### 预训练权重地址
pretrain_weights: https://paddledet.bj.bcebos.com/models/yolov3_darknet53_270e_coco.pdparams

###### coco_detection.yml
num_classes: 2 #实际类别数
TrainDataset:
 !COCODataSet
   image_dir: data/${dataset_id}/   # 图片地址
   anno_path: PretrainedModel/train_data_list.json # 标注文件
   dataset_dir: /home/work/ # 数据集根目录
   data_fields: ['image', 'gt_bbox', 'gt_class', 'is_crowd']

EvalDataset:
 !COCODataSet
   image_dir: data/${dataset_id}/   # 图片地址
   anno_path: PretrainedModel/eval_data_list.json   # 标注文件
   dataset_dir: /home/work/   # 数据集根目录

###### optimizer_270e.yml
epoch: 50 # 迭代轮数
LearningRate:
 base_lr: 0.0001 # 学习率
 schedulers:
 - !PiecewiseDecay
   gamma: 0.1
   milestones:
   - 30
   - 45
 - !LinearWarmup
   start_factor: 0.
   steps: 400
```

3、训练模型。

在终端中执行以下命令，开始模型训练。

```
cd /home/work/PaddleDetection/
python tools/train.py -c configs/yolov3/yolov3_darknet53_270e_coco.yml --eval
```

注意：如果报错 `No module named 'lap'` 和 `No module named 'motmetrics'` ，则需要通过如下命令安装相关依赖包，再运行 `coversion.py` 代码。（如果缺失其他模块，也可用类似命令下载安装）

```
pip install lap motmetrics
```

4、模型评估

在终端中执行以下命令，开始模型评估。

```
python tools/eval.py -c configs/yolov3/yolov3_darknet53_270e_coco.yml \
                -o weights=output/yolov3_darknet53_270e_coco/model_final
```

运行完成输出如下结果：

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.279
 Average Precision  (AP) @[ IoU=0.50       | area=   all | maxDets=100 ] = 0.534
 Average Precision  (AP) @[ IoU=0.75       | area=   all | maxDets=100 ] = 0.251
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.298
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.086
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.331
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.351
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.351
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
[01/30 02:28:27] ppdet.engine INFO: Total sample number: 150, averge FPS: 25.36864530446337
```

5、模型预测。

在终端中执行以下命令，开始模型预测（注意修改图片路径）。

```
python tools/infer.py -c configs/yolov3/yolov3_darknet53_270e_coco.yml \
                --infer_img=/home/work/data/${task_id}/xxx.jpeg \
                --output_dir=infer_output/ \
                --draw_threshold=0.5 \
                -o weights=output/yolov3_darknet53_270e_coco/model_final
```

6、导出模型。

在终端中执行以下命令，将最佳模型转为可以用于发布的 inference 模型

```
python tools/export_model.py -c configs/yolov3/yolov3_darknet53_270e_coco.yml \
        --output_dir=/home/work/PretrainedModel/ \
        -o weights=output/yolov3_darknet53_270e_coco/model_final
```

在终端中执行以下命令，将导出模型移至 `/home/work/PretrainedModel/` 目录。

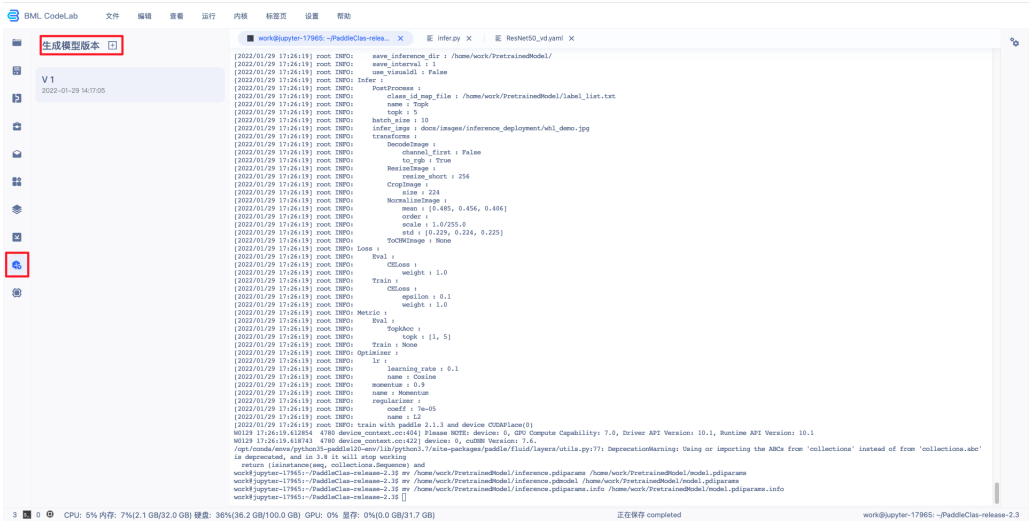
```
mv /home/work/PretrainedModel/yolov3_darknet53_270e_coco/* /home/work/PretrainedModel/
```

7、生成模型版本。

Notebook 中的模型文件只有生成模型版本后，才可以执行发布和部署功能：

- 请确保要保存的模型文件在`/home/work/PretrainedModel`目录下。模型支持版本管理功能，在保存时可以生成新版本也可以覆盖已有的且尚未部署的模型版本，每个版本的模型都可以独立部署。每个模型版本中保存的模型文件大小上限为1.5GB。
- 在保存模式时也可以将训练模型的代码一并保存。代码支持版本管理功能，用户再次启动Notebook时，可以使用指定的代码版本来初始化Notebook工作空间即/home/work目录下data以外的空间。每个代码版本中保存的文件大小上限为150M。

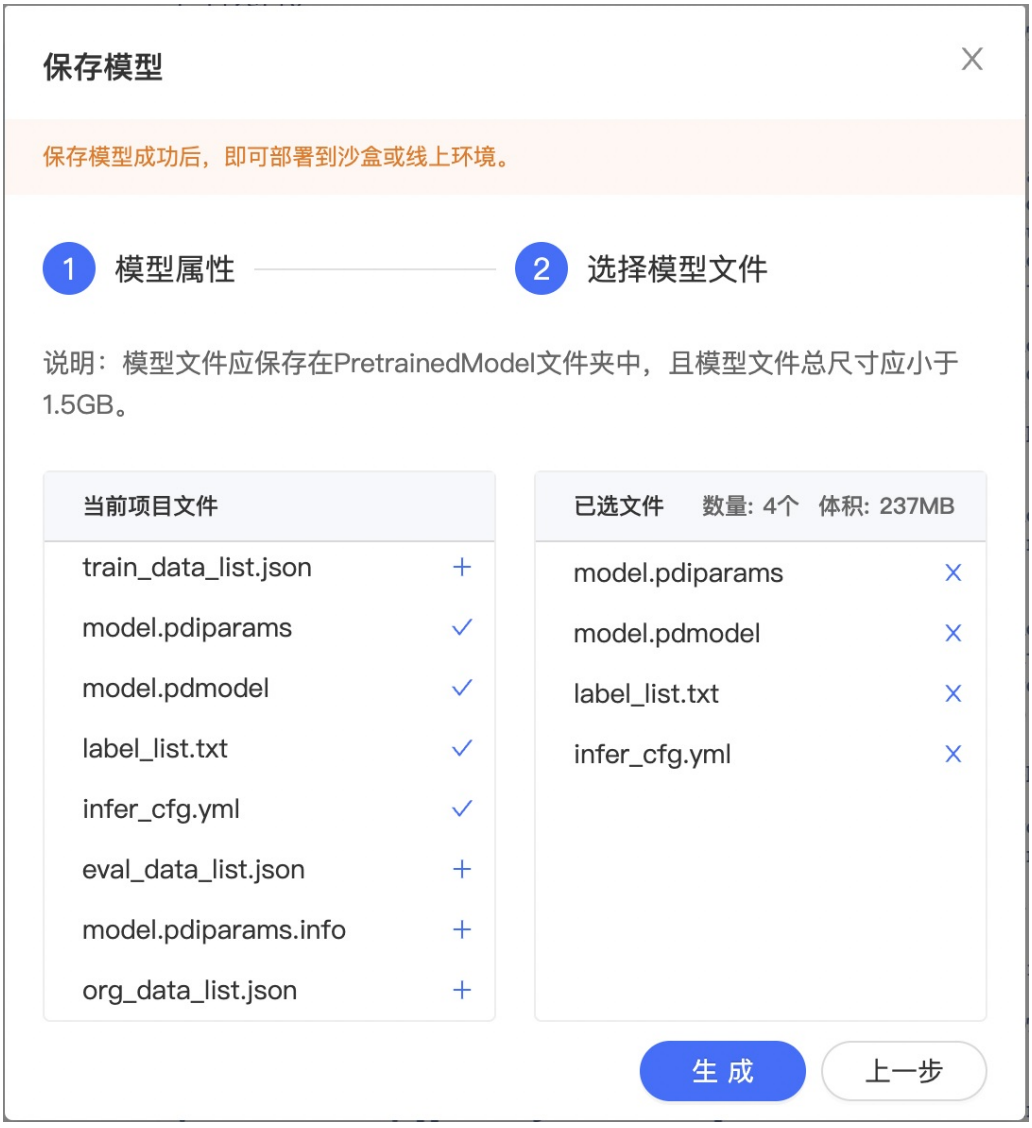
点击左侧导航栏中的生成模型版本组件，打开弹窗填写信息。



模型属性-选择 AI 框架选择 PaddlePaddle2.0.0，若上一次操作中进行了代码保存，可在“代码版本”选择对应的代码版本。



选择模型文件-选择 label_list.txt、model.pdiparams、model.pdmodel 、 infer_cfg.yaml 文件。



点击『生成』即可生成模型版本，生成模型版本一般需要数十秒，请耐心等待。

配置并发布模型

BML NoteBook 的物体检测模板产出的模型支持进行部署，下面以 PaddleDetection 的模型为例，详细介绍如何配置模型：

1、查看前置条件是否满足：需要训练完成，并生成了相应的模型生成版本（详见训练模型的第六步）。

2、回到 BML Notebook 列表页，点击对应 Notebook 项目的『模型发布列表』即可进入配置页面。



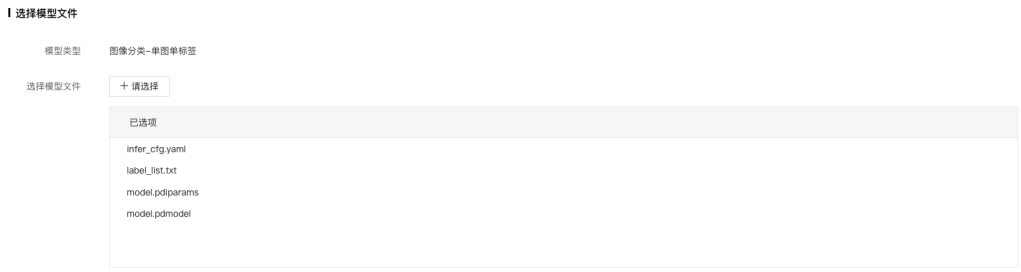
3、点击配置，即可进入配置流程。



4、填写模型信息。



5、选择待发布的模型文件，点击确定按钮。



对于 Paddle2.x 的模型而言：

- 网络结构文件 `model.pdmodel`：必需选择，且名字固定。
- 网络参数文件 `model.pdiparams`：必需选择，且名字固定。
- 模型标签文件：`label_list.txt`，非必须，主要看自定义逻辑代码是如何实现的。本文中在【配置出入参及数据处理逻辑脚本】的脚本代码里面会用到，所以需要选择。
- 预/后处理配置文件等其他文件：`infer_cfg.yaml`，非必须，主要看自定义逻辑代码是如何实现的。本文中在【配置出入参及数据处理逻辑脚本】的脚本代码里面会用到，所以需要选择。

6、配置出入参及数据逻辑处理。

这部分配置主要实现自定义的模型预处理和后处理逻辑。该脚本用于将选择的模型文件发布成模型服务。用户可以通过修改 `PredictWrapper` 的 `preprocess` 和 `postprocess` 方法来实现自定义的请求预处理和预测结果后处理。当提交该脚本时，系统会根据用户选择的模型文件和脚本内容，来验证是否可以启动模型服务，只有验证通过，才可以进行模型效果校验以及将模型发布到模型仓库。

实现脚本有一些建议和限制：

- `CustomException` 必需存在且是异常类；在自定义的逻辑中，建议当处理进入错误的分支时，抛出 `CustomException` 并指定 `message`，指定

的 message 在请求回包中会作为 error_msg 返回。

- PredictWrapper 类必需存在，且必需包含 preprocess 和 postprocess 两个方法。
- PredictWrapper 的 preprocess 和 postprocess 方法，是用户自定义模型服务请求预处理和预测结果后处理的入口。
- preprocess 方法接收的第一个参数为用户请求的 json 字典，对于图像类服务，传入图像的参数 key 必须是"image"，且传入的是图片的 base64 编码。
- 系统会根据 postprocess 方法的返回结果 result 类型的不同，做以下处理：

```
 - dict: 不作修改
- list: 转换为 {"results": result}
```

  - 其他: 转换为 {"result": result}
  - 最终的处理结果可以转换为json字符串，以确保能够正常将结果作为请求回包返回

- 单击“提交”，完成模型配置。 提交后，系统会自动对当前版本模型进行代码验证，通过后模型会更新为“有效”状态。

配置出入参及数据处理逻辑脚本：实现图片的预处理和模型输出结果后处理的逻辑；这一步是比较关键，但也比较复杂的一步。上面介绍了脚本实现时的限制和建议。这里针对PaddleDetection套件，实现了一套对应的脚本代码（在后面），可以点击立即编辑，将脚本代码拷贝替换掉。



注：

可以看下平台预置代码文件，以及各个类及函数的注释了解实现细节。这里贴了PaddleDetection套件对应的脚本文件，整个代码比较长，但大部分内容都拷贝于PaddleDetection套件的推理示例。如果自身训练模型比较特殊，当前脚本支持不了，需要自己去PaddleDetection套件中寻找逻辑，并更新到该脚本中。

PaddleDetection脚本样例：

```
 ###### !/usr/bin/env python
###### -*- coding: utf-8 -*-
###### ********************************************************************************
#
###### Copyright (c) 2020 Baidu.com, Inc. All Rights Reserved
#
###### ********************************************************************************

###### 注意事项：
###### 该脚本用于将通过notebook训练产出的模型发布成模型服务
###### 用户可以通过修改PredictWrapper的preprocess和postprocess方法来实现自定义的请求预处理和预测结果后处理
###### 当在EasyDL提交该脚本时，系统会根据用户选择的模型文件和脚本内容，来验证是否可以启动模型服务，如果验证通过，即可进行模型效果校验以及部署模型服务
###### 下面是修改脚本的一些限制和建议：
###### 1. CustomException必须存在且是异常类；在自定义的逻辑中，建议当处理进入错误的分支时,抛出CustomException并指定message，指定的message在请求回包中会作为error_msg返回；
###### 2. PredictWrapper类必须存在，且必需包含preprocess和postprocess两个方法；
###### 3. PredictWrapper的preprocess和postprocess方法，是用户自定义模型服务请求预处理和预测结果后处理的入口；
###### 4. preprocess方法接收的第一个参数为用户请求的json字典，对于图像类服务，传入图像的参数key必须是"image"，且传入的是图片的base64编码
###### 5. 系统会根据postprocess方法的返回结果`result`类型的不同，做以下处理：
###### dict: 不作修改
###### list: 转换为 {"results": result}
###### 其他: 转换为 {"result": result}
###### 处理结果将转换为json字符串，以确保系统能正常将结果作为请求回包返回。

import base64
import cv2
```

```
import numpy as np
import yaml

DEFAULT_THRESHOLD = 0.5


class CustomException(RuntimeError):
    """
    进行模型验证和部署服务必需的异常类，缺少该类在代码验证时将会失败
    在处理异常数据或者请求时，推荐在`PredictWrapper`中的自定义预处理preprocess和后处理postprocess函数中抛出`CustomException`类，
    并为`message`指定准确可读的错误信息，以便在服务响应包中的`error_msg`参数中返回。
    """
    def __init__(self, message, orig_error=None):
        """ 根据`message`初始化 """
        super(CustomException, self).__init__(message)
        self.orig_error = orig_error


class PredictWrapper(object):
    """ 模型服务预测封装类，支持用户自定义对服务请求数据的预处理和模型预测结果的后处理函数 """

    def __init__(self, model_path):
        """
        根据`model_path`初始化`PredictWrapper`类，如解析label_list.txt，加载模型输出标签id和标签名称的映射关系
        :param model_path: 该目录下存放了用户选择的模型版本中包含的所有文件
        """
        # 加载推理配置文件，获取【预处理配置】及【标签id和名称的映射关系】
        conf_path = '{model_path}/{conf_file}'.format(model_path=model_path, conf_file='infer_cfg.yml')
        with open(conf_path) as conf_fin:
            infer_conf = yaml.load(conf_fin, Loader=yaml.FullLoader)
            self._preprocess_args = infer_conf["Preprocess"]
            self._label_list = infer_conf["label_list"]

    def preprocess(self, request_body, **preprocess_args):
        """
        自定义对请求体的预处理，针对图像类模型服务，包括对图片对图像的解析、转化等
        :param request_body: 请求体的json字典
        :param preprocess_args: 从`{model_path}/preprocess_args.json`中加载的预处理参数字典，json文件不存在时，传入为空字典
        :return:
            data: 用于模型预测的输入。注意：data结构为dict，key为模型输入节点的名称，value为对应需要喂入的值，batch只能为1
            infer_args: 用于模型预测的其他参数
            request_context: 透传给自定义后处理函数`postprocess`的参数，例如指定返回预测结果的top N，过滤低score的阈值threshold.
        """
        # decode image from base64 string in request
        try:
            image_b64 = request_body['image']
            img_bin = base64.b64decode(image_b64)
        except KeyError:
            raise CustomException('Missing required parameter')
        except Exception:
            raise CustomException('Invalid BASE64')

        data = np.frombuffer(img_bin, dtype='uint8')
        im = cv2.imdecode(data, 1)  # BGR mode, but need RGB mode
        im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)

        # paddle detection preprocess
        # code: https://github.com/PaddlePaddle/PaddleDetection/blob/develop/deploy/python/infer.py#L99
        try:
            preprocess_ops = []
            for op_info in self._preprocess_args:
                new_op_info = op_info.copy()
                op_type = new_op_info.pop('type')
                preprocess_ops.append(eval(op_type)(**new_op_info))

            input_im_lst = []
            input_im_info_lst = []
            im, im_info = preprocess(im, preprocess_ops)
            input_im_lst.append(im)
```

```
            input_im_info_lst.append(im_info)
            input_info = create_inputs(input_im_lst, input_im_info_lst)
        except Exception:
            raise CustomException('Failed decoding input')

        return input_info, {}, {"threshold": request_body.get("threshold", DEFAULT_THRESHOLD)}

    def postprocess(self, infer_result, request_context, **postprocess_args):
        """
        自定义对图像分类模型输出结果的后处理，例如根据score对label进行排序，获取top N分类结果等
        :param infer_result: fluid模型的预测结果
        :param request_context: 自定义预处理函数中返回的`request context`
        :param postprocess_args: 从`{model_path}/postprocess_args.json`中加载的后处理参数字典，json文件不存在时，传入为空字典
        :return: request results 请求的处理结果
        """
        threshold = request_context["threshold"]
        boxes = infer_result[0]
        bboxes_num = int(infer_result[1][0])

        results = []
        for i in range(bboxes_num):
            box_info = boxes[i, :]
            label = int(box_info[0])
            score = float(box_info[1])
            x1 = int(box_info[2])
            y1 = int(box_info[3])
            x2 = int(box_info[4])
            y2 = int(box_info[5])

            if score < threshold:
                continue
            results.append({
                'name': self._label_list[label],
                'score': score,
                'location': {
                    'left': x1,
                    'top': y1,
                    'width': x2 - x1,
                    'height': y2 - y1,
                }
            })
        return results


###### copy from PaddleDetection: https://github.com/PaddlePaddle/PaddleDetection/blob/develop/deploy/python/preprocess.py
###### note: Only copy used preprocess OPs, if model used other op, show copy into this file
def preprocess(im, preprocess_ops):
    # process image by preprocess_ops
    im_info = {
        'scale_factor': np.array(
            [1., 1.], dtype=np.float32),
        'im_shape': None,
    }
    im, im_info = decode_image(im, im_info)
    for operator in preprocess_ops:
        im, im_info = operator(im, im_info)
    return im, im_info


def decode_image(im_file, im_info):
    """read rgb image
    Args:
        im_file (str|np.ndarray\bytes): input can be image path or np.ndarray
        im_info (dict): info of image
    Returns:
        im (np.ndarray):  processed image (np.ndarray)
        im_info (dict): info of processed image
    """
    if isinstance(im_file, str):
        with open(im_file, 'rb') as f:
```

```
            im_read = f.read()
        data = np.frombuffer(im_read, dtype='uint8')
        im = cv2.imdecode(data, 1)  # BGR mode, but need RGB mode
        im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
    else:
        im = im_file
    im_info['im_shape'] = np.array(im.shape[:2], dtype=np.float32)
    im_info['scale_factor'] = np.array([1., 1.], dtype=np.float32)
    return im, im_info


class Resize(object):
    """resize image by target_size and max_size
    Args:
        target_size (int): the target size of image
        keep_ratio (bool): whether keep_ratio or not, default true
        interp (int): method of resize
    """

    def __init__(self, target_size, keep_ratio=True, interp=cv2.INTER_LINEAR):
        if isinstance(target_size, int):
            target_size = [target_size, target_size]
        self.target_size = target_size
        self.keep_ratio = keep_ratio
        self.interp = interp

    def __call__(self, im, im_info):
        """
        Args:
            im (np.ndarray): image (np.ndarray)
            im_info (dict): info of image
        Returns:
            im (np.ndarray):  processed image (np.ndarray)
            im_info (dict): info of processed image
        """
        assert len(self.target_size) == 2
        assert self.target_size[0] > 0 and self.target_size[1] > 0
        im_channel = im.shape[2]
        im_scale_y, im_scale_x = self.generate_scale(im)
        im = cv2.resize(
            im,
            None,
            None,
            fx=im_scale_x,
            fy=im_scale_y,
            interpolation=self.interp)
        im_info['im_shape'] = np.array(im.shape[:2]).astype('float32')
        im_info['scale_factor'] = np.array(
            [im_scale_y, im_scale_x]).astype('float32')
        return im, im_info

    def generate_scale(self, im):
        """
        Args:
            im (np.ndarray): image (np.ndarray)
        Returns:
            im_scale_x: the resize ratio of X
            im_scale_y: the resize ratio of Y
        """
        origin_shape = im.shape[:2]
        im_c = im.shape[2]
        if self.keep_ratio:
            im_size_min = np.min(origin_shape)
            im_size_max = np.max(origin_shape)
            target_size_min = np.min(self.target_size)
            target_size_max = np.max(self.target_size)
            im_scale = float(target_size_min) / float(im_size_min)
            if np.round(im_scale * im_size_max) > target_size_max:
                im_scale = float(target_size_max) / float(im_size_max)
            im_scale_x = im_scale
```

```
            im_scale_y = im_scale
        else:
            resize_h, resize_w = self.target_size
            im_scale_y = resize_h / float(origin_shape[0])
            im_scale_x = resize_w / float(origin_shape[1])
        return im_scale_y, im_scale_x


class NormalizeImage(object):
    """normalize image
    Args:
        mean (list): im - mean
        std (list): im / std
        is_scale (bool): whether need im / 255
        is_channel_first (bool): if True: image shape is CHW, else: HWC
    """

    def __init__(self, mean, std, is_scale=True):
        self.mean = mean
        self.std = std
        self.is_scale = is_scale

    def __call__(self, im, im_info):
        """
        Args:
            im (np.ndarray): image (np.ndarray)
            im_info (dict): info of image
        Returns:
            im (np.ndarray):  processed image (np.ndarray)
            im_info (dict): info of processed image
        """
        im = im.astype(np.float32, copy=False)
        mean = np.array(self.mean)[np.newaxis, np.newaxis, :]
        std = np.array(self.std)[np.newaxis, np.newaxis, :]

        if self.is_scale:
            im = im / 255.0
        im -= mean
        im /= std
        return im, im_info


class Permute(object):
    """permute image
    Args:
        to_bgr (bool): whether convert RGB to BGR
        channel_first (bool): whether convert HWC to CHW
    """

    def __init__(self, ):
        super(Permute, self).__init__()

    def __call__(self, im, im_info):
        """
        Args:
            im (np.ndarray): image (np.ndarray)
            im_info (dict): info of image
        Returns:
            im (np.ndarray):  processed image (np.ndarray)
            im_info (dict): info of processed image
        """
        im = im.transpose((2, 0, 1)).copy()
        return im, im_info


class PadStride(object):
    """ padding image for model with FPN, instead PadBatch(pad_to_stride) in original config
    Args:
        stride (bool): model with FPN need image shape % stride == 0
    """
```

```
    def __init__(self, stride=0):
        self.coarsest_stride = stride

    def __call__(self, im, im_info):
        """
        Args:
            im (np.ndarray): image (np.ndarray)
            im_info (dict): info of image
        Returns:
            im (np.ndarray):  processed image (np.ndarray)
            im_info (dict): info of processed image
        """
        coarsest_stride = self.coarsest_stride
        if coarsest_stride <= 0:
            return im, im_info
        im_c, im_h, im_w = im.shape
        pad_h = int(np.ceil(float(im_h) / coarsest_stride) * coarsest_stride)
        pad_w = int(np.ceil(float(im_w) / coarsest_stride) * coarsest_stride)
        padding_im = np.zeros((im_c, pad_h, pad_w), dtype=np.float32)
        padding_im[:, :im_h, :im_w] = im
        return padding_im, im_info


class LetterBoxResize(object):
    def __init__(self, target_size):
        """
        Resize image to target size, convert normalized xywh to pixel xyxy
        format ([x_center, y_center, width, height] -> [x0, y0, x1, y1]).
        Args:
            target_size (int|list): image target size.
        """
        super(LetterBoxResize, self).__init__()
        if isinstance(target_size, int):
            target_size = [target_size, target_size]
        self.target_size = target_size

    def letterbox(self, img, height, width, color=(127.5, 127.5, 127.5)):
        # letterbox: resize a rectangular image to a padded rectangular
        shape = img.shape[:2]  # [height, width]
        ratio_h = float(height) / shape[0]
        ratio_w = float(width) / shape[1]
        ratio = min(ratio_h, ratio_w)
        new_shape = (round(shape[1] * ratio),
                     round(shape[0] * ratio))  # [width, height]
        padw = (width - new_shape[0]) / 2
        padh = (height - new_shape[1]) / 2
        top, bottom = round(padh - 0.1), round(padh + 0.1)
        left, right = round(padw - 0.1), round(padw + 0.1)

        img = cv2.resize(
            img, new_shape, interpolation=cv2.INTER_AREA)  # resized, no border
        img = cv2.copyMakeBorder(
            img, top, bottom, left, right, cv2.BORDER_CONSTANT,
            value=color)  # padded rectangular
        return img, ratio, padw, padh

    def __call__(self, im, im_info):
        """
        Args:
            im (np.ndarray): image (np.ndarray)
            im_info (dict): info of image
        Returns:
            im (np.ndarray):  processed image (np.ndarray)
            im_info (dict): info of processed image
        """
        assert len(self.target_size) == 2
        assert self.target_size[0] > 0 and self.target_size[1] > 0
        height, width = self.target_size
        h, w = im.shape[:2]
```

```
        im, ratio, padw, padh = self.letterbox(im, height=height, width=width)

        new_shape = [round(h * ratio), round(w * ratio)]
        im_info['im_shape'] = np.array(new_shape, dtype=np.float32)
        im_info['scale_factor'] = np.array([ratio, ratio], dtype=np.float32)
        return im, im_info


def create_inputs(imgs, im_info):
    """generate input for different model type
    Args:
        imgs (list(numpy)): list of images (np.ndarray)
        im_info (list(dict)): list of image info
    Returns:
        inputs (dict): input of model
    """
    inputs = {}

    im_shape = []
    scale_factor = []
    if len(imgs) == 1:
        inputs['image'] = np.array((imgs[0], )).astype('float32')
        inputs['im_shape'] = np.array(
            (im_info[0]['im_shape'], )).astype('float32')
        inputs['scale_factor'] = np.array(
            (im_info[0]['scale_factor'], )).astype('float32')
        return inputs

    for e in im_info:
        im_shape.append(np.array((e['im_shape'], )).astype('float32'))
        scale_factor.append(np.array((e['scale_factor'], )).astype('float32'))

    inputs['im_shape'] = np.concatenate(im_shape, axis=0)
    inputs['scale_factor'] = np.concatenate(scale_factor, axis=0)

    imgs_shape = [[e.shape[1], e.shape[2]] for e in imgs]
    max_shape_h = max([e[0] for e in imgs_shape])
    max_shape_w = max([e[1] for e in imgs_shape])
    padding_imgs = []
    for img in imgs:
        im_c, im_h, im_w = img.shape[:]
        padding_im = np.zeros(
            (im_c, max_shape_h, max_shape_w), dtype=np.float32)
        padding_im[:, :im_h, :im_w] = img
        padding_imgs.append(padding_im)
    inputs['image'] = np.stack(padding_imgs, axis=0)
    return inputs
```

7、点击提交即可进入模型验证阶段，验证时间一般需要数十秒，请耐心等待。

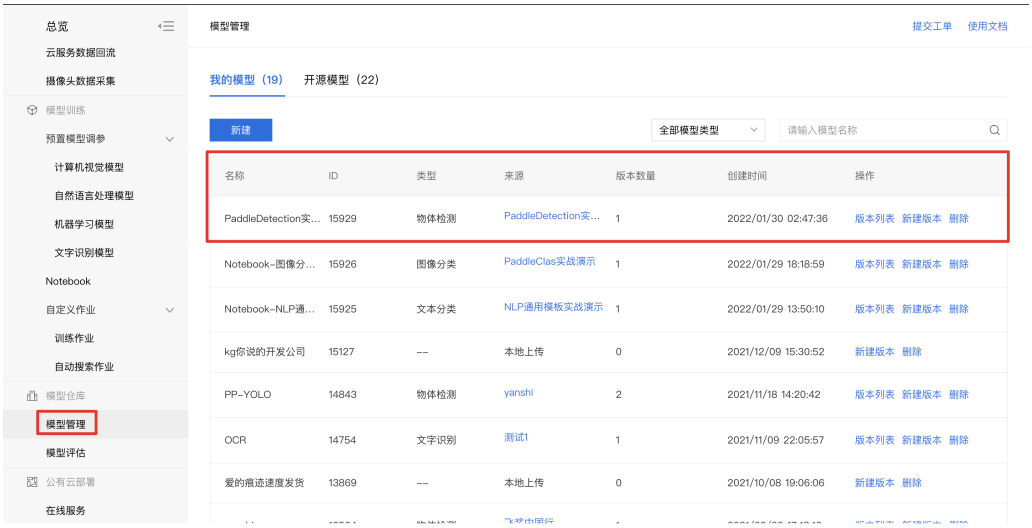


验证通过后，显示有效。



8、点击发布，填写相关信息后，即可发布成功。

9、点击左侧导航栏模型管理，即可查看发布成功的模型。



校验模型

1、点击『版本列表』。



2、点击『校验模型』。



3、点击『启动模型校验』，启动约需5分钟，请耐心等待。



4、上传图像即可开始校验，示例如下：



部署在线服务

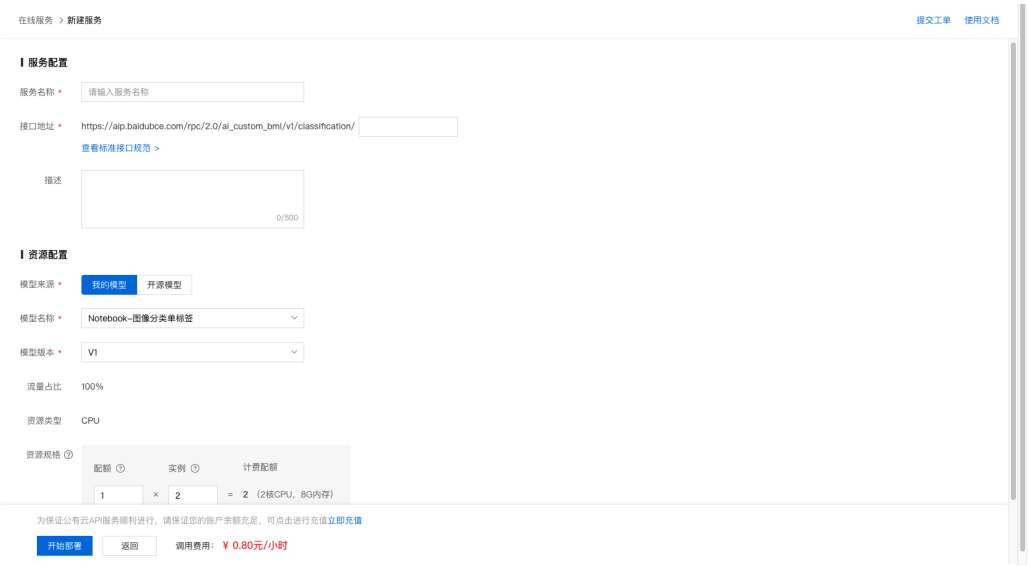
1、点击『版本列表』。



2、点击部署-在线服务。



3、完成信息填写及资源规格选择后，即可开始部署。



4、部署过程需要数十秒时间，请耐心等待。部署完成后，示例如下：



5、API调用方法请参考 公有云部署管理。

总结

- 更多丰富的内容与示例可以参考PaddleDetection的github与教程文档。
  - PaddleDetection github地址：https://github.com/PaddlePaddle/PaddleDetection/
  - PaddleDetection教程文档地址：https://github.com/PaddlePaddle/PaddleDetection/blob/release/2.3/docs/tutorials/GETTING_STARTED_cn.md
- 如果在使用PaddleDetection的过程中遇到问题，欢迎去PaddleDetection的github上提issue：https://github.com/PaddlePaddle/PaddleDetection/issues/new

基于 Notebook 的通用模板使用指南

目录

1.创建并启动Notebook
2.训练物体检测模型

3.配置并发布模型
4.校验模型
5.部署在线服务

基于 Notebook 的通用模板使用指南

本文采用物体检测模型的开发过程为例，介绍通用模板从创建 Notebook 任务到引入数据、训练模型，再到保存模型、部署模型的全流程。

创建并启动Notebook

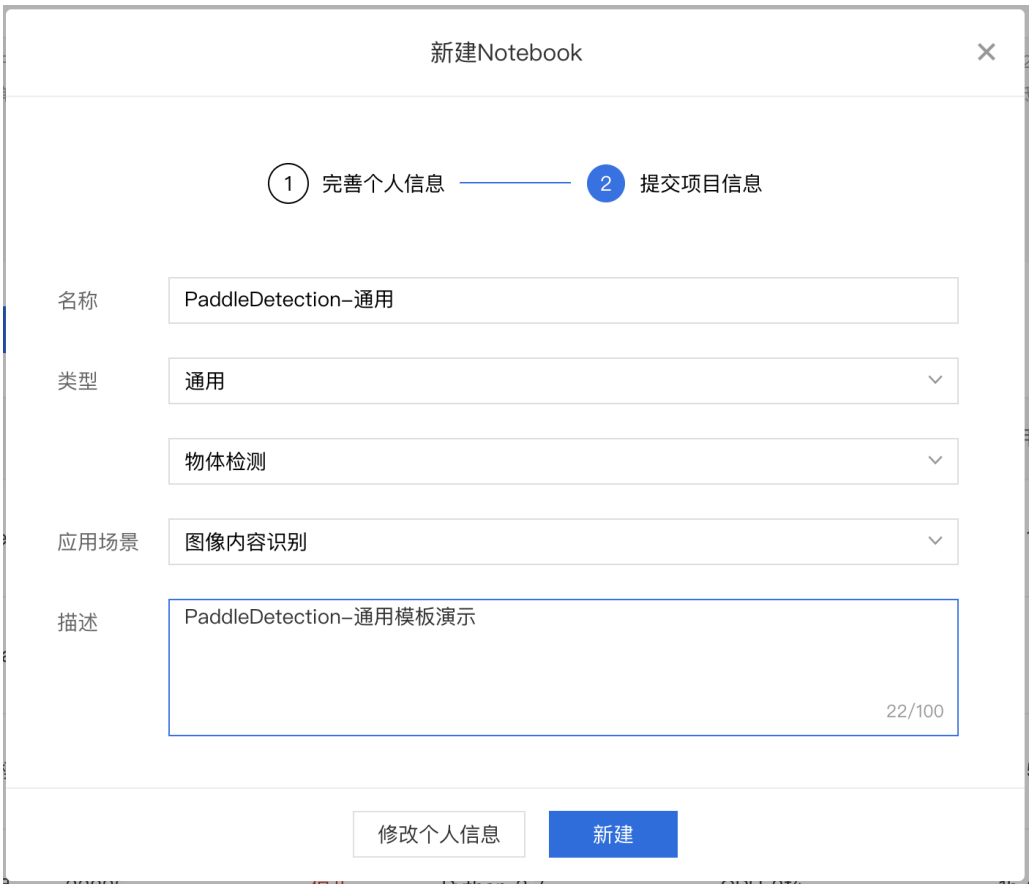
1、在 BML 左侧导航栏中点击『Notebook』

2、在 Notebook 页面点击『新建』，在弹出框中填写公司/个人信息以及项目信息，示例如下：

填写基础信息



填写项目信息



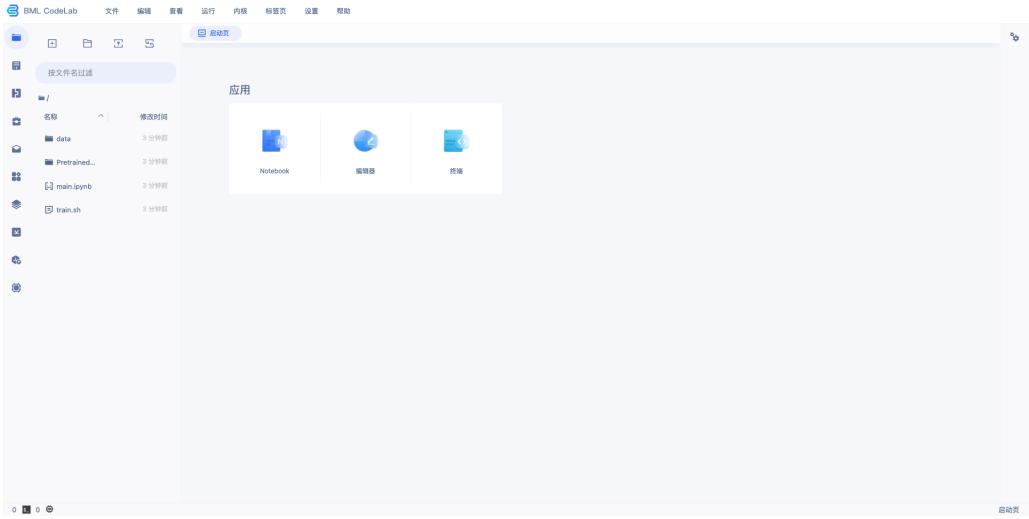
3、对 Notebook 任务操作入口中点击『配置』进行资源配置，示例如下：

选择开发语言、AI 框架，由于本次采用 PaddleClas 进行演示，所以需要选择 python3.7、PaddlePaddle2.0.0。选择资源规格，由于深度学习所需的训练资源一般较多，需要选择GPU V100的资源规格。



完成配置后点击『确认并启动』，即可启动 Notebook，启动过程中需要完成资源的申请以及实例创建，请耐心等待。

4、等待 Notebook 启动后，点击『打开』，页面跳转到 Notebook，即完成 Notebook 的创建与启动，示例如下：



训练物体检测模型

下载 PaddleDetection 套件

打开进入 Notebook，点击进入终端，输入如下命令切换到 `/home/work/` 目录。

```
cd /home/work/
```

本文以 `PaddleDetection` 代码库 `release/2.3` 分支为例，输入如下命令克隆PaddleDetection代码库并切换至release/2.3分支。整个过程需要数十秒，请耐心等待。

```
###### gitee 国内下载比较快
git clone https://gitee.com/paddlepaddle/PaddleDetection.git -b release/2.3
###### github
###### git clone https://github.com/PaddlePaddle/PaddleDetection.git -b release/2.3
```

安装环境

在终端环境中，安装该版本的 `PaddleDetection` 代码包依赖的 `paddlepaddle-gpu`，执行如下命令：

```
python -m pip install paddlepaddle-gpu==2.1.3.post101 -f https://www.paddlepaddle.org.cn/whl/linux/mkl/avx/stable.html
```

安装完成后，使用 `python` 或 `python3` 进入python解释器，输入 `import paddle` ，再输入 `paddle.utils.run_check()`

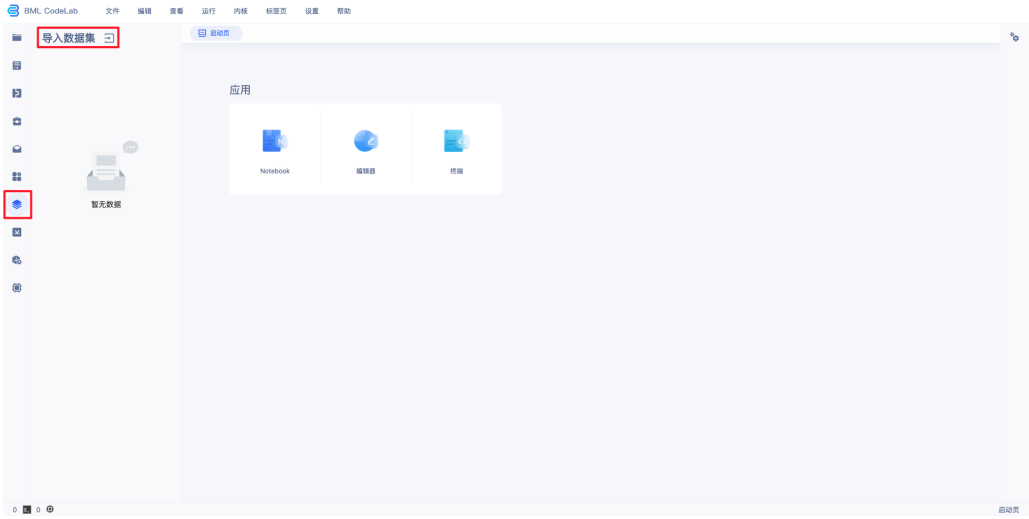
如果出现 `PaddlePaddle is installed successfully!`，说明成功安装。

准备训练数据

训练数据是模型生产的重要条件，优质的数据集可以很大程度上的提升模型训练效果，准备数据可以参考链接。本文所用的安全帽检测数据集可前往此链接进行下载：下载链接。

1、导入用户数据。

在 Notebook 中并不能直接访问您在 BML 中创建的数据集，需要通过左边选择栏的导入数据集选项，进行数据集导入。导入的数据位于用户目录的 `data/` 文件夹（当原始数据集有更新时，不会自动同步，需要手工进行同步）。



注：若在BML中未创建数据集，请先参考 数据服务 ，创建、上传、标注数据集。

2、数据转换。

`PaddleDetection` 训练所需要的数据格式与 `BML` 默认的数据格式有所不同，所以需要利用脚本将导入的数据转为 `PaddleDetection` 支持的数据格式，并进行3：7切分。

`PaddleDetection` 默认支持的标注格式为 `COCO格式`，转换脚本如下：

```
 import os
import cv2
import json
import glob
import codecs
import random
from pycocotools.coco import COCO

def parse_bml_json(json_file):
    """
    解析BML标注文件
    :return:
    """
    annos = json.loads(codecs.open(json_file).read())
    labels = annos['labels']
    bboxes = []
    for label in labels:
        x1 = label["x1"]
        y1 = label["y1"]
        x2 = label["x2"]
        y2 = label["y2"]
        id = label["name"]
        bboxes.append([x1, y1, x2, y2, id])
    return bboxes


def bbox_transform(box):
    """
    x1, y1, x2, y2 转为 x1, y1, width, height
    :return
    """
    box = list(map(lambda x: float(x), box))
    box[2] = box[2] - box[0]
    box[3] = box[3] - box[1]
    return box


def parse_label_list(src_data_dir, save_dir):
    """
    遍历标注文件，获取label_list
    :return:
    """
    label_list = []
```

```
    label_list = []
    anno_files = glob.glob(src_data_dir + "*.json")
    for anno_f in anno_files:
        annos = json.loads(codecs.open(anno_f).read())
        for lb in annos["labels"]:
            label_list.append(lb["name"])
    label_list = list(set(label_list))
    with codecs.open(os.path.join(save_dir, "label_list.txt"), 'w', encoding="utf-8") as f:
        for id, label in enumerate(label_list):
            f.writelines("%s:%s\n" % (id, label))
    return len(label_list), label_list


def bml2coco(src_dir, coco_json_file):
    """
    BML标注格式转为COCO标注格式
    :return:
    """
    coco_images = []
    coco_annotations = []

    image_id = 0
    anno_id = 0
    image_list = glob.glob(src_dir + "*.[jJPpBb][PpNnMm]*")
    for image_file in image_list:
        anno_f = image_file.split(".")[0] + ".json"
        if not os.path.isfile(anno_f):
            continue
        bboxes = parse_bml_json(anno_f)
        im = cv2.imread(image_file)
        h, w, _ = im.shape
        image_i = {"file_name": os.path.basename(image_file), "id": image_id, "width": w, "height": h}
        coco_images.append(image_i)
        for id, bbox in enumerate(bboxes):
            # bbox : [x1, y1, x2, y2, label_name]
            anno_i = {"image_id": image_id, "bbox": bbox_transform(bbox[:4]), 'category_id': label_list.index(bbox[4]),
                      'id': anno_id, 'area': 1.1, 'iscrowd': 0, "segmentation": None}
            anno_id += 1
            coco_annotations.append(anno_i)

        image_id += 1

    coco_categories = [{"id": id, "name": label_name} for id, label_name in enumerate(label_list)]
    coco_dict = {"info": "info", "licenses": "BMLCloud", "images": coco_images, "annotations": coco_annotations,
                 "categories": coco_categories}
    with open(coco_json_file, 'w', encoding="utf-8") as fin:
        json.dump(coco_dict, fin, ensure_ascii=False)


def split_det_origin_dataset(
        origin_file_path,
        train_file_path,
        eval_file_path,
        ratio=0.7):
    """
    按比例切分物体检测原始数据集
    :return:
    """
    coco = COCO(origin_file_path)
    img_ids = coco.getImgIds()
    items_num = len(img_ids)
    train_indexes, eval_indexes = random_split_indexes(items_num, ratio)
    train_items = [img_ids[i] for i in train_indexes]
    eval_items = [img_ids[i] for i in eval_indexes]

    dump_det_dataset(coco, train_items, train_file_path)
    dump_det_dataset(coco, eval_items, eval_file_path)

    return items_num, len(train_items), len(eval_items)
```

```
def random_split_indexes(items_num, ratio=0.7):
    """
    按比例分割整个list的index
    :return:分割后的两个index子列表
    """
    offset = round(items_num * ratio)
    full_indexes = list(range(items_num))
    random.shuffle(full_indexes)
    sub_indexes_1 = full_indexes[:offset]
    sub_indexes_2 = full_indexes[offset:]

    return sub_indexes_1, sub_indexes_2


def dump_det_dataset(coco, img_id_list, save_file_path):
    """
    物体检测数据集保存
    :return:
    """
    imgs = coco.loadImgs(img_id_list)
    img_anno_ids = coco.getAnnIds(imgIds=img_id_list, iscrowd=0)
    instances = coco.loadAnns(img_anno_ids)
    cat_ids = coco.getCatIds()
    categories = coco.loadCats(cat_ids)
    common_dict = {
        "info": coco.dataset["info"],
        "licenses": coco.dataset["licenses"],
        "categories": categories
    }
    img_dict = {
        "image_nums": len(imgs),
        "images": imgs,
        "annotations": instances
    }
    img_dict.update(common_dict)

    json_file = open(save_file_path, 'w', encoding='UTF-8')
    json.dump(img_dict, json_file)


class_nums, label_list = parse_label_list("/home/work/data/${dataset_id}/", "/home/work/PretrainedModel/")
bml2coco("/home/work/data/${dataset_id}/", "/home/work/PretrainedModel/org_data_list.json")
split_det_origin_dataset("/home/work/PretrainedModel/org_data_list.json", "/home/work/PretrainedModel/train_data_list.json",
"/home/work/PretrainedModel/eval_data_list.json")
```

将上述脚本存放为 `coversion.py` 代码脚本，并将脚本最后两行的 `${dataset_id}` 替换为所指定数据集的 ID（下图红框中的ID），在终端中运行即可。



运行代码。

```
python coversion.py
```

注意：如果报错 `No module named 'pycocotools'`，需要通过如下命令安装相关依赖包，再运行 `coversion.py` 代码。

```
pip install pycocotools
```

运行 `coversion.py` 代码成功之后将在 `PretrainedModel/` 文件夹下生成对应的数据文件，包括 `label_list.txt`、`train_data_list.json`、`eval_data_list.json`、`org_data_list.json`。



训练模型

1、在终端中打开 `PaddleDetection` 目录。

```
cd /home/work/PaddleDetection
```

2、修改yaml配置文件。

在PaddleDetection 2.0后续版本，采用了模块解耦设计，用户可以组合配置模块实现检测器，并可自由修改覆盖各模块配置，本文以 `configs/yolov3/yolov3_darknet53_270e_coco.yml` 为例：

```
 yolov3_darknet53_270e_coco.yml 主配置入口文件
coco_detection.yml 主要说明了训练数据和验证数据的路径
runtime.yml 主要说明了公共的运行参数，比如说是否使用GPU、每多少个epoch存储checkpoint等
optimizer_270e.yml 主要说明了学习率和优化器的配置。
yolov3_darknet53.yml 主要说明模型、和主干网络的情况。
yolov3_reader.yml 主要说明数据读取器配置，如batch size，并发加载子进程数等，同时包含读取后预处理操作，如resize、数据增强等等
```

需要修改/覆盖的参数均可写在主配置入口文件中，主要修改点为训练、验证数据集路径、运行epoch数、学习率等，修改后的主配置文件如下（注释行即为需要修改的点）：

```
 _BASE_: [
  '../datasets/coco_detection.yml',
  '../runtime.yml',
  '_base_/optimizer_270e.yml',
  '_base_/yolov3_darknet53.yml',
  '_base_/yolov3_reader.yml',
]

snapshot_epoch: 5
weights: output/yolov3_darknet53_270e_coco/model_final

###### 预训练权重地址
pretrain_weights: https://paddledet.bj.bcebos.com/models/yolov3_darknet53_270e_coco.pdparams

###### coco_detection.yml
num_classes: 2 #实际类别数
TrainDataset:
  !COCODataSet
    image_dir: data/${dataset_id}/   # 图片地址
    anno_path: PretrainedModel/train_data_list.json # 标注文件
    dataset_dir: /home/work/ # 数据集根目录
    data_fields: ['image', 'gt_bbox', 'gt_class', 'is_crowd']

EvalDataset:
  !COCODataSet
    image_dir: data/${dataset_id}/   # 图片地址
    anno_path: PretrainedModel/eval_data_list.json   # 标注文件
    dataset_dir: /home/work/   # 数据集根目录

###### optimizer_270e.yml
epoch: 50 # 迭代轮数
LearningRate:
  base_lr: 0.0001 # 学习率
  schedulers:
  - !PiecewiseDecay
    gamma: 0.1
    milestones:
    - 30
    - 45
  - !LinearWarmup
    start_factor: 0.
    steps: 400
```

3、训练模型。

在终端中执行以下命令，开始模型训练。

```
 cd /home/work/PaddleDetection/
python tools/train.py -c configs/yolov3/yolov3_darknet53_270e_coco.yml --eval
```

注意：如果报错 `No module named 'lap'` 和 `No module named 'motmetrics'` ，则需要通过如下命令安装相关依赖包，再运行 `coversion.py` 代码。（如果缺失其他模块，也可用类似命令下载安装）

```
pip install lap motmetrics
```

4、模型评估

在终端中执行以下命令，开始模型评估。

```
python tools/eval.py -c configs/yolov3/yolov3_darknet53_270e_coco.yml \
                -o weights=output/yolov3_darknet53_270e_coco/model_final
```

运行完成输出如下结果：

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.279
 Average Precision  (AP) @[ IoU=0.50       | area=   all | maxDets=100 ] = 0.534
 Average Precision  (AP) @[ IoU=0.75       | area=   all | maxDets=100 ] = 0.251
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.298
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.086
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.331
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.351
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.351
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
[01/30 02:28:27] ppdet.engine INFO: Total sample number: 150, averge FPS: 25.36864530446337
```

5、模型预测。

在终端中执行以下命令，开始模型预测（注意修改图片路径）。

```
python tools/infer.py -c configs/yolov3/yolov3_darknet53_270e_coco.yml \
                --infer_img=/home/work/data/${task_id}/xxx.jpeg \
                --output_dir=infer_output/ \
                --draw_threshold=0.5 \
                -o weights=output/yolov3_darknet53_270e_coco/model_final
```

6、导出模型。

在终端中执行以下命令，将最佳模型转为可以用于发布的 inference 模型

```
python tools/export_model.py -c configs/yolov3/yolov3_darknet53_270e_coco.yml \
        --output_dir=/home/work/PretrainedModel/ \
        -o weights=output/yolov3_darknet53_270e_coco/model_final
```

在终端中执行以下命令，将导出模型移至 `/home/work/PretrainedModel/` 目录。

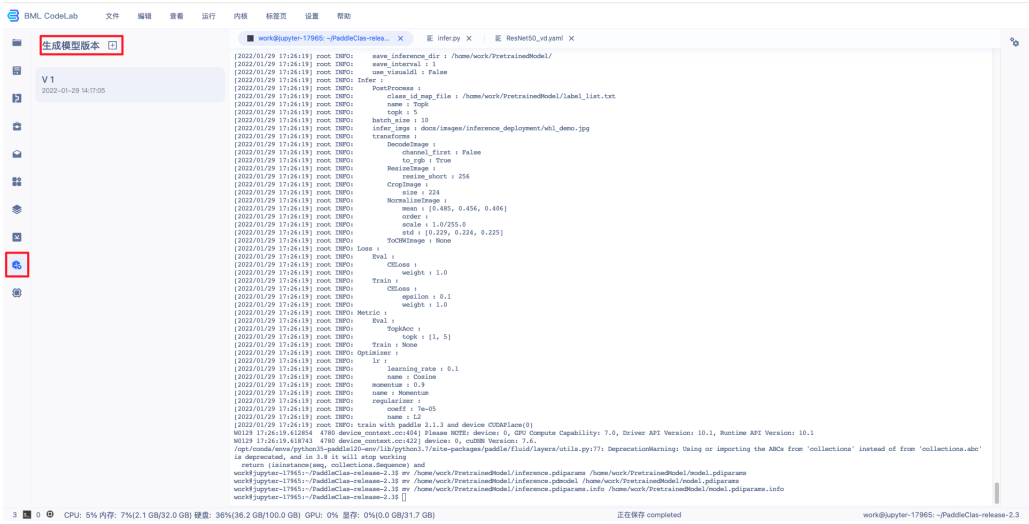
```
mv /home/work/PretrainedModel/yolov3_darknet53_270e_coco/* /home/work/PretrainedModel/
```

7、生成模型版本。

Notebook中的模型文件只有生成模型版本后，才可以执行发布和部署功能：

- 请确保要保存的模型文件在`/home/work/PretrainedModel`目录下。模型支持版本管理功能，在保存时可以生成新版本也可以覆盖已有的且尚未部署的模型版本，每个版本的模型都可以独立部署。每个模型版本中保存的模型文件大小上限为1.5GB。
- 在保存模式时也可以将训练模型的代码一并保存。代码支持版本管理功能，用户再次启动Notebook时，可以使用指定的代码版本来初始化Notebook工作空间即/home/work目录下data以外的空间。每个代码版本中保存的文件大小上限为150M。

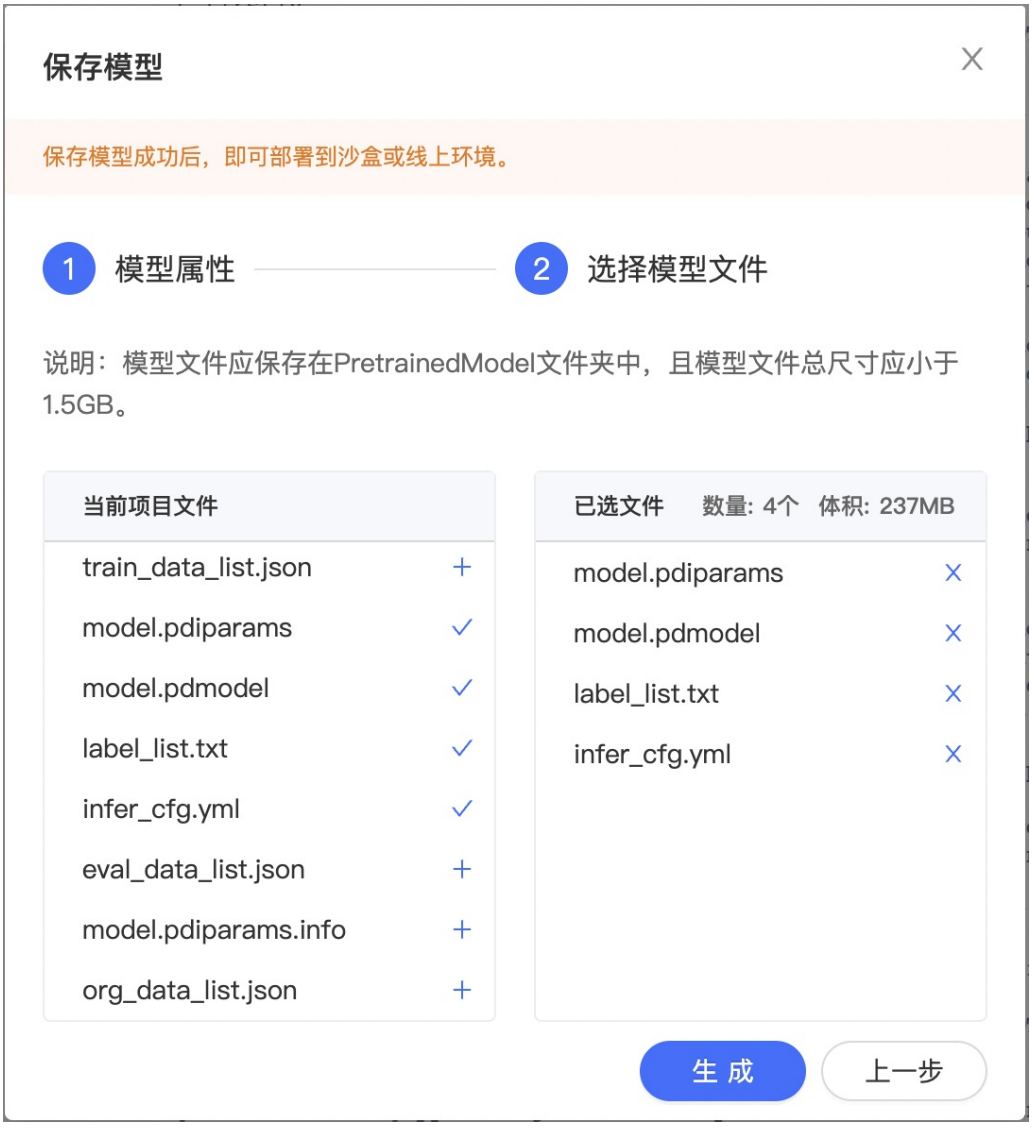
点击左侧导航栏中的生成模型版本组件，打开弹窗填写信息。



模型属性-选择 AI 框架选择 PaddlePaddle2.0.0，若上一次操作中进行了代码保存，可在“代码版本”选择对应的代码版本。



选择模型文件-选择 label_list.txt、model.pdiparams、model.pdmodel 、 infer_cfg.yaml 文件。

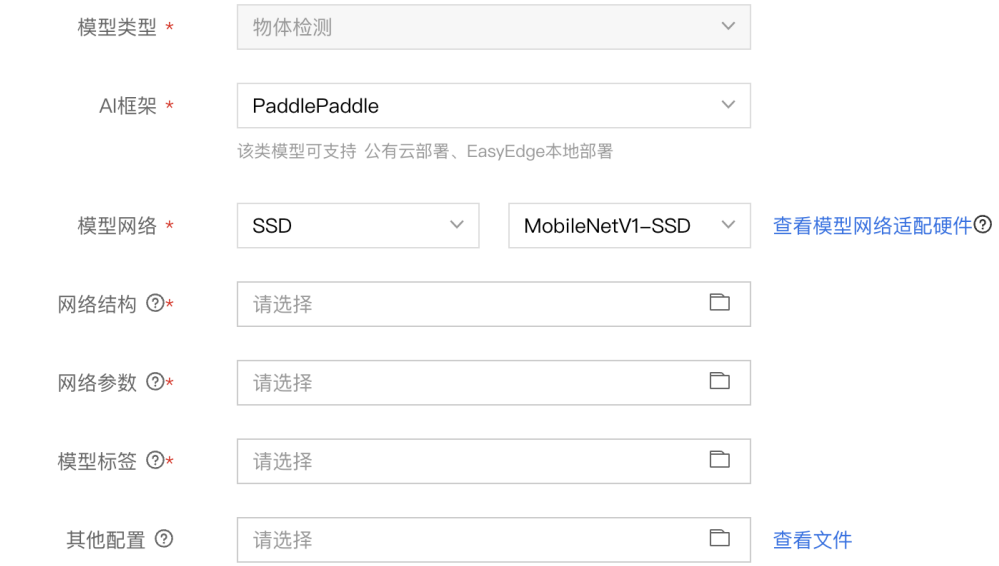


点击『生成』即可生成模型版本，生成模型版本一般需要数十秒，请耐心等待。

配置并发布模型

本文以 PaddlePaddle 的 物体检测模型为例，详细介绍配置说明如下：

选择模型文件



根据平台截图可以看出，主要需要设置一下几个配置

- **模型网络**：保存模型对应网络结构名称，例如SSD, YOLO等；
- **网络结构**：PaddlePaddle 保存为推理模型（save_inference_model）时，保存的网络结构文件，默认名字是model.pdparams。具体参考：PaddlePaddle模型保存 PaddlePaddle save_inference_model API
- **网络参数**：PaddlePaddle 保存为推理模型（save_inference_model）时，网络参数可以保存为单独的**params**文件，或者分散的多个独立文件。对于模型参数文件，请上传单独的**params**文件，或者将多个分散文件打包成zip（不带子目录）上传。具体参考：PaddlePaddle模型保存 PaddlePaddle save_inference_model API
- **模型标签**：模型识别的所有类别的标识，支持字母/数字/下划线/中划线/点；
- **其他配置**：其他配置包括图片预处理、后处理以及网络输入输出层的选择等配置。

对于其他应用方向及框架的模型配置，都是类似的，**但配置项可能存在差异。** 具体使用时，可以根据自身模型的训练框架和应用类型，进行选择查看需要提供的配置项。而且配置项旁边有hover提示，如果不清楚，可以查看提示进一步了解。

操作步骤如下

1、查看前置条件是否满足：需要训练完成，并生成了相应的模型生成版本（详见训练模型的第六步）。

2、回到 BML Notebook 列表页，点击『模型发布列表』即可进入配置页面。



3、点击配置，即可进入配置流程。



4、填写模型信息。



5、选择待发布的模型文件，点击确定按钮。



本文采用了PaddlePaddle中的yoloV3模型，故而选择如图。

另外，对 Paddle2.x 的模型而言：

- 网络结构文件 `model.pdmodel`：必需选择，且名字固定。
- 网络参数文件 `model.pdiparams`：必需选择，且名字固定。
- 模型标签文件：`label_list.txt`，必须选择。（注：因为`paddle detection`套件默认将标签存放在`infer_cfg.yaml`了，因此需要在notebook环境里面手动存储为txt文件，每行代表一个标签。）

7、点击提交即可进入模型验证阶段，验证时间一般需要数十秒，请耐心等待。



验证通过后，显示有效。



8、点击发布，填写相关信息后，即可发布成功。

9、点击左侧导航栏模型管理，即可查看发布成功的模型。



校验模型

1、点击『版本列表』。



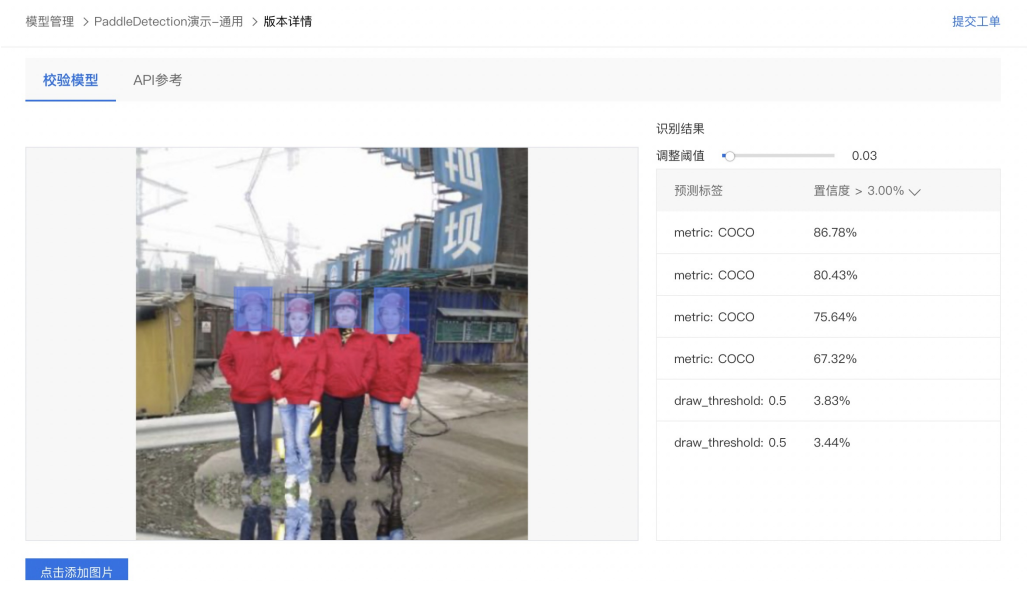
2、点击『校验模型』。



3、点击『启动模型校验』，启动约需5分钟，请耐心等待。



4、上传图像即可开始校验，示例如下：



部署在线服务

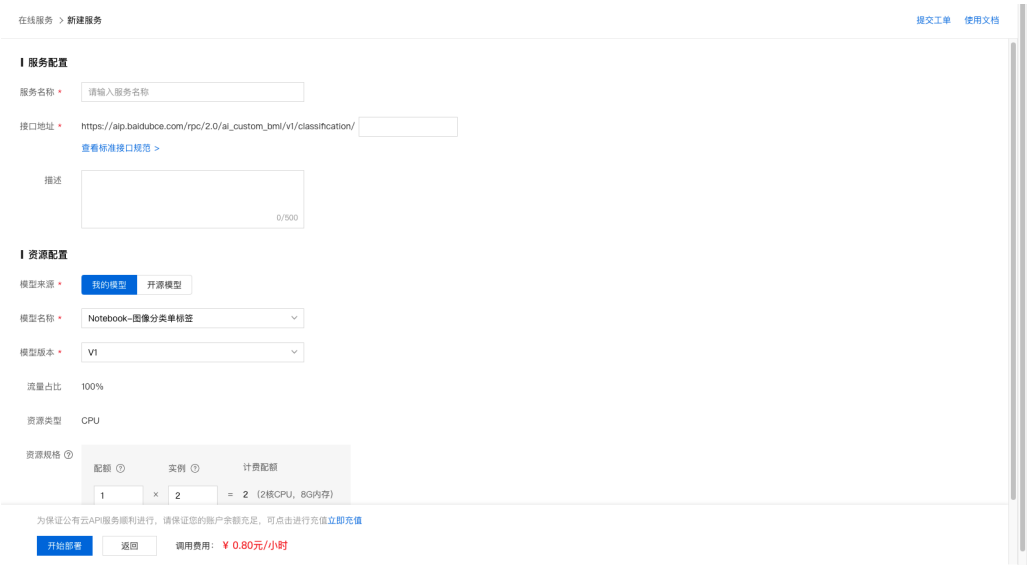
1、点击『版本列表』。



2、点击部署-在线服务。



3、完成信息填写及资源规格选择后，即可开始部署。



4、部署过程需要数十秒时间，请耐心等待。部署完成后，示例如下：



5、API调用方法请参考 公有云部署管理。

总结

- 更多丰富的内容与示例可以参考PaddleDetection的github与教程文档。
  - PaddleDetection github地址：https://github.com/PaddlePaddle/PaddleDetection/
  - PaddleDetection教程文档地址：https://github.com/PaddlePaddle/PaddleDetection/blob/release/2.3/docs/tutorials/GETTING_STARTED_cn.md
- 如果在使用PaddleDetection的过程中遇到问题，欢迎去PaddleDetection的github上提issue：https://github.com/PaddlePaddle/PaddleDetection/issues/new

Codelab Notebook自定义环境部署最佳实践

由于 Codelab Notebook 具备很强的开发灵活性，用户可能在 Codelab Notebook 开发各种各样的模型，为了保障用户自定义的模型可以顺利部署， Codelab Notebook 提供了自定义环境部署的功能。

注意：自定义环境部署功能仅支持通用型 Notebook 开发的模型，并且该流程仅支持公有云服务部署的方式。

本文介绍了从创建 Notebook 任务到训练模型，再到保存模型、部署模型的全流程。

从【飞桨精选案例】创建项目

1、在 BML 左侧导航栏中点击『飞桨精选案例』，注意查看开发语言和AI框架版本，后面需要选择对应的训练环境。



名称	类型	开发语言	AI框架	来源	贡献者	操作
基于PaddleOCR的轻量级车牌识别范例	通用-自定义方向	python3.7	Paddle2.3.0	飞桨行业	PaddlePaddle	查看详情 复制到我的项目
使用PP-HumanSegV2进行人像分割	通用-自定义方向	python3.7	Paddle2.2.2	飞桨行业	PaddlePaddle	查看详情 复制到我的项目
多类别车辆跟踪	通用-物体检测	python3.7	Paddle2.2.2	飞桨行业	AIStudio96069	查看详情 复制到我的项目
电梯内电瓶车检测全流程	通用-物体检测	python3.7	Paddle2.2.2	飞桨行业	chumley	查看详情 复制到我的项目
基于PP-YOLOv2的火灾/烟雾检测	通用-物体检测	python3.7	Paddle2.2.2	飞桨行业	PaddleEdu	查看详情 复制到我的项目
基于图像分类网络ResNet50_vd实现桃子分类	通用-物体检测	python3.7	Paddle2.1.2	飞桨行业	PaddleEdu	查看详情 复制到我的项目
基于YOLOv3实现跌倒检测	通用-物体检测	python3.7	Paddle2.1.2	飞桨行业	PaddleEdu	查看详情 复制到我的项目
基于PaddleX实现的安全帽检测	通用-物体检测	python3.7	Paddle2.2.2	飞桨行业	PaddlePaddle	查看详情 复制到我的项目
基于PP-PicoDet的通信塔识别及Android端部署	通用-物体检测	python3.7	Paddle2.2.2	飞桨行业	PaddleEdu	查看详情 复制到我的项目
基于图像识别的智慧零售商品识别	通用-图像分类单标签	python3.7	Paddle2.2.2	飞桨行业	Thomas-yanxin	查看详情 复制到我的项目

2、在『飞桨精选案例』中选中某一个感兴趣的项目，点击复制到我的项目，示例如下：



3、完成填写后点击下一步即可配置项目资源，支持选择训练环境和资源规格：



4、完成配置后点击确认并启动即可。

训练模型

1、点击进入项目查看案例内容。



2、案例中提供了配置依赖和下载数据集的链接，并且提供了训练代码，逐个 cell 点击运行即可完成整个案例的运行，并且可以得到训练出来的模型。



保存模型

1、进入到保存模型的路径（在训练的输出信息里面会显示模型保存的路径），将模型移动到下图中的 pretrainedmodel 文件夹下。





2、点击生成模型版本。



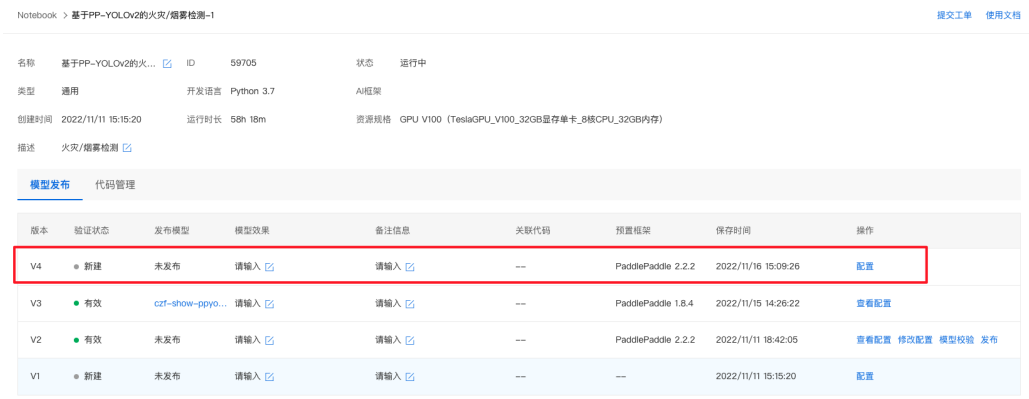
3、填写相关信息并选中模型文件。


保存模型
保存模型成功后，即可发起部署。
1 模型属性
2 选择模型文件
* 生成方式
生成新版本
覆盖现有可升级模型
* AI框架
PaddlePaddle 2.2.2
代码版本
(暂无版本可选)
下一步


保存模型
保存模型成功后，即可发起部署。
1 模型属性
2 选择模型文件
说明：模型文件应保存在PretrainedModel文件夹中，且模型文件总尺寸应小于1.5GB。不同AI框架需要提供不同的模型文件，点击参考规范
当前项目文件
epoch_1
model.pdparams
model.yml
eval_details.json
model.pdopt
已选文件 数量: 4个 体积: 424MB
epoch_1
model.pdparams
model.yml
eval_details.json
model.pdopt
生 成
上一步



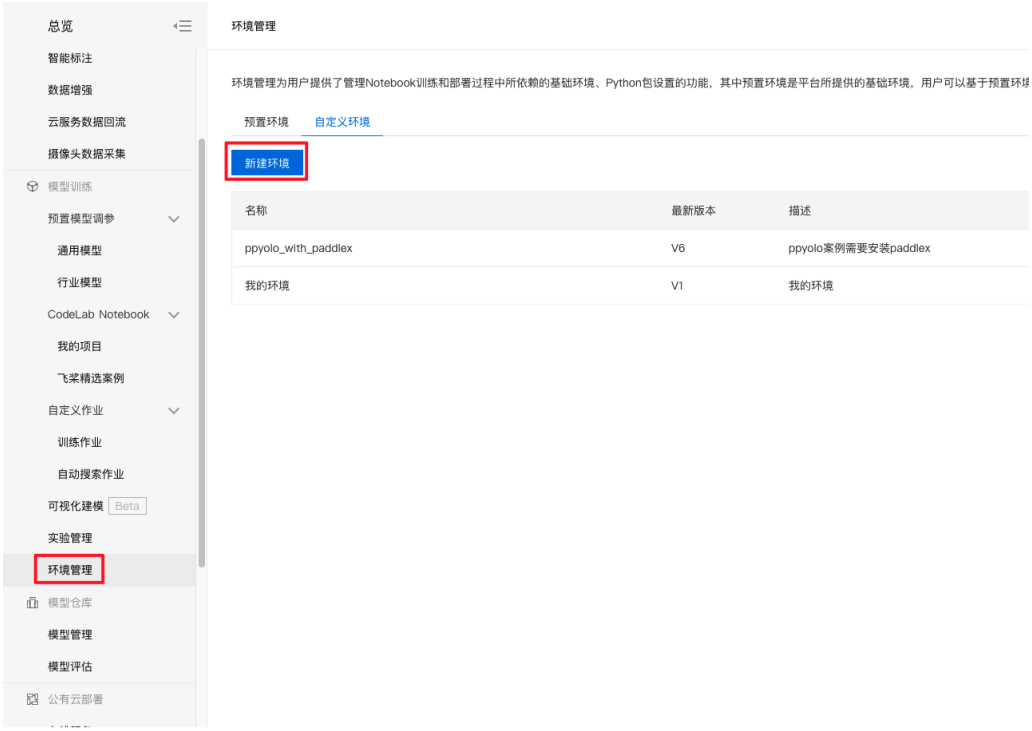
4、生成的模型版本将会在『模型列表』中出现。



制作自定义环境

由于在模型的训练过程中安装了一系列依赖，所以需要在平台环境的基础上安装这些依赖制作自定义环境才可进行后续的部署动作。

1、点击左侧导航栏的『环境管理』，并且进入『自定义环境』，点击新建环境。



2、填写环境名称和描述。



3、点击新建版本。



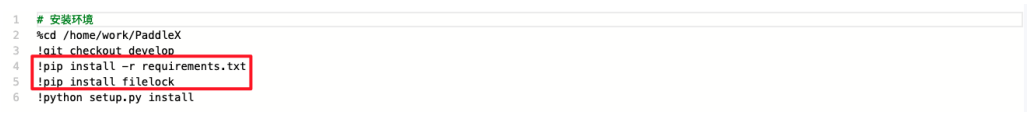
4、选择基础环境，这里注意选择对应框架版本的CPU环境。由于本最佳实践所使用的是paddle2.2.2环境，所以选择paddle2.2.2的CPU环境即可。



5、填写环境内容，这里主要是填写该模型运行所需要的依赖项，依赖项可以查看原项目中的内容。



如下图，该项目中主要依赖requirements中的内容和filelock。



找到requirements中的内容加以复制，添加进环境内容中。（注意，环境内容的原理是和requirements文件一样的，在自定义环境的过程中，平台会依据环境内容通过pip install安装在这里列出的所有组件。）



本项目的示例依赖：

```
 paddlex==2.1.0
tqdm
scipy
colorama
cython
pycocotools
visualdl >= 2.1.1
paddleslim == 2.2.1
shapely
paddlepaddle-gpu >= 2.2.0
opencv-python
lap
scikit-learn==0.23.2
motmetrics
matplotlib
chardet
openpyxl
filelock
```

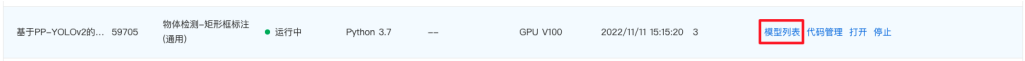
6、填写环境验证命令，此处的原理是根据用户需要，填入一些命令来验证一些关键的包是否安装成功，可以按照示例填写也可以按需填写。点击确认即可进入构建流程。



7、构建和验证的过程时间较长，验证成功后即可开始使用。

配置模型

1、在 BML 左侧导航栏中点击『Codelab Notebook』，进入对应项目的『模型列表』。

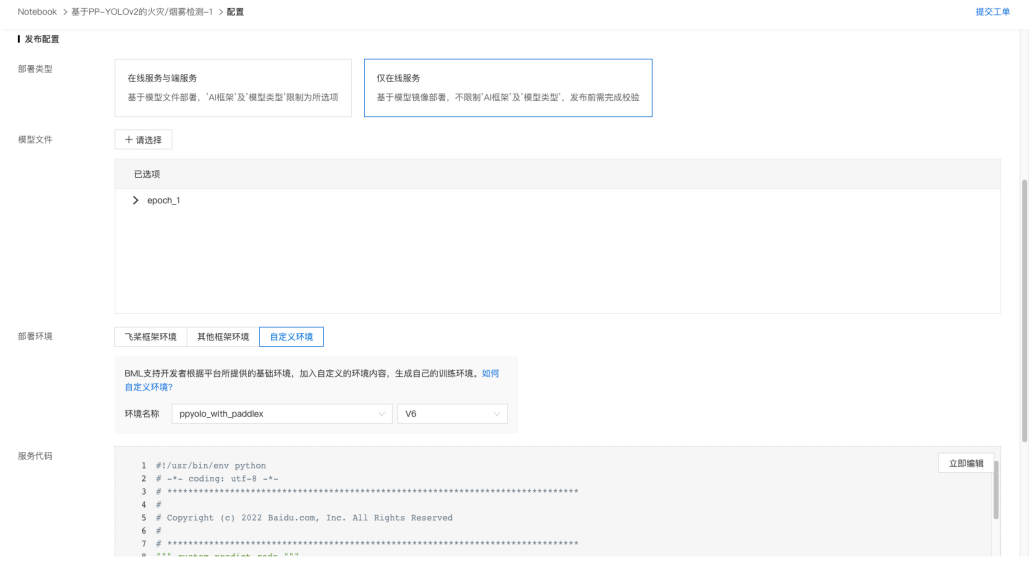


2、找到对应的模型进行配置。





3、选择『仅在线服务』，选择对应的模型文件以及自定义的环境。



4、编辑服务代码，此处主要是加载模型和构造一个请求体（此处的请求体会影响后续部署为公有云服务后的调用代码，需要根据自身情况撰写），可以参考项目中的predict代码。



本项目的示例代码：

```
 ###### !/usr/bin/env python
###### -*- coding: utf-8 -*-
###### ********************************************************************************
#
###### Copyright (c) 2022 Baidu.com, Inc. All Rights Reserved
#
###### ********************************************************************************
""" custom predict code """
import base64
import numpy as np
import cv2
class PyModel(object):
    """ 模型服务预测封装类，支持用户实现自定义的服务请求数据处理和模型推理逻辑。"""
    def __init__(self, model_path, *args, **kwargs):
        """
        根据`model_path`初始化`PyModel`类，如加载标签文件、预处理参数、加载模型 等等
        :param model_path: 该目录下存放了在页面上选择的模型版本中包含的所有文件，并保留原有的目录结构；
        :param *args/**kwargs: 其他信息(预留字段，不可以删除)，系统可能会在未来传入一些其他参数
        """
        # 加载模型
        import paddlex as pdx
        self._model = pdx.load_model(model_path + "/epoch_1/")
    def predict(self, request_body):
        """
        自定义对请求体的处理，包括请求处理、模型预测、预测结果封装等
        :param request_body: 调用接口的请求体json字典
        :return:
            response_body: 请求的处理结果，需要能够被json序列化
        """
        # 请求解析、模型推理、推理结果封装
        # 请求入参： {"image": image base64字符串}
        image = request_body["image"]
        image_bin = base64.b64decode(image)
        # 图像解码
        image = np.frombuffer(image_bin, dtype='uint8')
        image = cv2.imdecode(image, 1)
        # 模型推理计算，参考 https://github.com/PaddlePaddle/PaddleX/blob/develop/docs/apis/models/detection.md#predict
        result = self._model.predict(image)
        format_result = []
        for target in result:
            format_result.append({
                "label_index": target["category_id"],
                "name": target["category"],
                "score": target["score"],
                "location": {
                    "left": target["bbox"][0],
                    "top": target["bbox"][1],
                    "width": target["bbox"][2],
                    "height": target["bbox"][3]
                }
            })
        return {"results": format_result}

class CustomException(RuntimeError):
    """
    进行模型验证和部署服务必需的异常类，缺少该类在代码验证时将会失败
    在处理异常数据或者请求时，推荐在`PyModel`中的自定义推理predict函数中抛出`CustomException`异常，
    并为`message`指定准确可读的错误信息，以便在服务响应包中的`error_msg`参数中返回。
    """
    def __init__(self, error_code, message, orig_error=None):
        """ init with error_code, message and origin exception """
        super(CustomException, self).__init__(message)
        self.error_code = error_code
        self.orig_error = orig_error
```

5、点击保存和提交。

模型校验

1、配置完成的模型处于待校验状态，点击进行模型校验。



2、模型校验支持传入测试数据的base64格式的json（由于用户可能传入多种格式的数据，所以采用json格式传入更具有适配性）。



本项目中将对应的测试图片通过代码转换为了base64的json格式。

```
 import numpy as np
import cv2
image_bin = open("/home/work/dataset/images/fire_000001.jpg", "rb").read()
import base64
print(base64.b64encode(image_bin))
```

示例：

数据类型	json格式
图片	{"image": "xx（此处粘贴转写为base64后的内容）"}

3、点击发起校验，即可查看预测结果，如果与预期一致，点击发布即可。





4、发布完成将会在模型管理中新增一行记录。



模型部署

1、点击即可进入部署阶段。



2、填写相关信息即可完成部署。



Notebook简介

BML基于Jupyter提供了在线的交互式开发环境。Notebook不但免去了用户准备开发环境的过程，而且可以帮助用户轻松的实现数据集、代码以及模型的管理。

基本概念

- 数据集：通过导入操作可以将BML中已创建的数据集下载到Notebook的data目录中。
- 类型：为简化用户开发，平台根据用户选择的不同类型预置了不同的示例代码，只需简单进行数据集适配，即可进行模型训练。
- 模型：用户可以将Notebook中产生的模型文件进行保存，并版本进行管理。在Notebook中保存模型可以生成新的模型版本。
- 代码库：用于保存Notebook中除data以外的工作空间中的内容。代码库也通过版本管理，在保存模型时可以将生成模型的代码保存为新版本的代码库。在启动Notebook时可以使用指定版本的代码库来对Notebook的工作空间进行初始化。

基本流程

使用Notebook创建模型的流程如下所示：



说明：当前“通用”Notebook暂不支持配置模型与校验模型功能。

Notebook类型及支持的开发语言和AI框架

Notebook支持如下几种类型：

- 图像分类-单图单标签、图像分类-单图多标签、物体检测
  - 语言：
    - Python3.7
    - Python2.7
  - 框架：
    - PaddlePaddle 1.8.4
    - PaddlePaddle 1.7.1
    - PaddlePaddle 1.6.2
    - PaddlePaddle 1.5.1
  - 数据集：可以导入相应类型的数据集
- 通用
  - 语言：
    - Python3.7
  - 框架：
    - PaddlePaddle 2.0.0
    - PaddlePaddle 1.8.4
    - PyTorch 1.7.1
    - PyTorch 1.5.1
    - TensorFlow 1.13.2
    - TensorFlow 2.3.0
    - Sklearn 0.23.2-Xgboost 1.3.1
    - Blackhole 0.3.2
  - 数据集：可以导入所有类型的数据集

创建并启动Notebook

创建并启动Notebook

在BML中，Notebook支持多种语言以及不同的AI框架，且可灵活切换。

创建Notebook

BML提供了多种Notebook类型，每种类型都支持多语言及多框架。

1. 在BML左侧导航栏中单击“Notebook”。
2. 在“Notebook”页面单击“新建”，在弹出框中填写详细信息。示例如下：



注意：创建通用类型时，可支持选择PaddlePaddle、Tensorflow、Pytorch等多框架；创建图像分类-单图单标签、图像分类-单图多标签等非通用类型时，为保障示例代码的正常运行，仅支持选择PaddlePaddle框架。

3. 完成信息填写后单击“下一步”，配置notebook启动实例。Notebook运行在Docker容器中，用户可以在启动Notebook前配置适合自己的代码执行环境。

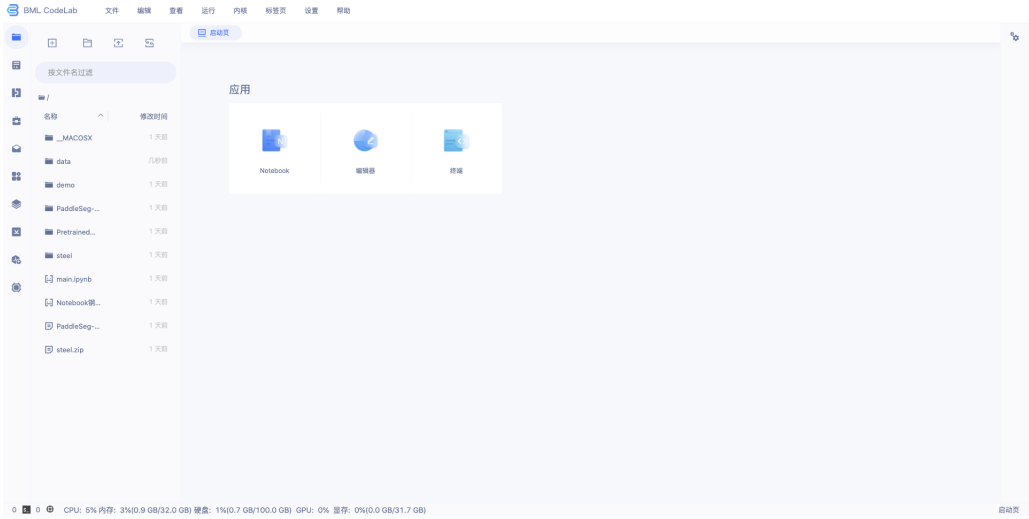
   在配置启动实例时，可支持选择多种资源池类型，如果选择公共资源池未降低您使用成本，可配置最大运行时长，若未配置默认按一小时误操作自动停止notebook任务，请悉知



完成配置后，单击“确认并启动”。

> Notebook启动过程中需要完成资源的申请以及实例创建，请耐心等待。

4. 等待Notebook启动后，单击“打开”，页面跳转到Notebook，如下所示：



- 文件夹：对应Notebook运行环境的“/home/work”目录，各预置的文件夹作用如下：
  - PretrainedModel：用于存储用户训练好的模型文件，可以将该文件夹下的模型文件保存为不同的模型版本。
  - data：用户导入的数据集保存在该文件夹下。
  - demo：该文件夹下保存了示例代码。
- 代码片段库：包含平台预置的公共代码片段库，支持用户直接进行复用。同时用户也可将自己的代码片段保存下来。
- 包管理：预置了常用的200+的Python安装包，包含PaddlePaddle、transformer等安装包。
- 正在运行的终端和内核：监测正在运行的终端和内核。
- 目录：显示相关目录。
- 环境信息：用于查看环境和配置。
- 数据集：用于导入用户已在EasyDL专业版中已创建的数据集。
- 生成代码版本：可生成并保存代码版本，进行版本管理。
- 生成模型版本：可生成并保存模型版本，进行版本管理。
- 资源监控：用于查看资源和配置

使用Notebook开发模型

使用Notebook开发模型

在Notebook可以导入已创建的数据集并将其用于模型训练。 **预置代码说明**

当您创建的是图像分类、物体检测类型的Notbeook时，系统会自动生成一份代码，您只需要导入数据集并将代码做简单的适配即可快速创建。

当您创建的是通用型的Notebook时，系统会根据您使用的框架来生成一份示例代码。

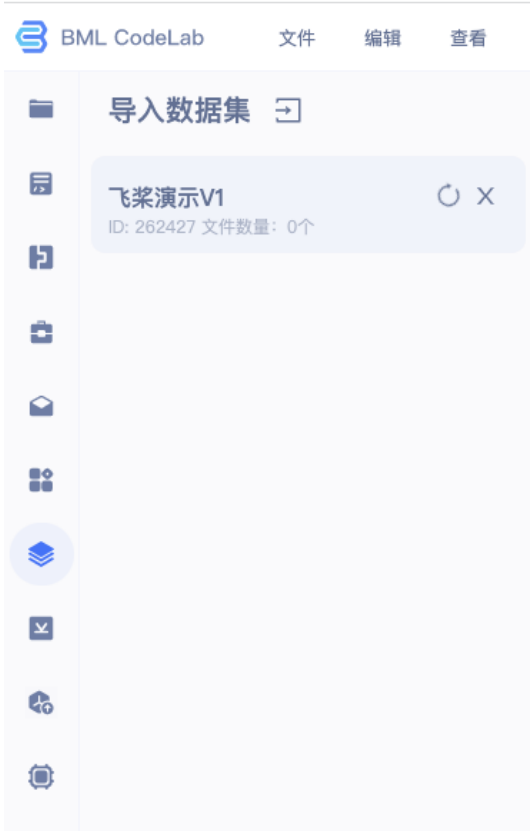
导入数据集

通过导入操作可以将已创建的数据集下载到Notebook的运行环境中，导入的数据集会保存在“data”目录下以数据集ID命名的文件夹中，对于已标注的图片，下载内容包括原始图片文件和同名的json标注文件。

1. 单击Notebook左侧页签中的“数据集”。
2. 在操作面板中，单击“导入数据集”，系统在弹出框中显示已创建的数据集及其标注情况，如下所示：



3. 勾选待导入数据集前的复选框并单击“确认”，开始导入数据集，如下所示：



数据集导入操作是单向的，且当数据服务中的数据集有更新时，不会自动同步，需要手动重新刷新。

训练模型

用户可以在Notebook或终端中编写模型训练代码，并完成模型的训练。

本节给出如何使用示例代码完成导入数据集的建模。示例代码结构如下所示：



用户只需要执行train.sh脚本即可启动模型训练，其输入参数如下所示：

`train.sh <数据目录> <模型保存目录>`

- 数据目录：数据集保存目录的绝对路径
- 模型保存目录：请设置为/home/work/PretrainedModel/，否则会造成生成新模型版本时无法读取模型文件。

上述代码结构不是必须的，用户可以自行编写代码训练模型，只需要将模型保存到/home/work/PretrainedModel/目录即可。 使用示例代码完成模型训练的步骤如下所示：

使用示例代码完成模型训练的步骤如下所示：

1. 在Notebook中执行如下代码查看已导入的数据集。

   `! ls /home/work/data/`

2. 在Notebook中执行train.sh脚本进行模型训练。

   `! bash demo/train.sh /home/work/data/96159/ /home/work/PretrainedModel/`

请根据实际情况修改上述代码中的数据集路径。

3. 查看生成的模型文件。

   在Notebook左侧导航栏中单击“文件夹”，并打开“Pretrainedmodel”文件夹，如下所示：



保存Notebook中的模型

保存Notebook中的模型

模型只有保存为特定版本后，才可以执行检验，校验通过后可以将其发布到模型仓库。

请确保要保存的模型文件在/home/work/PretrainedModel目录下。模型支持版本管理功能，在保存时可以生成新版本也可以覆盖已有的且尚未部署的模型版本，每个版本的模型都可以独立部署。每个模型版本中保存的模型文件大小上限为1.5GB。

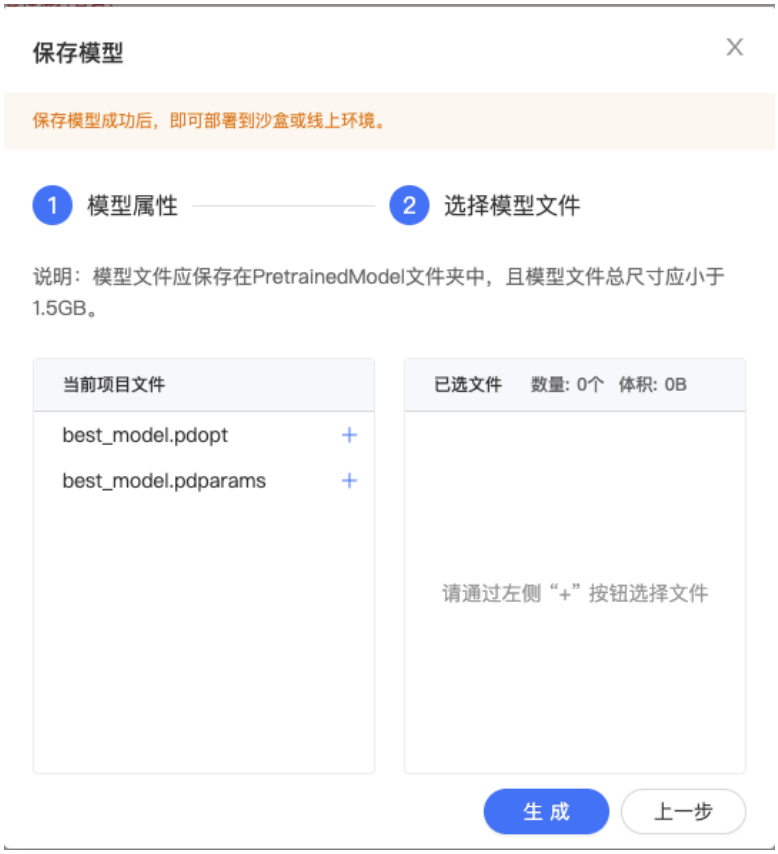
在保存模式时也可以将训练模型的代码一并保存。代码支持版本管理功能，用户再次启动Notebook时，可以使用指定的代码版本来初始化Notebook工作空间即/home/work目录下data以外的空间。每个代码版本中保存的文件大小上限为150M。

1. 单击Notebook页面左侧工具栏中的“生成模型版本”。
2. 在“保存模型”对话框中配置模型属性，如下所示：



- 生成新版本：生成一个新的模型版本，版本号由系统自动生成，若当前最高版本为Vn，则新生成版本为Vn+1。注意需要保存模型网络结构文件、参数文件以及标签文件（如：label.txt）。
- AI框架：即生成模型时使用的Paddle版本。
- 代码版本：即生成模型时使用的代码版本。

3. 单击“下一步”，选择要保存的模型文件：



当前项目文件为PreitraindModel目录下的文件列表。

4. 单击“生成”，完成模型保存。

在完成模型保存后，可以关闭Notebook，从而避免产生不必要的费用。

配置模型

配置模型

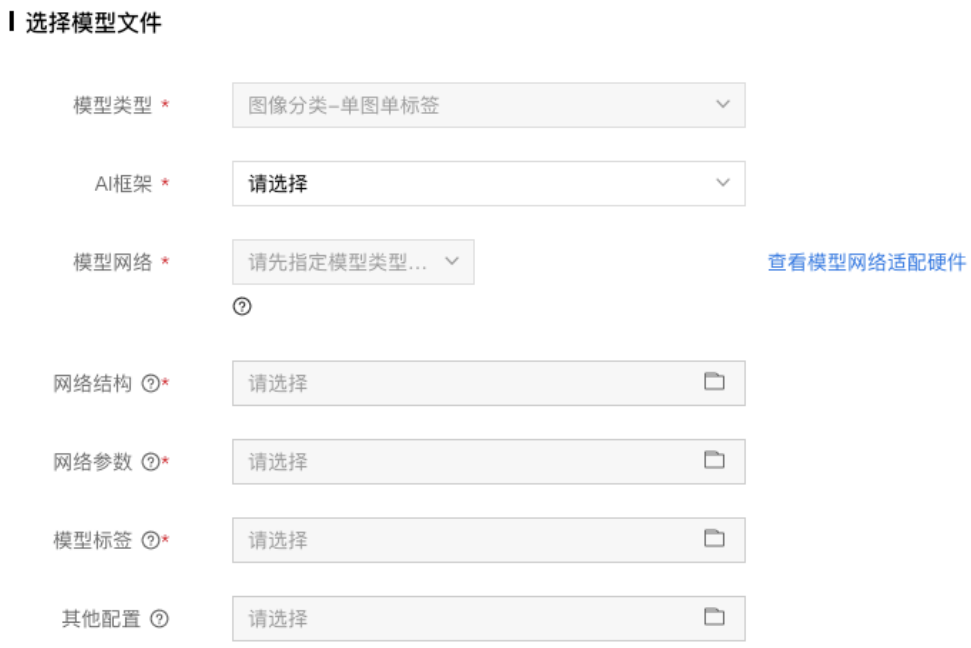
在BML中，Notebook不仅为用户提供了开发环境，而且支持用户将Notebook中开发的模型部署发布到模型仓库以及部署为在线服务。在发布前需要先对已保存的模型配置相应的出入参及数据处理逻辑，从而保证后续模型可被部署。

1. 在左侧导航栏中单击“Notebook”。
2. 在已创建的“dogcat分类模型”的模型发布列表中，单击待部署版本模型所在行的“模型发布列表”。
3. 在模型发布列表中，单击已保存模型所在行的“配置”，如下所示：



- 模型效果：建议记录准确率、召回率等评估指标，便于后续查阅以及进行多版本模型的对比。
- 模型备注：建议记录模型训练使用的超参数等信息，便于后续查阅。

4. 选择模型文件，如下所示：



可选项为生成当前模型版本时保存的模型文件。

注意事项

- 该脚本用于将选择的模型文件发布成模型服务。

- 用户可以通过修改PredictWrapper的preprocess和postprocess方法来实现自定义的请求预处理和预测结果后处理。
- 当提交该脚本时，系统会根据用户选择的模型文件和脚本内容，来验证是否可以启动模型服务，只有验证通过，才可以进行模型效果校验以及将模型发布到模型仓库。

限制和建议

1. CustomException必需存在且是异常类；在自定义的逻辑中，建议当处理进入错误的分支时,抛出CustomException并指定message，指定的message在请求回包中会作为error_msg返回。
2. PredictWrapper类必需存在，且必需包含preprocess和postprocess两个方法。
3. PredictWrapper的preprocess和postprocess方法，是用户自定义模型服务请求预处理和预测结果后处理的入口。
4. preprocess方法接收的第一个参数为用户请求的json字典，对于图像类服务，传入图像的参数key必须是"image"，且传入的是图片的base64编码。
5. 系统会根据postprocess方法的返回结果result类型的不同，做以下处理：
   - dict: 不作修改
   - list: 转换为 {"results": result}
   - 其他: 转换为 {"result": result}
   - 最终的处理结果可以转换为json字符串，以确保能够正常将结果作为请求回包返回
6. 单击“提交”，完成模型配置。

   提交后，系统会自动对当前版本模型进行代码验证，通过后模型会更新为“有效”状态。

发布模型

发布模型

当模型已完成配置且通过校验后，可以将其发布到模型仓库中。

说明：图像分类、物体检测类的Notbeook支持该操作，通用型Notebook不支持该操作。

1. 在左侧导航栏中单击“Notebook”。
2. 在已创建的“dogcat分类模型”的模型发布列表中，单击“有效”状态模型版本所在行的“发布”。
3. 在“发布”页面，完成相关配置，如下所示：



4. 单击“确定”。

在模型列表中可以查看模型的发布状态，在发布后，可以查看其发布后在模型仓库中生成的模型以及版本号。

常见问题

Q1.什么是Notebook？

Notebook 是一个 Jupyter 笔记本环境，不需要进行任何设置就可以使用，适用于进行机器学习深度学习的培训和开发.

Q2.支持哪些浏览器？

Notebook 在 Chrome/ Firefox/ Safari的各个桌面版本上进行了最全面的测试，**不支持** IE以及IE内核浏览器。

Q3.Jupyter 和 Notebook 有什么区别？如何使用Jupyter？

Jupyter Notebook是一个开放源代码项目，定义的基于web的交互式编程方法已经逐渐成为全球数据科学/机器学习/深度学习领域的前端标准. BML中Notebook 是在 Jupyter Notebook 基础之上开发的。通过BML中的Notebook，您可以使用 Jupyter Notebook，完全不需要在您自己的计算机上下载、安装或运行任何内容，只要有浏览器就可以使用. 通过查阅Jupyter Notebook使用手册，您可以充分了解Juypter指令使用方法.

Q4.我的代码在哪里执行？如果关闭浏览器窗口，我的执行状态会怎样？

代码会在专供您使用的docker容器中执行。docker容器闲置一段时间后会被回收，并且系统为docker容器强制设置了最长有效期.关闭浏览器窗口期间的代码执行的标准输出不会在屏幕窗口中展示，因此，建议您如果需要在关闭浏览器窗口的情况下继续运行代码，可以把程序标准输出/标准错误输出存储到文件(非/home/work/data目录下文件)中.

Q5.Python有哪些基础类库，我应该如何学习？

您可以参考Python3.5手册和Python2.7手册. 因为Python2.7官方已不再支持，推荐您使用Python3.5及以上版本.

Q6. 我在Notebook中pip install的Python package在下次运行时为什么不见了？

如果需要进行持久化安装, 需要使用持久化路径, 如下方代码示例:

```
 !mkdir /home/work/external-libraries
!pip install beautifulsoup4 -t /home/work/external-libraries
```

同时添加如下代码, 这样每次环境(kernel)启动的时候只要运行下方代码即可:

```
import sys
sys.path.append('/home/work/external-libraries')
```

除上述操作外，在保存模型时，需要将'/home/work/external-libraries'及其包含的文件作为代码库进行保存，并在下次启动Notebook时使用该代码库，从而在启动Notebook后才可以使用已安装的Python库。

Q7. 为什么我的项目打开后, 文件不见了?

Data目录为非持久化目录, 在Data下进行的文件操作, 重启之后会被自动恢复.

Q8. BML中Notebook除了PaddlePaddle外，是否支持TensorFlow等其他框架?

框架的支持需要大量的硬件适配工作, 而我们暂无人力开展此项工作. 用户自行安装的框架往往不能在GPU环境下正常运行, 从而对我平台稳定性产生质疑. 因此最终决定暂停支持. 平台对PaddlePaddle进行了充分的适配工作，您可以尝试使用PaddlePaddle. 附带一份TensorFlow和PaddlePaddle的API映射表: https://aistudio.baidu.com/aistudio/projectdetail/80548

Q9. 为什么我使用PaddlePaddle显存占用这么大?

PaddlePaddle在分配requested_size大小的显存时，先定义一个显存池的大小，记为chunk_size，chunk_size由环境变量FLAGS_fraction_of_gpu_memory_to_use 确定，表征chunk_size在全部显存的占比，默认值为0.92，**即框架预先分配显卡92%的显存** 注：若GPU卡上有其他任务占用显存，可以适当调整chunk的占比，保证框架能预分配到合适的chunk，比如可以分配40%的显存可以这样设置：

`exportFLAGS_fraction_of_gpu_memory_to_use=0.4 # 预先40%的GPU显存`

提醒：chunk占比应该尽可能大，只有在想测量网络的实际显存占用量时，可以设置该占比为0，观察nvidia-smi显示的显存占用情况。

Q10. Notebook中如何同步git内容?

您可以在code cell中执行如下命令来同步git `!git clone https://yourgitaddress.git`

请注意, 如果git网址为http (而不是https), 有可能被要求再次确认, 而Notebook中缺乏该类交互能力.

如有必要, 请使用Terminal功能.

Q11. 为什么在Notebook中同步GitHub会比较慢?

经测试, 主要是中国访问海外资源的总带宽有限.

可以尝试使用国内代码托管服务来替代.

或通过其他方式将GitHub资源下载后, 再上传至AI Studio.

Q12. 为什么不能运行turtle或tkinter?

turtle/tkinter运行时会创建一个GUI, 但是BML中Notebook是一个运行在云端的网页服务. 它无法在你的机器上打开一个窗口.

如果确实有必要, 请自己修改turtle或tkinter的代码, 对其GUI行为进行限制, 或使得它们直接支持Notebook.

Q13. 是否支持导入ipynb文件?

为了系统安全及稳定性,Notebook对ipynb文件进行了隐藏, 同时一个项目仅支持一个活动的ipynb文件, 作为整个项目的启动文件.

用户可以上传并替换默认的ipynb文件。

Q14. 如何通过apt-get或apt-install安装库?

限于安全策略, 我们暂不支持通过这种方式进行安装. 也不建议编译模式(make)进行安装. 如果的确有需求, 请通过mailto: aistudio@baidu.com 申请.

Q15. 如何移动Notebook中的文件?

可使用终端命令来进行文件位置操作. 例如:

```
 ##### !/bin/bash
direc="%%1" #$(pwd)
for dir2mv in $direc/* ; do
if [ -d $dir2mv ]; then
  mv $dir2mv "%%2"
fi
done
```

Q16. 平台中的Notebook模式使用的v100资源是几张卡呢？

目前是1张

Q17. Notebook项目中能运行的代码，在脚本任务中需要改吗?

需要进行修改. 在创建脚本任务时会提供一些范例，请按范例说明进行修改.

Q18. 我想把本地项目文件上传至Notebook项目中，但文件数量比较多，怎么上传?

a. 请将项目文件打成zip包后，在Notebook环境中上传.

如果项目文件较大(>150mb)，请使用数据集功能上传，然后挂载到项目中.

b. 最后在项目中通过unzip命令进行解压缩(请注意需要解压到work目录下)

c. 如执意需要.rar包，由于该格式为RAR共享软件独有，请自行百度解压命令及用法.

Q19. 我上传文件时，系统告诉我上传失败，怎么办?

已知上传失败有3种情况:

1. 您本地的系统时间和互联网标准时间差异过大，此时您需要调整本地电脑时钟，使之和互联网标准时间基本一致，然后重新上传.

2. 您使用了VPN软件，这类软件可能会导致上传失败. 此时您需要关闭VPN软件，然后重新上传.

3. Notebook内部服务异常. 此时您可发送问题页面url及截图，请提交工单联系我们.

Q20. 我在卸载或者安装包时，输出进程提示输入（y/n）没有输入框。不知从哪输入y或者n

部分情况，在Notebook的输出区有对应输入位置. 如确定没有，则请使用Notebook中的"终端"功能.

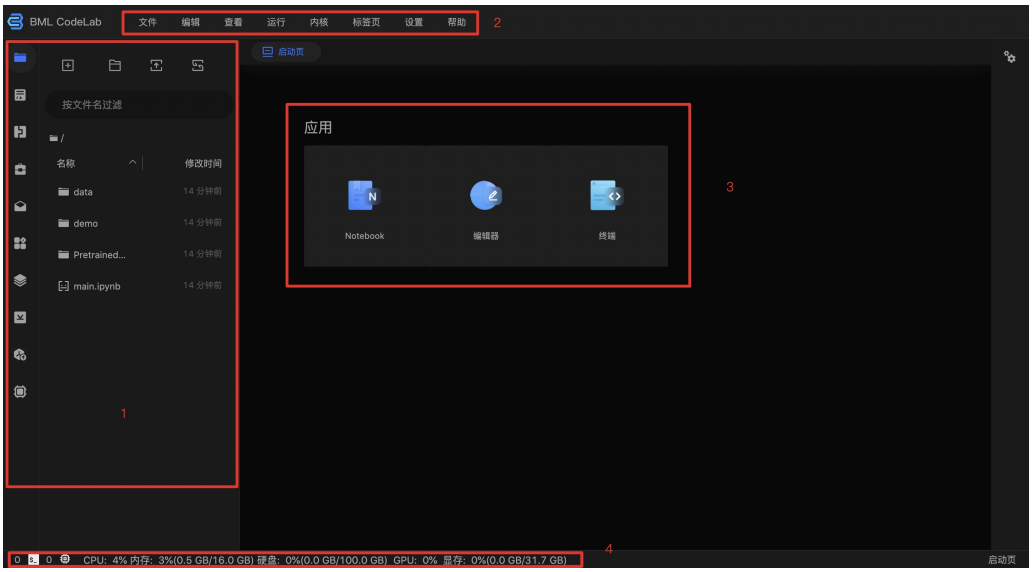
Q21. 如何在Notebook中进行调试?

Notebook环境自启动后，其实就已经在调试状态了. 点击每个Cell前面的运行按钮,可以视为Step-Over的执行状态. 如果需要添加断点等功能，请使用Python自带的调试器: PDB.

Notebook使用参考

页面概览

BML对原生Notebook进行了定制与扩展，用户可以使用新的Notebook完成代码开发、代码执行、结果查看以及模型保存等功能，其包含如下区域：



各区域依次为：

1. 侧边栏：提供文件、数据、环境管理、包管理等多项功能
2. 菜单栏：提供文件、模型、运行过程的管理功能
3. 编辑区：提供Notebook、编辑器、终端三种编写模式
4. 状态监控区：包括变量监控和已执行代码查看功能

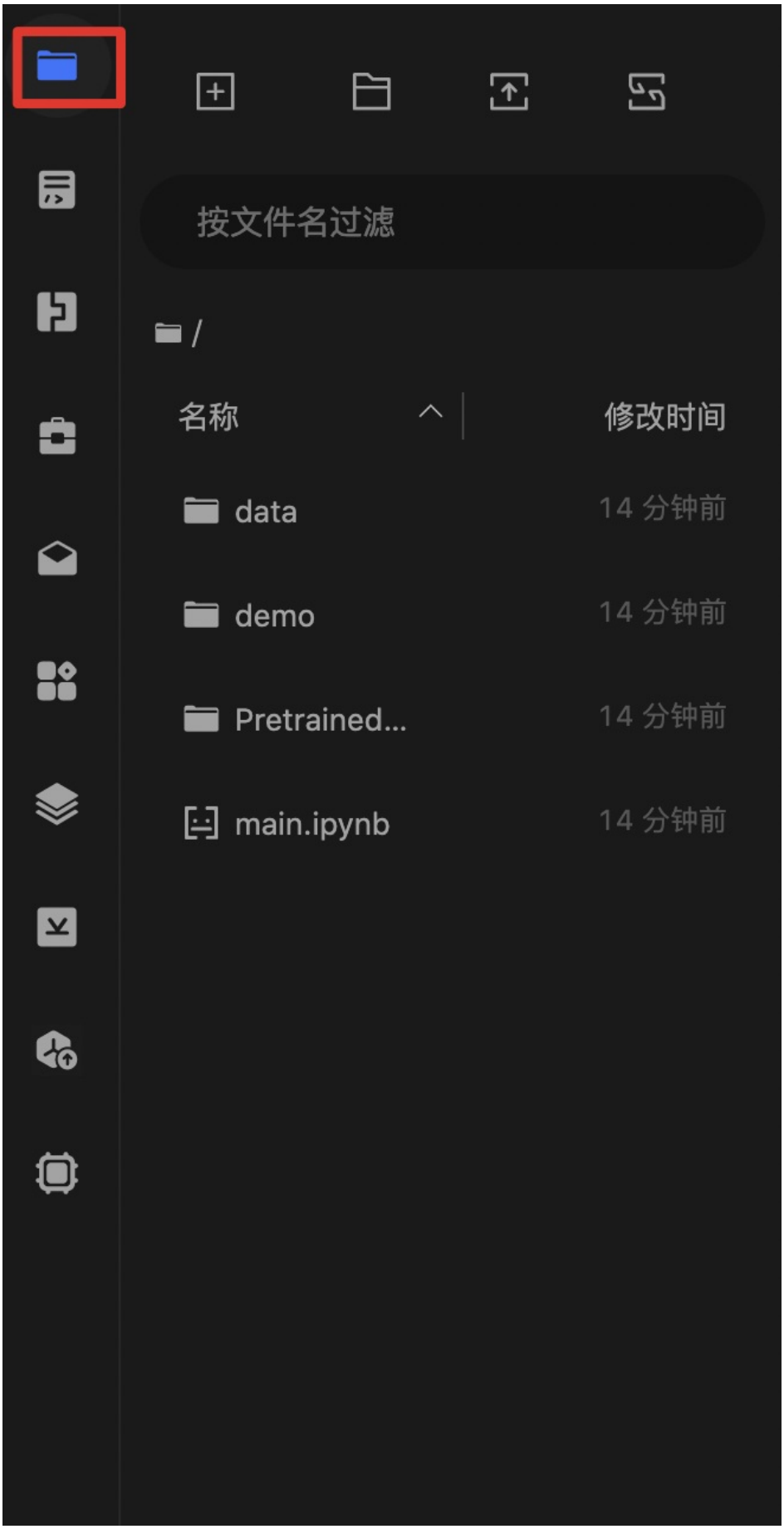
注意：BML的Notebook环境支持暗色和亮色两种主题，本文示例图均以暗色主题呈现。

侧边栏

侧边栏提供文件、数据、环境管理、包管理等多项功能。

文件夹

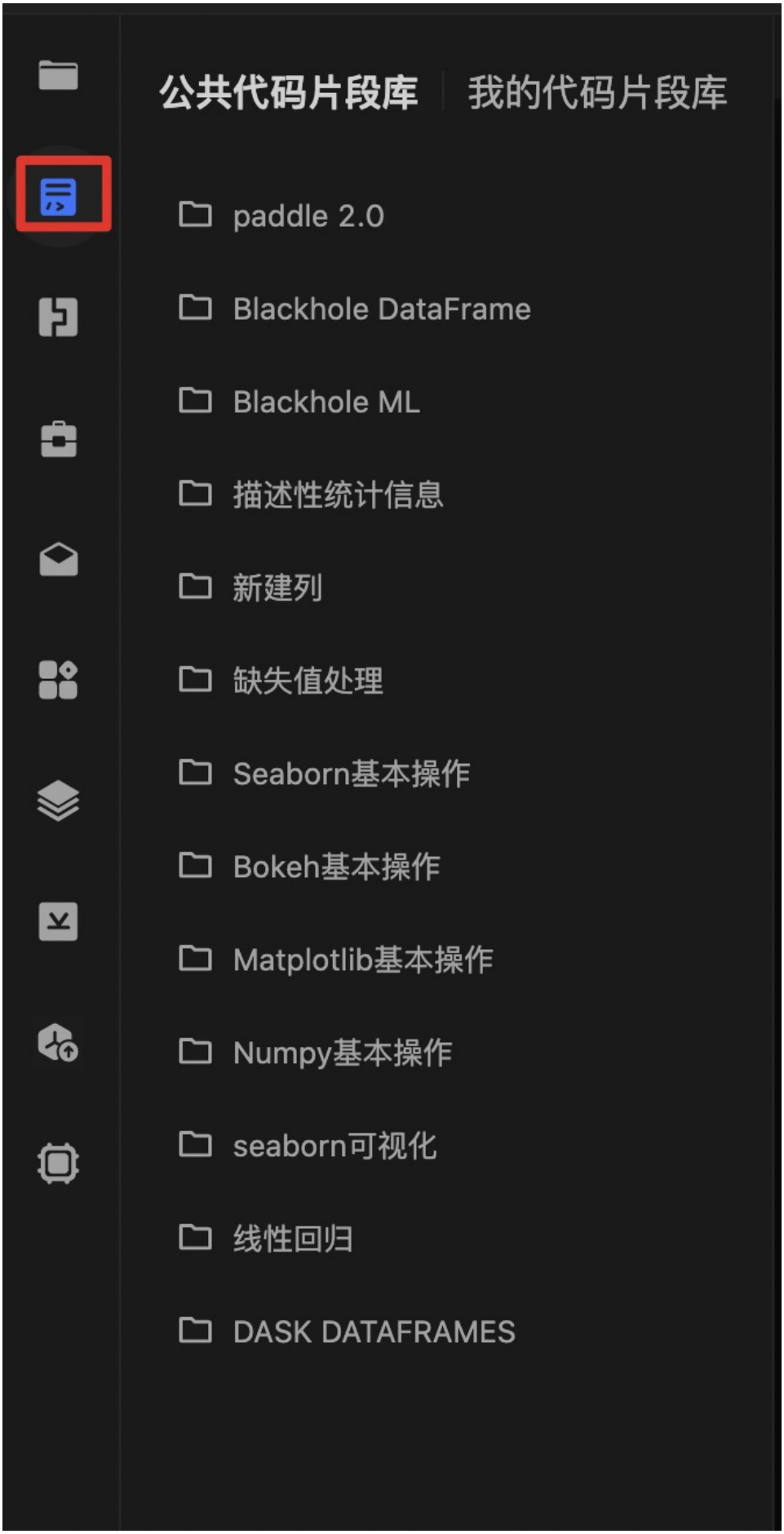
文件夹提供“/home/work”目录下文件的可视化管理功能，用户可以上下传文件、创建或删除其中的文件夹等。Notebook启动之后，会将导入过的数据集同步到data目录下，其它目录下会使用用户已保存过的代码版本或草稿版本进行初始化。



- 文件夹操作: 包括文件夹创建、删除、重命名以及路径复制功能。
- 文件操作: 包括文件创建、上传、下载、删除、重命名、熟悉查看等功能。其中上传时，支持的最大文件为30M。
- 刷新：若代码在运行过程中更新了磁盘中的文件，可以通过刷新操作更新文件状态。

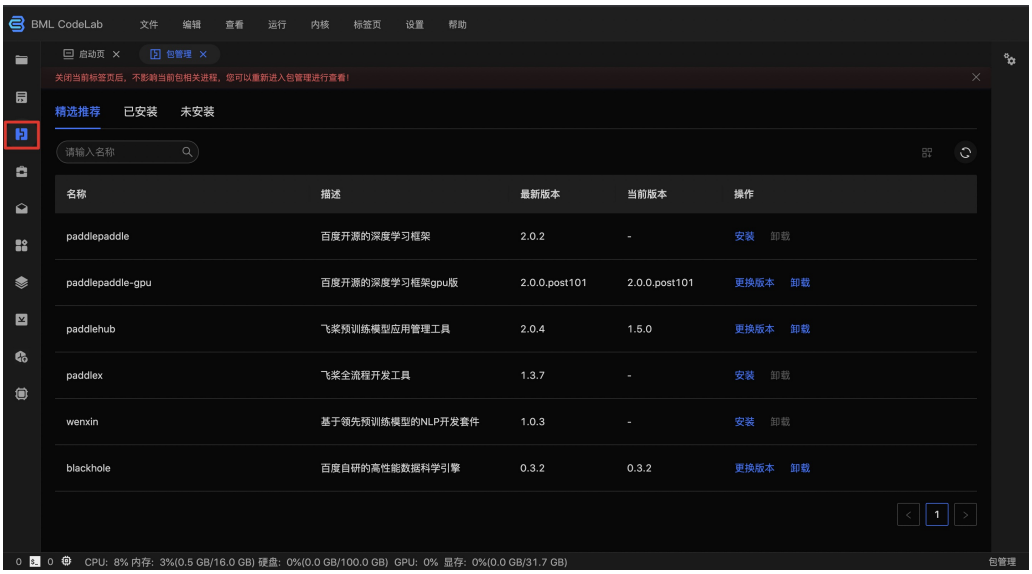
代码库

代码库提供了丰富的公共代码片段供用户使用，同时也支持用户自行上传片段至代码库。



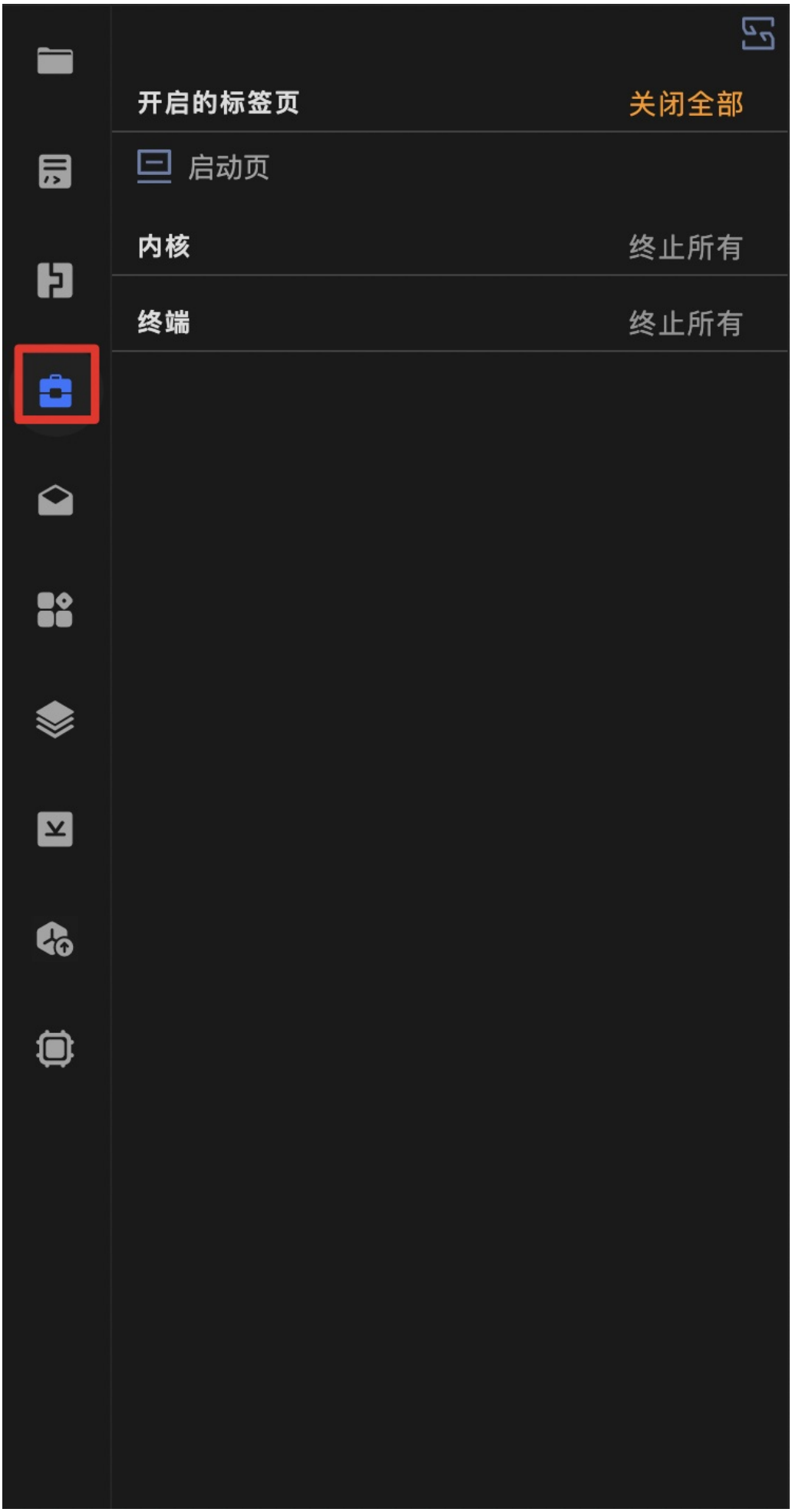
包管理

包管理提供了包括飞桨系列的海量的安装包供用户引入安装。



进程管理

进程管理为用户提供了进程管理的功能。



环境信息

用于查看当前Notebook运行的软硬件信息以及创建新的终端连接。



数据集

用于将在BML的数据服务中创建的数据集同步到Notebook运行的容器中。

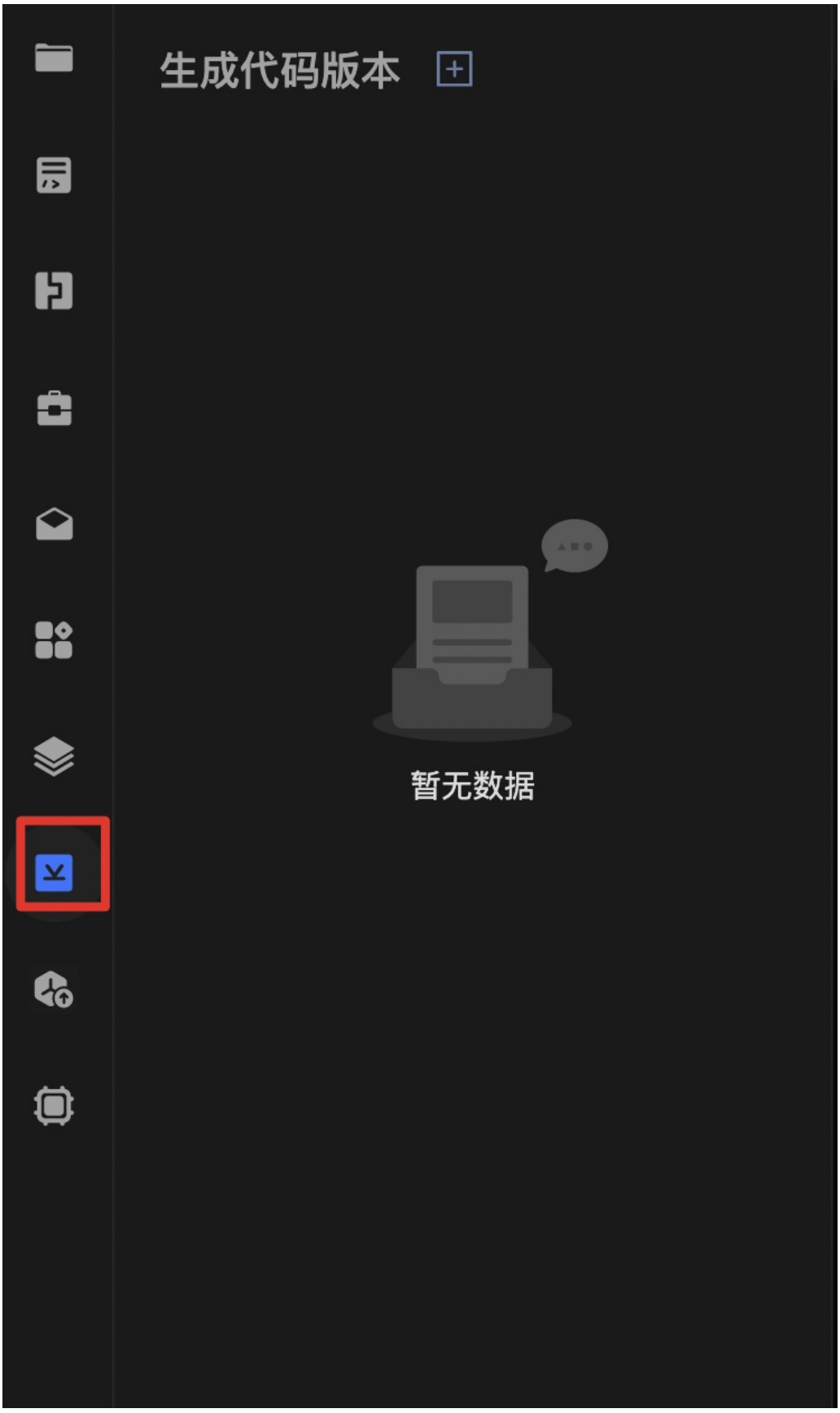


- 在导入数据集时，系统会自动在Notebook运行环境的“/home/data/”目录下创建以“数据集ID”为名的文件夹，并将数据集的原始文件以及标注文件复制到该文件夹中。
- 当原始数据集有更新时，data目录下的数据不会自动同步，需要手工进行同步。

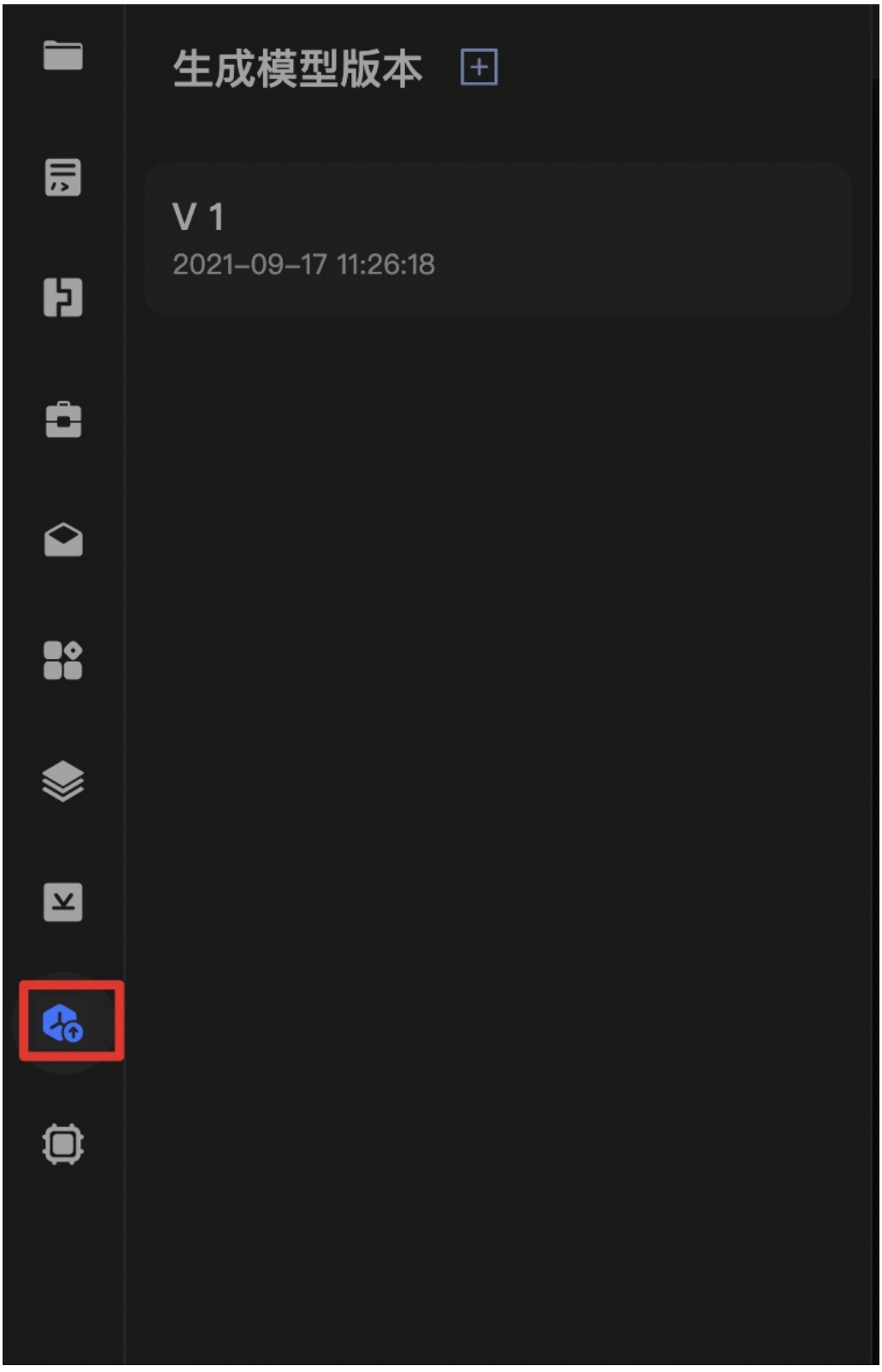
环境 用于查看当前Notebook运行的软硬件信息以及创建新的终端连接。


文件夹
数据集
环境
设置
硬件信息
CPU 8
RAM 8GB
磁盘 100GB
环境配置
Python版本 python3.7
框架版本 paddlepaddle1.7.1
终端连接（1/3）
终端 1
关闭终端
新建终端

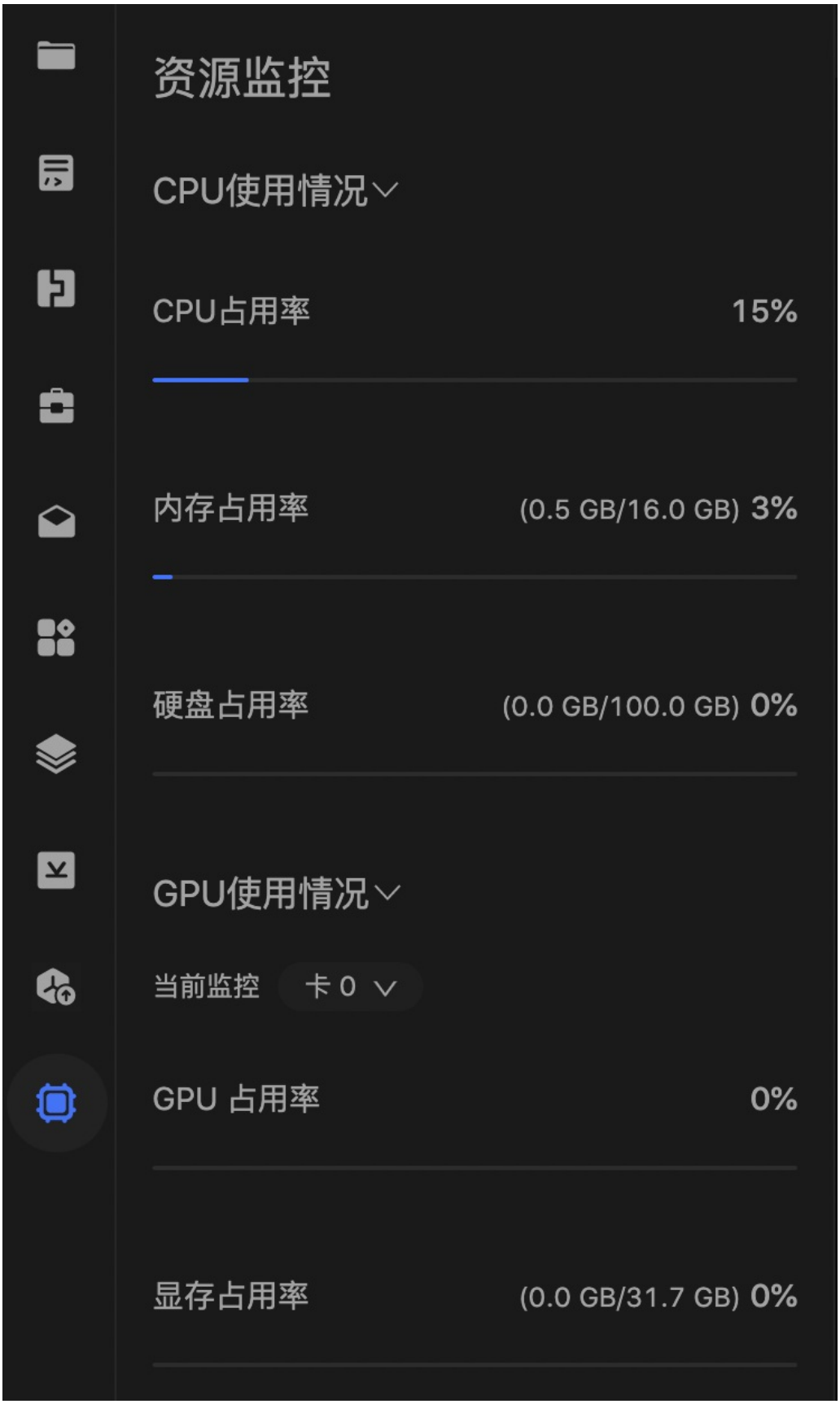
生成代码版本



生成模型版本



资源监控

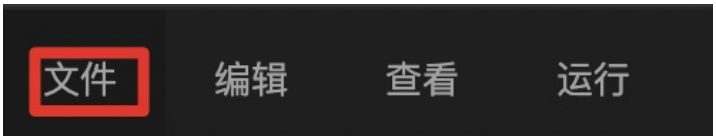


菜单栏

菜单栏包括文件、编辑、查看、运行、内核、标签页、设置、帮助几个功能。

文件

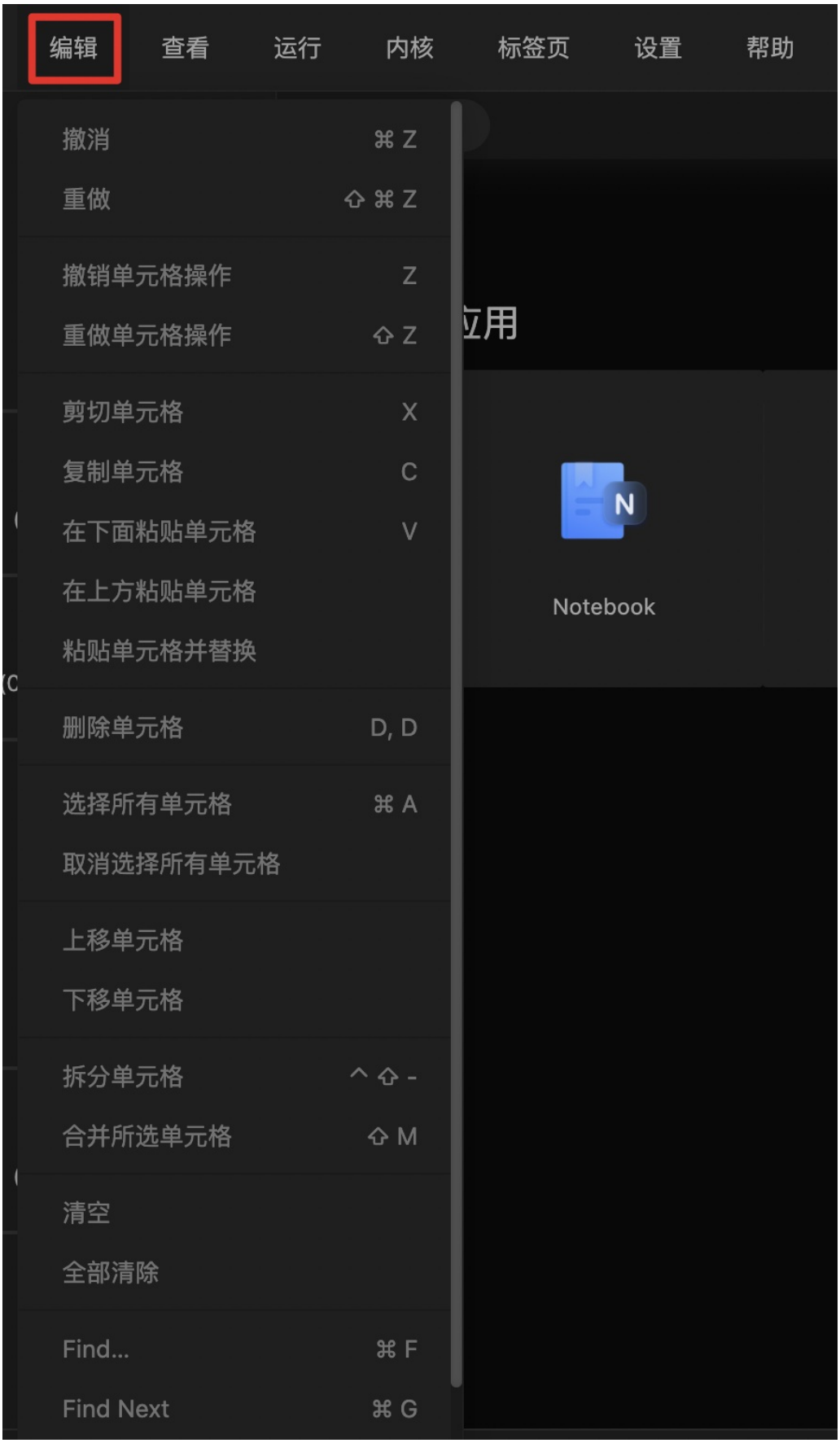
提供对Notebook文件的快捷操作。




新建
新建启动页 ⇧ ⌘ L
从路径打开...
的新视图
新建活动控制台
关闭标签页 ⌥ W
关闭并停止服务进程 ^ ⇧ Q
关闭所有标签
保存 ⌘ S
将 另存为... ⇧ ⌘ S
保存全部
从磁盘重新加载
将 恢复到检查点
重命名 ...
下载
导出笔记本为...
将当前工作区另存为...
保存当前工作区
打印... ⌘ P

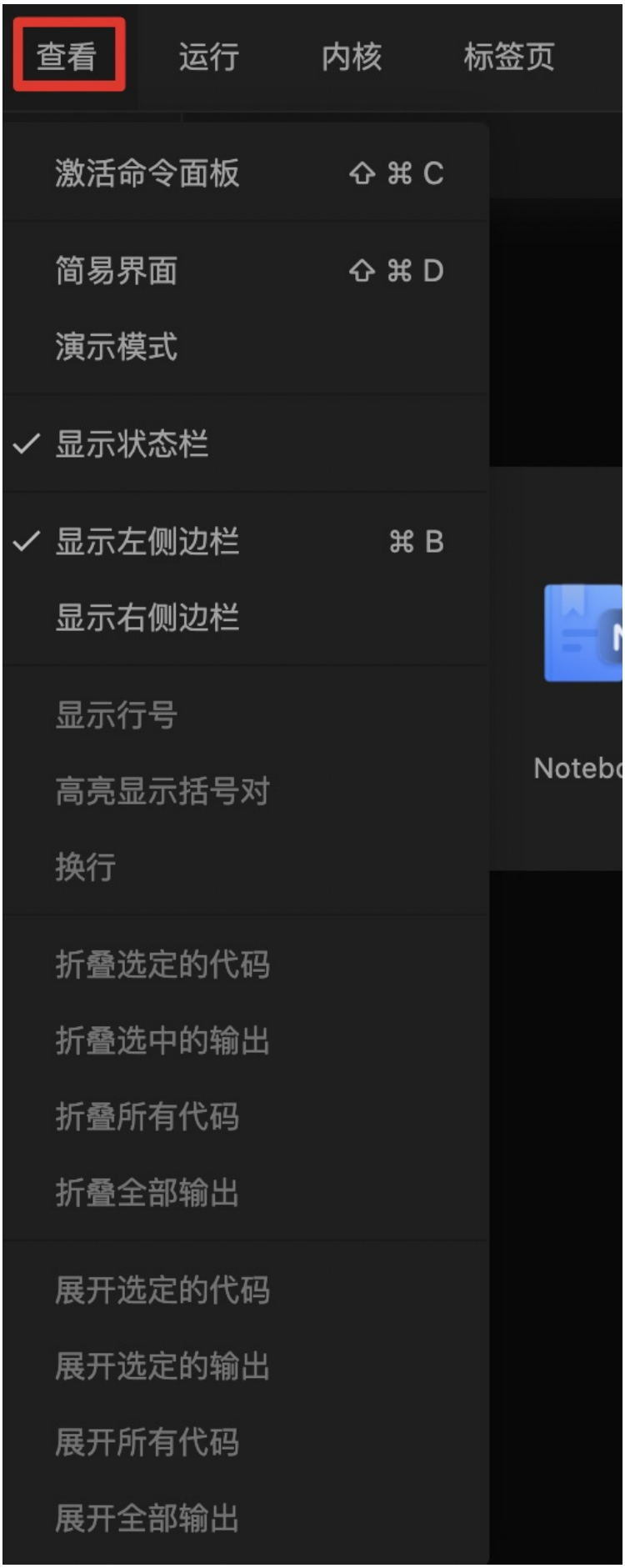
编辑

提供对Cell的各种快捷操作。



查看

提供查看的各种快捷操作。



运行

提供多种Cell运行方式。



内核

提供多种内核操作方式。



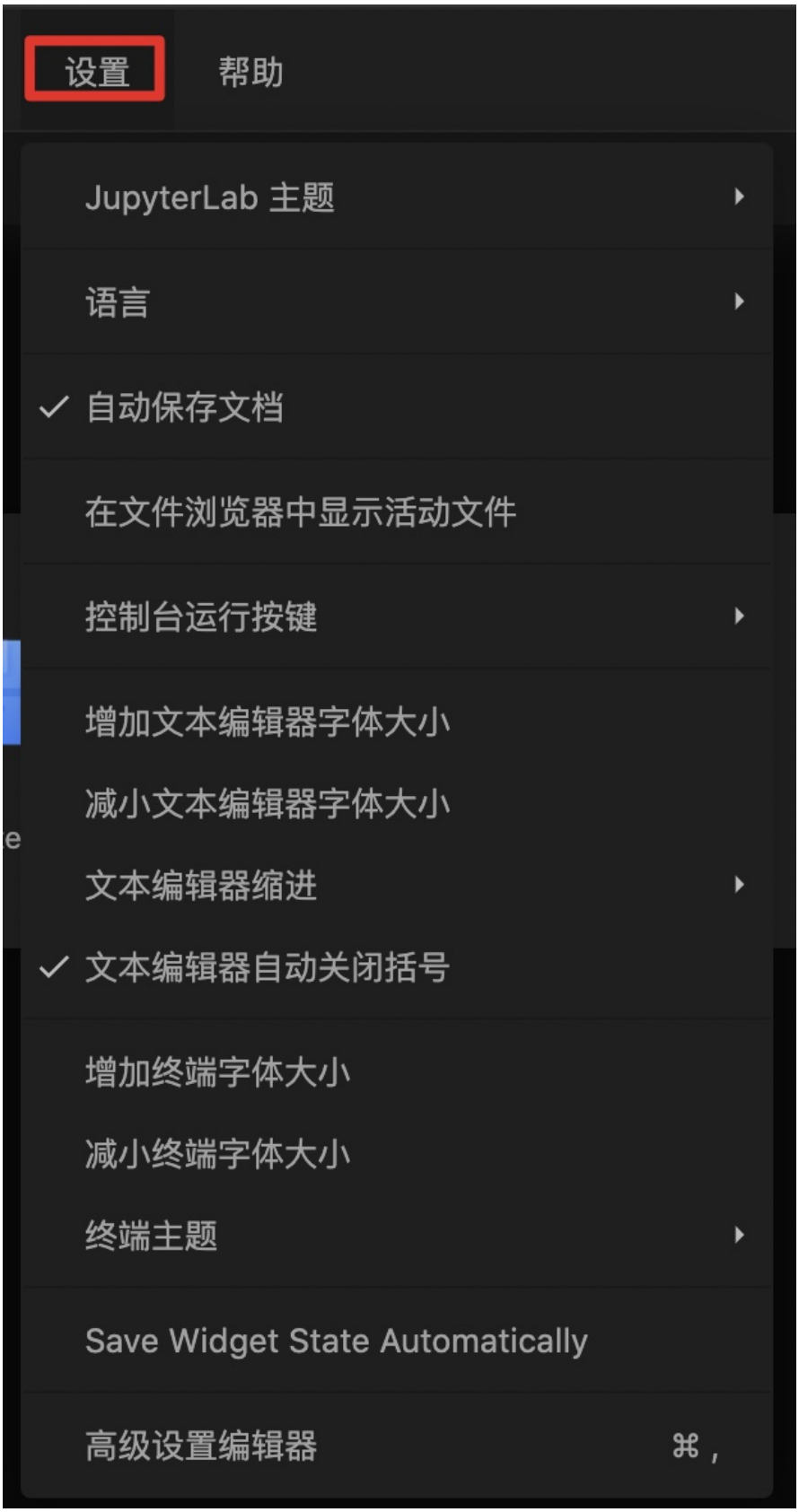
标签页

提供多种标签页操作方式。



设置

提供丰富的设置模式，包括主题、语言等设置。



帮助

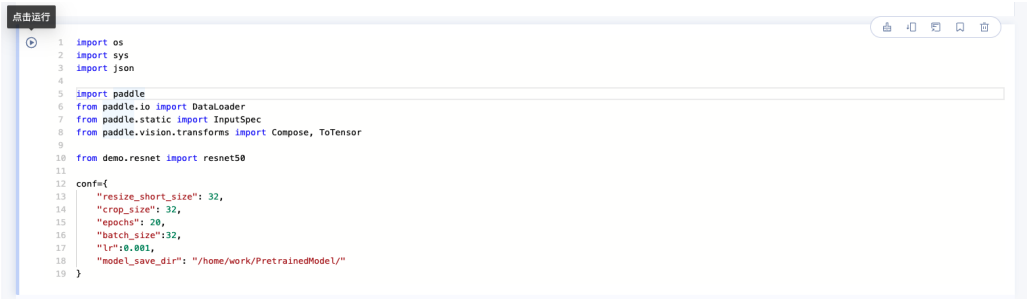
使用帮助，包括快捷键、使用说明、常见问题和飞桨API：



Code编辑区

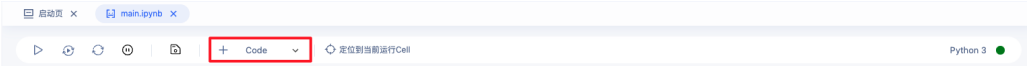
Code Cell是Notebook的代码编写单元。用户在Code Cell内编写代码（支持Python2、Python3）和shell命令，代码/命令在云端执行，并返回结果到Code Cell.

Code单元格操作

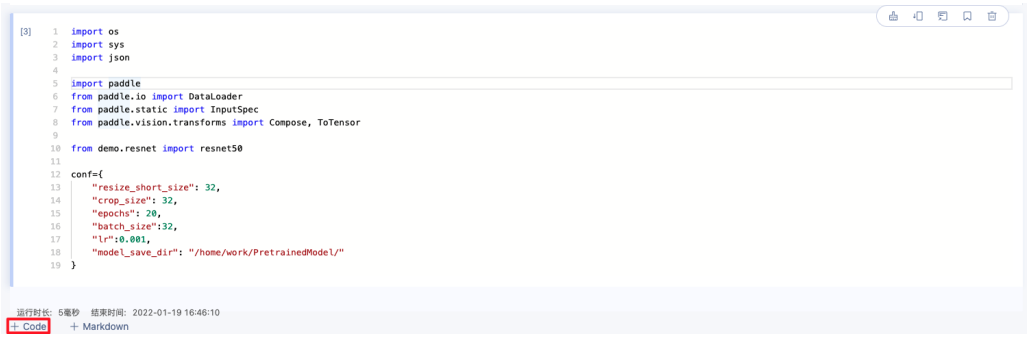


- 新建Cell

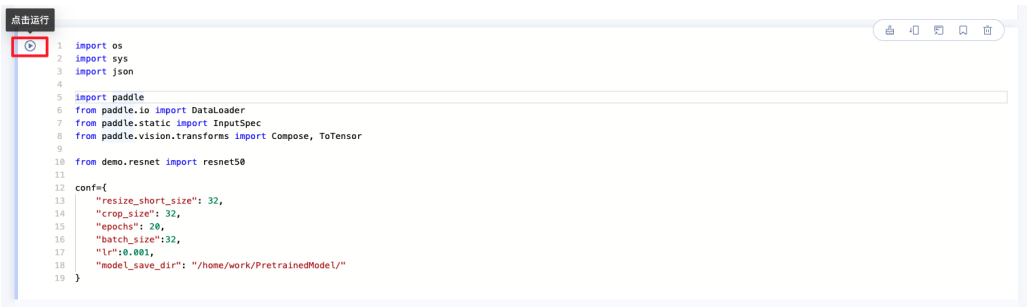
  从快捷工具栏中新建Code Cell



hover在Cell间隙区域新建Code Cell



- 运行Cell



运行中状态：



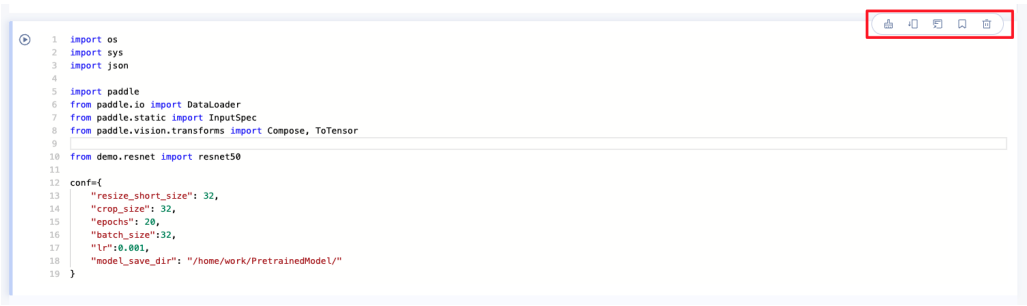
运行后结果输出：



- 中止Cel运行 如果发现代码并不尽如人意，可以点击中断按钮，打断所有Cell代码的执行



- 其他操作：依次为清空输出、折叠输入、折叠输出、收藏代码片段、删除



- Linux命令

  运行Linux命令的方式是在Linux命令前加一个`!`，就可以在块里运行

  示例:安装第三方包

- Magic关键字

  Magic关键字是可以运行特殊的命令. Magic 命令的前面带有一个或两个百分号(% 或 %%), 分别代表行Magic命令和Cell Magic命令. 行Magic命令仅应用于编写Magic命令时所在的行, 而Cell Magic命令应用于整个Cell.

Magic关键字	含义
%timeit	测试单行语句的执行时间
%%timeit	测试整个块中代码的执行时间
%matplotlib inline	显示 matplotlib 包生成的图形
%run	调用外部python脚本
%pdb	调试程序
%pwd	查看当前工作目录
%ls	查看目录文件列表
%reset	清除全部变量
%who	查看所有全局变量的名称，若给定类型参数，只返回该类型的变量列表
%whos	显示所有的全局变量名称、类型、值/信息
%xmode Plain	设置为当异常发生时只展示简单的异常信息
%xmode Verbose	设置为当异常发生时展示详细的异常信息
%debug	bug调试，输入quit退出调试
%bug	调试，输入quit退出调试
%env	列出全部环境变量

示例1: 使用 %%timeit测算整个块的运行时间.

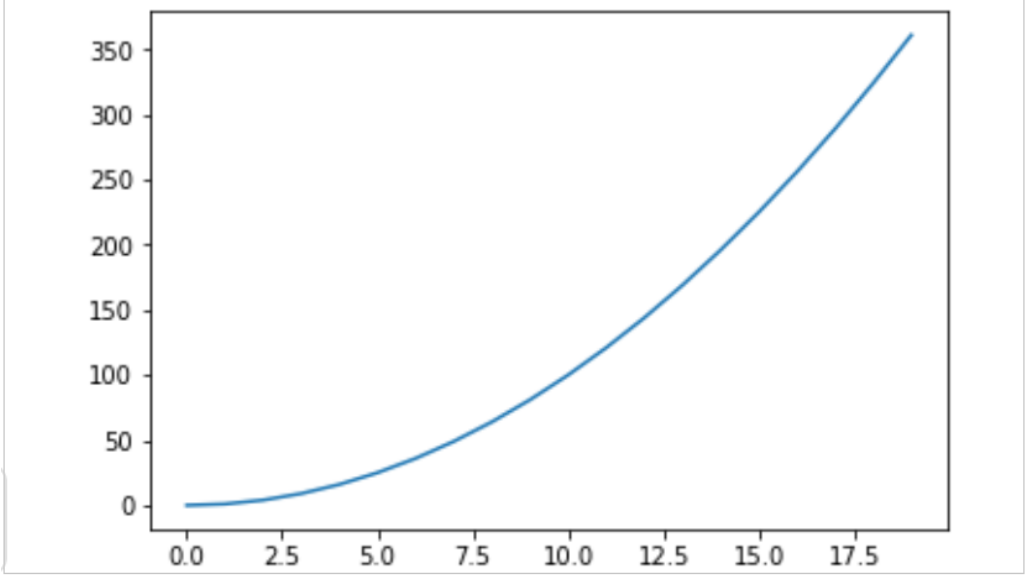
```
[3]
1  %%timeit
2
3  import random
4  prize = 0
5  for i in range(100):
6      roll = random.randint(1,6)
7      if roll % 2 == 0:
8          prize += roll
9      else:
10         prize -= 1

96.8 µs ± 245 ns per loop (mean ± std. dev. of 7 runs, 10000 loops each)
```

示例2: 块可集成Matplotlib,从而进行绘图, 但需要注意绘图前需要输入%Matplotlib inline并运行, 否则即使运行终端可用的绘图代码段, cell也只会返回一个文件说明, 如下图所示

```
[4]
1 %matplotlib inline
2
3 import matplotlib.pyplot as plt
4 import numpy as np
5
6 x = np.arange(20)
7 y = x**2
8 plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x7ff0901c9590>]
```



示例3: 查看所有支持的Magic关键字.

```
1 %lsmagic
```

```
Available line magics:
%alias %alias_magic %autoawait %autocall %automagic %autosave %bookmark %cat %cd %clear %colors %config %connect_info %cp %debug %dhist %dirs
%doctest_mode %ed %edit %env %gui %hist %history %killbgscripts %ldir %less %lf %lk %ll %load %load_ext %loadpy %logoff %logon %logstart %
logstate %logstop %ls %lsmagic %lx %macro %magic %man %matplotlib %mkdir %more %mv %notebook %page %pastebin %pdb %pdef %pdoc %pfile %pinf
o %pinfo2 %popd %pprint %precision %prun %psearch %psource %pushd %pwd %pycat %pylab %qtconsole %quickref %recall %rehashx %reload_ext %rep
%rerun %reset %reset_selective %rm %rmdir %run %save %sc %set_env %store %sx %system %tb %time %timeit %unalias %unload_ext %who %who_ls %
whos %xdel %xmode

Available cell magics:
%%! %%HTML %%SVG %%bash %%capture %%debug %%file %%html %%javascript %%js %%latex %%markdown %%perl %%prun %%pypy %%python %%python2 %%pytho
n3 %%ruby %%script %%sh %%svg %%sx %%system %%time %%timeit %%writefile

Automagic is ON, % prefix IS NOT needed for line magics.
```

Markdown编辑区

Markdown Cell是Notebook中文本编辑单元，通过在Markdown Cell中输入Markdown格式的文本，可以编写文字教程说明.

命令/编辑模式

绿色代表块内容可编辑状态-编辑模式（比如输入文字），**蓝色**代表块可操作状态-命令模式(预览md展示样式)，编辑模式和命令模式之间可以用Esc和Enter来切换。

- 编辑模式

```
### 读取数据且打印训练的中间信息

PaddlePaddle提供一个
[reader机制](https://github.com/PaddlePaddle/Paddle/tree/develop/doc/design/reader)
来读取数据。 Reader返回的数据可以包括多列，我们需要一个Python dict把列
序号映射到网络里的数据层。
```

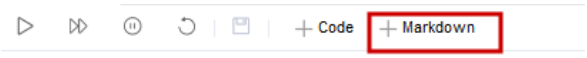
- 命令模式(预览样式)

读取数据且打印训练的中间信息

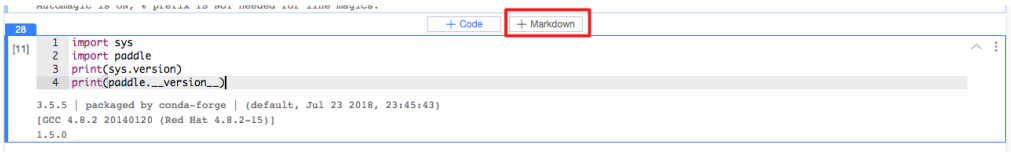
PaddlePaddle提供一个 reader机制 来读取数据。 Reader返回的数据可以包括多列，我们需要一个Python dict把列 序号映射到网络里的数据层。

Markdown单元格操作

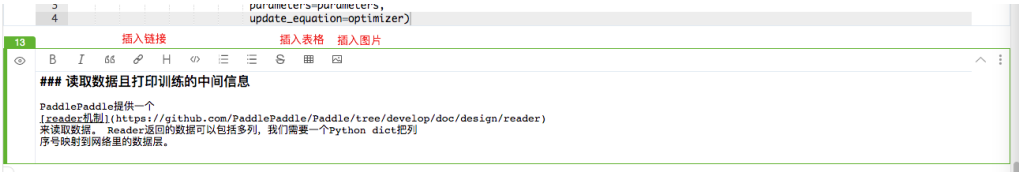
- 新建Cell 从快捷工具栏中新建Markdown Cell



hover在Cell间隙区域新建Markdown Cell



- 编辑Cell 支持插入公式、表格、图片、音乐、视频、网页等. 相关Markdown用法可以参考Markdown官网.



终端

用户可以使用终端来进行一些操作, 与Notebook主界面并行, 例如查询GPU占用率. 或进行文本编辑.

注意:

1. Terminal未开放sudo权限.
2. 最多可同时使用3个终端.

快捷键操作

快捷键列表

Notebook的命令模式和编辑模式都提供了丰富的快捷键，查看Notebook快捷键。

常用操作列表

模式	内容	快捷键(Windows)	快捷键(Mac)
命令模式 (Esc切换)	运行块	Shift-Enter	Shift-Enter
命令模式	在下方插入块	B	B
命令模式	在上方插入块	A	A
命令模式	删除块	d-d	d-d
命令模式	切换到编辑模式	Enter	Enter
编辑模式 (Enter切换)	运行块	Shift-Enter	Shift-Enter
编辑模式	缩进	Clrl-]	Command-]
编辑模式	取消缩进	Ctrl-[	Command-[
编辑模式	注释	Ctrl-/	Command-/
编辑模式	函数内省	Tab	Tab

如果遇到Notebook环境异常，或者保存代码失败。请提交工单联系群管解决

数据模型可视化功能说明

当前BML Notebook已经集成VisualDL工具以实现数据模型可视化，您可在可视化tab中启动VisualDL服务。

VisualDL工具

VisualDL是一个面向深度学习任务设计的可视化工具，利用丰富的图表来展示数据，用户可以更直观、清晰地查看数据的特征与变化趋势，有助于分析数据、及时发现错误，进而改进神经网络模型的设计。目前，VisualDL 支持 scalar, image, audio, graph, histogram, pr curve, high dimensional 七个组件，项目正处于高速迭代中，敬请期待新组件的加入。

组件名称	展示图表	作用
Scalar	折线图	动态展示损失函数值、准确率等标量数据
Image	图片可视化	显示图片，可显示输入图片和处理后的结果，便于查看中间过程的变化
Audio	音频可视化	播放训练过程中的音频数据，监控语音识别与合成等任务的训练过程
Graph	网络结构	展示网络结构、节点属性及数据流向，辅助学习、优化网络结构
Histogram	直方图	展示训练过程中权重、梯度等张量的分布
PR Curve	折线图	权衡精度与召回率之间的平衡关系
High Dimensional	数据降维	将高维数据映射到 2D/3D 空间来可视化嵌入，便于观察不同数据的相关性

Scalar--标量组件

介绍

Scalar 组件的输入数据类型为标量，该组件的作用是将训练参数以折线图形式呈现。将损失函数值、准确率等标量数据作为参数传入 scalar 组件，即可画出折线图，便于观察变化趋势。

记录接口

Scalar 组件的记录接口如下：

`add_scalar(tag, value, step, walltime=None)`

接口参数说明如下：

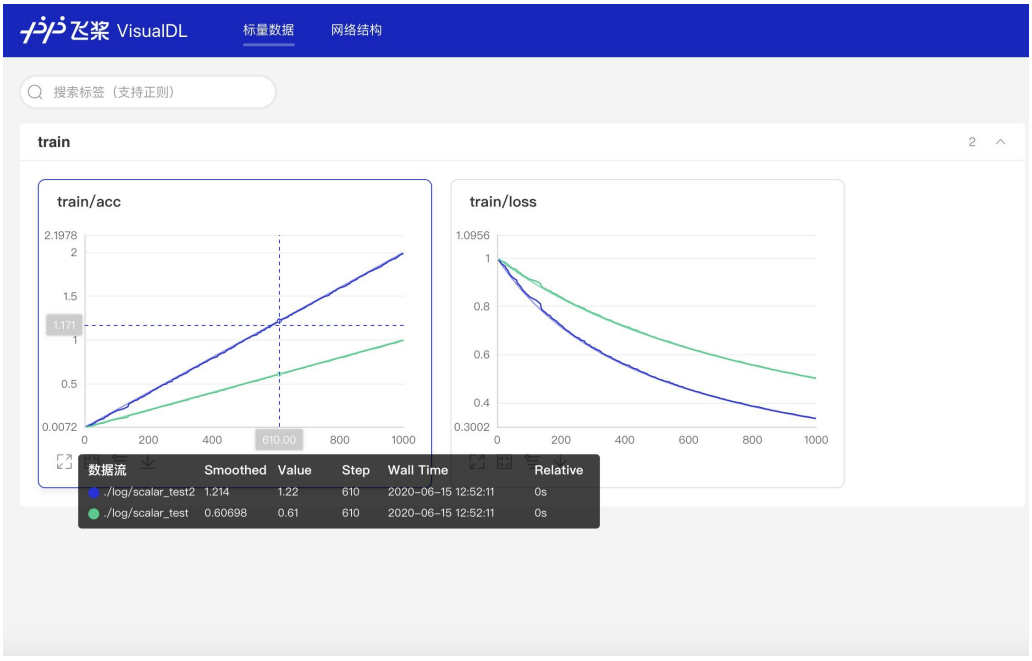
参数	格式	含义
tag	string	记录指标的标志，如`train/loss`，不能含有`%`
value	float	要记录的数据值
step	int	记录的步数
walltime	int	记录数据的时间戳，默认为当前时间戳

*注意tag的使用规则为：

1、第一个/前的为父tag，并作为一栏图片的tag
2、第一个/后的为子tag，子tag的对应图片将显示在父tag下
3、可以使用多次/，但一栏图片的tag依旧为第一个/前的tag

具体使用参见以下三个例子：

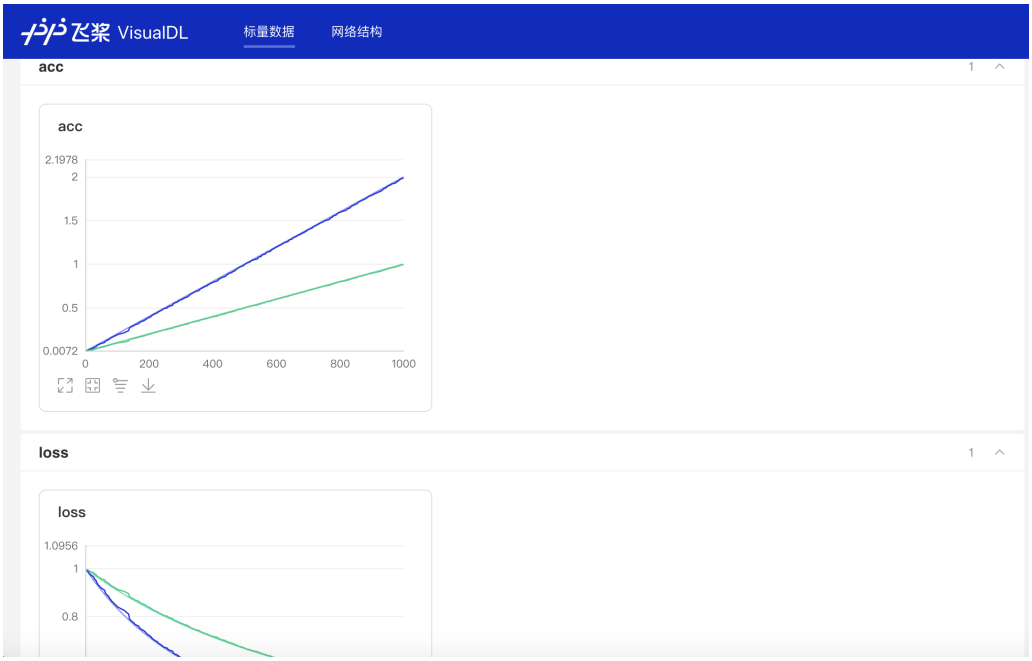
- 创建train为父tag，acc和loss为子tag：train/acc、 train/loss，即创建了tag为train的图片栏，包含acc和loss两张图片：



- 创建train为父tag，test/acc和test/loss为子tag：train/test/acc、 train/test/loss，即创建了tag为train的图片栏，包含test/acc和test/loss两张图片：



- 创建两个父tag：`acc`、 `loss`，即创建了tag分别为acc和loss的两个图片栏：



Demo

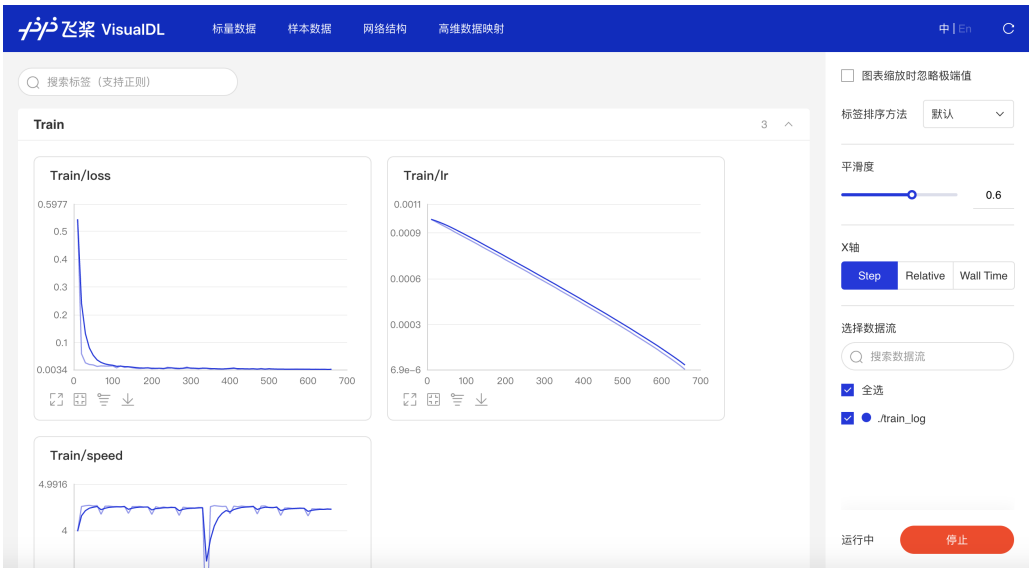
- 基础使用

下面展示了使用 Scalar 组件记录数据的示例：

```
 from visualdl import LogWriter

if __name__ == '__main__':
    value = [i/1000.0 for i in range(1000)]
    # 初始化一个记录器
    with LogWriter(logdir="./log/scalar_test/train") as writer:
        for step in range(1000):
            # 向记录器添加一个tag为`acc`的数据
            writer.add_scalar(tag="acc", step=step, value=value[step])
            # 向记录器添加一个tag为`loss`的数据
            writer.add_scalar(tag="loss", step=step, value=1/(value[step] + 1))
```

运行上述程序后，点击可视化选择相应日志文件即可查看可视化结果：



- 多组实验对比

下面展示了使用Scalar组件实现多组实验对比

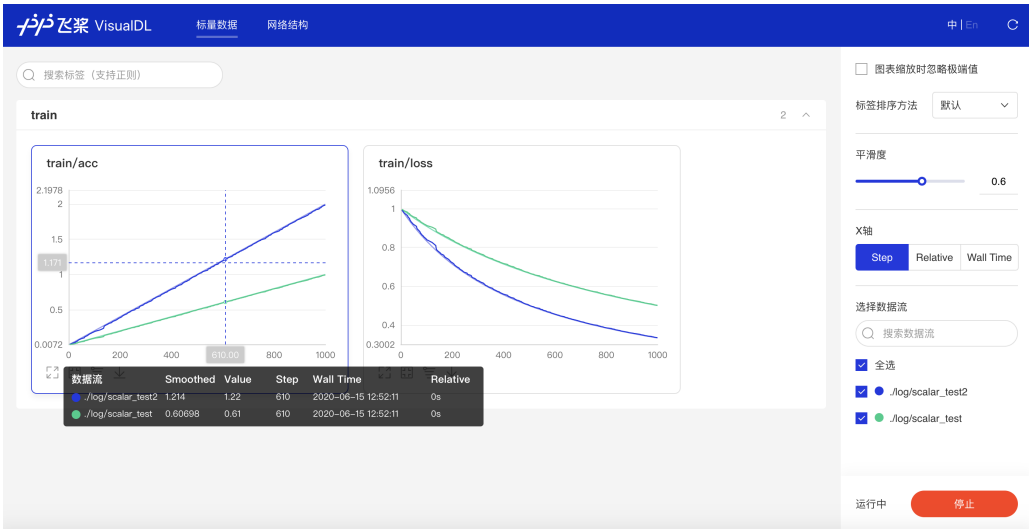
多组实验对比的实现分为两步：

1、创建子日志文件储存每组实验的参数数据 2、将数据写入scalar组件时，使用相同的tag，即可实现对比不同实验的同一类型参数

```
from visualdl import LogWriter

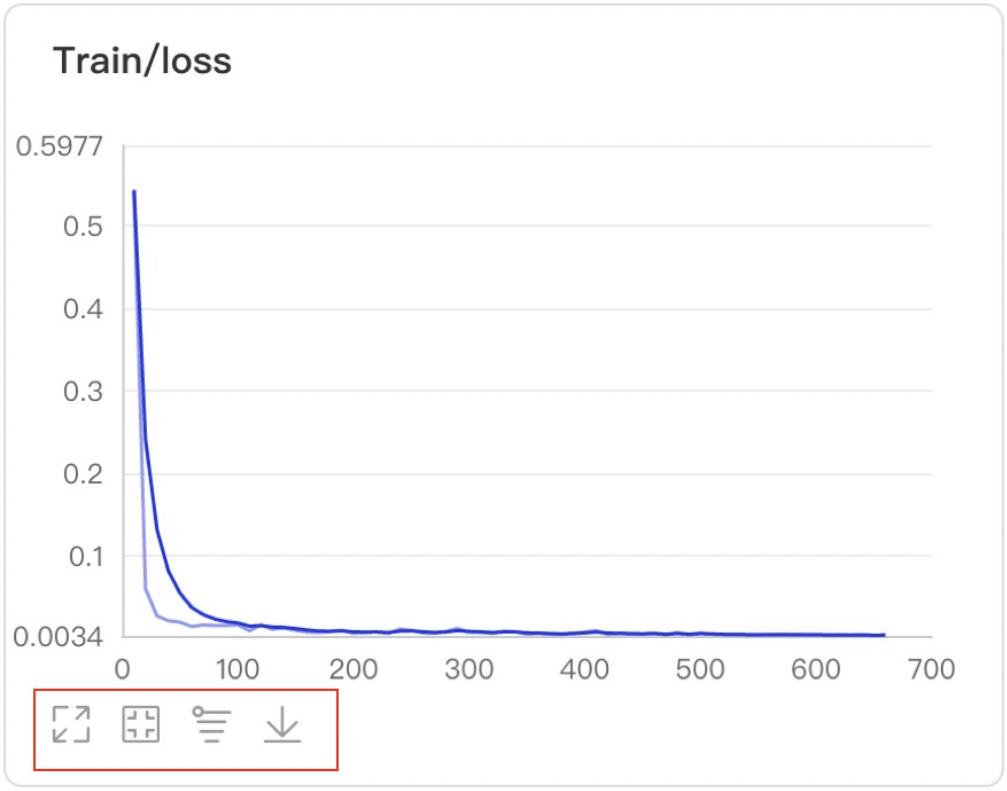
if __name__ == '__main__':
    value = [i/1000.0 for i in range(1000)]
    # 步骤一：创建父文件夹：log与子文件夹：scalar_test
    with LogWriter(logdir="./log/scalar_test") as writer:
        for step in range(1000):
            # 步骤二：向记录器添加一个tag为`train/acc`的数据
            writer.add_scalar(tag="train/acc", step=step, value=value[step])
            # 步骤二：向记录器添加一个tag为`train/loss`的数据
            writer.add_scalar(tag="train/loss", step=step, value=1/(value[step] + 1))
    # 步骤一：创建第二个子文件夹scalar_test2
    value = [i/500.0 for i in range(1000)]
    with LogWriter(logdir="./log/scalar_test2") as writer:
        for step in range(1000):
            # 步骤二：在同样名为`train/acc`下添加scalar_test2的accuracy的数据
            writer.add_scalar(tag="train/acc", step=step, value=value[step])
            # 步骤二：在同样名为`train/loss`下添加scalar_test2的loss的数据
            writer.add_scalar(tag="train/loss", step=step, value=1/(value[step] + 1))
```

运行上述程序后，点击可视化选择相应日志文件即可查看以下折线图，观察scalar_test和scalar_test2的accuracy和loss的对比。

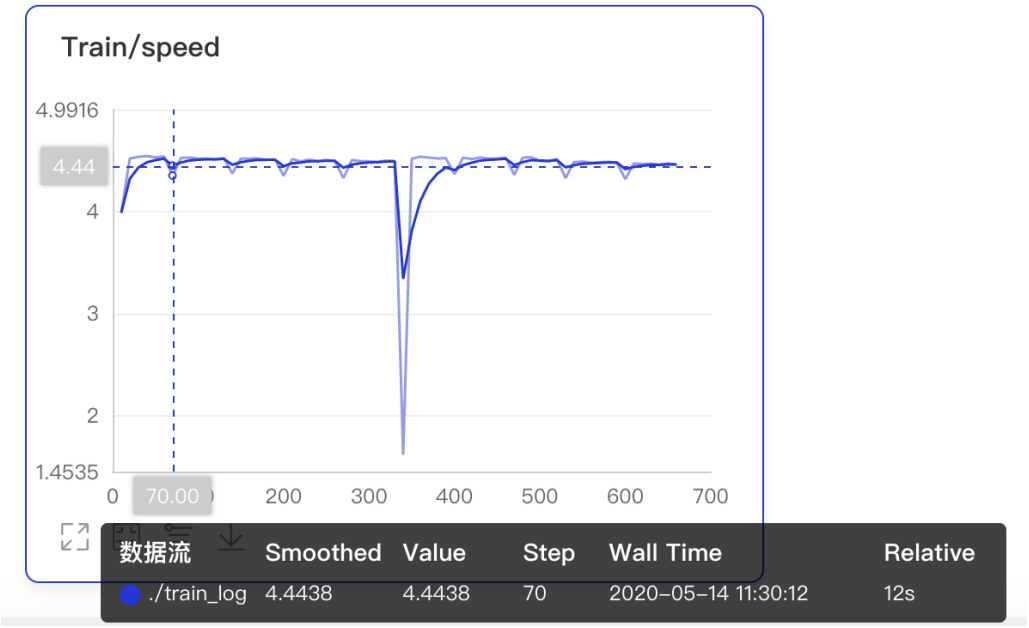


功能操作说明

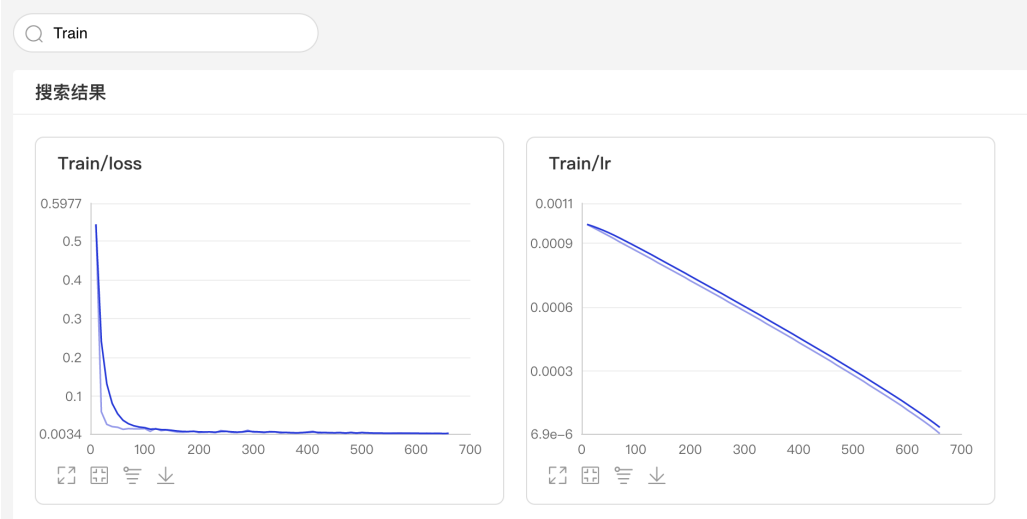
- 支持数据卡片「最大化」、「还原」、「坐标系转化」（y轴对数坐标）、「下载」折线图



- 数据点Hover展示详细信息



- 可搜索卡片标签，展示目标图像



- 可搜索打点数据标签，展示特定数据



- X轴有三种衡量尺度

1、Step：迭代次数
2、Walltime：训练绝对时间
3、Relative：训练时长



- 可调整曲线平滑度，以便更好的展现参数整体的变化趋势



Image--图片可视化组件

介绍

Image 组件用于显示图片数据随训练的变化。在模型训练过程中，将图片数据传入 Image 组件，就可在 VisualDL 的前端网页查看相应图片。

记录接口

Image 组件的记录接口如下：

`add_image(tag, img, step, walltime=None)`

接口参数说明如下：

参数	格式	含义
tag	string	记录指标的标志，如`train/loss`，不能含有`%`
img	numpy.ndarray	以ndarray格式表示的图片
step	int	记录的步数
walltime	int	记录数据的时间戳，默认为当前时间戳

Demo

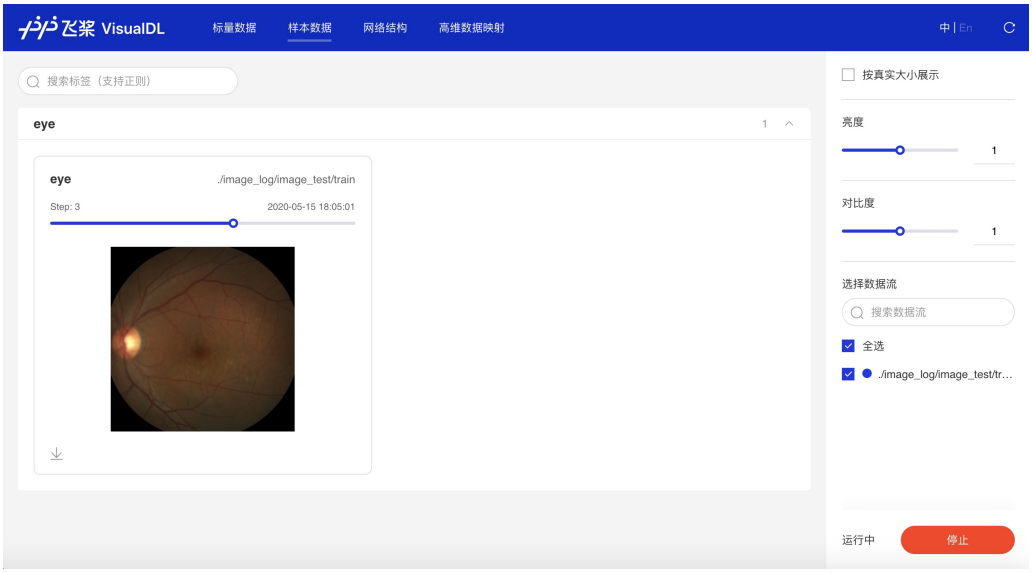
下面展示了使用 Image 组件记录数据的示例：

```
 import numpy as np
from PIL import Image
from visualdl import LogWriter


def random_crop(img):
    """获取图片的随机 100x100 分片
    """
    img = Image.open(img)
    w, h = img.size
    random_w = np.random.randint(0, w - 100)
    random_h = np.random.randint(0, h - 100)
    r = img.crop((random_w, random_h, random_w + 100, random_h + 100))
    return np.asarray(r)

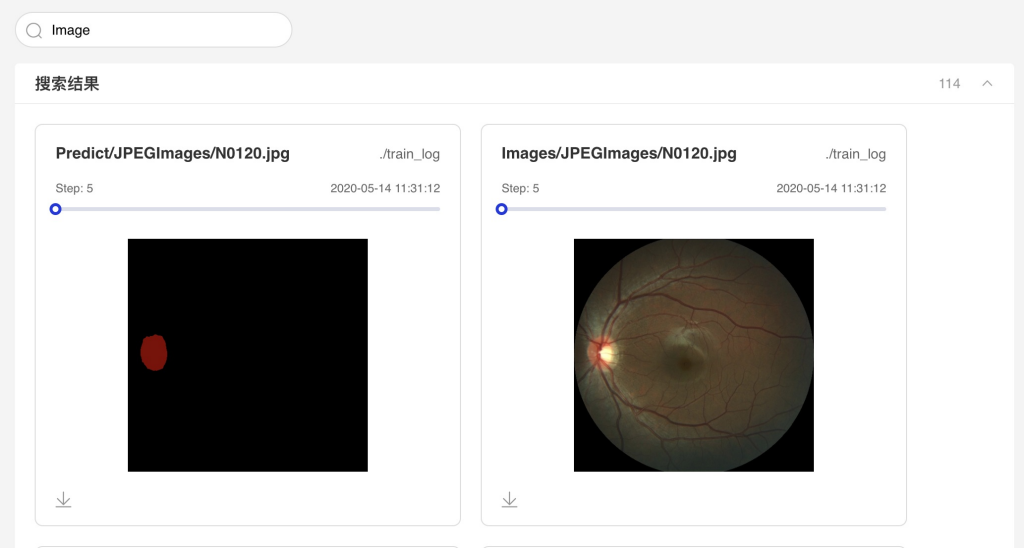

if __name__ == '__main__':
    # 初始化一个记录器
    with LogWriter(logdir="./log/image_test/train") as writer:
        for step in range(6):
            # 添加一个图片数据
            writer.add_image(tag="eye",
                             img=random_crop("../../docs/images/eye.jpg"),
                             step=step)
```

运行上述程序后，点击可视化选择相应日志文件即可查看可视化结果。

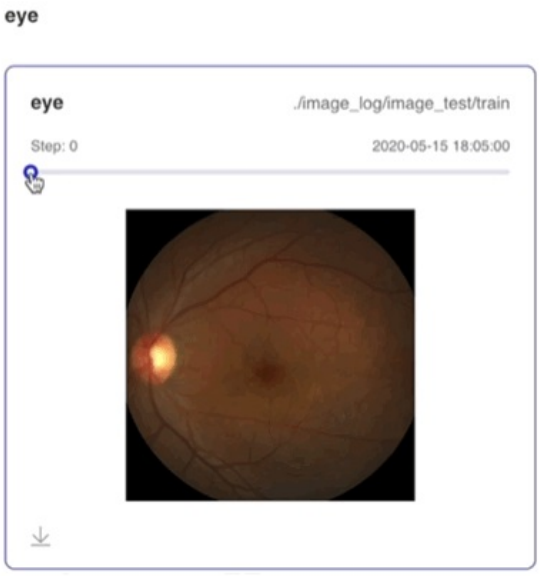


功能操作说明

- 可搜索图片标签显示对应图片数据



- 支持滑动Step/迭代次数查看不同迭代次数下的图片数据



Audio--音频播放组件

介绍

Audio组件实时查看训练过程中的音频数据，监控语音识别与合成等任务的训练过程。

记录接口

Audio 组件的记录接口如下：

`add_audio(tag, audio_array, step, sample_rate)`

接口参数说明如下：

参数	格式	含义
tag	string	记录指标的标志，如`train/loss`，不能含有`%`
audio_arry	numpy.ndarray	以ndarray格式表示的音频
step	int	记录的步数
sample_rate	int	采样率，*注意正确填写对应音频的采样率*

Demo

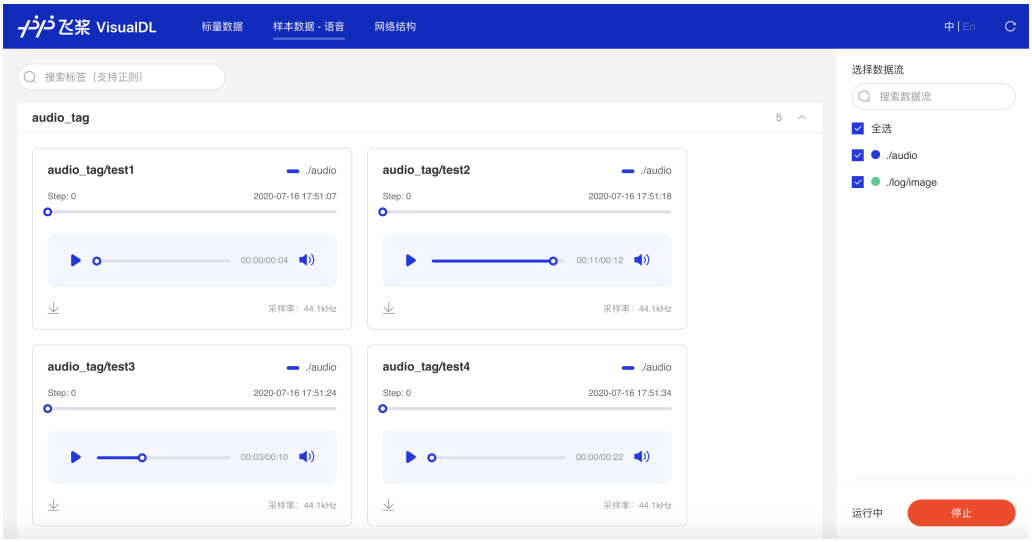
下面展示了使用 Image 组件记录数据的示例：

```
 from visualdl import LogWriter
import numpy as np
import wave


def read_audio_data(audio_path):
    """
    Get audio data.
    """
    CHUNK = 4096
    f = wave.open(audio_path, "rb")
    wavdata = []
    chunk = f.readframes(CHUNK)
    while chunk:
        data = np.frombuffer(chunk, dtype='uint8')
        wavdata.extend(data)
        chunk = f.readframes(CHUNK)
    # 8k sample rate, 16bit frame, 1 channel
    shape = [8000, 2, 1]
    return shape, wavdata

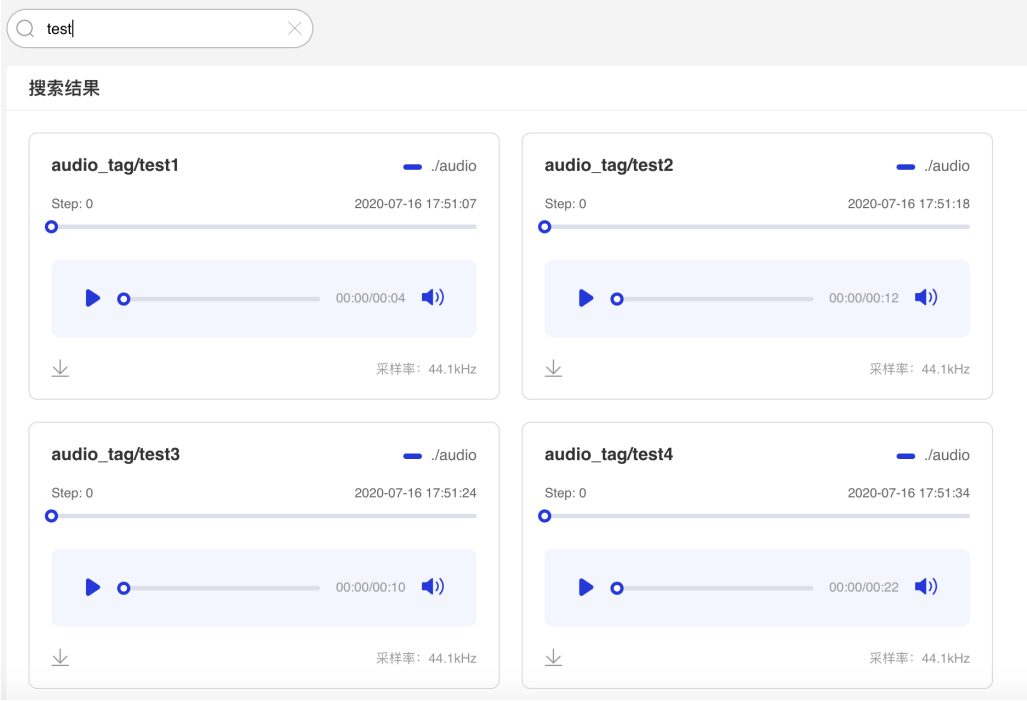

if __name__ == '__main__':
    with LogWriter(logdir="./log") as writer:
        audio_shape, audio_data = read_audio_data("./testing.wav")
        audio_data = np.array(audio_data)
        writer.add_audio(tag="audio_tag",
                         audio_array=audio_data,
                         step=0,
                         sample_rate=8000)
```

运行上述程序后，点击可视化选择相应日志文件即可查看可视化结果。



功能操作说明

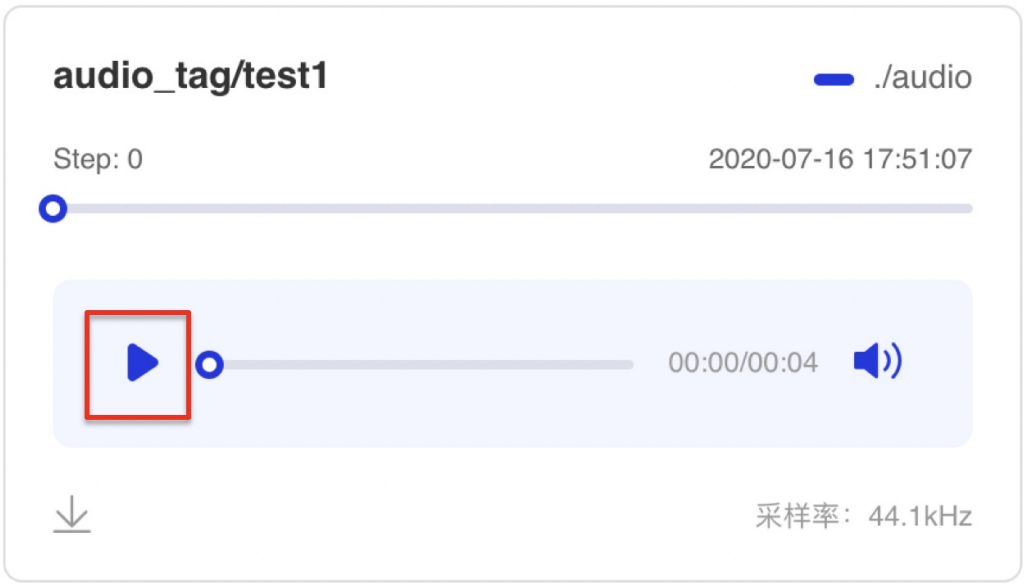
- 可搜索音频标签显示对应音频数据



- 支持滑动Step/迭代次数查看不同迭代次数下的音频数据



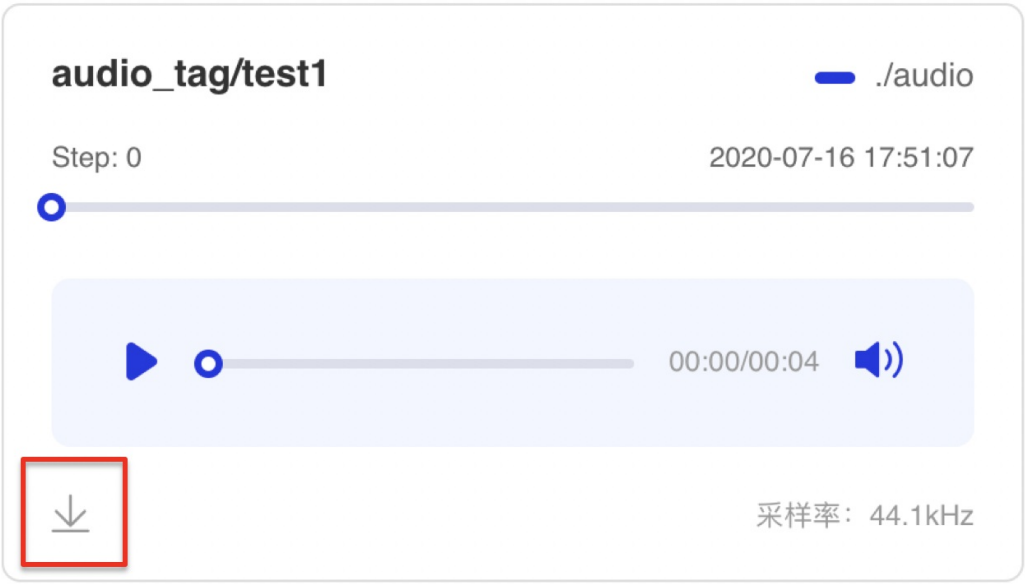
- 支持播放/暂停音频数据



- 支持音量调节



- 支持音频下载

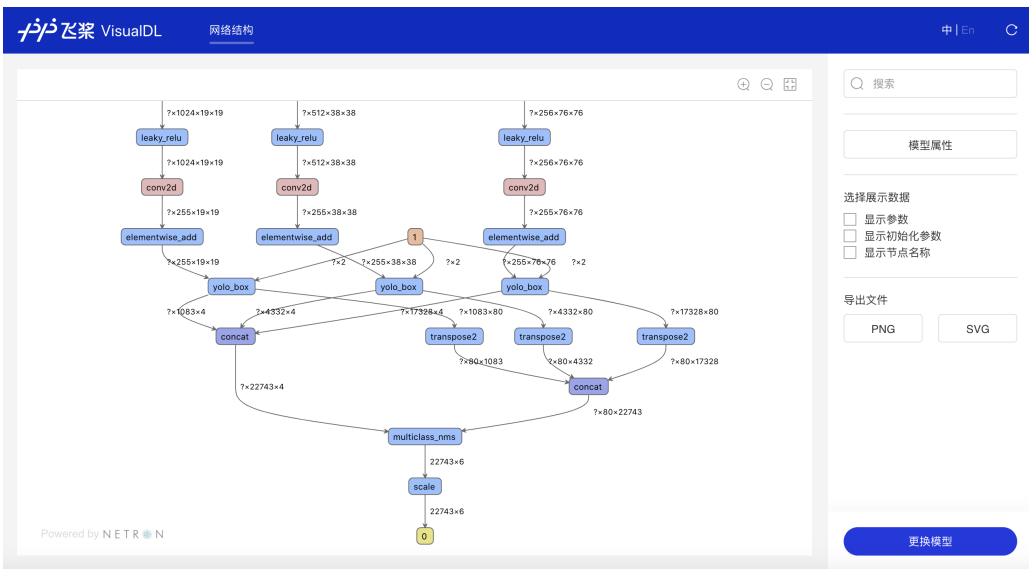


Graph--网络结构组件

介绍

Graph组件一键可视化模型的网络结构。用于查看模型属性、节点信息、节点输入输出等，并进行节点搜索，协助开发者们快速分析模型结构与了解数据流向。

在生成Model文件后，在可视化模块中选择模型文件，启动后即可查看网络结构可视化：



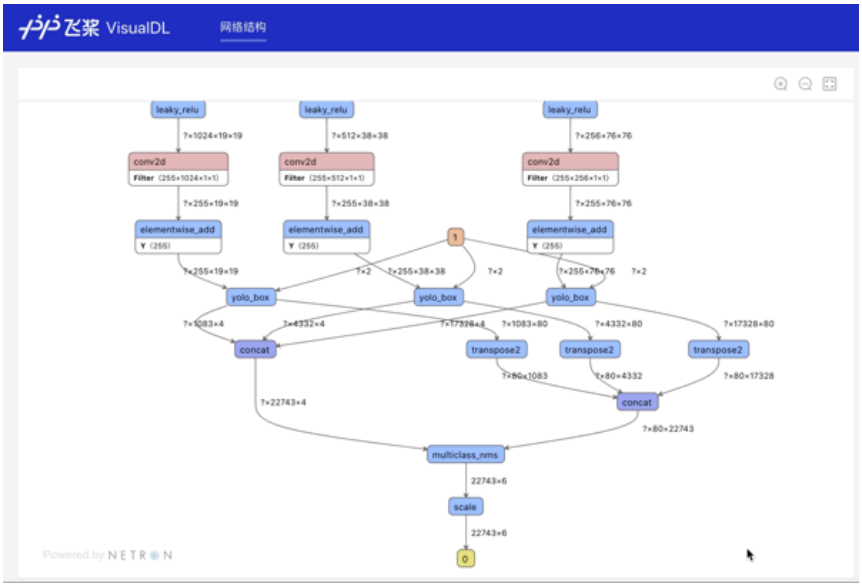
功能操作说明

- 一键上传模型
  - 支持模型格式：PaddlePaddle、ONNX、Keras、Core ML、Caffe、Caffe2、Darknet、MXNet、ncnn、TensorFlow Lite
  - 实验性支持模型格式：TorchScript、PyTorch、Torch、 ArmNN、BigDL、Chainer、CNTK、Deeplearning4j、MediaPipe、ML.NET、MNN、OpenVINO、Scikit-learn、Tengine、TensorFlow.js、TensorFlow



VisualDL支持：	PaddlePaddle、ONNX、Keras、Core ML、Caffe、Caffe2、Darknet、MXNet、ncnn、TensorFlow Lite
VisualDL实验性支持：	TorchScript、PyTorch、Torch、 ArmNN、BigDL、Chainer、CNTK、Deeplearning4j、MediaPipe、ML.NET、MNN、OpenVINO、Scikit-learn、Tengine、TensorFlow.js、TensorFlow

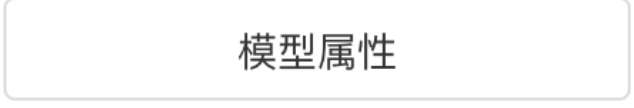
- 支持上下左右任意拖拽模型、放大和缩小模型



- 搜索定位到对应节点

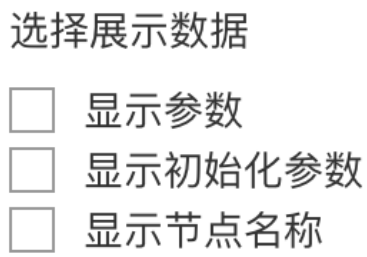


- 点击查看模型属性





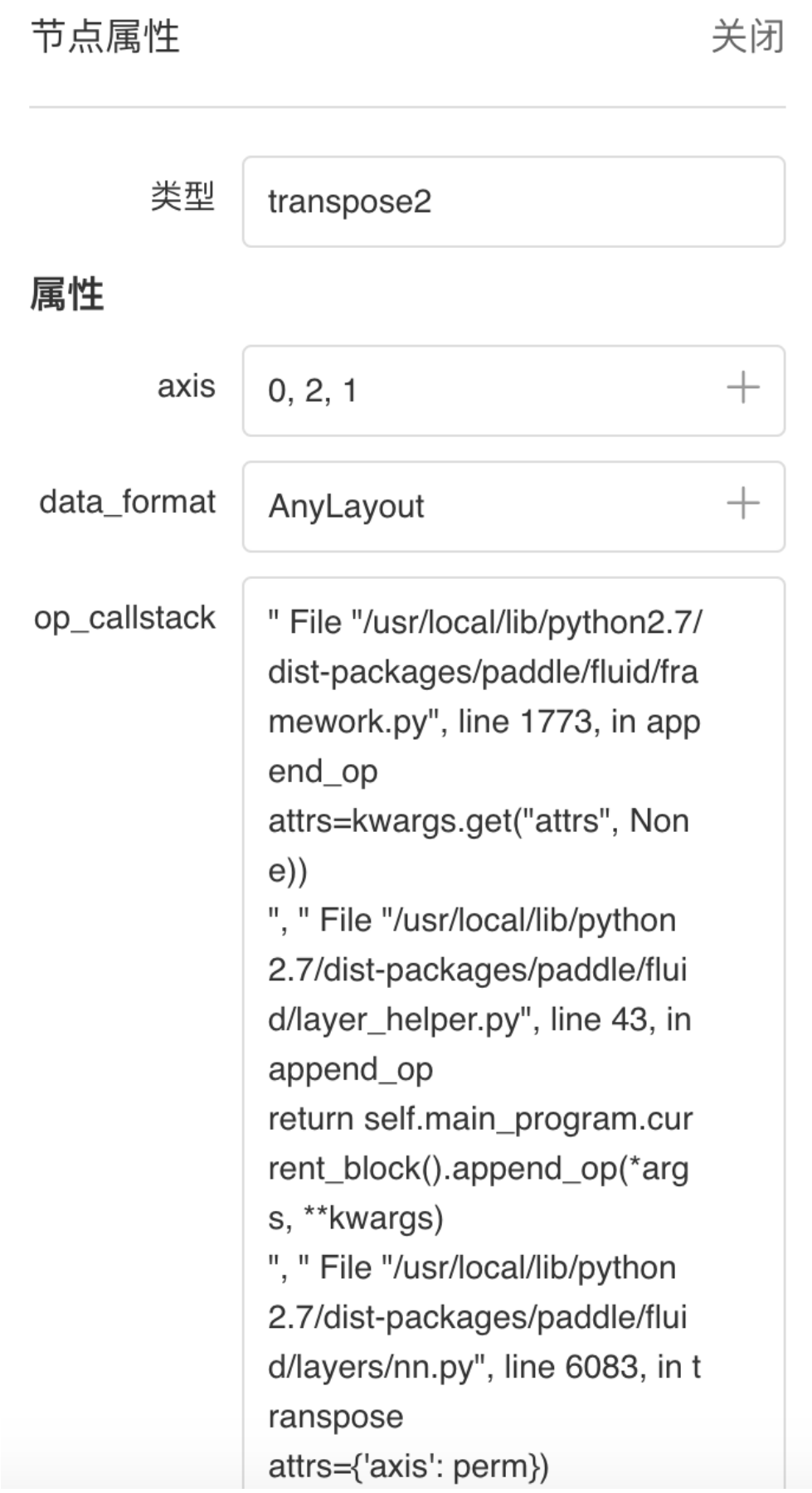
- 支持选择模型展示的信息



- 支持以PNG、SVG格式导出文件



- 点击节点即可展示对应属性信息



- 支持一键更换模型

更换模型

Histogram--直方图组件

介绍

Histogram组件以直方图形式展示Tensor（weight、bias、gradient等）数据在训练过程中的变化趋势。深入了解模型各层效果，帮助开发者精准调整模型结构。

记录接口

Histogram 组件的记录接口如下：

`add_histogram(tag, values, step, walltime=None, buckets=10)`

接口参数说明如下：

参数	格式	含义
tag	string	记录指标的标志，如`train/loss`，不能含有`%`
values	numpy.ndarray or list	以ndarray或list格式表示的数据
step	int	记录的步数
walltime	int	记录数据的时间戳，默认为当前时间戳
buckets	int	生成直方图的分段数，默认为10

Demo

下面展示了使用 Histogram 组件记录数据的示例：

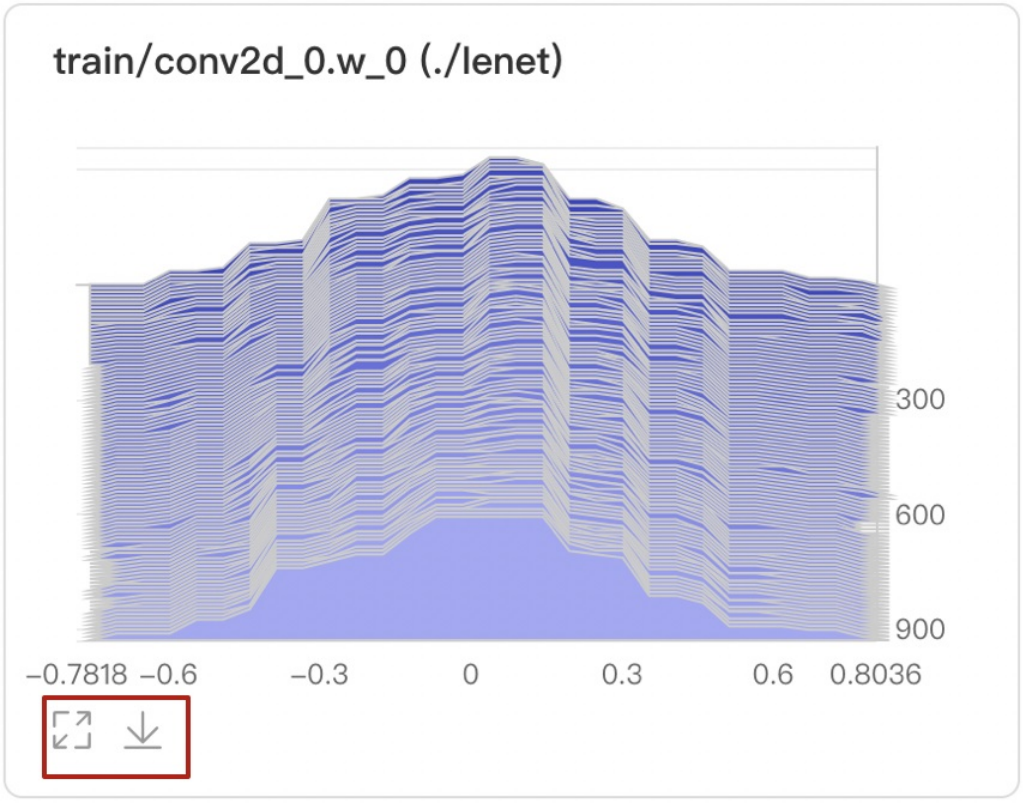
```
  from visualdl import LogWriter
import numpy as np


if __name__ == '__main__':
    values = np.arange(0, 1000)
    with LogWriter(logdir="./log/histogram_test/train") as writer:
        for index in range(1, 101):
            interval_start = 1 + 2 * index / 100.0
            interval_end = 6 - 2 * index / 100.0
            data = np.random.uniform(interval_start, interval_end, size=(10000))
            writer.add_histogram(tag='default tag',
                                 values=data,
                                 step=index,
                                 buckets=10)
```

运行上述程序后，点击可视化选择相应日志文件即可查看可视化结果。

功能操作说明

- 支持数据卡片「最大化」、「下载」直方图

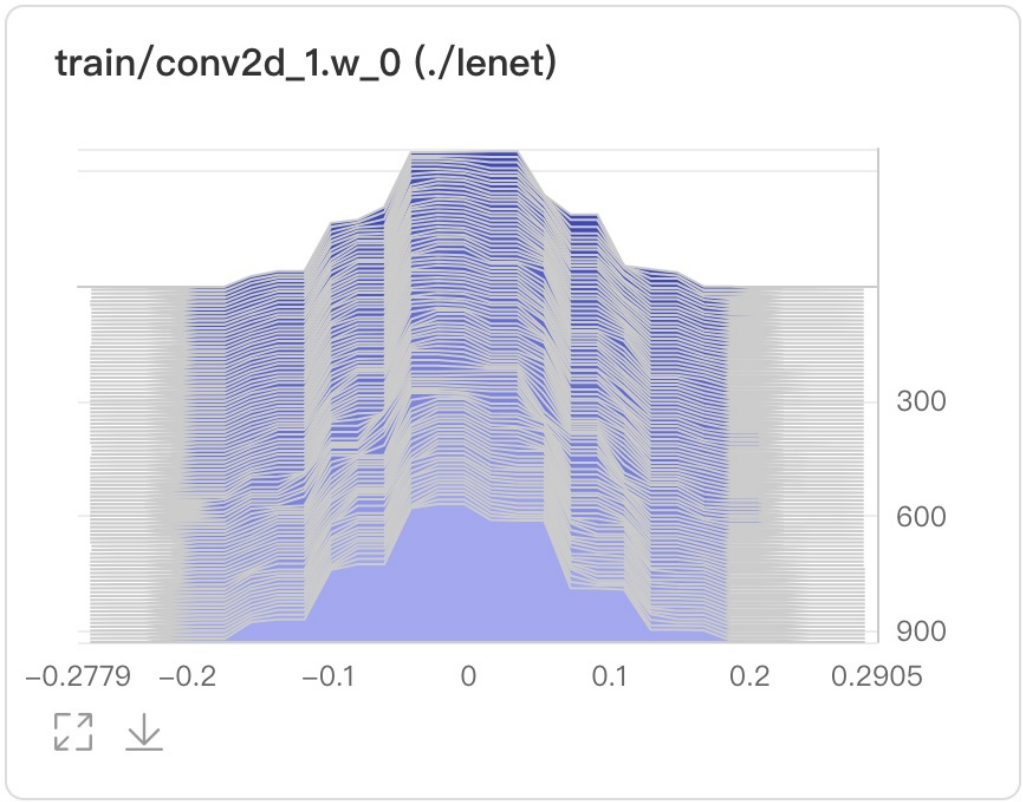


- 可选择Offset或Overlay模式

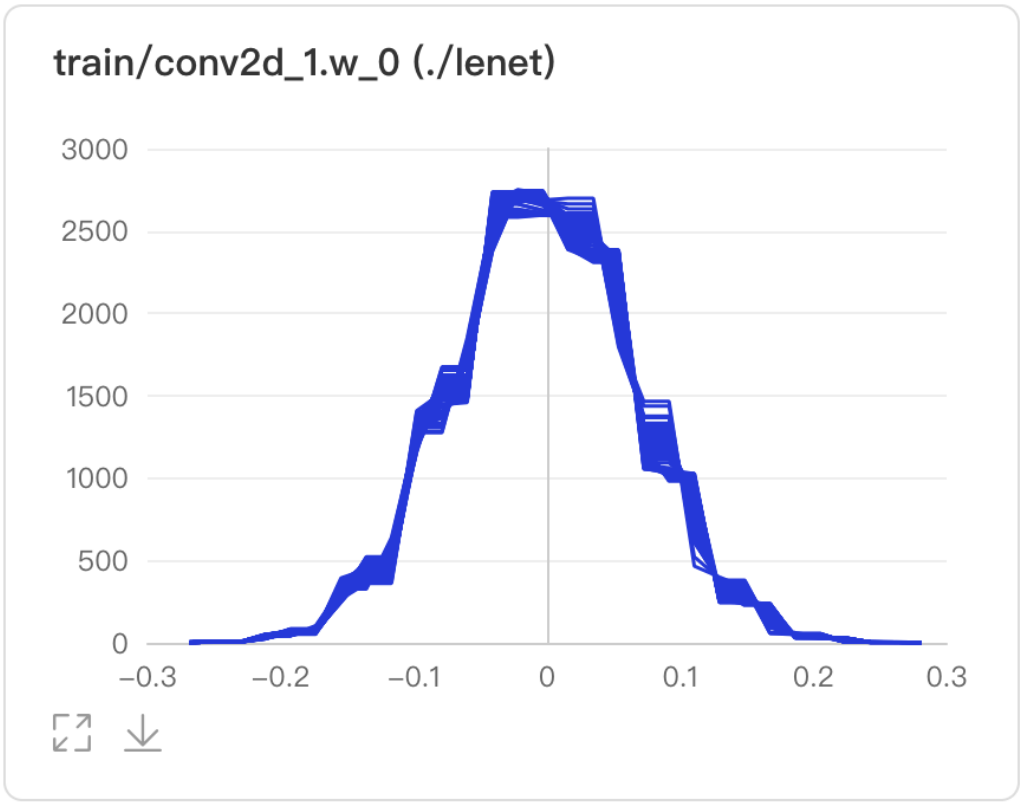
直方图模式



1. Offset模式

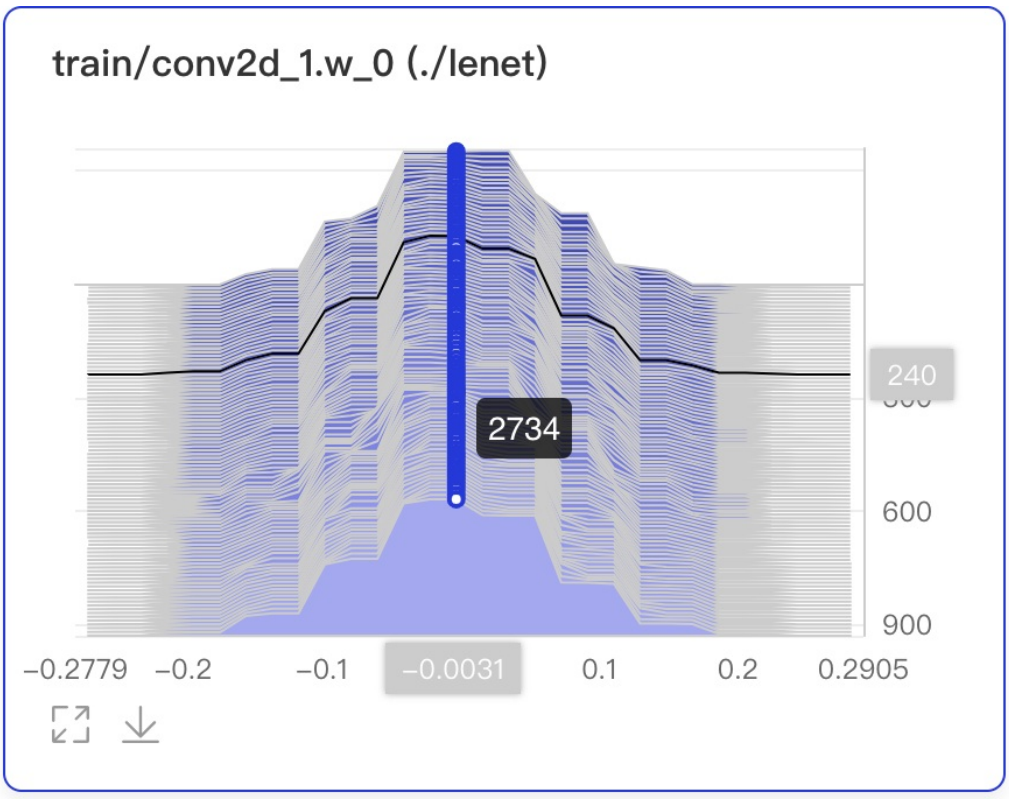


2. Overlay模式



- 数据点Hover展示参数值、训练步数、频次

1. 在第240次训练步数时，权重为-0.0031，且出现的频次是2734次



- 可搜索卡片标签，展示目标直方图



- 可搜索打点数据标签，展示特定数据流



PR Curve--PR曲线组件

介绍

PR Curve以折线图形式呈现精度与召回率的权衡分析，清晰直观了解模型训练效果，便于分析模型是否达到理想标准。

记录接口

PR Curve 组件的记录接口如下：

`add_pr_curve(tag, labels, predictions, step=None, num_thresholds=10)`

接口参数说明如下：

参数	格式	含义
tag	string	记录指标的标志，如`train/loss`，不能含有`%`
values	numpy.ndarray or list	以ndarray或list格式表示的实际类别
predictions	numpy.ndarray or list	以ndarray或list格式表示的预测类别
step	int	记录的步数
num_thresholds	int	阈值设置的个数，默认为10，最大值为127

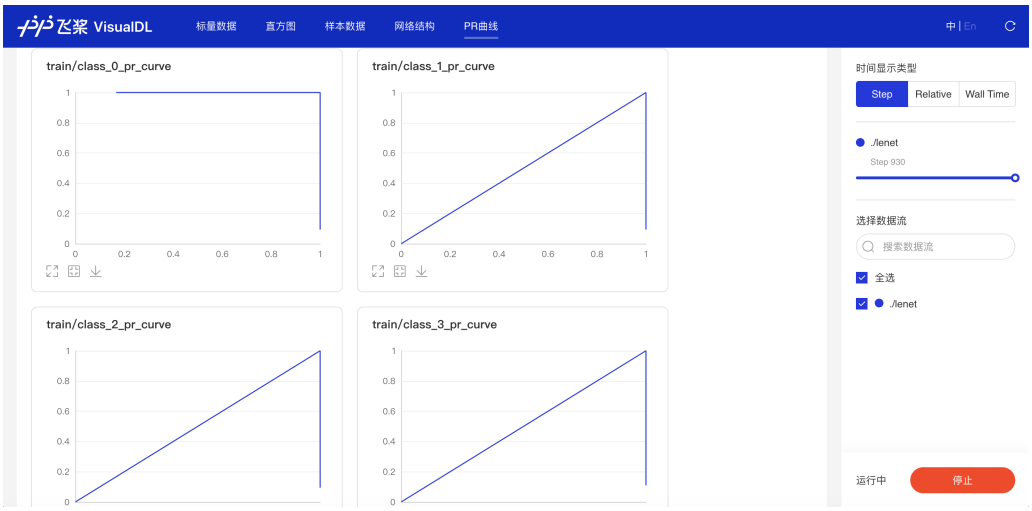
Demo

下面展示了使用 PR Curve 组件记录数据的示例：

```
 from visualdl import LogWriter
import numpy as np

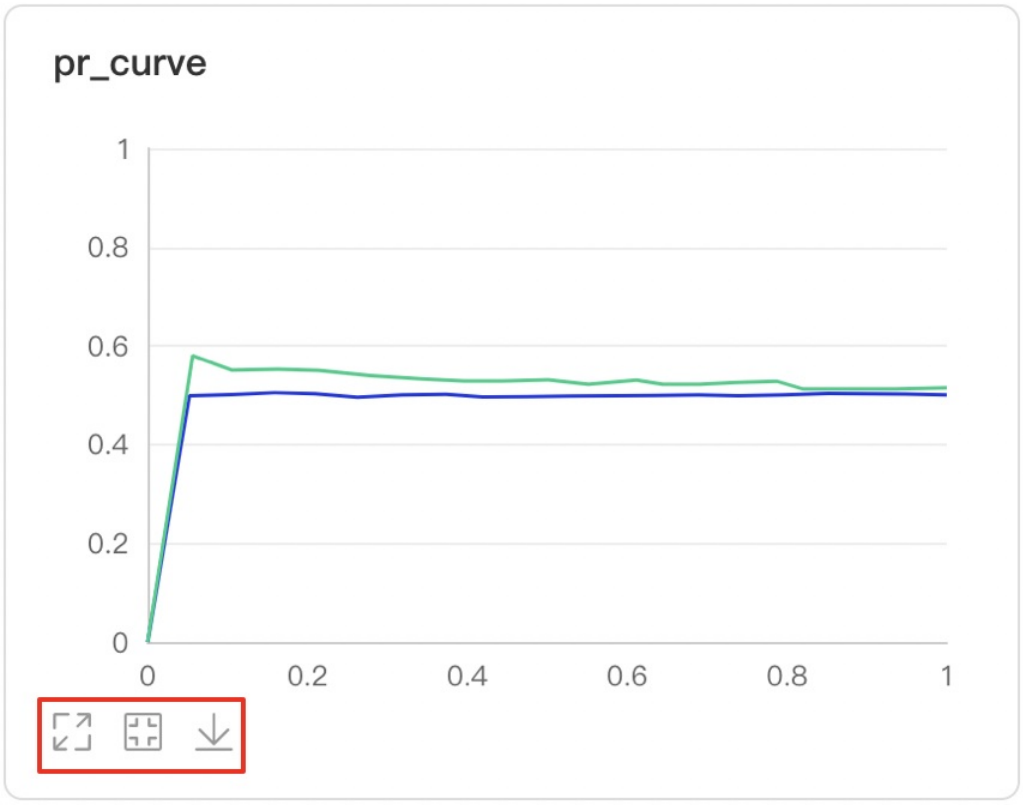
with LogWriter("./log/pr_curve_test/train") as writer:
    for step in range(3):
        labels = np.random.randint(2, size=100)
        predictions = np.random.rand(100)
        writer.add_pr_curve(tag='pr_curve',
                            labels=labels,
                            predictions=predictions,
                            step=step,
                            num_thresholds=5)
```

运行上述程序后，点击可视化选择相应日志文件即可查看可视化结果。

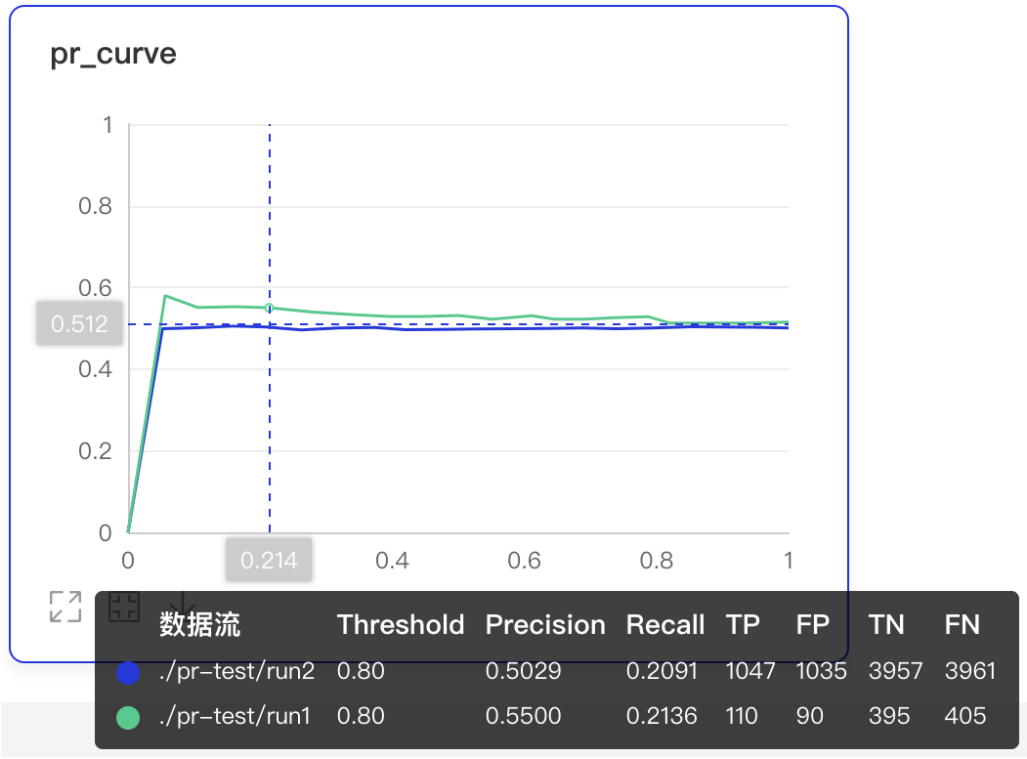


功能操作说明

- 支持数据卡片「最大化」、「还原」、「下载」PR曲线



- 数据点Hover展示详细信息：阈值对应的TP、TN、FP、FN



数据流	Threshold	Precision	Recall	TP	FP	TN	FN
./pr-test/run2	0.80	0.5029	0.2091	1047	1035	3957	3961
./pr-test/run1	0.80	0.5500	0.2136	110	90	395	405

- 可搜索卡片标签，展示目标图表



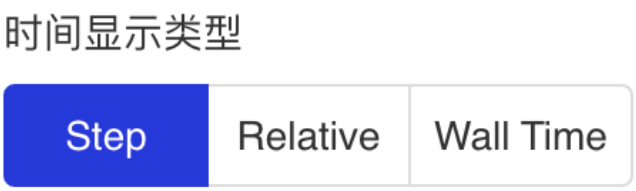
- 可搜索打点数据标签，展示特定数据



- 支持查看不同训练步数下的PR曲线



- X轴-时间显示类型有三种衡量尺度
  - Step：迭代次数
  - Walltime：训练绝对时间
  - Relative：训练时长



High Dimensional--数据降维组件

介绍

High Dimensional 组件将高维数据进行降维展示，用于深入分析高维数据间的关系。目前支持以下两种降维算法：

- PCA：Principle Component Analysis 主成分分析
- t-SNE：t-distributed stochastic neighbor embedding t-分布式随机领域嵌入

记录接口

High Dimensional 组件的记录接口如下：

`add_embeddings(tag, labels, hot_vectors, walltime=None)`

接口参数说明如下：

参数	格式	含义
tag	string	记录指标的标志，如`default`，不能含有`%`
labels	numpy.ndarray or list	一维数组表示的标签，每个元素是一个string类型的字符串
hot_vectors	numpy.ndarray or list	与labels一一对应，每个元素可以看作是某个标签的特征
walltime	int	记录数据的时间戳，默认为当前时间戳

Demo

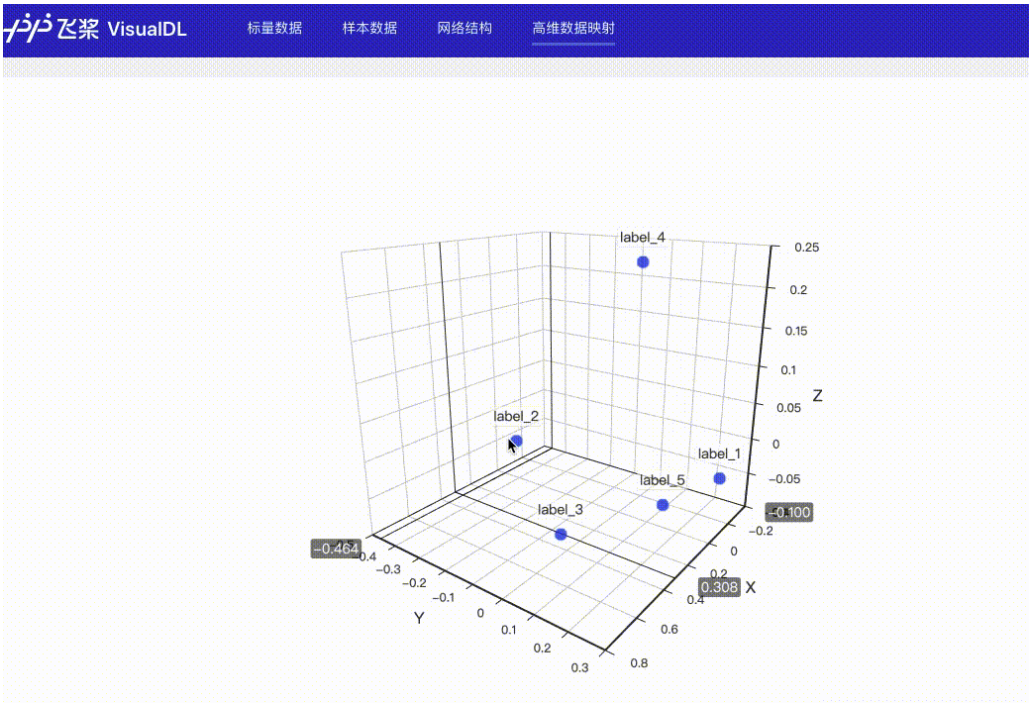
下面展示了使用 High Dimensional 组件记录数据的示例。

```
from visualdl import LogWriter


if __name__ == '__main__':
    hot_vectors = [
        [1.3561076367500755, 1.3116267195134017, 1.6785401875616097],
        [1.1039614644440658, 1.8891609992484688, 1.32030488587171],
        [1.9924524852447711, 1.9358920727142739, 1.2124401279391606],
        [1.4129542689796446, 1.7372166387197474, 1.7317806077076527],
        [1.3913371800587777, 1.4684674577930312, 1.5214136352476377]]

    labels = ["label_1", "label_2", "label_3", "label_4", "label_5"]
    # 初始化一个记录器
    with LogWriter(logdir="./log/high_dimensional_test/train") as writer:
        # 将一组labels和对应的hot_vectors传入记录器进行记录
        writer.add_embeddings(tag='default',
                              labels=labels,
                              hot_vectors=hot_vectors)
```

运行上述程序后，点击可视化选择相应日志文件即可查看可视化结果。

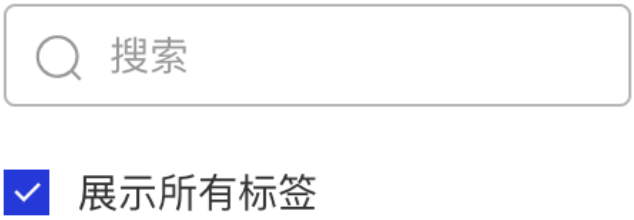


功能操作说明

- 支持展示特定打点数据



- 可搜索展示特定数据标签或展示所有数据标签



- 支持「二维」或「三维」展示高维数据分布



- 可选择「PCA」或「T-SNE」作为降维方式

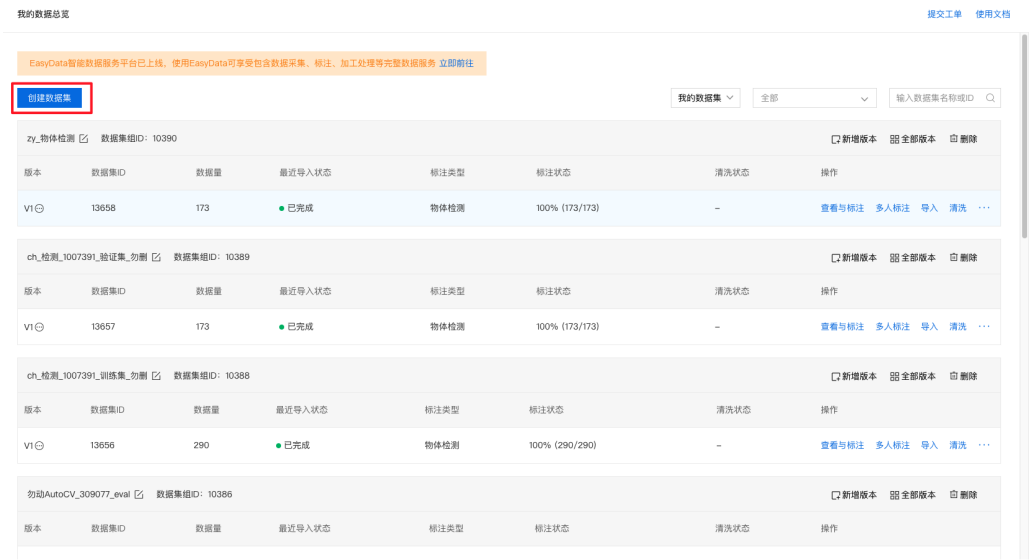


Notebook导入数据集

BML基于Jupyter提供了在线的交互式开发环境，在此基础上也为用户提供了便捷导入数据集的方式。

通过平台导入

先点击数据总览，并点击创建数据集，将数据集上传到 BML 平台进行纳管。



进入Codelab Notebook，点击导入数据集插件，并选择从平台数据集的导入方式，选择已经在平台纳管的数据集进行导入。



通过本地文件导入

进入Codelab Notebook，点击导入数据集插件，并选择从本地文件的导入方式，选择本地存储的数据集zip压缩包进行导入，单次上传不超过5GB。



通过BOS目录导入

进入Codelab Notebook，点击导入数据集插件，并选择从BOS目录的导入方式，选择BOS目录下保存的数据集zip压缩包进行导入。



通过Jupyter原生上传文件

进入Codelab Notebook，点击上传文件按钮，即可选择本地存储的文件进行上传。这种方式仅可上传150MB以下大小的文件。



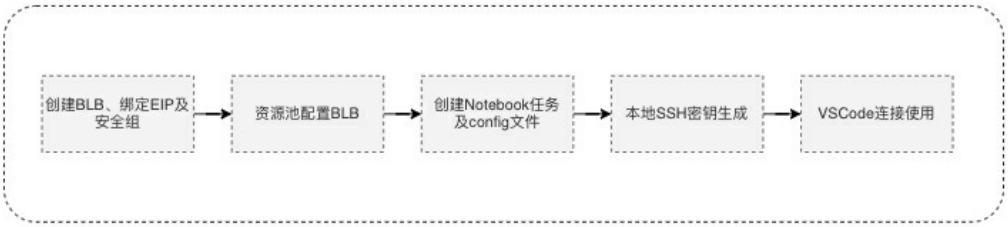
如何使用Notebook SSH 功能

什么是SSH功能

如果您想要使用本地的开发环境，把任务提交到线上的资源运行。可以使用SSH连接的功能。

如何使用SSH功能

使用流程：



创建BLB、EIP 如果您需要使用Notebook SSH功能，您需要创建EIP。EIP为您提供公网带宽服务。创建方法见：https://cloud.baidu.com/doc/EIP/s/rjx1m4njs

另外您需要创建应用型BLB实例，并保证BLB所在VPC和用户资源池在同一个VPC下。 创建方法见：https://cloud.baidu.com/doc/BLB/s/Njwvxnt8e

并需要在BLB创建时，将BLB和EIP绑定。



此外，出于安全考虑，建议您创建企业安全组，并和BLB进行绑定。



资源池配置 创建资源池-配置BLB

SSH功能需要在用户资源池中使用。您可以在资源池管理中创建资源池。“选择可用模块”时，如果选择“Notebook”可以发现，需要添加企业版BLB才可以使用SSH功能。



修改资源池-配置BLB

如果创建资源池时，未配置BLB，并需要使用SSH功能，您可以修改资源池可用模块，并选择绑定了EIP的应用型BLB。



如果不再使用SSH，可选择“暂不选择”，即可解除绑定/卸载BLB。需要注意的是，如果解绑前存在Notebook任务使用SSH功能，取消BLB可能会导致任务异常。



创建Notebook

如果需要使用SSH功能，需要在创建Notebook任务时，选择用户资源池后，并开启SSH开关。



如果资源池中未配置BLB信息，不能使用SSH相关功能。需要现在资源池管理中配置SSH连接的BLB实例。



如果BLB绑定异常，需要在资源池中重新绑定BLB实例（如果您卸载该BLB相关SVC会强制解绑EIP，因此，需要手动从BLB控制台关联EIP，关联后，重新绑定BLB实例）。



创建后，可以在Notebook详情中查看SSH命令。



进入Notebook 点击Notebook列表页的“打开”按钮，可点击进入到Notebook中。



进入到Notebook后，可点击进入终端。点击输入如下命令：

```
 mkdir ~/.ssh
ssh-keygen -t dsa -f ~/.ssh/ssh_host_dsa_key -N ''
ssh-keygen -t rsa -f ~/.ssh/ssh_host_rsa_key -N ''
```

接下来需要在目录创建sshd_config文件，内容为：

```
 Port 8022 #设置需要监听的端口

ListenAddress 0.0.0.0     #设置在哪个地址上监听，通常多网卡时须设置

HostKey /home/work/.ssh/ssh_host_dsa_key      #设置 HostKey
HostKey /home/work/.ssh/ssh_host_rsa_key

UsePAM yes      #是否需要使用 Linux-PAM

KexAlgorithms=+diffie-hellman-group1-sha1       # 如需兼容低版本
```

即可创建sshd_config文件，可在左侧文件插件中看到。

终端输入如下命令启动sshd服务：

```
 /usr/sbin/sshd -f ./sshd_config
```

如果需要重新配置，需要kill之前的进程，重新启动。

本地环境密钥生成

本地生成密钥，放入Notebook中。

本地环境生成ssh pubkey：

mac/linux：

```
 ssh-keygen -t rsa
##### 公钥文件在 ~/.ssh/id_rsa.pub
```

win：

```
ssh-keygen -t rsa
##### 公钥文件在 C:\Users\username\.ssh\id_rsa.pub
```

将~/.ssh/id_rsa.pub中的公钥放入Notebook用户目录中.ssh/authorized_keys的文本文件中。

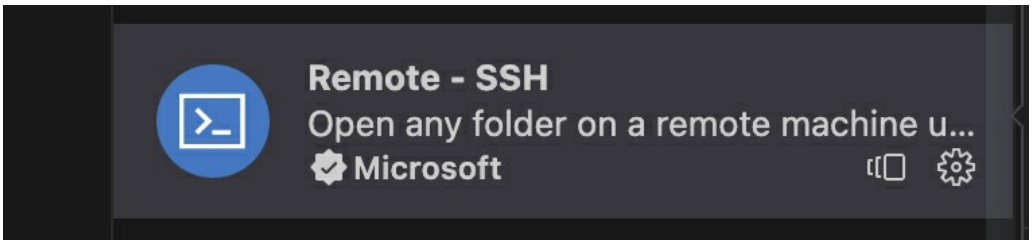
如果多用户访问同一个Notebook，需要将密钥按行放入即可。

VSCode连接使用 下载安装VSCode，建议使用vscode 1.85.1及以上。

下载地址: https://code.visualstudio.com/updates/v1_85

安装Remote-SSH插件，并配置Notebook详情中复制的SSH命令。

“扩展”中搜索Remote-SSH插件，并安装。



安装后，配置Notebook中复制的SSH连接命令即可连接。

Remote-SSH插件使用参考：https://code.visualstudio.com/docs/remote/ssh

终端使用 终端使用 ssh work@XXX.XXX.XXX.XXX -p XXXXX



或者指定密钥路径使用

ssh -i ssh_host_dsa_key work@XXX.XXX.XXX.XXX -p XXXXX

自定义作业建模

自定义作业简介

自定义作业简介

自定义作业提供高性能的计算环境，用于进行大规模分布式的模型训练及优化。您可以使用不同框架，编写代码进行多轮训练和迭代，将结果输出到BOS对象存储，或者发布模型到模型仓库。

训练方式简介 自定义作业预置的训练环境支持机器学习和深度学习常用的python库。您可以将代码文件和数据集存储在您的BOS对象存储中，并且在编辑自定义作业时从BOS选取这些文件进行训练。训练好的模型会输出到您选取的BOS路径中，并且支持发布到模型仓库。

算法框架简介 自定义作业支持多种主流的机器学习和深度学习框架，满足不同训练场景下的使用需求：

算法框架	版本
PaddlePaddle	2.0.0rc
TensorFlow	1.13.2
TensorFlow	2.3.0
Pytorch	1.7.1
Sklearn	0.23.2
XGBoost	1.6.2
Blackhole	1.0.0

训练作业

创建训练作业

创建训练作业 前提条件

1. 自定义作业需要依赖于BOS对象存储读取输入文件，创建自定义作业之前需要保证您已经开通了BOS对象存储的服务。

2. 授权自定义作业读写您的BOS对象存储，以顺利进行自定义作业的配置。

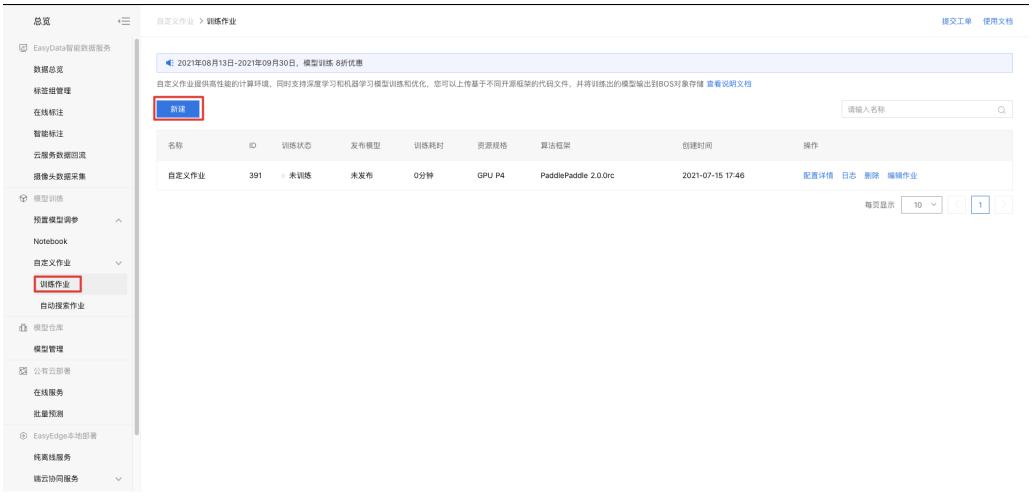


4. 在BOS中存储创建Bucket，并且存储用于训练的代码文件和数据集，创建一个空文件夹用于输出文件的存储

新建作业

1. 在导航栏选择『自定义作业-训练作业』，进入训练作业的列表页。

2. 点击『新建作业』，进入配置自定义作业流程



使用训练作业训练模型

使用训练作业训练模型 训练作业训练提供多种开源框架和优质的训练资源。您可以上传代码文件，数据集到BOS对象存储，通过训练作业完成训练后，训练结果会输出到BOS中的指定位置。 **基本信息** 填写作业名称和备注信息。作业名称填写完成后，训练作业才可以保存。



算法配置

1. 选择算法框架：选择训练代码文件使用的算法框架，目前BML支持Paddle，TensorFlow，Pytorch，Sklearn，XGBoost，Blackhole六种主流算法框架

2. 选择代码文件：从BOS对象存储中选取代码文件，完成代码录入。单击显示框任何位置，都可以打开选择代码文件弹窗，从弹窗中选择bucke及文件夹。双击bucket或者单击『>』图标，即可进入下一级



3. 代码文件要求如下：
    - 选取对象可以是一个文件或者文件夹
    - 所选代码文件必须和所选算法框架对应
4. 输入启动命令，支持python和shell两类脚本：
    - 当代码文件项选择一个单独文件时，启动命令指向该启动文件
    - 当代码文件项选择一个文件夹时，启动命令需指向该文件夹下的特定启动脚本，例如bash bml_job.sh
5. 选择输出路径：选择输出路径存放训练产生的模型和日志文件。为避免出错，建议您选择空文件夹作为输出路径。日志文件需要打印到标准输出，才能保证系统中『日志』功能能够正常读取。
6. 可选高级配置。提交训练作业时，可以通过高级配置来自定义环境变量。
    - 手动配置：通过手动填写环境变量来配置，包括配置变量名称与对应取值。



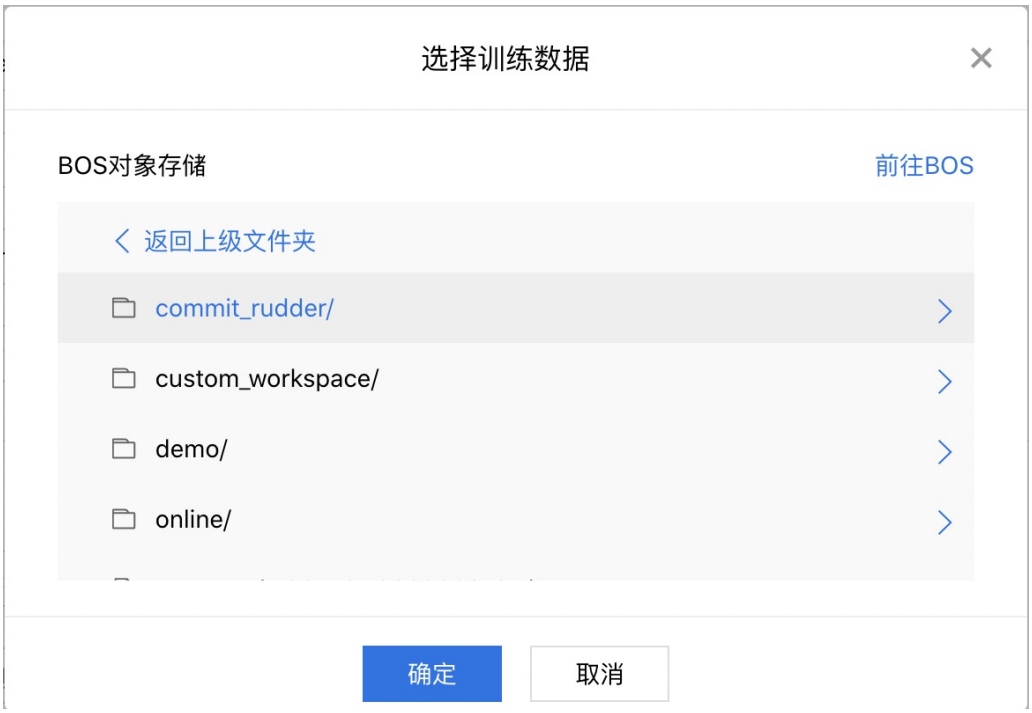
    - YAML文件：通过上传YAML文件来配置环境变量，支持本地上传和BOS存储导入两种方式。注意：环境变量最多配置100组，平台将在解析时对超出部分做自动截断处理。

      此外，BML平台亦提供预置环境变量，由BML平台预先定义，不建议您自行覆盖修改。

预置环境变量

作业类型	变量名	变量含义	示例	默认值
Pytorch 作业	PYTHONUNBUFFERED	控制Python的标准输出和标准错误输出是否被缓冲。'1'代表输出将不会被缓冲	0	0
	PET_RDZV_ENDPOINT	分布式任务同步服务的地址	job-qcu5m5-1984-master-0:23456	MASTER_ADDR:MASTER_PORT
	PET_NNODES	{实例数}:{实例数} ，代表训练任务的节点个数	2:2	无
	PET_MAX_RESTARTS	训练进程的最大重启次数	100	0
	PET_NPROC_PER_NODE	当前节点创建的进程数	10	无
	MASTER_PORT	分布式任务主节点的端口	23456	23456
	MASTER_ADDR	分布式任务主节点的地址	job-qcu5m5-1984-master-0	主节点名称
	NVIDIA_VISIBLE_DEVICES	训练节点可见的gpu卡设备	GPU-ae58e6cc-1dec-bcb9-820c-e433d01afda6	无
PaddlePaddle 作业	POD_IP	当前节点的IP	172.16.5.9	无
	PADDLE_TRAINER_ID	当前节点的ID，从0开始编号，取值为 0,1,...,(PADDLE_TRAINERS_NUM-1)	0	无
	TRAINING_ROLE	当前节点的角色	PSERVER或TRAINER	无
	PADDLE_TRAINING_ROLE	当前节点的角色	PSERVER或TRAINER	无
	PADDLE_TRAINERS_NUM	分布式训练节点的数量	2	无
	PADDLE_TRAINERS	所有分布式训练节点的ip，使用逗号 ',' 分隔	172.16.5.9,172.16.5.1	无
	PADDLE_TRAINER_ENDPOINTS	所有分布式训练节点的ip、端口，取值为【node1-ip:端口,node2-ip:端口】	172.16.5.9:43817,172.16.5.1:45492	无
	NVIDIA_VISIBLE_DEVICES	训练节点可见的gpu卡设备	GPU-ae58e6cc-1dec-bcb9-820c-e433d01afda6	无
Horovod 作业	K_MPI_JOB_ROLE	当前节点的角色，取值为worker或者launcher	worker或launcher	无
	OMPI_MCA_orte_keep_fqdn_hostnames	是否保留FQDN hostnames（全限定域名）	true	true
	OMPI_MCA_orte_default_hostfile	mpirun命令指定hostfile，在平台中可以自动生成hostfile，无需手动修改。	/etc/mpi/hostfile	/etc/mpi/hostfile
	OMPI_MCA_plm_rsh_args	训练节点之间ssh连接使用的参数配置	"-o ConnectionAttempts=10"	"-o ConnectionAttempts=10"
	NVIDIA_VISIBLE_DEVICES	训练节点可见的gpu卡设备	GPU-ae58e6cc-1dec-bcb9-820c-e433d01afda6	无
	OMPI_MCA_plm_rsh_agent	指定远程代理命令，用于launcher节点给worker节点发送命令。在平台中可以自动生成，无需手动修改。	/etc/mpi/kubexec.sh	/etc/mpi/kubexec.sh

数据集配置 训练作业数据集配置方式是从BOS中选取数据集。在弹出的对话框中选择数据集对应的bucket和文件夹。



资源配置

BML提供CPU和GPU两类训练机型。
CPU机型供算法框架为sklearn，XGBoost，Blackhole时使用，用于机器学习训练：

机型	规格说明
CPU 4核	CPU 4核16GB内存
CPU 16核	CPU 16核64GB内存

GPU机型供算法框架为Paddle，TensorFlow，Pytorch时使用，用于深度学习训练：

机型	规格说明
GPU V100	TeslaGPU_V100_32G显存单卡_12核CPU_56G内存
GPU P4	TeslaGPU_P4_8G显存单卡_12核CPU_40G内存

温馨提示： 未开通付费的情况下，可选的运行环境为CPU 4核，GPU P4，我们为每位用户提供了CPU 4核环境下100（小时×节点），GPU P4运行环境下100（小时×节点）的免费算力支持，超出后请您付费购买。详见页面提示的价格说明。

查看作业结果 作业运行完成后，会将训练结果与运行日志存储到BOS中指定的输出路径，您可以前往BOS查看或下载作业运行结果。日志文件打印到标准输出时可以直接在日志界面查看。
两种情况下，会导致作业结果及日志无法保存：1. 手动终止作业；2. 作业运行超时被自动终止

发布模型

发布模型 训练作业训练完成后支持直接发布模型到模型仓库。

1. 模型列表中，训练完成模型支持『发布』操作，点击『发布』进入发布界面



2. 在发布界面，完成相关配置，如下所示：

2.1 填写下图所示基本信息。『模型类型』和『版本』由系统自动生成，不支持修改。重新发布时，『模型名称和『模型描述』会自动填充，无需再次填写。



2.2 根据算法框架的不同，您还需要在发布界面填写响应的额外配置项信息

算法框架	额外配置项
Paddle	无
TensorFlow	模型文件格式：自定义作业支发布pb格式的模型文件到模型仓库
Pytorch	从BOS中选择源代码并输入源代码的主文件名 **说明：**选取的代码文件必须是当前自定义作业任务对应的代码文件
Sklearn/XGBoost	模型文件格式：支持发布pickle格式或joblib格式模型到模型仓库 模型文件：从BOS选择当前自定义作业训练输出的模型文件 **说明：**选取的模型文件必须是当前自定义作业任务对应的模型文件

3.发布成功后，关联模型的名称与版本会显示在训练作业的任务列表中，点击即可进入模型仓库查看模型详情。



训练作业代码示例

TensorFlow 1.13.2

TensorFlow 基于tensorflow框架的MNIST图像分类任务示例代码，训练数据集点击这里下载

单机训练（计算节点数为1），示例代码如下：

```
 import os
import tensorflow as tf
import numpy as np
from tensorflow import keras
layers = tf.layers
tf.logging.set_verbosity(tf.logging.INFO)
def conv_model(feature, target, mode):
 """2-layer convolution model."""
 # Convert the target to a one-hot tensor of shape (batch_size, 10) and
 # with a on-value of 1 for each one-hot vector of length 10.
 target = tf.one_hot(tf.cast(target, tf.int32), 10, 1, 0)
 # Reshape feature to 4d tensor with 2nd and 3rd dimensions being
 # image width and height final dimension being the number of color channels.
 feature = tf.reshape(feature, [-1, 28, 28, 1])
 # First conv layer will compute 32 features for each 5x5 patch
 with tf.variable_scope('conv_layer1'):
```

```
    h_conv1 = layers.conv2d(feature, 32, kernel_size=[5, 5],
                            activation=tf.nn.relu, padding="SAME")
    h_pool1 = tf.nn.max_pool(
        h_conv1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
  # Second conv layer will compute 64 features for each 5x5 patch.
  with tf.variable_scope('conv_layer2'):
    h_conv2 = layers.conv2d(h_pool1, 64, kernel_size=[5, 5],
                            activation=tf.nn.relu, padding="SAME")
    h_pool2 = tf.nn.max_pool(
        h_conv2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
    # reshape tensor into a batch of vectors
    h_pool2_flat = tf.reshape(h_pool2, [-1, 7 * 7 * 64])
  # Densely connected layer with 1024 neurons.
  h_fc1 = layers.dropout(
    layers.dense(h_pool2_flat, 1024, activation=tf.nn.relu),
    rate=0.5, training=mode == tf.estimator.ModeKeys.TRAIN)
  # Compute logits (1 per class) and compute loss.
  logits = layers.dense(h_fc1, 10, activation=None)
  loss = tf.losses.softmax_cross_entropy(target, logits)
  return tf.argmax(logits, 1), loss
def train_input_generator(x_train, y_train, batch_size=64):
  assert len(x_train) == len(y_train)
  while True:
    p = np.random.permutation(len(x_train))
    x_train, y_train = x_train[p], y_train[p]
    index = 0
    while index <= len(x_train) - batch_size:
      yield x_train[index:index + batch_size], \
          y_train[index:index + batch_size],
      index += batch_size
def main(_):
  work_path = os.getcwd()
  # Download and load MNIST dataset.
  (x_train, y_train), (x_test, y_test) = \
    keras.datasets.mnist.load_data('%s/train_data/mnist.npz' % work_path)
  # The shape of downloaded data is (-1, 28, 28), hence we need to reshape it
  # into (-1, 784) to feed into our network. Also, need to normalize the
  # features between 0 and 1.
  x_train = np.reshape(x_train, (-1, 784)) / 255.0
  x_test = np.reshape(x_test, (-1, 784)) / 255.0
  # Build model...
  with tf.name_scope('input'):
    image = tf.placeholder(tf.float32, [None, 784], name='image')
    label = tf.placeholder(tf.float32, [None], name='label')
  predict, loss = conv_model(image, label, tf.estimator.ModeKeys.TRAIN)
  opt = tf.train.RMSPropOptimizer(0.001)
  global_step = tf.train.get_or_create_global_step()
  train_op = opt.minimize(loss, global_step=global_step)
  hooks = [
    tf.train.StopAtStepHook(last_step=20000),
    tf.train.LoggingTensorHook(tensors={'step': global_step, 'loss': loss},
                               every_n_iter=10),
  ]
  # Horovod: pin GPU to be used to process local rank (one GPU per process)
  config = tf.ConfigProto()
  config.gpu_options.allow_growth = True
  config.gpu_options.visible_device_list = '0'
  # Horovod: save checkpoints only on worker 0 to prevent other workers from
  # corrupting them.
  checkpoint_dir = './checkpoints'
  training_batch_generator = train_input_generator(x_train,
                                                   y_train, batch_size=100)
  # The MonitoredTrainingSession takes care of session initialization,
  # restoring from a checkpoint, saving to a checkpoint, and closing when done
  # or an error occurs.
  with tf.train.MonitoredTrainingSession(checkpoint_dir=checkpoint_dir,
                                         hooks=hooks,
                                         config=config) as mon_sess:
    while not mon_sess.should_stop():
      # Run a training step synchronously.
      image, label = next(training_batch_generator)
```

```
        image_, label_ = next(training_batch_generator)
        mon_sess.run(train_op, feed_dict={image: image_, label: label_})
  checkpoint_file = tf.train.latest_checkpoint(checkpoint_dir)
  saver = tf.train.Saver()
  inputs_classes = tf.saved_model.utils.build_tensor_info(image)
  outputs_classes = tf.saved_model.utils.build_tensor_info(predict)
  signature = (tf.saved_model.signature_def_utils.build_signature_def(
      inputs={tf.saved_model.signature_constants.CLASSIFY_INPUTS: inputs_classes},
      outputs={tf.saved_model.signature_constants.CLASSIFY_OUTPUT_CLASSES: outputs_classes},
      method_name=tf.saved_model.signature_constants.CLASSIFY_METHOD_NAME))
  os.system("rm -rf ./output")
  with tf.Session() as sess:
    sess.run([tf.local_variables_initializer(), tf.tables_initializer()])
    saver.restore(sess, checkpoint_file)
    builder = tf.saved_model.builder.SavedModelBuilder('./output')
    legacy_init_op = tf.group(tf.tables_initializer(), name='legacy_init_op')
    builder.add_meta_graph_and_variables(sess, [tf.saved_model.tag_constants.SERVING], signature_def_map={'predict_images': signature},
legacy_init_op=legacy_init_op)
    builder.save()
if __name__ == "__main__":
  tf.app.run()
```

分布式训练（计算节点数大于1），示例代码如下：

说明：demo分布式程序没有做数据的分片操作，仅供参考

```
  import os
import tensorflow as tf
import horovod.tensorflow as hvd
import numpy as np
from tensorflow import keras
layers = tf.layers
tf.logging.set_verbosity(tf.logging.INFO)
def conv_model(feature, target, mode):
    """2-layer convolution model."""
    # Convert the target to a one-hot tensor of shape (batch_size, 10) and
    # with a on-value of 1 for each one-hot vector of length 10.
    target = tf.one_hot(tf.cast(target, tf.int32), 10, 1, 0)
    # Reshape feature to 4d tensor with 2nd and 3rd dimensions being
    # image width and height final dimension being the number of color channels.
    feature = tf.reshape(feature, [-1, 28, 28, 1])
    # First conv layer will compute 32 features for each 5x5 patch
    with tf.variable_scope('conv_layer1'):
        h_conv1 = layers.conv2d(feature, 32, kernel_size=[5, 5],
                                activation=tf.nn.relu, padding="SAME")
        h_pool1 = tf.nn.max_pool(
            h_conv1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
    # Second conv layer will compute 64 features for each 5x5 patch.
    with tf.variable_scope('conv_layer2'):
        h_conv2 = layers.conv2d(h_pool1, 64, kernel_size=[5, 5],
                                activation=tf.nn.relu, padding="SAME")
        h_pool2 = tf.nn.max_pool(
            h_conv2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
        # reshape tensor into a batch of vectors
        h_pool2_flat = tf.reshape(h_pool2, [-1, 7 * 7 * 64])
    # Densely connected layer with 1024 neurons.
    h_fc1 = layers.dropout(
        layers.dense(h_pool2_flat, 1024, activation=tf.nn.relu),
        rate=0.5, training=mode == tf.estimator.ModeKeys.TRAIN)
    # Compute logits (1 per class) and compute loss.
    logits = layers.dense(h_fc1, 10, activation=None)
    loss = tf.losses.softmax_cross_entropy(target, logits)
    return tf.argmax(logits, 1), loss
def train_input_generator(x_train, y_train, batch_size=64):
    assert len(x_train) == len(y_train)
    while True:
        p = np.random.permutation(len(x_train))
        x_train, y_train = x_train[p], y_train[p]
        index = 0
        while index <= len(x_train) - batch_size:
```

```
        yield x_train[index:index + batch_size], \
              y_train[index:index + batch_size],
        index += batch_size
def main(_):
    # Horovod: initialize Horovod.
    hvd.init()
    work_path = os.getcwd()
    # Download and load MNIST dataset.
    (x_train, y_train), (x_test, y_test) = \
        keras.datasets.mnist.load_data('%s/train_data/mnist.npz' % work_path)
    # The shape of downloaded data is (-1, 28, 28), hence we need to reshape it
    # into (-1, 784) to feed into our network. Also, need to normalize the
    # features between 0 and 1.
    x_train = np.reshape(x_train, (-1, 784)) / 255.0
    x_test = np.reshape(x_test, (-1, 784)) / 255.0
    # Build model...
    with tf.name_scope('input'):
        image = tf.placeholder(tf.float32, [None, 784], name='image')
        label = tf.placeholder(tf.float32, [None], name='label')
    predict, loss = conv_model(image, label, tf.estimator.ModeKeys.TRAIN)
    serve_graph_file = "./serve_graph.meta"
    tf.train.export_meta_graph(serve_graph_file, as_text=True)
    # Horovod: adjust learning rate based on number of GPUs.
    opt = tf.train.RMSPropOptimizer(0.001 * hvd.size())
    # Horovod: add Horovod Distributed Optimizer.
    opt = hvd.DistributedOptimizer(opt)
    global_step = tf.train.get_or_create_global_step()
    train_op = opt.minimize(loss, global_step=global_step)
    hooks = [
        # Horovod: BroadcastGlobalVariablesHook broadcasts initial variable states
        # from rank 0 to all other processes. This is necessary to ensure consistent
        # initialization of all workers when training is started with random weights
        # or restored from a checkpoint.
        hvd.BroadcastGlobalVariablesHook(0),
        # Horovod: adjust number of steps based on number of GPUs.
        tf.train.StopAtStepHook(last_step=10000 // hvd.size()),
        tf.train.LoggingTensorHook(tensors={'step': global_step, 'loss': loss},
                                   every_n_iter=10),
    ]
    # Horovod: pin GPU to be used to process local rank (one GPU per process)
    config = tf.ConfigProto()
    config.gpu_options.allow_growth = True
    config.gpu_options.visible_device_list = str(hvd.local_rank())
    # Horovod: save checkpoints only on worker 0 to prevent other workers from
    # corrupting them.
    checkpoint_dir = './checkpoints' if hvd.rank() == 0 else None
    training_batch_generator = train_input_generator(x_train,
                                                     y_train, batch_size=100)
    # The MonitoredTrainingSession takes care of session initialization,
    # restoring from a checkpoint, saving to a checkpoint, and closing when done
    # or an error occurs.
    with tf.train.MonitoredTrainingSession(checkpoint_dir=checkpoint_dir,
                                           hooks=hooks,
                                           config=config) as mon_sess:
        while not mon_sess.should_stop():
            # Run a training step synchronously.
            image_, label_ = next(training_batch_generator)
            mon_sess.run(train_op, feed_dict={image: image_, label: label_})
    if hvd.rank() != 0:
        return
    checkpoint_file = tf.train.latest_checkpoint(checkpoint_dir)
    tf.reset_default_graph()
    saver = tf.train.import_meta_graph(serve_graph_file)
    inputs_classes = tf.saved_model.utils.build_tensor_info(image)
    outputs_classes = tf.saved_model.utils.build_tensor_info(predict)
    signature = (tf.saved_model.signature_def_utils.build_signature_def(
        inputs={tf.saved_model.signature_constants.CLASSIFY_INPUTS: inputs_classes},
        outputs={tf.saved_model.signature_constants.CLASSIFY_OUTPUT_CLASSES: outputs_classes},
        method_name=tf.saved_model.signature_constants.CLASSIFY_METHOD_NAME))
    os.system("rm -rf ./output")
```

```
    with tf.Session() as sess:
        sess.run([tf.local_variables_initializer(), tf.tables_initializer()])
        saver.restore(sess, checkpoint_file)
        builder = tf.saved_model.builder.SavedModelBuilder('./output')
        legacy_init_op = tf.group(tf.tables_initializer(), name='legacy_init_op')
        builder.add_meta_graph_and_variables(sess, [tf.saved_model.tag_constants.SERVING], signature_def_map={'predict_images':
signature}, legacy_init_op=legacy_init_op)
        builder.save()
if __name__ == "__main__":
    tf.app.run()

##### TensorFlow 2.3.0

**TensorFlow**
 基于tensorflow框架的MNIST图像分类任务示例代码，训练数据集点击[这
里](https://doc.bce.baidu.com/bce-documentation/BML/tf_train_data2.zip)下载<br>
单机训练（计算节点数为1），示例代码如下：
  ```python
"""
tf train demo
"""
import tensorflow as tf
import os
mnist = tf.keras.datasets.mnist
work_path = os.getcwd()
(x_train, y_train), (x_test, y_test) = mnist.load_data('%s/train_data/mnist.npz' % work_path)
x_train, x_test = x_train / 255.0, x_test / 255.0
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test, verbose=2)
model.save('./output/')
```

分布式训练（计算节点数大于1），示例代码如下：

说明：demo分布式程序没有做数据的分片操作，仅供参考

```
  """
tf horovod train demo
"""
import tensorflow as tf
import horovod.tensorflow as hvd
import os
**Horovod: initialize Horovod.**
hvd.init()
**Horovod: pin GPU to be used to process local rank (one GPU per process)**
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)
if gpus:
    tf.config.experimental.set_visible_devices(gpus[hvd.local_rank()], 'GPU')
work_path = os.getcwd()
(mnist_images, mnist_labels), _ = \
    tf.keras.datasets.mnist.load_data('%s/train_data/mnist.npz' % work_path)
dataset = tf.data.Dataset.from_tensor_slices(
    (tf.cast(mnist_images[..., tf.newaxis] / 255.0, tf.float32),
            tf.cast(mnist_labels, tf.int64))
)
dataset = dataset.repeat().shuffle(10000).batch(128)
mnist_model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, [3, 3], activation='relu'),
    tf.keras.layers.Conv2D(64, [3, 3], activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
```
```

```
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(10, activation='softmax')
])
loss = tf.losses.SparseCategoricalCrossentropy()
**Horovod: adjust learning rate based on number of GPUs.**
opt = tf.optimizers.Adam(0.001 * hvd.size())
@tf.function
def training_step(images, labels, first_batch):
    """
    :param images:
    :param labels:
    :param first_batch:
    :return:
    """
    with tf.GradientTape() as tape:
        probs = mnist_model(images, training=True)
        loss_value = loss(labels, probs)
    # Horovod: add Horovod Distributed GradientTape.
    tape = hvd.DistributedGradientTape(tape)
    grads = tape.gradient(loss_value, mnist_model.trainable_variables)
    opt.apply_gradients(zip(grads, mnist_model.trainable_variables))
    # Horovod: broadcast initial variable states from rank 0 to all other processes.
    # This is necessary to ensure consistent initialization of all workers when
    # training is started with random weights or restored from a checkpoint.
    #
    # Note: broadcast should be done after the first gradient step to ensure optimizer
    # initialization.
    if first_batch:
        hvd.broadcast_variables(mnist_model.variables, root_rank=0)
        hvd.broadcast_variables(opt.variables(), root_rank=0)
    return loss_value
**Horovod: adjust number of steps based on number of GPUs.**
for batch, (images, labels) in enumerate(dataset.take(10000 // hvd.size())):
    loss_value = training_step(images, labels, batch == 0)
    if batch % 10 == 0 and hvd.local_rank() == 0:
        print('Step #%d\tLoss: %.6f' % (batch, loss_value))
**Horovod: save model only on worker 0 to prevent other workers from**
**corrupting it.**
if hvd.rank() == 0:
    mnist_model.save('./output/')

##### Blackhole 1.0.0

**Blackhole 1.0.0**
Blackhole是百度自研的高性能数据科学引擎，CodeLab中内嵌了该引擎。通过异构加速计算、超大数据处理、高效数据存储等技术，单机Blackhole在数据分析和机器学习等场景相比开源Pandas/Sklearn性能可提升7倍以上、拥有TB级的单机超大数据处理能力，同时提供和Pandas、Sklearn基本一致的易用接口。参考文档点击[这里](https://cloud.baidu.com/doc/BML/s/2l2svt3we)查看。<br>
本文使用Blackhole中随机森林算法对希格斯玻色子的信号进行预测，并采用准确率评估指标对模型性能进行评估。参考kaggle竞赛-希格斯玻色子机器学习挑战，HIGGS数据集由加利福尼亚大学机器学习与智能系统中心提供，用于预测希格斯玻色子的信号。
训练数据集点击[这里](https://codelab-dataset.cdn.bcebos.com/small/competition/higgs.zip)下载。<br>
单机训练（计算节点数为1），示例代码如下：

```python
**Blackhole train demo**
**!/usr/bin/env python**
**-*- coding: utf-8 -*-**
"""
    Copyright (c) 2021 Baidu.com, Inc. All Rights Reserved
    File: demo.py
    Blackhole demo
    本demo背景参考kaggle竞赛-希格斯玻色子机器学习挑战，HIGGS数据集由加利福尼亚大学机器学习与智能系统中心提供，用于预测希格斯玻色子的信号。
    本demo使用blackhole中随机森林算法对希格斯玻色子的信号进行预测，并采用准确率评估指标对模型性能进行评估。
    数据集中第1列为标签列，其后28列为特征列.
    数据集地址： https://archive.ics.uci.edu/ml/datasets/HIGGS
            https://codelab-dataset.cdn.bcebos.com/small/competition/higgs.zip
    竞赛地址： https://www.kaggle.com/c/higgs-boson/overview
```
```

```
    竞赛地址：https://www.kaggle.com/c/higgs-boson/overview
"""
import os
import logging
import shutil
import blackhole
import blackhole.gibbons as pd
from blackhole.ml.metrics import accuracy_score
from blackhole.ml.model_selection import train_test_split
from blackhole.ml.ensemble import RandomForestClassifier
logging.basicConfig(format='%(asctime)s - %(pathname)s[line:%(lineno)d] - %(levelname)s: %(message)s',
                    level=logging.INFO)


def load_data(csv_file):
    """
    :param csv_file:
    :return:
    """
    assert os.path.exists(csv_file), "%s not exists" % csv_file
    logging.info("Load data from %s" % csv_file)
    col_names = ['label'] + ["col-{}".format(i) for i in range(2, 30)]  # Assign column names
    data = pd.read_csv(csv_file, names=col_names)
    return data


def split_data(data, test_ratio=0.3):
    """
    :param data: dataframe
    :param test_ratio: test data ratio
    :return:
    """
    logging.info("Split_data, train ratio: %s, split_ratio: %s" % (1.0 - test_ratio, test_ratio))
    X, y = data[data.columns.difference(['label'])], data['label']  # Separate data into X and y
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=int(len(data) * test_ratio))
    logging.info("After split, train_data: %s, test_data: %s" % (len(X_train), len(X_test)))
    train_test_data_dict = {
        "X_train": X_train,
        "X_test": X_test,
        "y_train": y_train,
        "y_test": y_test
    }
    return train_test_data_dict


def create_model(model, params):
    """
    :param model:
    :param params:
    :return:
    """
    logging.info("Create model from %s, params: %s" % (model, params))
    bh_model = model(**params)
    return bh_model


def fit(model_instance, X_train, y_train, X_test, y_test):
    """
    :param model_instance:
    :param X_train:
    :param y_train:
    :param X_test:
    :param y_test:
    :return:
    """
    logging.info("Fit model...")
    model_instance.fit(X_train, y_train)
    pre = model_instance.predict(X_test)
    accuracy = accuracy_score(pre, y_test)
    logging.info("Train accuracy: %s" % accuracy)
    return model_instance
```

```
    return model_instance


def save_model(model_instance, output_path):
    """
    :param model_instance:
    :param output_path:
    :return:
    """
    shutil.rmtree(output_path, ignore_errors=True)
    blackhole.ml.save_model(model_instance, output_path)
    logging.info("Save model to %s" % output_path)
    return output_path


def predict(model_path, X_test):
    """
    :param model_path:
    :param X_test: dataframe, should not contain label
    :return:
    """
    assert os.path.exists(model_path), "%s not exists" % model_path
    logging.info("Load model from %s, predict ..." % model_path)
    model = blackhole.ml.load_model(model_path)
    pred = model.predict(X_test)
    logging.info("Predict_result number: %s, show top 5: \n%s" % (len(pred), pred.head()))
    return pred


def main():
    """
    :return:
    File directory example:
    |-- demo.py
    |-- output
    |   |-- bhml.meta
    |   `-- bhml.model
    |-- test_data
    |   `-- HIGGS.csv
    `-- train_data
        `-- HIGGS.csv
    """
    # step1, load and split data
    train_csv_file = "./train_data/HIGGS.csv"  # csv file is in train_data folder
    train_data = load_data(train_csv_file)
    train_test_data_dict = split_data(train_data)
    X_train = train_test_data_dict['X_train']
    X_test = train_test_data_dict['X_test']
    y_train = train_test_data_dict['y_train']
    y_test = train_test_data_dict['y_test']

    # step 2, create model、fit and save
    model_params = {
        'n_estimators': 25,
        'max_depth': 13,
    }
    bh_RandomForestClassifier = create_model(RandomForestClassifier, model_params)
    bh_RandomForestClassifier = fit(bh_RandomForestClassifier, X_train, y_train, X_test, y_test)
    saved_model_path = "./output/"  # output path is in ./output
    save_model(bh_RandomForestClassifier, saved_model_path)

    # step 3, predict
    test_csv_file = "./test_data/HIGGS.csv"  # csv file is in train_data folder
    test_data = load_data(test_csv_file)
    y_predict = predict(saved_model_path, test_data)


if __name__ == "__main__":
    main()
```

Pytorch 1.7.1

Pytorch

训练代码

基于Pytorch框架的MNIST图像分类示例代码，数据集请点击这里下载。

单机训练时（计算节点等于1），示例代码如下：

```
 import argparse
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torch.utils.data as data
from torchvision import transforms
import codecs
import errno
import gzip
import numpy as np
import os
from PIL import Image
**Training settings**
parser = argparse.ArgumentParser(description='PyTorch MNIST Example')
parser.add_argument('--train-dir', type=str, default='./train_data',
                    help='input data dir for training (default: ./train_data)')
parser.add_argument('--test-dir', type=str, default='./test_data',
                    help='input data dir for test (default: ./test_data)')
parser.add_argument('--output-dir', type=str, default='./output',
                    help='output dir for custom job (default: ./output)')
parser.add_argument('--batch-size', type=int, default=64, metavar='N',
                    help='input batch size for training (default: 64)')
parser.add_argument('--test-batch-size', type=int, default=64, metavar='N',
                    help='input batch size for testing (default: 64)')
parser.add_argument('--epochs', type=int, default=10, metavar='N',
                    help='number of epochs to train (default: 10)')
parser.add_argument('--lr', type=float, default=0.01, metavar='LR',
                    help='learning rate (default: 0.01)')
parser.add_argument('--momentum', type=float, default=0.5, metavar='M',
                    help='SGD momentum (default: 0.5)')
parser.add_argument('--no-cuda', action='store_true', default=False,
                    help='disables CUDA training')
parser.add_argument('--log-interval', type=int, default=10, metavar='N',
                    help='how many batches to wait before logging training status')
**定义MNIST数据集的dataset**
class MNIST(data.Dataset):
    """
    MNIST dataset
    """
    training_file = 'training.pt'
    test_file = 'test.pt'
    classes = ['0 - zero', '1 - one', '2 - two', '3 - three', '4 - four',
               '5 - five', '6 - six', '7 - seven', '8 - eight', '9 - nine']
    def __init__(self, root, train=True, transform=None, target_transform=None):
        self.root = os.path.expanduser(root)
        self.transform = transform
        self.target_transform = target_transform
        self.train = train  # training set or test set
        self.preprocess(root, train, False)
        if self.train:
            data_file = self.training_file
        else:
            data_file = self.test_file
        self.data, self.targets = torch.load(os.path.join(self.processed_folder, data_file))

    def __getitem__(self, index):
        """
        Args:
            index (int): Index
        Returns:
```

```
            tuple: (image, target) where target is index of the target class.
        """
        img, target = self.data[index], int(self.targets[index])
        # doing this so that it is consistent with all other datasets
        # to return a PIL Image
        img = Image.fromarray(img.numpy(), mode='L')
        if self.transform is not None:
            img = self.transform(img)
        if self.target_transform is not None:
            target = self.target_transform(target)

        return img, target
    def __len__(self):
        return len(self.data)

    @property
    def raw_folder(self):
        """
        raw folder
        """
        return os.path.join('/tmp', 'raw')

    @property
    def processed_folder(self):
        """
        processed folder
        """
        return os.path.join('/tmp', 'processed')

    # data preprocessing
    def preprocess(self, train_dir, train, remove_finished=False):
        """
        preprocess
        """
        makedir_exist_ok(self.raw_folder)
        makedir_exist_ok(self.processed_folder)
        train_list = ['train-images-idx3-ubyte.gz', 'train-labels-idx1-ubyte.gz']
        test_list = ['t10k-images-idx3-ubyte.gz', 't10k-labels-idx1-ubyte.gz']
        zip_list = train_list if train else test_list
        for zip_file in zip_list:
            print('Extracting {}'.format(zip_file))
            zip_file_path = os.path.join(train_dir, zip_file)
            raw_folder_path = os.path.join(self.raw_folder, zip_file)
            with open(raw_folder_path.replace('.gz', ''), 'wb') as out_f, gzip.GzipFile(zip_file_path) as zip_f:
                out_f.write(zip_f.read())
            if remove_finished:
                os.unlink(zip_file_path)
        if train:
            training_set = (
                read_image_file(os.path.join(self.raw_folder, 'train-images-idx3-ubyte')),
                read_label_file(os.path.join(self.raw_folder, 'train-labels-idx1-ubyte'))
            )
            with open(os.path.join(self.processed_folder, self.training_file), 'wb') as f:
                torch.save(training_set, f)
        else:
            test_set = (
                read_image_file(os.path.join(self.raw_folder, 't10k-images-idx3-ubyte')),
                read_label_file(os.path.join(self.raw_folder, 't10k-labels-idx1-ubyte'))
            )
            with open(os.path.join(self.processed_folder, self.test_file), 'wb') as f:
                torch.save(test_set, f)

def get_int(b):
    """
    get int
    """
    return int(codecs.encode(b, 'hex'), 16)
def read_label_file(path):
    """
    read label file
```

```
    """
    with open(path, 'rb') as f:
        data = f.read()
        assert get_int(data[:4]) == 2049
        length = get_int(data[4:8])
        parsed = np.frombuffer(data, dtype=np.uint8, offset=8)
        return torch.from_numpy(parsed).view(length).long()
def read_image_file(path):
    """
    read image file
    """
    with open(path, 'rb') as f:
        data = f.read()
        assert get_int(data[:4]) == 2051
        length = get_int(data[4:8])
        num_rows = get_int(data[8:12])
        num_cols = get_int(data[12:16])
        parsed = np.frombuffer(data, dtype=np.uint8, offset=16)
        return torch.from_numpy(parsed).view(length, num_rows, num_cols)
def makedir_exist_ok(dirpath):
    """
    Python2 support for os.makedirs(.., exist_ok=True)
    """
    try:
        os.makedirs(dirpath)
    except OSError as e:
        if e.errno == errno.EEXIST:
            pass
        else:
            raise
**定义网络模型**
class Net(nn.Module):
    """
    Net
    """
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 10, kernel_size=5)
        self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)
    def forward(self, x):
        """
        forward
        """
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
        x = x.view(-1, 320)
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        x = self.fc2(x)
        return F.log_softmax(x)
def train(epoch):
    """
    train
    """
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        if args.cuda:
            data, target = data.cuda(), target.cuda()
        optimizer.zero_grad()
        output = model(data) # 获取预测值
        loss = F.nll_loss(output, target) # 计算loss
        loss.backward()
        optimizer.step()
        if batch_idx % args.log_interval == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx, len(train_loader),
                100. * batch_idx / len(train_loader), loss.item()))
```

```
def test():
    """
    test
    """
    model.eval()
    test_loss = 0.
    test_accuracy = 0.
    for data, target in test_loader:
        if args.cuda:
            data, target = data.cuda(), target.cuda()
        output = model(data)
        # sum up batch loss
        test_loss += F.nll_loss(output, target, size_average=False).item()
        # get the index of the max log-probability
        pred = output.data.max(1, keepdim=True)[1]
        test_accuracy += pred.eq(target.data.view_as(pred)).cpu().float().sum()
    test_loss /= len(test_loader) * args.test_batch_size
    test_accuracy /= len(test_loader) * args.test_batch_size
    print('\nTest set: Average loss: {:.4f}, Accuracy: {:.2f}%\n'.format(
        test_loss, 100. * test_accuracy))
def save():
    """
    save
    """
    if not os.path.exists(args.output_dir):
        os.makedirs(args.output_dir)
    # 保存模型
    torch.save(model.state_dict(), os.path.join(args.output_dir, 'model.pkl'))
if __name__ == '__main__':
    args = parser.parse_args()
    args.cuda = not args.no_cuda and torch.cuda.is_available()
    # 若无测试集，训练集做验证集
    if not os.path.exists(args.test_dir) or not os.listdir(args.test_dir):
        args.test_dir = args.train_dir
    # 将数据进行转化，从PIL.Image/numpy.ndarray的数据进转化为torch.FloadTensor
    trans = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.1307,), (0.3081,))])
    train_set = MNIST(root=args.train_dir, train=True, transform=trans)
    test_set = MNIST(root=args.test_dir, train=False, transform=trans)
    # 定义data reader
    train_loader = torch.utils.data.DataLoader(
                    dataset=train_set,
                    batch_size=args.batch_size,
                    shuffle=True)
    test_loader = torch.utils.data.DataLoader(
                    dataset=test_set,
                    batch_size=args.test_batch_size,
                    shuffle=False)
    # 选择模型
    model = Net()
    if args.cuda:
        # Move model to GPU.
        model.cuda()
    print(model)
    # 选择优化器
    optimizer = optim.SGD(model.parameters(), lr=args.lr,
                          momentum=args.momentum)
    for epoch in range(1, args.epochs + 1):
        train(epoch)
        test()
        save()
```

分布式训练时（计算节点大于1），示例代码如下：

说明：demo分布式程序没有做数据的分片操作，仅供参考

```
 import argparse
import torch
import torch.nn as nn
import torch.nn.functional as F
```

```
import torch.optim as optim
import torch.utils.data as data
from torchvision import datasets, transforms
import codecs
import errno
import gzip
import numpy as np
import os
from PIL import Image
import torch.multiprocessing as mp
import torch.utils.data.distributed
import horovod.torch as hvd
**Training settings**
parser = argparse.ArgumentParser(description='PyTorch MNIST Example')
parser.add_argument('--train-dir', type=str, default='./train_data',
                    help='input data dir for training (default: ./train_data)')
parser.add_argument('--test-dir', type=str, default='./test_data',
                    help='input data dir for test (default: ./test_data)')
parser.add_argument('--output-dir', type=str, default='./output',
                    help='output dir for custom job (default: ./output)')
parser.add_argument('--batch-size', type=int, default=64, metavar='N',
                    help='input batch size for training (default: 64)')
parser.add_argument('--test-batch-size', type=int, default=64, metavar='N',
                    help='input batch size for testing (default: 64)')
parser.add_argument('--epochs', type=int, default=10, metavar='N',
                    help='number of epochs to train (default: 10)')
parser.add_argument('--lr', type=float, default=0.01, metavar='LR',
                    help='learning rate (default: 0.01)')
parser.add_argument('--momentum', type=float, default=0.5, metavar='M',
                    help='SGD momentum (default: 0.5)')
parser.add_argument('--no-cuda', action='store_true', default=False,
                    help='disables CUDA training')
parser.add_argument('--seed', type=int, default=42, metavar='S',
                    help='random seed (default: 42)')
parser.add_argument('--log-interval', type=int, default=10, metavar='N',
                    help='how many batches to wait before logging training status')
parser.add_argument('--fp16-allreduce', action='store_true', default=False,
                    help='use fp16 compression during allreduce')
parser.add_argument('--use-adasum', action='store_true', default=False,
                    help='use adasum algorithm to do reduction')
parser.add_argument('--gradient-predivide-factor', type=float, default=1.0,
                    help='apply gradient predivide factor in optimizer (default: 1.0)')
**定义MNIST数据集的dataset**
class MNIST(data.Dataset):
    """
    MNIST dataset
    """
    training_file = 'training.pt'
    test_file = 'test.pt'
    classes = ['0 - zero', '1 - one', '2 - two', '3 - three', '4 - four',
               '5 - five', '6 - six', '7 - seven', '8 - eight', '9 - nine']
    def __init__(self, root, train=True, transform=None, target_transform=None):
        self.root = os.path.expanduser(root)
        self.transform = transform
        self.target_transform = target_transform
        self.train = train  # training set or test set
        self.preprocess(root, train, False)
        if self.train:
            data_file = self.training_file
        else:
            data_file = self.test_file
        self.data, self.targets = torch.load(os.path.join(self.processed_folder, data_file))

    def __getitem__(self, index):
        """
        Args:
            index (int): Index
        Returns:
            tuple: (image, target) where target is index of the target class.
        """
```

```
        img, target = self.data[index], int(self.targets[index])
        # doing this so that it is consistent with all other datasets
        # to return a PIL Image
        img = Image.fromarray(img.numpy(), mode='L')
        if self.transform is not None:
            img = self.transform(img)
        if self.target_transform is not None:
            target = self.target_transform(target)

        return img, target
    def __len__(self):
        return len(self.data)

    @property
    def raw_folder(self):
        """
        raw folder
        """
        return os.path.join('/tmp', 'raw')

    @property
    def processed_folder(self):
        """
        processed folder
        """
        return os.path.join('/tmp', 'processed')

    # data preprocessing
    def preprocess(self, train_dir, train, remove_finished=False):
        """
        preprocess
        """
        makedir_exist_ok(self.raw_folder)
        makedir_exist_ok(self.processed_folder)
        train_list = ['train-images-idx3-ubyte.gz', 'train-labels-idx1-ubyte.gz']
        test_list = ['t10k-images-idx3-ubyte.gz', 't10k-labels-idx1-ubyte.gz']
        zip_list = train_list if train else test_list
        for zip_file in zip_list:
            print('Extracting {}'.format(zip_file))
            zip_file_path = os.path.join(train_dir, zip_file)
            raw_folder_path = os.path.join(self.raw_folder, zip_file)
            with open(raw_folder_path.replace('.gz', ''), 'wb') as out_f, gzip.GzipFile(zip_file_path) as zip_f:
                out_f.write(zip_f.read())
            if remove_finished:
                os.unlink(zip_file_path)
        if train:
            training_set = (
                read_image_file(os.path.join(self.raw_folder, 'train-images-idx3-ubyte')),
                read_label_file(os.path.join(self.raw_folder, 'train-labels-idx1-ubyte'))
            )
            with open(os.path.join(self.processed_folder, self.training_file), 'wb') as f:
                torch.save(training_set, f)
        else:
            test_set = (
                read_image_file(os.path.join(self.raw_folder, 't10k-images-idx3-ubyte')),
                read_label_file(os.path.join(self.raw_folder, 't10k-labels-idx1-ubyte'))
            )
            with open(os.path.join(self.processed_folder, self.test_file), 'wb') as f:
                torch.save(test_set, f)
def get_int(b):
    """
    get int
    """
    return int(codecs.encode(b, 'hex'), 16)
def read_label_file(path):
    """
    read label file
    """
    with open(path, 'rb') as f:
        data = f.read()
```

```
        assert get_int(data[:4]) == 2049
        length = get_int(data[4:8])
        parsed = np.frombuffer(data, dtype=np.uint8, offset=8)
        return torch.from_numpy(parsed).view(length).long()
def read_image_file(path):
    """
    read image file
    """
    with open(path, 'rb') as f:
        data = f.read()
        assert get_int(data[:4]) == 2051
        length = get_int(data[4:8])
        num_rows = get_int(data[8:12])
        num_cols = get_int(data[12:16])
        parsed = np.frombuffer(data, dtype=np.uint8, offset=16)
        return torch.from_numpy(parsed).view(length, num_rows, num_cols)
def makedir_exist_ok(dirpath):
    """
    Python2 support for os.makedirs(.., exist_ok=True)
    """
    try:
        os.makedirs(dirpath)
    except OSError as e:
        if e.errno == errno.EEXIST:
            pass
        else:
            raise
**定义网络模型**
class Net(nn.Module):
    """
    Net
    """
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 10, kernel_size=5)
        self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)
    def forward(self, x):
        """
        forward
        """
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
        x = x.view(-1, 320)
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        x = self.fc2(x)
        return F.log_softmax(x)
def train(epoch):
    """
    train
    """
    model.train()
    # Horovod: set epoch to sampler for shuffling.
    train_sampler.set_epoch(epoch)
    for batch_idx, (data, target) in enumerate(train_loader):
        if args.cuda:
            data, target = data.cuda(), target.cuda()
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()
        if batch_idx % args.log_interval == 0:
            # Horovod: use train_sampler to determine the number of examples in
            # this worker's partition.
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_sampler),
```

```
                100. * batch_idx / len(train_loader), loss.item()))
def metric_average(val, name):
    """
    metric average
    """
    tensor = torch.tensor(val)
    avg_tensor = hvd.allreduce(tensor, name=name)
    return avg_tensor.item()
def test():
    """
    test
    """
    model.eval()
    test_loss = 0.
    test_accuracy = 0.
    for data, target in test_loader:
        if args.cuda:
            data, target = data.cuda(), target.cuda()
        output = model(data)
        # sum up batch loss
        test_loss += F.nll_loss(output, target, size_average=False).item()
        # get the index of the max log-probability
        pred = output.data.max(1, keepdim=True)[1]
        test_accuracy += pred.eq(target.data.view_as(pred)).cpu().float().sum()
    # Horovod: use test_sampler to determine the number of examples in
    # this worker's partition.
    test_loss /= len(test_sampler)
    test_accuracy /= len(test_sampler)
    # Horovod: average metric values across workers.
    test_loss = metric_average(test_loss, 'avg_loss')
    test_accuracy = metric_average(test_accuracy, 'avg_accuracy')
    # Horovod: print output only on first rank.
    if hvd.rank() == 0:
        print('\nTest set: Average loss: {:.4f}, Accuracy: {:.2f}%\n'.format(
            test_loss, 100. * test_accuracy))
def save():
    """
    save
    """
    if not os.path.exists(args.output_dir):
        os.makedirs(args.output_dir)
    # 保存模型
    # Horovod: save model only on first rank.
    if hvd.rank() == 0:
        torch.save(model.state_dict(), os.path.join(args.output_dir, 'model.pkl'))
if __name__ == '__main__':
    args = parser.parse_args()
    args.cuda = not args.no_cuda and torch.cuda.is_available()
    # Horovod: initialize library.
    hvd.init()
    torch.manual_seed(args.seed)
    if args.cuda:
        # Horovod: pin GPU to local rank.
        torch.cuda.set_device(hvd.local_rank())
        torch.cuda.manual_seed(args.seed)
    # Horovod: limit # of CPU threads to be used per worker.
    torch.set_num_threads(1)
    kwargs = {'num_workers': 1, 'pin_memory': True} if args.cuda else {}
    # When supported, use 'forkserver' to spawn dataloader workers instead of 'fork' to prevent
    # issues with Infiniband implementations that are not fork-safe
    if (kwargs.get('num_workers', 0) > 0 and hasattr(mp, '_supports_context') and
            mp._supports_context and 'forkserver' in mp.get_all_start_methods()):
        kwargs['multiprocessing_context'] = 'forkserver'

    # 若无测试集，训练集做验证集
    if not os.path.exists(args.test_dir) or not os.listdir(args.test_dir):
        args.test_dir = args.train_dir
     # 将数据进行转化，从PIL.Image/numpy.ndarray的数据进转化为torch.FloadTensor
    trans = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.1307,), (0.3081,))])
    train_set = MNIST(root=args.train_dir, train=True, transform=trans)
```

```
    test_set = MNIST(root=args.test_dir, train=False, transform=trans)
    # Horovod: use DistributedSampler to partition the training data.
    train_sampler = torch.utils.data.distributed.DistributedSampler(
        train_set, num_replicas=hvd.size(), rank=hvd.rank())
    # Horovod: use DistributedSampler to partition the test data.
    test_sampler = torch.utils.data.distributed.DistributedSampler(
        test_set, num_replicas=hvd.size(), rank=hvd.rank())
    # 定义data reader
    train_loader = torch.utils.data.DataLoader(
                    dataset=train_set,
                    batch_size=args.batch_size,
                    sampler=train_sampler,
                    **kwargs)
    test_loader = torch.utils.data.DataLoader(
                    dataset=test_set,
                    batch_size=args.test_batch_size,
                    sampler=test_sampler,
                    **kwargs)
    model = Net()
    # By default, Adasum doesn't need scaling up learning rate.
    lr_scaler = hvd.size() if not args.use_adasum else 1
    if args.cuda:
        # Move model to GPU.
        model.cuda()
        # If using GPU Adasum allreduce, scale learning rate by local_size.
        if args.use_adasum and hvd.nccl_built():
            lr_scaler = hvd.local_size()
    # Horovod: scale learning rate by lr_scaler.
    optimizer = optim.SGD(model.parameters(), lr=args.lr * lr_scaler,
                          momentum=args.momentum)
    # Horovod: broadcast parameters & optimizer state.
    hvd.broadcast_parameters(model.state_dict(), root_rank=0)
    hvd.broadcast_optimizer_state(optimizer, root_rank=0)
    # Horovod: (optional) compression algorithm.
    compression = hvd.Compression.fp16 if args.fp16_allreduce else hvd.Compression.none
    # Horovod: wrap optimizer with DistributedOptimizer.
    optimizer = hvd.DistributedOptimizer(optimizer,
                                         named_parameters=model.named_parameters(),
                                         compression=compression,
                                         op=hvd.Adasum if args.use_adasum else hvd.Average,
                                         gradient_predivide_factor=args.gradient_predivide_factor)
    for epoch in range(1, args.epochs + 1):
        train(epoch)
        test()
        save()
```

推理代码

Pytorch模型在发布到模型仓库时，需要上传用于启动服务的自定义代码，并且在**主文件名**指定的py模块中实现：模型加载『model_fn』、请求预处理『input_fn』和预测结果后处理『output_fn』函数。

示例代码：

```
 **!/usr/bin/env python**
**-*- coding: utf-8 -*-**
"""
@license: Copyright (c) 2019 Baidu.com, Inc. All Rights Reserved.
@desc: 图像预测算法示例
"""
import logging
import torch
import torch.nn as nn
import torch.nn.functional as F
import base64
import json
from PIL import Image
from io import BytesIO
from torchvision import datasets, models, transforms
MODEL_FILE_NAME = 'model.pkl'  # 模型文件名称
def get_image_transform():
    """获取图片处理的transform
```

```
    获取图片处理的transform
    Args:
        data_type: string, type of data(train/test)
    Returns:
        torchvision.transforms.Compose
    """
    trans = transforms.Compose([transforms.Resize((28, 28)),
                                transforms.ToTensor(),
                                transforms.Normalize((0.5,), (1.0,))])
    return trans
def model_fn(model_dir):
    """模型加载
    Args:
        model_dir: 模型路径，该目录存储的文件为在自定义作业选择的输出路径下产出的文件
    Returns:
        加载好的模型对象
    """
    class Net(nn.Module):
        """
        Net
        """
        def __init__(self):
            super(Net, self).__init__()
            self.conv1 = nn.Conv2d(1, 10, kernel_size=5)
            self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
            self.conv2_drop = nn.Dropout2d()
            self.fc1 = nn.Linear(320, 50)
            self.fc2 = nn.Linear(50, 10)
        def forward(self, x):
            """
            forward
            """
            x = F.relu(F.max_pool2d(self.conv1(x), 2))
            x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
            x = x.view(-1, 320)
            x = F.relu(self.fc1(x))
            x = F.dropout(x, training=self.training)
            x = self.fc2(x)
            return F.log_softmax(x)
    model = Net()
    meta_info_path = "%s/%s" % (model_dir, MODEL_FILE_NAME)
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model.load_state_dict(torch.load(meta_info_path, map_location=device))
    model.to(device)
    logging.info("device type: " + str(device))
    return model
def input_fn(request):
    """对输入进行格式化，处理为预测需要的输入格式
    Args:
        request: api请求的json
    Returns:
        预测需要的输入数据，一般为tensor
    """
    instances = request['instances']
    transform_composes = get_image_transform()
    arr_tensor_data = []
    for instance in instances:
        decoded_data = base64.b64decode(instance['data'].encode("utf8"))
        byte_stream = BytesIO(decoded_data)
        roiImg = Image.open(byte_stream)
        target_data = transform_composes(roiImg)
        arr_tensor_data.append(target_data)
    tensor_data = torch.stack(arr_tensor_data, dim=0)
    return tensor_data
def output_fn(predict_result):
    """进行输出格式化
    Args:
        predict_result: 预测结果
    Returns:
        格式化后的预测结果，需能够json序列化以便接口返回
    """
```

```
    js_str = None
    if type(predict_result) == torch.Tensor:
        list_prediction = predict_result.detach().cpu().numpy().tolist()
        js_str = json.dumps(list_prediction)
    return js_str
```

Sklearn 0.23.2

Sklearn

sklearn框架下，自定义作业支持发布保存模型为pickle和joblib格式，并且在发布至模型仓库时需要选择相应的模型文件。使用下面代码进行模型训练时，训练程序可以自行加载数据，训练数据选择空文件夹即可。

pickle格式模型示例代码:

```
 **-*- coding:utf-8 -*-**
""" sklearn train demo """
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
def load_data():
    """ load data """
    # 共150条数据，训练120条，测试30条，进行2,8分进行模型训练
    # 每条数据类型为 x{nbarray} [6.4, 3.1, 5.5, 1.8]
    inputdata = datasets.load_iris()
    # 切分，测试训练2,8分
    x_train, x_test, y_train, y_test = \
        train_test_split(inputdata.data, inputdata.target, test_size = 0.2, random_state=0)
    return x_train, x_test, y_train, y_test
def save_model(model):
    """ save model with pickle format """
    import pickle
    with open('output/clf.pickle','wb') as f:
        pickle.dump(model, f)
def save_model_joblib(model):
    """ save model with joblib format """
    try:
        import joblib
    except:
        from sklearn.externals import joblib
    joblib.dump(model, 'output/clf.pkl')
def main():
    """ main """
    # 训练集x ,测试集x,训练集label,测试集label
    x_train, x_test, y_train, y_test = load_data()
    # l2为正则项
    model = LogisticRegression(penalty='l2')
    model.fit(x_train, y_train)
    save_model(model)
    print("w: %s" % model.coef_)
    print("b: %s" % model.intercept_)
    # 准确率
    print("precision: %s" % model.score(x_test, y_test))
    print("MSE: %s" % np.mean((model.predict(x_test) - y_test) ** 2))
if __name__ == '__main__':
    main()
```

joblib格式示例代码:

```
 **-*- coding:utf-8 -*-**
import numpy as np

from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

def load_data():
    # 共150条数据，训练120条，测试30条，进行2,8分进行模型训练
    # 每条数据类型为 x{nbarray} [6.4, 3.1, 5.5, 1.8]
    inputdata = datasets.load_iris()
    # 切分，测试训练2,8分
    x_train, x_test, y_train, y_test = \
        train_test_split(inputdata.data, inputdata.target, test_size = 0.2, random_state=0)
    return x_train, x_test, y_train, y_test


def save_model(model):
    import pickle
    with open('output/clf.pickle','wb') as f:
        pickle.dump(model, f)


def save_model_joblib(model):
    from sklearn.externals import joblib
    joblib.dump(model, 'output/clf.pkl')


def main():
    # 训练集x ,测试集x,训练集label,测试集label
    x_train, x_test, y_train, y_test = load_data()
    # l2为正则项
    model = LogisticRegression(penalty='l2')
    model.fit(x_train, y_train)

    save_model_joblib(model)
    print("w: %s" % model.coef_)
    print("b: %s" % model.intercept_)
    # 准确率
    print("precision: %s" % model.score(x_test, y_test))
    print("MSE: %s" % np.mean((model.predict(x_test) - y_test) ** 2))

if __name__ == '__main__':
    main()
```

XGBoost 1.3.1

XGBoost XGBoost框架下，自定义作业支持发布保存模型为pickle和joblib格式，并且在发布至模型仓库时需要选择相应的模型文件。使用下面代码进行模型训练时，训练程序可以自行加载数据，训练数据选择空文件夹即可。

pickle格式示例代码

```
 **-*- coding:utf-8 -*-**
""" xgboost train demo """
import xgboost as xgb
import numpy as np
def save_model(model):
    """ save model with pickle format """
    import pickle
    with open('output/clf.pickle', 'wb') as f:
        pickle.dump(model, f)
def save_model_joblib(model):
    """ save model with joblib format """
    import joblib
    joblib.dump(model, 'output/clf.pkl')
def main():
    """ main """
    rawData = np.array([[2, 4], [3, 4], [1, 2], [4, 5], [7, 8]])
    label = np.array([6, 7, 3, 9, 15])
    dtrain = xgb.DMatrix(rawData, label=label)
    deval = xgb.DMatrix(np.array([[3, 5], [3, 6]]), label=np.array([8, 9]))
    param = {'max_depth': 2, 'eta': 1, 'silent': 1, 'objective': 'reg:linear'}
    evallist = [(deval, 'eval'), (dtrain, 'train')]
    num_round = 10
    bst = xgb.train(param, dtrain, num_round, evallist)
    dtest = xgb.DMatrix(np.array([[2, 4], [7, 8]]))
    ypred = bst.predict(dtest)
    print(ypred)
    save_model_joblib(bst)
if __name__ == '__main__':
    main()
```

PaddlePaddle 2.0.0rc

Paddle 此处提供基于Paddle框架的MNIST图像分类示例代码，数据集请点击这里下载。

单机训练时（计算节点等于1），示例代码如下：

```
 import os
import numpy
import paddle # 导入paddle模块
import paddle.fluid as fluid
import gzip
import struct
work_path = os.getcwd()
cluster_train_dir = "%s/train_data" % work_path
def load_data(file_dir, is_train=True):
    """
    :param file_dir:
    :param is_train:
    :return:
    """
    if is_train:
        image_path = file_dir + '/train-images-idx3-ubyte.gz'
        label_path = file_dir + '/train-labels-idx1-ubyte.gz'
    else:
        image_path = file_dir + '/t10k-images-idx3-ubyte.gz'
        label_path = file_dir + '/t10k-labels-idx1-ubyte.gz'
    with open(image_path.replace('.gz', ''), 'wb') as out_f, gzip.GzipFile(image_path) as zip_f:
        out_f.write(zip_f.read())
        os.unlink(image_path)
    with open(label_path.replace('.gz', ''), 'wb') as out_f, gzip.GzipFile(label_path) as zip_f:
        out_f.write(zip_f.read())
        os.unlink(label_path)
    with open(label_path[:-3], 'rb') as lbpath:
        magic, n = struct.unpack('>II', lbpath.read(8))
        labels = numpy.fromfile(lbpath, dtype=numpy.uint8)
    with open(image_path[:-3], 'rb') as imgpath:
        magic, num, rows, cols = struct.unpack('>IIII', imgpath.read(16))
        images = numpy.fromfile(imgpath, dtype=numpy.uint8).reshape(len(labels), 784)
    return images, labels
```

```
    return images, labels
def reader_creator(file_dir, is_train=True, buffer_size=100):
    """
    :param file_dir:
    :param is_train:
    :param buffer_size:
    :return:
    """
    images, labels = load_data(file_dir, is_train)
    def reader():
        """
        :return:
        """
        for num in range(int(len(labels) / buffer_size)):
            for i in range(buffer_size):
                yield images[num * buffer_size + i, :], int(labels[num * buffer_size + i])
    return reader
def softmax_regression():
    """
    定义softmax分类器：
        一个以softmax为激活函数的全连接层
    Return:
        predict_image -- 分类的结果
    """
    # 输入的原始图像数据，大小为28*28*1
    img = fluid.layers.data(name='img', shape=[1, 28, 28], dtype='float32')
    # 以softmax为激活函数的全连接层，输出层的大小必须为数字的个数10
    predict = fluid.layers.fc(
        input=img, size=10, act='softmax')
    return predict
def multilayer_perceptron():
    """
    定义多层感知机分类器：
        含有两个隐藏层（全连接层）的多层感知器
        其中前两个隐藏层的激活函数采用 ReLU，输出层的激活函数用 Softmax
    Return:
        predict_image -- 分类的结果
    """
    # 输入的原始图像数据，大小为28*28*1
    img = fluid.layers.data(name='img', shape=[1, 28, 28], dtype='float32')
    # 第一个全连接层，激活函数为ReLU
    hidden = fluid.layers.fc(input=img, size=200, act='relu')
    # 第二个全连接层，激活函数为ReLU
    hidden = fluid.layers.fc(input=hidden, size=200, act='relu')
    # 以softmax为激活函数的全连接输出层，输出层的大小必须为数字的个数10
    prediction = fluid.layers.fc(input=hidden, size=10, act='softmax')
    return prediction
def convolutional_neural_network():
    """
    定义卷积神经网络分类器：
        输入的二维图像，经过两个卷积-池化层，使用以softmax为激活函数的全连接层作为输出层
    Return:
        predict -- 分类的结果
    """
    # 输入的原始图像数据，大小为28*28*1
    img = fluid.layers.data(name='img', shape=[1, 28, 28], dtype='float32')
    # 第一个卷积-池化层
    # 使用20个5*5的滤波器，池化大小为2，池化步长为2，激活函数为Relu
    conv_pool_1 = fluid.nets.simple_img_conv_pool(
        input=img,
        filter_size=5,
        num_filters=20,
        pool_size=2,
        pool_stride=2,
        act="relu")
    conv_pool_1 = fluid.layers.batch_norm(conv_pool_1)
    # 第二个卷积-池化层
    # 使用20个5*5的滤波器，池化大小为2，池化步长为2，激活函数为Relu
    conv_pool_2 = fluid.nets.simple_img_conv_pool(
        input=conv_pool_1,
        filter_size=5
```

```
        num_filters=50,
        pool_size=2,
        pool_stride=2,
        act="relu")
    # 以softmax为激活函数的全连接输出层，输出层的大小必须为数字的个数10
    prediction = fluid.layers.fc(input=conv_pool_2, size=10, act='softmax')
    return prediction
def train_program():
    """
    配置train_program
    Return:
        predict -- 分类的结果
        avg_cost -- 平均损失
        acc -- 分类的准确率
    """
    paddle.enable_static()
    # 标签层，名称为label,对应输入图片的类别标签
    label = fluid.layers.data(name='label', shape=[1], dtype='int64')
    # predict = softmax_regression() # 取消注释将使用 Softmax回归
    # predict = multilayer_perceptron() # 取消注释将使用 多层感知器
    predict = convolutional_neural_network() # 取消注释将使用 LeNet5卷积神经网络
    # 使用类交叉熵函数计算predict和label之间的损失函数
    cost = fluid.layers.cross_entropy(input=predict, label=label)
    # 计算平均损失
    avg_cost = fluid.layers.mean(cost)
    # 计算分类准确率
    acc = fluid.layers.accuracy(input=predict, label=label)
    return predict, [avg_cost, acc]
def optimizer_program():
    """
    :return:
    """
    return fluid.optimizer.Adam(learning_rate=0.001)
**一个minibatch中有64个数据**
BATCH_SIZE = 64
**每次读取训练集中的500个数据并随机打乱，传入batched reader中，batched reader 每次 yield 64个数据**
train_reader = paddle.batch(
        paddle.reader.shuffle(
            reader_creator(cluster_train_dir, is_train=True, buffer_size=100), buf_size=500),
        batch_size=BATCH_SIZE)
**读取测试集的数据，每次 yield 64个数据**
test_reader = paddle.batch(
            reader_creator(cluster_train_dir, is_train=False, buffer_size=100), batch_size=BATCH_SIZE)
def event_handler(pass_id, batch_id, cost):
    # 打印训练的中间结果，训练轮次，batch数，损失函数
    print("Pass %d, Batch %d, Cost %f" % (pass_id, batch_id, cost))
**该模型运行在单个CPU上**
place = fluid.CPUPlace()
**调用train_program 获取预测值，损失值，**
prediction, [avg_loss, acc] = train_program()
**输入的原始图像数据，大小为28*28*1**
img = fluid.layers.data(name='img', shape=[1, 28, 28], dtype='float32')
**标签层，名称为label,对应输入图片的类别标签**
label = fluid.layers.data(name='label', shape=[1], dtype='int64')
**告知网络传入的数据分为两部分，第一部分是img值，第二部分是label值**
feeder = fluid.DataFeeder(feed_list=[img, label], place=place)
**选择Adam优化器**
optimizer = fluid.optimizer.Adam(learning_rate=0.001)
optimizer.minimize(avg_loss)
PASS_NUM = 1 #训练1轮
epochs = [epoch_id for epoch_id in range(PASS_NUM)]
**将模型参数存储在名为 save_dirname 的文件中**
save_dirname = "./output/"
def train_test(train_test_program,
               train_test_feed, train_test_reader):
    # 将分类准确率存储在acc_set中
    acc_set = []
    # 将平均损失存储在avg_loss_set中
    avg_loss_set = []
    # 将测试 reader yield 出的每一个数据传入网络中进行训练
```

```
    for test_data in train_test_reader():
        acc_np, avg_loss_np = exe.run(
            program=train_test_program,
            feed=train_test_feed.feed(test_data),
            fetch_list=[acc, avg_loss])
        acc_set.append(float(acc_np))
        avg_loss_set.append(float(avg_loss_np))
    # 获得测试数据上的准确率和损失值
    acc_val_mean = numpy.array(acc_set).mean()
    avg_loss_val_mean = numpy.array(avg_loss_set).mean()
    # 返回平均损失值，平均准确率
    return avg_loss_val_mean, acc_val_mean
exe = fluid.Executor(place)
exe.run(fluid.default_startup_program())
main_program = fluid.default_main_program()
test_program = fluid.default_main_program().clone(for_test=True)
lists = []
step = 0
for epoch_id in epochs:
    for step_id, data in enumerate(train_reader()):
        metrics = exe.run(main_program,
                          feed=feeder.feed(data),
                          fetch_list=[avg_loss, acc])
        if step % 100 == 0: #每训练100次 更新一次图片
            event_handler(step, epoch_id, metrics[0])
        step += 1
    # 测试每个epoch的分类效果
    avg_loss_val, acc_val = train_test(train_test_program=test_program,
                                       train_test_reader=test_reader,
                                       train_test_feed=feeder)
    print("Test with Epoch %d, avg_cost: %s, acc: %s" % (epoch_id, avg_loss_val, acc_val))
    lists.append((epoch_id, avg_loss_val, acc_val))
    # 保存训练好的模型参数用于预测
    if save_dirname is not None:
        fluid.io.save_inference_model(save_dirname,
                                      ["img"], [prediction], exe,
                                      model_filename='model',
                                      params_filename='params')
**选择效果最好的pass**
best = sorted(lists, key=lambda list: float(list[1]))[0]
print('Best pass is %s, testing Avgcost is %s' % (best[0], best[1]))
print('The classification accuracy is %.2f%%' % (float(best[2]) * 100))
```

分布式训练时（计算节点大于1），示例代码如下： **说明：demo分布式程序没有做数据的分片操作，仅供参考**

```
 **!/usr/bin/env python**
**-*- coding: utf-8 -*-**
"""
"""
import os
import gzip
import struct
import numpy as np
from PIL import Image
import time
import paddle
import paddle.distributed.fleet as fleet
import paddle.static.nn as nn
import paddle.fluid as fluid
from paddle.io import Dataset
TEST_IMAGE = 't10k-images-idx3-ubyte.gz'
TEST_LABEL = 't10k-labels-idx1-ubyte.gz'
TRAIN_IMAGE = 'train-images-idx3-ubyte.gz'
TRAIN_LABEL = 'train-labels-idx1-ubyte.gz'
class MNIST(Dataset):
    """
    MNIST
    """
```

```
    def __init__(self,
                 data_dir=None,
                 mode='train',
                 transform=None,
                 backend=None):
        assert mode.lower() in ['train', 'test'], \
            "mode should be 'train' or 'test', but got {}".format(mode)
        if backend is None:
            backend = paddle.vision.get_image_backend()
        if backend not in ['pil', 'cv2']:
            raise ValueError(
                "Expected backend are one of ['pil', 'cv2'], but got {}"
                .format(backend))
        self.backend = backend
        self.mode = mode.lower()
        if self.mode == 'train':
            self.image_path = os.path.join(data_dir, TRAIN_IMAGE)
            self.label_path = os.path.join(data_dir, TRAIN_LABEL)
        else:
            self.image_path = os.path.join(data_dir, TEST_IMAGE)
            self.label_path = os.path.join(data_dir, TEST_LABEL)
        self.transform = transform
        # read dataset into memory
        self._parse_dataset()
        self.dtype = paddle.get_default_dtype()
    def _parse_dataset(self, buffer_size=100):
        self.images = []
        self.labels = []
        with gzip.GzipFile(self.image_path, 'rb') as image_file:
            img_buf = image_file.read()
            with gzip.GzipFile(self.label_path, 'rb') as label_file:
                lab_buf = label_file.read()
                step_label = 0
                offset_img = 0
                # read from Big-endian
                # get file info from magic byte
                # image file : 16B
                magic_byte_img = '>IIII'
                magic_img, image_num, rows, cols = struct.unpack_from(
                    magic_byte_img, img_buf, offset_img)
                offset_img += struct.calcsize(magic_byte_img)
                offset_lab = 0
                # label file : 8B
                magic_byte_lab = '>II'
                magic_lab, label_num = struct.unpack_from(magic_byte_lab,
                                                          lab_buf, offset_lab)
                offset_lab += struct.calcsize(magic_byte_lab)
                while True:
                    if step_label >= label_num:
                        break
                    fmt_label = '>' + str(buffer_size) + 'B'
                    labels = struct.unpack_from(fmt_label, lab_buf, offset_lab)
                    offset_lab += struct.calcsize(fmt_label)
                    step_label += buffer_size
                    fmt_images = '>' + str(buffer_size * rows * cols) + 'B'
                    images_temp = struct.unpack_from(fmt_images, img_buf,
                                                     offset_img)
                    images = np.reshape(images_temp, (buffer_size, rows *
                                                      cols)).astype('float32')
                    offset_img += struct.calcsize(fmt_images)
                    for i in range(buffer_size):
                        self.images.append(images[i, :])
                        self.labels.append(
                            np.array([labels[i]]).astype('int64'))
    def __getitem__(self, idx):
        image, label = self.images[idx], self.labels[idx]
        image = np.reshape(image, [28, 28])
        if self.backend == 'pil':
            image = Image.fromarray(image.astype('uint8'), mode='L')
        if self.transform is not None:
```

```
            image = self.transform(image)
        if self.backend == 'pil':
            return image, label.astype('int64')
        return image.astype(self.dtype), label.astype('int64')
    def __len__(self):
        return len(self.labels)
def mlp_model():
    """
    mlp_model
    """
    x = paddle.static.data(name="x", shape=[64, 28, 28], dtype='float32')
    y = paddle.static.data(name="y", shape=[64, 1], dtype='int64')
    x_flatten = paddle.reshape(x, [64, 784])
    fc_1 = nn.fc(x=x_flatten, size=128, activation='tanh')
    fc_2 = nn.fc(x=fc_1, size=128, activation='tanh')
    prediction = nn.fc(x=[fc_2], size=10, activation='softmax')
    cost = paddle.fluid.layers.cross_entropy(input=prediction, label=y)
    acc_top1 = paddle.metric.accuracy(input=prediction, label=y, k=1)
    avg_cost = paddle.mean(x=cost)
    res = [x, y, prediction, avg_cost, acc_top1]
    return res
def train(epoch, exe, train_dataloader, cost, acc):
    """
    train
    """
    total_time = 0
    step = 0
    for data in train_dataloader():
        step += 1
        start_time = time.time()
        loss_val, acc_val = exe.run(
        paddle.static.default_main_program(),
        feed=data, fetch_list=[cost.name, acc.name])
        if step % 100 == 0:
            end_time = time.time()
            total_time += (end_time - start_time)
            print(
                "epoch: %d, step:%d, train_loss: %f, train_acc: %f, total time cost = %f, speed: %f"
              % (epoch, step, loss_val[0], acc_val[0], total_time,
                1 / (end_time - start_time) ))
def test(exe, test_dataloader, cost, acc):
    """
    test
    """
    total_time = 0
    step = 0
    for data in test_dataloader():
        step += 1
        start_time = time.time()
        loss_val, acc_val = exe.run(
        paddle.static.default_main_program(),
        feed=data, fetch_list=[cost.name, acc.name])
        if step % 100 == 0:
            end_time = time.time()
            total_time += (end_time - start_time)
            print(
                "step:%d, test_loss: %f, test_acc: %f, total time cost = %f, speed: %f"
              % (step, loss_val[0], acc_val[0], total_time,
                1 / (end_time - start_time) ))
def save(save_dir, feed_vars, fetch_vars, exe):
    """
    save
    """
    path_prefix = os.path.join(save_dir, 'model')
    if fleet.is_first_worker():
        paddle.static.save_inference_model(path_prefix, feed_vars, fetch_vars, exe)
if __name__ == '__main__':
    # 设置训练集路径
    train_data = './train_data'
    # 设置验证集路径
```

```
    test_data = './test_data'
    # 设置输出路径
    save_dir = './output'
    # 设置迭代轮数
    epochs = 10
    # 设置验证间隔轮数
    test_interval = 2
    # 设置模型保存间隔轮数
    save_interval = 2
    paddle.enable_static()
    paddle.vision.set_image_backend('cv2')
    # 训练数据集
    train_dataset = MNIST(data_dir=train_data, mode='train')
    # 验证数据集
    test_dataset = MNIST(data_dir=test_data, mode='test')
    # 设置模型
    [x, y, pred, cost, acc] = mlp_model()
    place = paddle.CUDAPlace(int(os.environ.get('FLAGS_selected_gpus', 0)))
    # 数据加载
    train_dataloader = paddle.io.DataLoader(
        train_dataset, feed_list=[x, y], drop_last=True,
        places=place, batch_size=64, shuffle=True, return_list=False)
    test_dataloader = paddle.io.DataLoader(
        test_dataset, feed_list=[x, y], drop_last=True,
        places=place, batch_size=64, return_list=False)
    # fleet初始化
    strategy = fleet.DistributedStrategy()
    fleet.init(is_collective=True, strategy=strategy)
    # 设置优化器
    optimizer = paddle.optimizer.Adam()
    optimizer = fleet.distributed_optimizer(optimizer)
    optimizer.minimize(cost)
    exe = paddle.static.Executor(place)
    exe.run(paddle.static.default_startup_program())
    prog = paddle.static.default_main_program()
    for epoch in range(epochs):
        train(epoch, exe, train_dataloader, cost, acc)
        if epoch % test_interval == 0:
            test(exe, test_dataloader, cost, acc)
        # save model
        if epoch % save_interval == 0:
            save(save_dir, [x], [pred], exe)
```

AIAK- Training Pytorch版

AIAK-Training 简介 AI 加速套件AI Accelerate Kit是基于百度云 IAAS 资源推出的 AI 加速能力，可用来加速基于 PyTorch 等深度学习框架的 AI 应用，详情可进一步查看 AI加速套件-AIAK 简介。



应用场景 AIAK-Training PyTorch版主要适用于计算机视觉模型，例如：Swin Transformer、Faster R-CNN、Mask R-CNN、YOLOV7、STDC等。 **使用示例**

1. 创建训练任务 选择PyTorch下的AIAK-Training镜像，并指定模型网络代码文件、数据集文件、输出文件等各自的BOS存储路径。



在输入启动命令时，通过传入约定参数来开启AIAK-Training的训练加速能力，以下为单机/多机场景下的启动命令示例。

```
 单机-不加速（1节点、1GPU）：bash run.sh -g 1 -e torch2.0
单机-AIAK-Training加速（1节点、1GPU）：bash run.sh -g 1 -e torch2.0 -c -pm -dp -f -m
多机-仅常规加速（2节点、2GPU）：bash run.sh -g 1 -n 2 -e torch2.0
多机-AIAK-Training加速（2节点、2GPU）：bash run.sh -g 1 -n 2 -e torch2.0 -c -pm -dp -f -m
```

参数含义如下：

```
 -e)
    TRAIN_CONDA_ENV=$2;
    选择conda环境版本，目前可支持torch2.0、torch1.12、torch1.13，默认base环境不支持训练。
-i)
    RANK=$2;
    单机训练默认为0，多机训练默认为0,1,2……
    如两机训练：第一台机器执行脚本时此参数为0，第二台机器执行脚本时此参数为1
-g)
    GPUS=$2;
    每台机器上的GPU卡数
-n)
    NNODES=$2;
    总节点数，单机训练为1，两机训练为2等等
-p)
    MASTER_PORT=$2;
    master节点的端口
-h)
    MASTER_ADDR=$2;
    master节点的地址
-c)
    CHANNEL_LAST="--use_channel_last";
    减少 nchw -> nhwc 转换的开销
-pm)
    PIN_MEMORY="--use_pin_memory";
    加速 H2D
-dp)
    DATA_PREFETCH="--use_data_prefetch";
    实现H2D和前向计算overlap，缩短训练耗时
-f)
    FUSED_OPTIMIZER="--fused_optimizer";
    通过算子融合，加速参数更新
-m)
    MULTI_EPOCHS_LOADER="--use_multi_epochs_loader";
    跨epoch dataloader优化
-a)
    AFFINITY="--set_affinity";
    实现CPU-GPU的亲和性绑定，消除跨numa传输带来的带宽损耗
-flb)
    FUSED_LINEAR_BIAS="--use_fused_linear";
    算子融合策略，支持 Module算子替换，减少kernel launch的时间
-fln)
    FUSED_LAYERNORM="--use_fused_layernorm";
    算子融合策略，支持 Module算子替换，减少kernel launch的时间
-d)
    DYNAMO="--use-dynamo";
    适用于PyTorch2.0的加速特性
```

2. 提交训练任务 选择运行环境，并按需配置计算节点数，提交训练任务。 训练完成后，通过训练任务的训练耗时或训练吞吐量对比，可对比AIAK-Training镜像所带来的训练加速效果提升。

自动搜索作业

自动搜索作业简介

自动搜索作业简介

超参搜索是机器学习/深度学习技术中的关键一环，无论是机器学习的树模型参数、特征选择、还是深度学习的学习率/权重衰减等等，甚至于网络结构的选择，都会涉及到搜索最优参数的需求。传统的人工超参搜索需要有经验的工程师耗费大量的时间和精力进行手动调优，而自动超参搜索能够在节省人力的情况下，自动地进行超参调优，更有效率地寻找最优解。

本模块提供的自动搜索作业，旨在帮助用户实现在各种AI框架以及各种应用场景下的超参搜索需求。用户只需上传数据及代码，指定训练框架、搜索算法以及需要搜索的超参数，即可完成超参搜索任务。

自动搜索作业目前支持以下5种超参搜索算法：

1.随机搜索
2.贝叶斯搜索

3.进化算法
4.进化-CMAES
5.粒子群算法

随机搜索

随机搜索作为最为常见的搜索算法之一，在搜索空间较大的场景中，拥有不错的表现。在每次试验中，算法会对每个参数进行随机取值并组合，作为新的一组建议超参用于训练。

算法参数	说明	取值参考
最大搜索次数/max_trial_num	算法给出的超参组合个数	int，默认值为10，越大，搜索时间越长，效果越好

贝叶斯搜索

贝叶斯搜索算法充分利用了之前试验的信息，假设超参数与最后的结果存在一个函数关系。通过历史试验的【超参数，对应的结果值】拟合一个目标函数，使目标函数最优的超参数组合作为新的一组建议超参用于训练。

算法参数	说明	取值参考
最大搜索次数/max_trial_num	算法给出的超参组合个数	int，默认值为10，越大，搜索时间越长，效果越好
初始点数量/n_startup_points	算法初始阶段采用随机搜索获取超参组合的个数	int，取值范围[1,20]，默认值为5，一般不建议用户修改
最大并发量/max_concurrent	搜索任务中同时运行的最大任务数	int，取值范围[1,20]，默认值为5，一般不建议用户修改

进化算法

进化算法采用的是PBT(Population based training)搜索算法，开始进行搜索时，会并行训练一批随机初始化超参数的模型。每隔一定的epoch，都会用表现较好的模型替换表现不好的模型，并且对超参数进行随机扰动，而表现较好的模型则继续训练。

算法参数	说明	取值参考
种群个体数量/population_num	算法每个轮次并行训练的模型数量	int，取值范围[1,10]，默认值为8，一般不建议用户修改
最大轮次/round	每个轮次的间隔会发生模型间的继承与超参数的扰动	int，取值范围[5,50]，默认值为10，一般不建议用户修改
扰动比例/quantile_frac	需要继承较好模型权重的较差模型比例	float，取值范围(0,0.5]，默认值为0.5，一般不建议用户修改，即较差的一半模型会继承较好一半模型的权重
扰动间隔/perturb_interval	一个轮次中训练的epoch数量，扰动间隔与最大轮次的乘积等于一个模型最终训练的epoch数量	float,取值范围[1,20],默认值为10，越大，训练时间越长
随机初始化概率/explore_prob	较差模型在继承较优模型的权重和超参后，会以一定的概率随机初始化超参数，否则，仅对超参数进行轻微扰动	float,取值范围(0,0.5],默认值为0.25，一般不建议用户修改

进化-CMAES

CMAES(Covariance Matrix Adaptation Evolutionary Strategies)是协方差矩阵自适应进化算法，是最先进的优化算法之一，在进化算法中避免了对种群规模的依赖，小种群可快速收敛，大种群可更好地全局搜索。

算法参数	说明	取值参考
种群个体数量/population_num	算法每个轮次并行训练的模型数量	int，取值范围[1,10]，默认值为8，一般不建议用户修改
最大轮次/round	进化轮次总数	int，取值范围[5,50]，默认值为10，一般不建议用户修改
学习步长/step_size	种群每次迭代更新时的学习率	float，取值范围(0,10]，默认值为1.0，一般不建议用户修改

粒子群算法

粒子群算法是一种源于对鸟群捕食行为的研究而发明的进化计算技术，算法初始化一组随机解，通过迭代搜寻最优解，不但具有全局寻优能力，而且具有较强的局部寻优能力。粒子群由 n个粒子组成，每个粒子代表一组超参数组合，在搜索空间中粒子以一定的速度飞行，这个速度根据它本身的飞行经验和同伴的飞行经验来动态调整，从而决定下一步的飞行方向和距离。

算法参数	说明	取值参考
种群个体数量/population_num	算法每个轮次并行训练的模型数量	int，取值范围[1,10]，默认值为8，一般不建议用户修改
最大轮次/round	进化轮次总数	int，取值范围[5,50]，默认值为10，一般不建议用户修改
惯性权重/inertia_weight	上一个批次的粒子速度占本批次的比例	float，取值范围(0,1]，默认值为0.5，其值越大，全局寻优能力越强，局部寻优能力越弱
全局加速度/global_acceleration	反应种群最佳值对个体速度影响的程度，也被称为群体认知项，是一个从当前点指向种群最好点的矢量，反映了粒子间的协同合作和知识共享	int，取值范围(0,4]，默认值为1.5，一般不建议用户修改
个体加速度/local_acceleration	反应个体最佳值对自身速度影响的程度，也被称为自身认知项，是从当前点指向粒子自身最好点的一个矢量，表示粒子的动作来源于自己经验的部分	int，取值范围(0,4]，默认值为1.5，一般不建议用户修改

创建自动搜索作业

创建自动搜索作业

1.前提条件
2.新建作业
3.使用自动搜索作业训练模型
3.1 基本信息
3.2 算法配置
3.3 数据集配置
3.4 自动搜索配置
3.5 资源配置
3.6 查看搜索结果及可视化
4.发布模型

前提条件

1. 自定义作业需要依赖于BOS对象存储读取输入文件，创建自定义作业之前需要保证您已经开通了BOS对象存储的服务。

2. 授权自定义作业读写您的BOS对象存储，以顺利进行自定义作业的配置。

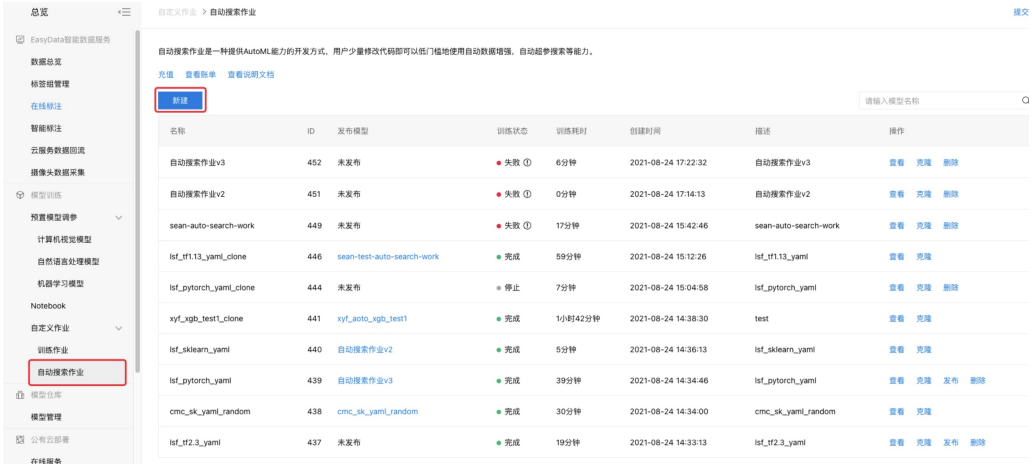


3. 在BOS中存储创建Bucket，并且存储用于训练的代码文件和数据集，创建一个空文件夹用于存储代码的输出文件。

新建作业

1. 在导航栏选择『自定义作业-自动搜索作业』，进入自动搜索作业的列表页。

2. 点击『新建作业』，进入配置自动搜索作业流程



使用自动搜索作业训练模型

自动搜索作业提供了多种开源框架、搜索算法以及优质的训练资源。您可以上传代码文件，数据集到BOS对象存储，通过自动搜索作业完成训练后，平台提供了每组超参数组合的参数值及其对应的模型评估指标、训练结果会输出到BOS中的指定输出目录，同时您也可以通过链接进入可视化界面查看具体搜索结果。

基本信息

填写作业名称和备注信息



算法配置



1. 选择算法框架：选择训练代码文件使用的算法框架，目前BML支持Paddle，TensorFlow，Pytorch，Sklearn，XGBoost五种主流算法框架。
2. 选择代码文件：从BOS对象存储中选取代码文件，完成代码录入。单击文件夹符号，从弹窗中选择bucket及文件夹。双击bucket或者单击『>』图标，即可进入下一级。



3. 代码文件要求如下：
   - 选取对象可以是一个文件或者文件夹
   - 所选代码文件必须与所选算法框架对应，所选路径必须包含适配自动搜索作业代码编写规范的代码文件，查看代码文件编写规范（链接到代码文件编写规范）。
4. 输入启动命令，支持python和shell两类脚本：
   - 当代码文件项选择一个单独文件时，启动命令指向该启动文件
   - 当代码文件项选择一个文件夹时，启动命令需指向该文件夹下的特定启动脚本，例如bash bml_job.sh
5. 可选高级配置。提交训练作业时，可以通过高级配置来自定义环境变量。
   - 手动配置：通过手动填写环境变量来配置，包括配置变量名称与对应取值。



   - YAML文件：通过上传YAML文件来配置环境变量，支持本地上传和BOS存储导入两种方式。注意：环境变量最多配置100组，平台将在解析时对超出部分做自动截断处理。

此外，BML平台亦提供预置环境变量，由BML平台预先定义，不建议您自行覆盖修改。

预置环境变量

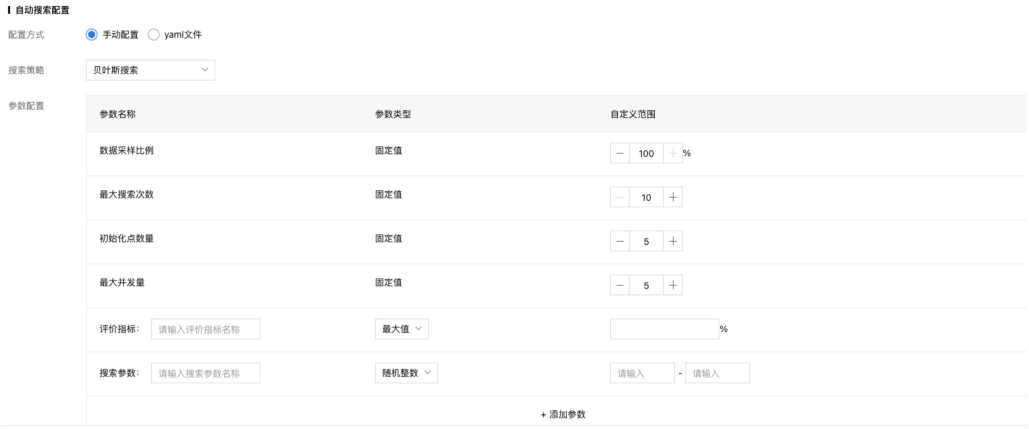
作业类型	变量名	变量含义	示例	默认值
自动搜索作业	MANAGER_ADDRESS	管理自动搜索作业节点exp-manager的svc地址	172.16.227.135:8888	无
	NVIDIA_VISIBLE_DEVICES	训练节点可见的gpu卡设备	GPU-ae58e6cc-1dec-bcb9-820c-e433d01afda6	无
	PYTHONPATH	训练任务节点汇报训练结果管理节点的sdk路径	/home/sdk/output	/home/sdk/output

数据集配置

自动搜索作业数据集配置方式是从BOS中选取数据集。在弹出的对话框中选择数据集对应的bucket和文件夹。注：选择的数据集路径与代码训练时传入代码的训练集、测试集路径一致。为保证训练效率，最好将数据存放在压缩包内。



自动搜索配置



1. 配置方式：

- 手动配置：用户需要在平台上手动选择搜索策略以及配置对应的参数。
- yaml文件：用户可以通过yaml文件来设置自动搜索配置，文件上传方式分为本地上传与从BOS中选取两种。



2. 搜索策略：

自动搜索作业目前提供了5种搜索算法，见搜索算法简介。

3. 参数配置：

- 选择搜索策略后，参数配置表格会提供对应的参数进行选择，其中关于搜索算法的参数说明见搜索算法简介。
- 非搜索参数的说明如下：
  - 数据采样比例：超参搜索需要进行多个模型的训练，因此为了提高效率，减少总的搜索耗时，可以设置训练集的数据采样比例。
  - 评价指标：用户在代码中可以自定义评价指标的计算方式，只需要在平台参数配置表格中填写评价指标的名称、优化方向（最大值/最小值）以及早停指标即可。下图是一个填写实例，评价指标：acc,最大值优化，当搜索实验的某个训练结果，acc的数值达到100%时，终止整个自动搜索任务。

  - 搜索参数：用户需要将待搜索的超参数填入显示框内，搜索参数的名称需要与代码中的名称保持严格一致！如搜索任务中有多个参数需要搜索，点击『+添加参数』即可，如下为搜索参数的参数类型以及对应的取值范围说明：
    - 随机整数：参数范围中填写搜索参数的上下界，算法在其中随机取整。
    - 离散值：参数范围中填写搜索参数的所有可能取值，以英文逗号隔开，算法会随机从中取值。
    - 平均采样：参数范围中填写搜索参数的上下界，算法在其中随机取值，包括整数和小数。
    - 对数平均采样：参数范围中填写搜索参数的上下界，算法在其对数尺度上随机取值，该参数类型适用于学习率等参数范围有尺度差异的超参数。



资源配置

BML提供CPU和GPU两类训练机型。

CPU机型供算法框架为sklearn，XGBoost时使用，用于机器学习训练：

机型	规格说明
CPU 4核	CPU 4核16GB内存
CPU 16核	CPU 16核64GB内存

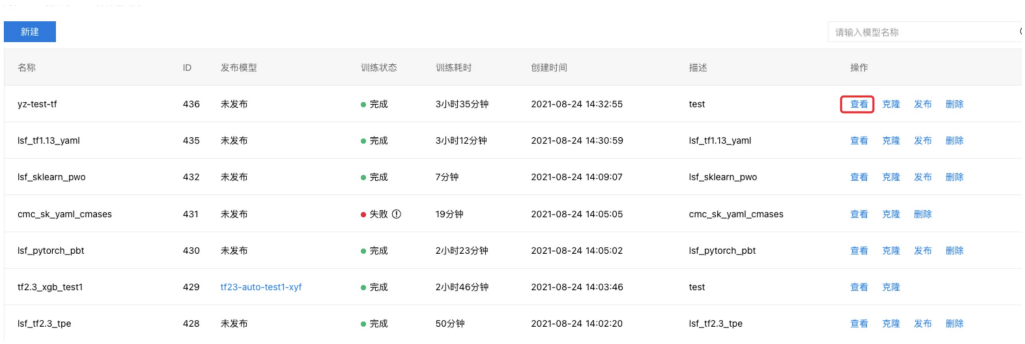
GPU机型供算法框架为Paddle，TensorFlow，Pytorch时使用，用于深度学习训练：

机型	规格说明
GPU V100	TeslaGPU_V100_32G显存单卡_12核CPU_56G内存
GPU P4	TeslaGPU_P4_8G显存单卡_12核CPU_40G内存

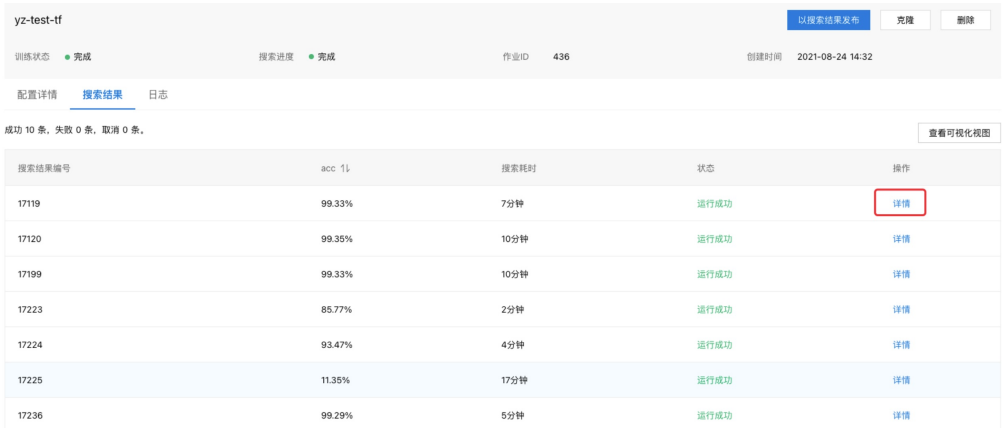
查看搜索结果及可视化

查看搜索结果

自动搜索作业运行完成后，在任务列表页中点击查看进入任务详情界面（任务运行时，也可点击查看实时信息）



点击搜索结果，表格中会显示所有试验的详细信息，如评价指标的值、搜索耗时，试验状态等；点击日志，可以查看每个试验的运行日志。



点击详情可以查看每个试验的超参数取值。



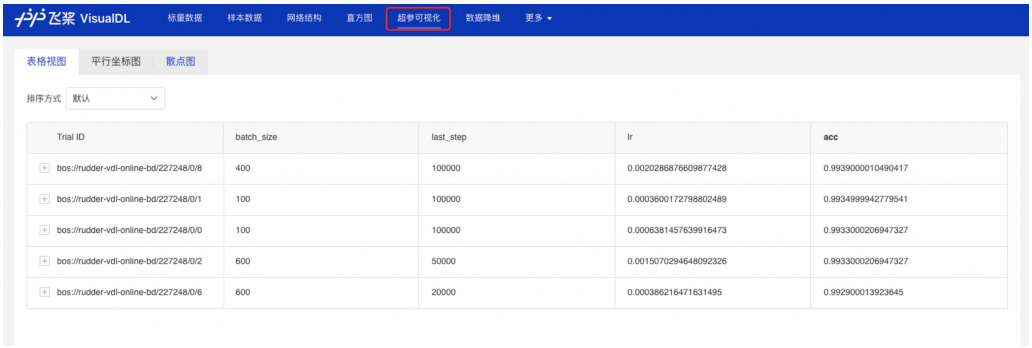
查看可视化视图

可视化界面目前仅保存了所有试验中指标最好的5个结果。

1. 点击查看可视化视图，可以进入飞桨VisualDL可视化界面。

2. 点击标题栏的超参可视化，自动搜索作业一共提供了三种图：

- 表格视图：搜索结果以表格的形式呈现。



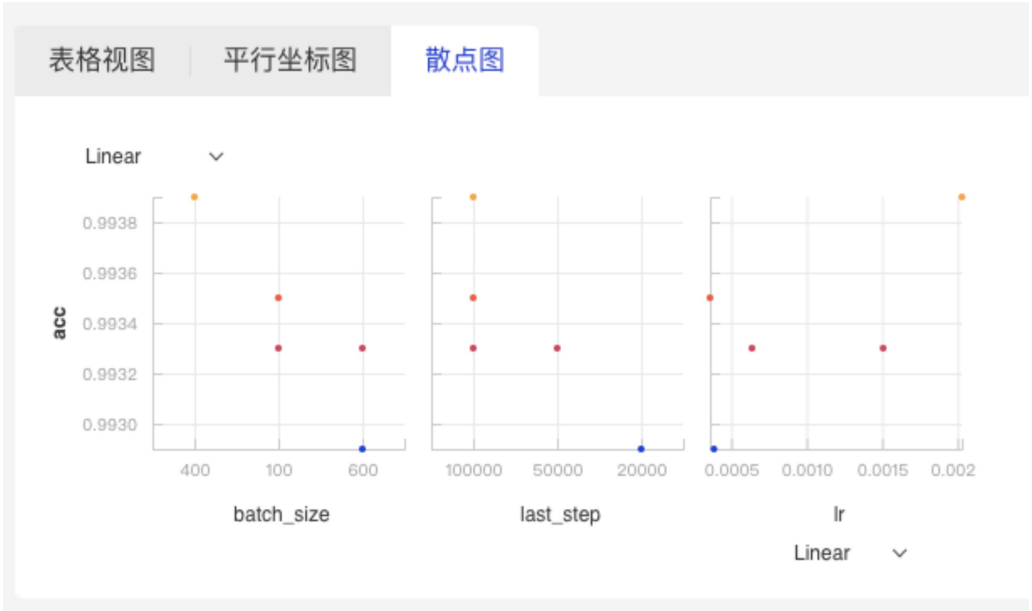
- 平行坐标图

  - 图中右方为柱状热力图，当在右上方选择一个参数时，平行坐标图便会以该参数作为基准结果，参数组合的折线颜色便会依据该组合在选择参数上的取值而改变。

  - 自动搜索作业中，选择评价指标（图中为acc）作为基准结果，然后观察各个超参数对其的影响大小。若某个超参数上相同颜色的折线较为集中，不同颜色有一定的距离，则说明该参数对于结果影响很大；而如果某个超参数上线条颜色混乱，则较大可能该参数对于结果的影响很小。



- 散点图：呈现超参数与指标之间的分布关系。



发布模型

自动搜索作业训练完成后支持将最好的5个模型直接发布到模型仓库。

1. 模型列表中，训练完成的模型支持『发布』操作，点击『发布』进入发布界面。



2. 在发布界面，完成相关配置，如下所示：

- 填写下图所示基本信息。『模型类型』和『版本』由系统自动生成，不支持修改。重新发布时，『模型名称和『模型描述』会自动填充，无需



再次填写。

- 根据算法框架的不同，您还需要在发布界面填写响应的额外配置项信息。

算法框架	额外配置项
Paddle Paddle	无
TensorFlow	无。说明：自动搜索作业支持发布pb格式的模型文件到模型仓库，因此代码中需要保存pb格式的模型
Pytorch	从BOS中选择源代码并输入源代码的主文件名。说明：选取的代码为启动服务的推理代码，具体说明点击这里（链接到pytorch代码示例中的推理代码部分）
Sklearn/XGBoost	模型文件格式：支持发布pickle格式或joblib格式模型到模型仓库。模型文件：选择一个搜索结果后，下方模型文件会自动填充对应试验的路径，用户只需要在对应路径下选中保存的模型即可。说明：其他框架在保存模型时，模型名称是固定的，而机器学习框架中通常是用户指定，因此在发布模型时需要手动选择

自动搜索作业代码编写规范

自动搜索作业代码编写规范

1.代码入参说明
2.必要接口说明

代码入参说明

自动搜索作业的实现过程：通过搜索算法获取多个超参数组合，每个组合都会通过训练得到一个评估结果，以此最终判断超参数组合的优劣，而用户编写的代码即是用于实现单次训练。

用户需要通过argparse模块接受在平台中填写的信息以及搜索算法反馈的超参数组合。

```
def parse_arg():
    """parse arguments"""
    parser = argparse.ArgumentParser(description='tensorflow2.3 mnist Example')
    parser.add_argument('--train_dir', type=str, default='./train_data',
                        help='input data dir for training (default: ./train_data)')
    parser.add_argument('--test_dir', type=str, default='./test_data',
                        help='input data dir for test (default: ./test_data)')
    parser.add_argument('--output_dir', type=str, default='./output',
                        help='output dir for auto_search job (default: ./output)')
    parser.add_argument('--job_id', type=str, default="job-1234",
                        help='auto_search job id (default: "job-1234")')
    parser.add_argument('--trial_id', type=str, default="0-0",
                        help='auto_search id of a single trial (default: "0-0")')
    parser.add_argument('--metric', type=str, default="acc",
                        help='evaluation metric of the model')
    parser.add_argument('--data_sampling_scale', type=float, default=1.0,
                        help='sampling ratio of the data (default: 1.0)')
    parser.add_argument('--batch_size', type=int, default=100,
                        help='number of images input in an iteration (default: 100)')
    parser.add_argument('--lr', type=float, default=0.001,
                        help='learning rate of the training (default: 0.001)')
    parser.add_argument('--epoch', type=int, default=5,
                        help='number of epochs to train (default: 5)')
    args = parser.parse_args()
    args.output_dir = os.path.join(args.output_dir, args.job_id, args.trial_id)
    if not os.path.exists(args.output_dir):
        os.makedirs(args.output_dir)
    print("job_id: {}, trial_id: {}".format(args.job_id, args.trial_id))
    return args
```

参数说明：

参数	说明
train_dir/test_dir/output_dir	用户在平台中填写的训练集路径、测试集路径、输出路径及其中的文件会同步到代码训练环境中，对应文件名称分别为argparse参数中的default默认值：trian_data/test_data/output，因此请用户切记不要修改这三处的default参数。
job-id	当前自动搜索作业的任务id，系统生成，用于构成输出路径的一部分
trial-id	任务中的每个试验都会拥有独立的试验id，argparse接受该参数用于构成模型输出路径的一部分。注：每个试验的模型保存路径为：output_dir/job-id/trial-id
metric	接受用户在平台上填写的评价指标的名称
data_sampling_scale	接受用户在平台上填写的数据采样比例

其余用户自定义的待搜索的超参数同样需要写在argparse模块中，且名称与平台中填写的完全一致，每个试验运行时搜索算法会将参数值传入代码。

必要接口说明

自动搜索作业给予了用户极高的自由度，但仍然存在一些必须实现的接口，否则无法在平台中完成作业的训练，在编写代码前，请务必仔细阅读下面的几点说明。

- 代码入参说明中提到的argparse模块是必须实现的。
- 模型训练部分的代码，用户可以自定义实现，pytorch框架比较特殊，需要额外实现推理代码，见代码示例(链接到pytorch推理代码处)，其余框

架仅需要实现训练代码即可，平台为用户提供了推理代码用于预测，因此，模型保存的路径与格式，需要统一。

- 模型必须保存到args.output_dir/args.job-id/args.trial-id中，这也是为了方便用户最终选择需要的模型。
- 所有示例代码中main函数均是按照获取参数、加载数据集、模型定义、模型训练、模型保存、模型评估、上报结果的形式进行封装的，每种框架的模型保存方式在代码示例中的模型保存函数已经详细给出，特别注意，用户编写的模型保存函数的名称可以是自定义的，但必须将对应格式的模型保存到对应的路径下！

```
def main(_):
    """main"""
    # 获取参数
    args = parse_arg()
    # 加载数据集
    train_test_data = load_data(args.data_sampling_scale)
    # 模型定义
    model = Model(args, train_test_data)
    # 模型训练
    model.run_train()
    # 模型保存
    model.save_model()
    # 模型评估
    acc = model.evaluate()
    # 上报结果
    report_final(args, metric=acc)
```

- 上报结果的函数实现：每次训练完成后，代码需要将评估结果汇报给搜索算法，用于下一个超参数组合的建议以及是否早停的判断。用户可以通过调用平台提供的SDK接口实现该功能：

```
from rudder_autosearch.sdk.amaas_tools import AMaasTools
```

所有示例均是通过report_final函数实现结果的上报，传入的参数为args以及评估结果的值，函数内部结构建议用户不要做任何修改！上报结果实际上是调用的AMaasTools的report_final_result函数，如下为函数入参说明： *metric：评价指标以字典的形式传入*。 export_model_path：保存模型文件的文件夹路径，2中已经进行了说明 * checkpoint_path：模型的权重，该参数一般设为空字符，只有当搜索算法选择进化算法PBT时，才会使用，该算法每次试验前会用到其余试验的模型权重，因此每次模型训练完需要将权重保存路径进行上传，见4小点。

```
def report_final(args, metric):
    """report_final_result"""
    # 结果上报sdk
    amaas_tools = AMaasTools(args.job_id, args.trial_id)
    metric_dict = {args.metric: metric}
    for i in range(3):
        flag, ret_msg = amaas_tools.report_final_result(metric=metric_dict,
                                                        export_model_path=args.output_
                                                        checkpoint_path="")
        print("End Report, metric:{}, ret_msg:{}".format(metric, ret_msg))
        if flag:
            break
        time.sleep(1)
    assert flag, "Report final result to manager failed! Please check whether manager'addre
                 "is ok! "
```

- 进化算法PBT：该搜索算法专门用于深度学习，详情见搜索算法简介，当采用该算法时，需要在代码中提供接受之前试验权重的接口，代码示例中的pytorch框架就是采用了进化算法PBT的搜索算法，点击这里查看，如下是PBT算法的关键步骤：
  - 首先在argparse模块中需要新增resume_checkpoint_path参数，在训练时，系统会传入之前试验的模型文件路径，如果该试验属于随机初始化的第一个批次，则会传入空字符。用户只需添加该参数，具体传入内容由搜索算法决定。

```
parser.add_argument('--resume_checkpoint_path', type=str, default="",
                    help='inherit the initial weight of the previous trial')
```

  - 接着需要实现模型加载的函数，从传入的模型路径中加载模型，从而实现继承之前试验中的模型权重的功能。

```
def load_state_dict(model, resume_checkpoint_path):
    """load_state_dict"""
    if resume_checkpoint_path:
        model.load_state_dict(torch.load(resume_checkpoint_path))
```

  - 下图为结果上报函数，checkpoint_path处不再是空字符，而是模型文件的保存路径。注意：export_model_path需要传入的是模型所在的文件夹路径，而checkpoint_path是模型文件的路径！

```
def report_final(args, metric):
    """report_final_result"""
    # 结果上报sdk
    amaas_tools = AMaasTools(args.job_id, args.trial_id)
    metric_dict = {args.metric: metric}
    checkpoint_path = os.path.join(args.output_dir, 'model.pkl')
    for i in range(3):
        flag, ret_msg = amaas_tools.report_final_result(metric=metric_dict,
                                                        export_model_path=args.output_dir,
                                                        checkpoint_path=checkpoint_path)
        print("End Report, metric:{}, ret_msg:{}".format(metric, ret_msg))
        if flag:
            break
        time.sleep(1)
    assert flag, "Report final result to manager failed! Please check whether manager'addre
                 "is ok! "
```

yaml文件编写规范

yaml文件编写规范

如下所示是进化算法pbt的yaml文件配置示例：

```
#搜索算法参数
search_strategy:
  algo: PBT_SEARCH #搜索策略：进化算法
  params:
    population_num: 8 #种群个体数量 | [1,10] int类型
    round: 10 #迭代轮数 |[5,50] int类型
    perturb_interval: 10 # 扰动间隔 | [1,20] int类型
    quantile_frac: 0.5 #扰动比例 | (0,0.5] float类型
    explore_prob: 0.25 #扰动概率 | (0,0.5] float类型

#单次训练时数据的采样比例，单位%
data_sampling_scale: 100 #| (0, 100] int类型

#评价指标参数
metrics:
  name: acc #评价指标 | 任意字符串 str类型
  goal: MAXIMIZE #最大值/最小值 | str类型 MAXIMIZE or MINIMIZE 必须为这两个之一（也即支持大写）
  expected_value: 100 #早停标准值，评价指标超过该值则结束整个超参搜索，单位% |无限制 int类型

#搜索参数空间
search_space:
  batch_size:
    htype: choice
    value: [64, 128, 256, 512]
  lr:
    htype: loguniform
    value: [0.0001, 0.1]
  momentum:
    htype: uniform
    value: [0.1, 0.9]
```

yaml文件中主要分为四大部分：

1. 搜索算法参数：

algo：指定选择的搜索策略,下表为各种搜索算法对应的字段。

搜索策略	英文字段
随机搜索	RANDOM_SEARCH
贝叶斯搜索	TPE_SEARCH
进化算法	PBT_SEARCH
进化-CMAES	CMAES_SEARCH
粒子群算法	PARTICLE_SEARCH

params：搜索算法的参数配置，详情见搜索算法简介。

2. 评价指标参数：

- name：评价指标的名称。
- goal：优化方向，可选MAXIMIZE/MINIMIZE。
- expected_value:早停标准值，单位为%。

3. 搜索参数空间：

- 和手动配置一样，需要填写参数的类型htype以及参数取值范围value，下表为参数类型对应的字段：

参数类型	英文字段
随机整数	randint
离散值	choice
平均采样	uniform
对数平均采样	loguniform

离散值的value为所有可能取值的列表，而其余参数类型的value只需要将上下界填入列表即可。

4. 其余参数：除上述主要的三部分外，还需填写的内容如下。

- data_sampling_scale：数据采样比例，单位为%。
- max_trial_num:最大搜索次数，随机搜索、贝叶斯搜索的yaml文件中需要配置该参数，而其余基于种群的进化算法不需要，最大搜索次数由种群个体数量与最大轮次的乘积决定。

自动搜索作业代码示例

TensorFlow 1.13.2代码规范

TensorFlow 1.13.2代码规范 基于TensorFlow1.13.2框架的MNIST图像分类，训练数据集tf_train_data2.zip点击这里下载。
如下所示是其超参搜索任务中一个超参数组合的训练代码，代码会通过argparse模块接受在平台中填写的信息，请保持一致。

tensorflow1.13.2_autosearch.py示例代码

```
 **-*- coding:utf-8 -*-**
""" tensorflow1 train demo """
import os
import tensorflow as tf
import numpy as np
import time
from tensorflow import keras
import os
import argparse
from rudder_autosearch.sdk.amaas_tools import AMaasTools

tf.logging.set_verbosity(tf.logging.INFO)

def parse_arg():
    """parse arguments"""
    parser = argparse.ArgumentParser(description='tensorflow1.13.2 mnist Example')
    parser.add_argument('--train_dir', type=str, default='./train_data',
                        help='input data dir for training (default: ./train_data)')
    parser.add_argument('--test_dir', type=str, default='./test_data',
                        help='input data dir for test (default: ./test_data)')
    parser.add_argument('--output_dir', type=str, default='./output',
                        help='output dir for auto_search job (default: ./output)')
    parser.add_argument('--job_id', type=str, default="job-1234",
                        help='auto_search job id (default: "job-1234")')
    parser.add_argument('--trial_id', type=str, default="0-0",
                        help='auto_search id of a single trial (default: "0-0")')
    parser.add_argument('--metric', type=str, default="acc",
                        help='evaluation metric of the model')
    parser.add_argument('--data_sampling_scale', type=float, default=1.0,
                        help='sampling ratio of the data (default: 1.0)')
    parser.add_argument('--batch_size', type=int, default=100,
                        help='number of images input in an iteration (default: 100)')
    parser.add_argument('--lr', type=float, default=0.001,
```

```
                        help='learning rate of the training (default: 0.001)')
    parser.add_argument('--last_step', type=int, default=20000,
                        help='number of steps to train (default: 20000)')
    args = parser.parse_args()
    args.output_dir = os.path.join(args.output_dir, args.job_id, args.trial_id)
    print("job_id: {}, trial_id: {}".format(args.job_id, args.trial_id))
    return args

def load_data(data_sampling_scale):
    """ load data """
    work_path = os.getcwd()
    (x_train, y_train), (x_test, y_test) = \
        keras.datasets.mnist.load_data('%s/train_data/mnist.npz' % work_path)
    # sample training data
    np.random.seed(0)
    sample_data_num = int(data_sampling_scale * len(x_train))
    idx = np.arange(len(x_train))
    np.random.shuffle(idx)
    x_train, y_train = x_train[0:sample_data_num], y_train[0:sample_data_num]
    # The shape of downloaded data is (-1, 28, 28), hence we need to reshape it
    # into (-1, 784) to feed into our network. Also, need to normalize the
    # features between 0 and 1.
    x_train = np.reshape(x_train, (-1, 784)) / 255.0
    x_test = np.reshape(x_test, (-1, 784)) / 255.0
    return (x_train, x_test), (y_train, y_test)

def train_input_generator(x_train, y_train, batch_size=64):
    """train_input_generator"""
    assert len(x_train) == len(y_train)
    while True:
        p = np.random.permutation(len(x_train))
        x_train, y_train = x_train[p], y_train[p]
        index = 0
        while index <= len(x_train) - batch_size:
            yield x_train[index:index + batch_size], \
                  y_train[index:index + batch_size],
            index += batch_size

def conv_model(feature, target, mode):
    """2-layer convolution model."""
    # Convert the target to a one-hot tensor of shape (batch_size, 10) and
    # with a on-value of 1 for each one-hot vector of length 10.
    target = tf.one_hot(tf.cast(target, tf.int32), 10, 1, 0)
    # Reshape feature to 4d tensor with 2nd and 3rd dimensions being
    # image width and height final dimension being the number of color channels.
    feature = tf.reshape(feature, [-1, 28, 28, 1])
    # First conv layer will compute 32 features for each 5x5 patch
    with tf.variable_scope('conv_layer1'):
        h_conv1 = tf.layers.conv2d(feature, 32, kernel_size=[5, 5],
                                   activation=tf.nn.relu, padding="SAME")
        h_pool1 = tf.nn.max_pool(
            h_conv1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
    # Second conv layer will compute 64 features for each 5x5 patch.
    with tf.variable_scope('conv_layer2'):
        h_conv2 = tf.layers.conv2d(h_pool1, 64, kernel_size=[5, 5],
                                   activation=tf.nn.relu, padding="SAME")
        h_pool2 = tf.nn.max_pool(
            h_conv2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
        # reshape tensor into a batch of vectors
        h_pool2_flat = tf.reshape(h_pool2, [-1, 7 * 7 * 64])
    # Densely connected layer with 1024 neurons.
    h_fc1 = tf.layers.dropout(
        tf.layers.dense(h_pool2_flat, 1024, activation=tf.nn.relu),
        rate=0.5, training=mode == tf.estimator.ModeKeys.TRAIN)

    # Compute logits (1 per class) and compute loss.
    logits = tf.layers.dense(h_fc1, 10, activation=None)
    loss = tf.losses.softmax_cross_entropy(target, logits)
    return tf.argmax(logits, 1), loss

def main():
```

```
class Model():
    def __init__(self, args, train_test_data):
        self.args = args
        self.create_model()
        (self.x_train, self.x_test), (self.y_train, self.y_test) = train_test_data

    def create_model(self):
        """create_model"""
        with tf.name_scope('input'):
            self.image = tf.placeholder(tf.float32, [None, 784], name='image')
            self.label = tf.placeholder(tf.float32, [None], name='label')
        self.predict, self.loss = conv_model(self.image, self.label, tf.estimator.ModeKeys.TRAIN)
        opt = tf.train.RMSPropOptimizer(self.args.lr)
        self.global_step = tf.train.get_or_create_global_step()
        self.train_op = opt.minimize(self.loss, global_step=self.global_step)

    def run_train(self):
        """run_train"""
        hooks = [
            tf.train.StopAtStepHook(last_step=self.args.last_step),
            tf.train.LoggingTensorHook(tensors={'step': self.global_step, 'loss': self.loss},
                                       every_n_iter=10),
        ]
        # Horovod: pin GPU to be used to process local rank (one GPU per process)
        config = tf.ConfigProto()
        config.gpu_options.allow_growth = True
        config.gpu_options.visible_device_list = '0'
        # Horovod: save checkpoints only on worker 0 to prevent other workers from
        # corrupting them.
        self.checkpoint_dir = '/checkpoints'
        os.system("rm -rf " + self.checkpoint_dir)
        training_batch_generator = train_input_generator(self.x_train,
                                                         self.y_train, batch_size=self.args.batch_size)
        # The MonitoredTrainingSession takes care of session initialization,
        # restoring from a checkpoint, saving to a checkpoint, and closing when done
        # or an error occurs.
        with tf.train.MonitoredTrainingSession(checkpoint_dir=self.checkpoint_dir,
                                               hooks=hooks,
                                               config=config) as mon_sess:
            while not mon_sess.should_stop():
                # Run a training step synchronously.
                image_, label_ = next(training_batch_generator)
                mon_sess.run(self.train_op, feed_dict={self.image: image_, self.label: label_})

    def save_model(self):
        """save_model"""
        saver = tf.train.Saver()
        inputs_classes = tf.saved_model.utils.build_tensor_info(self.image)
        outputs_classes = tf.saved_model.utils.build_tensor_info(self.predict)
        signature = (tf.saved_model.signature_def_utils.build_signature_def(
            inputs={tf.saved_model.signature_constants.CLASSIFY_INPUTS: inputs_classes},
            outputs={tf.saved_model.signature_constants.CLASSIFY_OUTPUT_CLASSES: outputs_classes},
            method_name=tf.saved_model.signature_constants.CLASSIFY_METHOD_NAME))

        with tf.Session() as sess:
            sess.run([tf.local_variables_initializer(), tf.tables_initializer()])
            saver.restore(sess, tf.train.latest_checkpoint(self.checkpoint_dir))
            model_output_dir = self.args.output_dir
            builder = tf.saved_model.builder.SavedModelBuilder(model_output_dir)
            legacy_init_op = tf.group(tf.tables_initializer(), name='legacy_init_op')
            builder.add_meta_graph_and_variables(sess, [tf.saved_model.tag_constants.SERVING],
                                                 signature_def_map={'predict_images': signature},
                                                 legacy_init_op=legacy_init_op)
            builder.save()

    def evaluate(self):
        """evaluate"""
        with tf.Session() as sess:
            sess.run([tf.local_variables_initializer(), tf.tables_initializer()])
            saver = tf.train.Saver()
```

```
            saver.restore(sess, tf.train.latest_checkpoint(self.checkpoint_dir))
            y_pred = sess.run(self.predict, feed_dict={self.image: self.x_test})
            self.acc = sum(y_pred == self.y_test) / len(y_pred)
        print("accuracy: %f" % self.acc)
        return self.acc

def report_final(args, metric):
    """report_final_result"""
    # 结果上报sdk
    amaas_tools = AMaasTools(args.job_id, args.trial_id)
    metric_dict = {args.metric: metric}
    for i in range(3):
        flag, ret_msg = amaas_tools.report_final_result(metric=metric_dict,
                                                        export_model_path=args.output_dir,
                                                        checkpoint_path="")
        print("End Report, metric:{}, ret_msg:{}".format(metric, ret_msg))
        if flag:
            break
        time.sleep(1)
    assert flag, "Report final result to manager failed! Please check whether manager'address or manager'status " \
                 "is ok! "

def main(_):
    """main"""
    # 获取参数
    args = parse_arg()
    # 加载数据集
    train_test_data = load_data(args.data_sampling_scale)
    # 模型定义
    model = Model(args, train_test_data)
    # 模型训练
    model.run_train()
    # 模型保存
    model.save_model()
    # 模型评估
    acc = model.evaluate()
    # 上报结果
    report_final(args, metric=acc)

if __name__ == "__main__":
    tf.app.run()
```

示例代码对应的yaml配置如下，请保持格式一致

tpe_search_demo.yml示例内容

```
 **搜索算法参数**
search_strategy:
  algo: TPE_SEARCH #搜索策略：贝叶斯搜索
  params:
    n_startup_points: 5 # 初始点数量   | [1,20] int类型
    max_concurrent: 5 #最大并发量     | [1,20] int类型

**单次训练时数据的采样比例，单位%**
data_sampling_scale: 100  # |  （0，100] int类型

**最大搜索次数**
max_trial_num: 10  #  | >0 int类型

**评价指标参数**
metrics:
  name: acc #评价指标   |  任意字符串 str类型
  goal: MAXIMIZE #最大值/最小值  |  str类型   MAXIMIZE or MINIMIZE   必须为这两个之一（也即支持大写）
  expected_value: 100 #早停标准值，评价指标超过该值则结束整个超参搜索，单位%   | 无限制 int类型

**搜索参数空间**
search_space:
  batch_size:
    htype: choice
    value: [100, 200, 300, 400, 500, 600]
  lr:
    htype: loguniform
    value: [0.0001, 0.1]
  last_step:
    htype: choice
    value: [20000, 50000, 100000]
```

Sklearn 0.23.2代码规范

Sklearn 0.23.2代码规范 基于Sklearn 0.23.2框架的结构化数据的多分类问题，训练数据集sklearn_train_data.zip点击这里下载。
如下所示是其超参搜索任务中一个超参数组合的训练代码，代码会通过argparse模块接受在平台中填写的信息，请保持一致。另外该框架支持发布保存模型为pickle和joblib格式，并且在发布至模型仓库时需要选择相应的模型文件。

sklearn0.23.2_autosearch.py示例代码

```
 **-*- coding:utf-8 -*-**
""" sklearn train demo """
import os
import argparse
import time
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn import svm
import pandas as pd
import numpy as np
from rudder_autosearch.sdk.amaas_tools import AMaasTools

def parse_arg():
    """parse arguments"""
    parser = argparse.ArgumentParser(description='Sklearn iris Example')
    parser.add_argument('--train_dir', type=str, default='./train_data',
                        help='input data dir for training (default: ./train_data)')
    parser.add_argument('--test_dir', type=str, default='./test_data',
                        help='input data dir for test (default: ./test_data)')
    parser.add_argument('--output_dir', type=str, default='./output',
                        help='output dir for auto_search job (default: ./output)')
    parser.add_argument('--job_id', type=str, default="job-1234",
                        help='auto_search job id')
    parser.add_argument('--trial_id', type=str, default="0-0",
                        help='auto_search id of a single trial')
    parser.add_argument('--metric', type=str, default="f1_score",
                        help='evaluation metric of the model')
    parser.add_argument('--data_sampling_scale', type=float, default=1.0,
                        help='sampling ratio of the dataset for auto_search (default: 1.0)')
```

```
    parser.add_argument('--kernel', type=str, default='linear',
                        help='kernel function (default: "linear")')
    parser.add_argument('--C', type=float, default=1,
                        help='penalty term (default: 1)')
    parser.add_argument('--gamma', type=float, default=0.5,
                        help='parameter of the kernel (default: 0.5)')

    args = parser.parse_args()
    args.output_dir = os.path.join(args.output_dir, args.job_id, args.trial_id)
    if not os.path.exists(args.output_dir):
        os.makedirs(args.output_dir)
    print("job_id: {}, trial_id: {}".format(args.job_id, args.trial_id))
    return args

def load_data(train_dir, data_sampling_scale):
    """ load data """
    # 共150条数据，训练120条，测试30条，进行2,8分进行模型训练
    # 每条数据类型为 x{nbarray} [6.4, 3.1, 5.5, 1.8]
    # 上传的数据储存在./train_data和./test_data中
    inputdata = pd.read_csv(train_dir + "/iris.csv")
    target = inputdata["Species"]
    inputdata = inputdata.drop(columns=["Species"])
    # 切分，测试训练2,8分
    x_train, x_test, y_train, y_test = train_test_split(inputdata, target, test_size=0.2, random_state=0)
    train_data = np.concatenate([x_train, y_train.ravel().reshape([-1, 1])], axis=1)
    np.random.seed(0)
    np.random.shuffle(train_data)
    train_data = train_data[0:int(data_sampling_scale * len(train_data))]
    x_train, y_train = train_data[:, 0:-1], train_data[:, -1]
    return (x_train, x_test), (y_train, y_test)

def save_model(model, output_dir):
    """ save model with pickle format """
    import pickle
    with open(output_dir + '/clf.pickle', 'wb') as f:
        pickle.dump(model, f)

def save_model_joblib(model, output_dir):
    """ save model with joblib format """
    try:
        import joblib
    except:
        from sklearn.externals import joblib
    joblib.dump(model, output_dir + '/clf.pkl')

def evaluate(model, x_test, y_test):
    """evaluate"""
    # 多分类f1_score指标
    predict = model.predict(x_test)
    f1 = f1_score(y_test, predict, average="micro")
    print("f1_score: %f" % f1)
    return f1

def report_final(args, metric):
    """report_final_result"""
    # 结果上报sdk
    amaas_tools = AMaasTools(args.job_id, args.trial_id)
    metric_dict = {args.metric: metric}
    for i in range(3):
        flag, ret_msg = amaas_tools.report_final_result(metric=metric_dict,
                                                        export_model_path=args.output_dir,
                                                        checkpoint_path="")
        print("End Report, metric:{}, ret_msg:{}".format(metric, ret_msg))
        if flag:
            break
        time.sleep(1)
    assert flag, "Report final result to manager failed! Please check whether manager'address or manager'status " \
                 "is ok! "

def main():
```

```
    """ main """
    # 获取参数
    args = parse_arg()
    # 加载数据集
    (x_train, x_test), (y_train, y_test) = load_data(args.train_dir, args.data_sampling_scale)
    # 模型定义
    model = svm.SVC(C=args.C, kernel=args.kernel, gamma=args.gamma)
    # 模型训练
    model.fit(x_train, y_train)
    # 模型保存
    save_model(model, args.output_dir)
    # 模型评估
    f1 = evaluate(model, x_test, y_test)
    # 上报结果
    report_final(args, metric=f1)

if __name__ == '__main__':
    main()
```

示例代码对应的yaml配置如下，请保持格式一致

random_search_demo.yml示例内容

```
  **搜索算法参数**
search_strategy:
  algo: RANDOM_SEARCH #搜索策略：随机搜索

**单次训练时数据的采样比例，单位%**
data_sampling_scale: 100 # | （0，100] int类型

**最大搜索次数**
max_trial_num: 10 #  | >0 int类型

**评价指标参数**
metrics:
  name: f1_score #评价指标  |  任意字符串 str类型
  goal: MAXIMIZE #最大值/最小值  |  str类型  MAXIMIZE or MINIMIZE  必须为这两个之一（也即支持大写）
  expected_value: 100 #早停标准值，评价指标超过该值则结束整个超参搜索，单位%  | 无限制 int类型

**搜索参数空间**
search_space:
  kernel: #核函数
    htype: choice
    value: ["linear", "rbf"]
  C: #惩罚项
    htype: loguniform
    value: [0.001, 1000]
  gamma: #核函数参数
    htype: loguniform
    value: [0.0001, 1]
```

Pytorch 1.7.1代码规范

Pytorch 1.7.1代码规范 基于Pytorch 1.7.1框架的MNIST图像分类，训练数据集pytorch_train_data.zip点击这里下载。
如下所示是其超参搜索任务中一个超参数组合的训练代码，代码会通过argparse模块接受在平台中填写的信息，请保持一致。
特别注意，示例采用的是进化算法进行超参搜索，每个试验在训练时会继承之前试验的权重，resume_checkpoint_path是权重的保存路径，由搜索算法自身提供，与job_id及trial_id一样，只需要在argparse中提供对应参数即可。

pytorch1.7.1_autosearch.py示例代码

```
  **-*- coding:utf-8 -*-**
""" pytorch train demo """
import argparse
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torch.utils.data as data
```

```
from torchvision import transforms
import codecs
import errno
import gzip
import numpy as np
import os
import time
from PIL import Image
from rudder_autosearch.sdk.amaas_tools import AMaasTools

def parse_arg():
    """parse arguments"""
    parser = argparse.ArgumentParser(description='PyTorch1.7.1 MNIST Example')
    parser.add_argument('--train_dir', type=str, default='./train_data',
                        help='input data dir for training (default: ./train_data)')
    parser.add_argument('--test_dir', type=str, default='./test_data',
                        help='input data dir for test (default: ./test_data)')
    parser.add_argument('--output_dir', type=str, default='./output',
                        help='output dir for auto_search job (default: ./output)')
    parser.add_argument('--job_id', type=str, default="job-1234",
                        help='auto_search job id (default: "job-1234")')
    parser.add_argument('--trial_id', type=str, default="0-0",
                        help='auto_search id of a single trial (default: "0-0")')
    parser.add_argument('--metric', type=str, default="acc",
                        help='evaluation metric of the model')
    parser.add_argument('--data_sampling_scale', type=float, default=1.0,
                        help='sampling ratio of the data (default: 1.0)')
    parser.add_argument('--batch_size', type=int, default=64,
                        help='number of images input in an iteration (default: 64)')
    parser.add_argument('--lr', type=float, default=0.01,
                        help='learning rate (default: 0.01)')
    parser.add_argument('--momentum', type=float, default=0.5,
                        help='SGD momentum (default: 0.5)')
    parser.add_argument('--no_cuda', action='store_true', default=False,
                        help='disables CUDA training')
    parser.add_argument('--log_interval', type=int, default=10,
                        help='how many batches to wait before logging training status')
    parser.add_argument('--perturb_interval', type=int, default=10,
                        help='number of epochs to train (default: 10)')
    parser.add_argument('--resume_checkpoint_path', type=str, default="",
                        help='inherit the initial weight of the previous trial')
    args = parser.parse_args()
    args.output_dir = os.path.join(args.output_dir, args.job_id, args.trial_id)
    if not os.path.exists(args.output_dir):
        os.makedirs(args.output_dir)
    args.cuda = not args.no_cuda and torch.cuda.is_available()


    print("job_id: {}, trial_id: {}".format(args.job_id, args.trial_id))
    return args

**定义MNIST数据集的dataset**
class MNIST(data.Dataset):
    """
    MNIST dataset
    """
    training_file = 'training.pt'
    test_file = 'test.pt'
    classes = ['0 - zero', '1 - one', '2 - two', '3 - three', '4 - four',
               '5 - five', '6 - six', '7 - seven', '8 - eight', '9 - nine']

    def __init__(self, root, train=True, transform=None, target_transform=None, data_sampling_scale=1):
        self.root = os.path.expanduser(root)
        self.transform = transform
        self.target_transform = target_transform
        self.train = train  # training set or test set
        self.data_sampling_scale = data_sampling_scale
        self.preprocess(root, train, False)
        if self.train:
            data_file = self.training_file
```

```
        else:
            data_file = self.test_file
        self.data, self.targets = torch.load(os.path.join(self.processed_folder, data_file))

    def __getitem__(self, index):
        """
        Args:
            index (int): Index
        Returns:
            tuple: (image, target) where target is index of the target class.
        """
        img, target = self.data[index], int(self.targets[index])
        # doing this so that it is consistent with all other datasets
        # to return a PIL Image
        img = Image.fromarray(img.numpy(), mode='L')
        if self.transform is not None:
            img = self.transform(img)
        if self.target_transform is not None:
            target = self.target_transform(target)

        return img, target

    def __len__(self):
        return len(self.data)

    @property
    def raw_folder(self):
        """
        raw folder
        """
        return os.path.join('/tmp', 'raw')

    @property
    def processed_folder(self):
        """
        processed folder
        """
        return os.path.join('/tmp', 'processed')

    # data preprocessing
    def preprocess(self, train_dir, train, remove_finished=False):
        """
        preprocess
        """
        makedir_exist_ok(self.raw_folder)
        makedir_exist_ok(self.processed_folder)
        train_list = ['train-images-idx3-ubyte.gz', 'train-labels-idx1-ubyte.gz']
        test_list = ['t10k-images-idx3-ubyte.gz', 't10k-labels-idx1-ubyte.gz']
        zip_list = train_list if train else test_list
        for zip_file in zip_list:
            print('Extracting {}'.format(zip_file))
            zip_file_path = os.path.join(train_dir, zip_file)
            raw_folder_path = os.path.join(self.raw_folder, zip_file)
            with open(raw_folder_path.replace('.gz', ''), 'wb') as out_f, gzip.GzipFile(zip_file_path) as zip_f:
                out_f.write(zip_f.read())
            if remove_finished:
                os.unlink(zip_file_path)
        if train:
            x_train = read_image_file(os.path.join(self.raw_folder, 'train-images-idx3-ubyte'))
            y_train = read_label_file(os.path.join(self.raw_folder, 'train-labels-idx1-ubyte'))
            np.random.seed(0)
            sample_data_num = int(self.data_sampling_scale * len(x_train))
            idx = np.arange(len(x_train))
            np.random.shuffle(idx)
            x_train, y_train = x_train[0:sample_data_num], y_train[0:sample_data_num]
            training_set = (x_train, y_train)
            with open(os.path.join(self.processed_folder, self.training_file), 'wb') as f:
                torch.save(training_set, f)
        else:
            test_set = (
```

```
            read_image_file(os.path.join(self.raw_folder, 't10k-images-idx3-ubyte')),
            read_label_file(os.path.join(self.raw_folder, 't10k-labels-idx1-ubyte'))
        )
        with open(os.path.join(self.processed_folder, self.test_file), 'wb') as f:
            torch.save(test_set, f)

def get_int(b):
    """
    get int
    """
    return int(codecs.encode(b, 'hex'), 16)

def read_label_file(path):
    """
    read label file
    """
    with open(path, 'rb') as f:
        data = f.read()
        assert get_int(data[:4]) == 2049
        length = get_int(data[4:8])
        parsed = np.frombuffer(data, dtype=np.uint8, offset=8)
        return torch.from_numpy(parsed).view(length).long()

def read_image_file(path):
    """
    read image file
    """
    with open(path, 'rb') as f:
        data = f.read()
        assert get_int(data[:4]) == 2051
        length = get_int(data[4:8])
        num_rows = get_int(data[8:12])
        num_cols = get_int(data[12:16])
        parsed = np.frombuffer(data, dtype=np.uint8, offset=16)
        return torch.from_numpy(parsed).view(length, num_rows, num_cols)

def makedir_exist_ok(dirpath):
    """
    Python2 support for os.makedirs(.., exist_ok=True)
    """
    try:
        os.makedirs(dirpath)
    except OSError as e:
        if e.errno == errno.EEXIST:
            pass
        else:
            raise

def load_data(args):
    """load_data"""
    # 若无测试集，训练集做验证集
    if not os.path.exists(args.test_dir) or not os.listdir(args.test_dir):
        args.test_dir = args.train_dir
    # 将数据进行转化，从PIL.Image/numpy.ndarray的数据进转化为torch.FloadTensor
    trans = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.1307,), (0.3081,))])
    train_set = MNIST(root=args.train_dir, train=True, transform=trans, data_sampling_scale=args.data_sampling_scale)
    test_set = MNIST(root=args.test_dir, train=False, transform=trans)
    # 定义data reader
    train_loader = torch.utils.data.DataLoader(
        dataset=train_set,
        batch_size=args.batch_size,
        shuffle=True)
    test_loader = torch.utils.data.DataLoader(
        dataset=test_set,
        batch_size=args.batch_size,
        shuffle=False)
    return train_loader, test_loader

**定义网络模型**
class Net(nn.Module):
```

```
    """
    Net
    """
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 10, kernel_size=5)
        self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)

    def forward(self, x):
        """
        forward
        """
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
        x = x.view(-1, 320)
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        x = self.fc2(x)
        return F.log_softmax(x)

def load_state_dict(model, resume_checkpoint_path):
    """load_state_dict"""
    if resume_checkpoint_path:
        model.load_state_dict(torch.load(resume_checkpoint_path))

def run_train(model, args, train_loader):
    """run_train"""
    if args.cuda:
        # Move model to GPU.
        model.cuda()
    # 选择优化器
    optimizer = optim.SGD(model.parameters(), lr=args.lr,
                          momentum=args.momentum)
    for epoch in range(1, args.perturb_interval + 1):
        train(model, args, train_loader, optimizer, epoch)

def train(model, args, train_loader, optimizer, epoch):
    """
    train
    """
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        if args.cuda:
            data, target = data.cuda(), target.cuda()
        optimizer.zero_grad()
        output = model(data)  # 获取预测值
        loss = F.nll_loss(output, target)  # 计算loss
        loss.backward()
        optimizer.step()
        if batch_idx % args.log_interval == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx, len(train_loader),
                100. * batch_idx / len(train_loader), loss.item()))

def evaluate(model, args, test_loader):
    """evaluate"""
    model.eval()
    test_loss = 0.
    test_accuracy = 0.
    for data, target in test_loader:
        if args.cuda:
            data, target = data.cuda(), target.cuda()
        output = model(data)
        # sum up batch loss
        test_loss += F.nll_loss(output, target, size_average=False).item()
        # get the index of the max log-probability
        pred = output.data.max(1, keepdim=True)[1]
```

```
            test_accuracy += pred.eq(target.data.view_as(pred)).cpu().float().sum()
        test_loss /= len(test_loader) * args.batch_size
        test_accuracy /= len(test_loader) * args.batch_size
        print('\nTest set: Average loss: {:.4f}, Accuracy: {:.2f}%\n'.format(
            test_loss, 100. * test_accuracy))
        return float(test_accuracy)

    def save(model, output_dir):
        """
        save
        """
        if not os.path.exists(output_dir):
            os.makedirs(output_dir)
        # 保存模型
        torch.save(model.state_dict(), os.path.join(output_dir, 'model.pkl'))

    def report_final(args, metric):
        """report_final_result"""
        # 结果上报sdk
        amaas_tools = AMaasTools(args.job_id, args.trial_id)
        metric_dict = {args.metric: metric}
        checkpoint_path = os.path.join(args.output_dir, 'model.pkl')
        for i in range(3):
            flag, ret_msg = amaas_tools.report_final_result(metric=metric_dict,
                                                            export_model_path=args.output_dir,
                                                            checkpoint_path=checkpoint_path)
            print("End Report, metric:{}, ret_msg:{}".format(metric, ret_msg))
            if flag:
                break
            time.sleep(1)
        assert flag, "Report final result to manager failed! Please check whether manager'address or manager'status " \
                     "is ok! "

    def main():
        """main"""
        # 获取参数
        args = parse_arg()
        # 加载数据集
        train_loader, test_loader = load_data(args)
        # 模型定义
        model = Net()
        # 继承之前实验的模型参数
        load_state_dict(model, args.resume_checkpoint_path)
        # 模型训练
        run_train(model, args, train_loader)
        # 模型保存
        save(model, args.output_dir)
        # 模型评估
        acc = evaluate(model, args, test_loader)
        # 上报结果
        report_final(args, metric=acc)

    if __name__ == '__main__':
        main()
```

示例代码对应的yaml配置如下，请保持格式一致

pbt_search_demo.yml示例内容

```
  **搜索算法参数**
search_strategy:
  algo: PBT_SEARCH #搜索策略：进化算法
  params:
    population_num: 8 #种群个体数量 | [1,10] int类型
    round: 10 #迭代轮数 | [5,50] int类型
    perturb_interval: 10 # 扰动间隔 | [1,20] int类型
    quantile_frac: 0.5 #扰动比例 | (0,0.5] float类型
    explore_prob: 0.25 #扰动概率 | (0,0.5] float类型

**单次训练时数据的采样比例，单位%**
data_sampling_scale: 100 # | （0，100] int类型

**评价指标参数**
metrics:
  name: acc #评价指标 | 任意字符串 str类型
  goal: MAXIMIZE #最大值/最小值 | str类型 MAXIMIZE or MINIMIZE 必须为这两个之一（也即支持大写）
  expected_value: 100 #早停标准值，评价指标超过该值则结束整个超参搜索，单位% | 无限制 int类型

**搜索参数空间**
search_space:
  batch_size:
    htype: choice
    value: [64, 128, 256, 512]
  lr:
    htype: loguniform
    value: [0.0001, 0.1]
  momentum:
    htype: uniform
    value: [0.1, 0.9]
```

pytorch_predict.py示例代码

```
  **!/usr/bin/env python**
**-*- coding: utf-8 -*-**
"""
@license: Copyright (c) 2019 Baidu.com, Inc. All Rights Reserved.
@desc: 图像预测算法示例
"""
import logging
import torch
import torch.nn as nn
import torch.nn.functional as F
import base64
import json
from PIL import Image
from io import BytesIO
from torchvision import datasets, models, transforms
MODEL_FILE_NAME = 'model.pkl'  # 模型文件名称
def get_image_transform():
    """获取图片处理的transform
    Args:
        data_type: string, type of data(train/test)
    Returns:
        torchvision.transforms.Compose
    """
    trans = transforms.Compose([transforms.Resize((28, 28)),
                                transforms.ToTensor(),
                                transforms.Normalize((0.5,), (1.0,))])
    return trans
def model_fn(model_dir):
    """模型加载
    Args:
        model_dir: 模型路径，该目录存储的文件为在自动搜索作业中选择的输出路径下产出的文件
    Returns:
        加载好的模型对象
    """
    class Net(nn.Module):
        """Net"""
```

```
        Net
        def __init__(self):
            super(Net, self).__init__()
            self.conv1 = nn.Conv2d(1, 10, kernel_size=5)
            self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
            self.conv2_drop = nn.Dropout2d()
            self.fc1 = nn.Linear(320, 50)
            self.fc2 = nn.Linear(50, 10)
        def forward(self, x):
            """
            forward
            """
            x = F.relu(F.max_pool2d(self.conv1(x), 2))
            x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
            x = x.view(-1, 320)
            x = F.relu(self.fc1(x))
            x = F.dropout(x, training=self.training)
            x = self.fc2(x)
            return F.log_softmax(x)
    model = Net()
    meta_info_path = "%s/%s" % (model_dir, MODEL_FILE_NAME)
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model.load_state_dict(torch.load(meta_info_path, map_location=device))
    model.to(device)
    logging.info("device type: " + str(device))
    return model
def input_fn(request):
    """对输入进行格式化，处理为预测需要的输入格式
    Args:
        request: api请求的json
    Returns:
        预测需要的输入数据，一般为tensor
    """
    instances = request['instances']
    transform_composes = get_image_transform()
    arr_tensor_data = []
    for instance in instances:
        decoded_data = base64.b64decode(instance['data'].encode("utf8"))
        byte_stream = BytesIO(decoded_data)
        roiImg = Image.open(byte_stream)
        target_data = transform_composes(roiImg)
        arr_tensor_data.append(target_data)
    tensor_data = torch.stack(arr_tensor_data, dim=0)
    return tensor_data
def output_fn(predict_result):
    """进行输出格式化
    Args:
        predict_result: 预测结果
    Returns:
        格式化后的预测结果，需能够json序列化以便接口返回
    """
    js_str = None
    if type(predict_result) == torch.Tensor:
        list_prediction = predict_result.detach().cpu().numpy().tolist()
        js_str = json.dumps(list_prediction)
    return js_str
```

Tensorflow2.3.0代码规范

Tensorflow 2.3.0代码规范 基于Tensorflow2.3.0框架的MNIST图像分类，训练数据集tf_train_data2.zip点击这里下载。

如下所示是其超参搜索任务中一个超参数组合的训练代码，代码会通过argparse模块接受在平台中填写的信息，请保持一致。

tensorflow2.3_autosearch.py示例代码

```
 **-*- coding:utf-8 -*-**
""" tensorflow2 train demo """
import tensorflow as tf
import os
import numpy as np
import time
```

```
import time
import argparse
from rudder_autosearch.sdk.amaas_tools import AMaasTools

def parse_arg():
    """parse arguments"""
    parser = argparse.ArgumentParser(description='tensorflow2.3 mnist Example')
    parser.add_argument('--train_dir', type=str, default='./train_data',
                        help='input data dir for training (default: ./train_data)')
    parser.add_argument('--test_dir', type=str, default='./test_data',
                        help='input data dir for test (default: ./test_data)')
    parser.add_argument('--output_dir', type=str, default='./output',
                        help='output dir for auto_search job (default: ./output)')
    parser.add_argument('--job_id', type=str, default="job-1234",
                        help='auto_search job id (default: "job-1234")')
    parser.add_argument('--trial_id', type=str, default="0-0",
                        help='auto_search id of a single trial (default: "0-0")')
    parser.add_argument('--metric', type=str, default="acc",
                        help='evaluation metric of the model')
    parser.add_argument('--data_sampling_scale', type=float, default=1.0,
                        help='sampling ratio of the data (default: 1.0)')
    parser.add_argument('--batch_size', type=int, default=100,
                        help='number of images input in an iteration (default: 100)')
    parser.add_argument('--lr', type=float, default=0.001,
                        help='learning rate of the training (default: 0.001)')
    parser.add_argument('--epoch', type=int, default=5,
                        help='number of epochs to train (default: 5)')
    args = parser.parse_args()
    args.output_dir = os.path.join(args.output_dir, args.job_id, args.trial_id)
    if not os.path.exists(args.output_dir):
        os.makedirs(args.output_dir)
    print("job_id: {}, trial_id: {}".format(args.job_id, args.trial_id))
    return args

def load_data(data_sampling_scale):
    """ load data """
    mnist = tf.keras.datasets.mnist
    work_path = os.getcwd()
    (x_train, y_train), (x_test, y_test) = mnist.load_data('%s/train_data/mnist.npz' % work_path)
    # sample training data
    np.random.seed(0)
    sample_data_num = int(data_sampling_scale * len(x_train))
    idx = np.arange(len(x_train))
    np.random.shuffle(idx)
    x_train, y_train = x_train[0:sample_data_num], y_train[0:sample_data_num]
    x_train, x_test = x_train / 255.0, x_test / 255.0
    return (x_train, x_test), (y_train, y_test)

def Model(learning_rate):
    """Model"""
    model = tf.keras.models.Sequential([
        tf.keras.layers.Flatten(input_shape=(28, 28)),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(10, activation='softmax')
    ])
    model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model

def evaluate(model, x_test, y_test):
    """evaluate"""
    loss, acc = model.evaluate(x_test, y_test, verbose=2)
    print("accuracy: %f" % acc)
    return acc

def report_final(args, metric):
    """report_final_result"""
    # 结果上报sdk
    amaas_tools = AMaasTools(args.job_id, args.trial_id)
```

```
    amaas_tools = AmaasTools(args.job_id, args.trial_id)
    metric_dict = {args.metric: metric}
    for i in range(3):
        flag, ret_msg = amaas_tools.report_final_result(metric=metric_dict,
                                                        export_model_path=args.output_dir,
                                                        checkpoint_path="")
        print("End Report, metric:{}, ret_msg:{}".format(metric, ret_msg))
        if flag:
            break
        time.sleep(1)
    assert flag, "Report final result to manager failed! Please check whether manager'address or manager'status " \
                 "is ok! "

def main():
    """main"""
    # 获取参数
    args = parse_arg()
    # 加载数据集
    (x_train, x_test), (y_train, y_test) = load_data(args.data_sampling_scale)
    # 模型定义
    model = Model(args.lr)
    # 模型训练
    model.fit(x_train, y_train, epochs=args.epoch, batch_size=args.batch_size)
    # 模型保存
    model.save(args.output_dir)
    # 模型评估
    acc = evaluate(model, x_test, y_test)
    # 上报结果
    report_final(args, metric=acc)

if __name__ == '__main__':
    main()
```

示例代码对应的yaml配置如下，请保持格式一致

pwo_search_demo.yml示例内容

```
  **搜索算法参数**
search_strategy:
  algo: PARTICLE_SEARCH #搜索策略：粒子群算法
  params:
    population_num: 8 #种群个体数量 | [1,10] int类型
    round: 10 #迭代轮数 | [5,50] int类型
    inertia_weight: 0.5 # 惯性权重 | (0，1] float类型
    global_acceleration: 1.5 #全局加速度 | (0,4] float类型
    local_acceleration: 1.5 #个体加速度 | (0,4] float类型

**单次训练时数据的采样比例，单位%**
data_sampling_scale: 100 # | （0，100] int类型

**评价指标参数**
metrics:
  name: acc #评价指标 | 任意字符串 str类型
  goal: MAXIMIZE #最大值/最小值 | str类型 MAXIMIZE or MINIMIZE 必须为这两个之一（也即支持大写）
  expected_value: 100 #早停标准值，评价指标超过该值则结束整个超参搜索，单位% | 无限制 int类型

**搜索参数空间**
search_space:
  batch_size:
    htype: choice
    value: [100, 200, 300, 400, 500, 600]
  lr:
    htype: loguniform
    value: [0.0001, 0.9]
  epoch:
    htype: choice
    value: [5, 10, 12]
```

PaddlePaddle 2.1.1代码规范

PaddlePaddle 2.1.1代码规范 基于PaddlePaddle2.1.1框架的MNIST图像分类，训练数据集paddle_train_data.zip点击这里下载。
如下所示是其超参搜索任务中一个超参数组合的训练代码，代码会通过argparse模块接受在平台中填写的信息，请保持一致。

PaddlePaddle2.1.1_autosearch.py示例代码

```
 **-*- coding:utf-8 -*-**
""" paddle train demo """
import os
import numpy as np
import paddle # 导入paddle模块
import paddle.fluid as fluid
import gzip
import struct
import argparse
import time
from rudder_autosearch.sdk.amaas_tools import AMaasTools

def parse_arg():
    """parse arguments"""
    parser = argparse.ArgumentParser(description='paddle2.1.1 mnist Example')
    parser.add_argument('--train_dir', type=str, default='./train_data',
                        help='input data dir for training (default: ./train_data)')
    parser.add_argument('--test_dir', type=str, default='./test_data',
                        help='input data dir for test (default: ./test_data)')
    parser.add_argument('--output_dir', type=str, default='./output',
                        help='output dir for auto_search job (default: ./output)')
    parser.add_argument('--job_id', type=str, default="job-1234",
                        help='auto_search job id (default: "job-1234")')
    parser.add_argument('--trial_id', type=str, default="0-0",
                        help='auto_search id of a single trial (default: "0-0")')
    parser.add_argument('--metric', type=str, default="acc",
                        help='evaluation metric of the model')
    parser.add_argument('--data_sampling_scale', type=float, default=1.0,
                        help='sampling ratio of the data (default: 1.0)')
    parser.add_argument('--batch_size', type=int, default=64,
                        help='number of images input in an iteration (default: 64)')
    parser.add_argument('--lr', type=float, default=0.001,
                        help='learning rate of the training (default: 0.001)')
    parser.add_argument('--epoch', type=int, default=5,
                        help='number of epochs to train (default: 5)')
    args = parser.parse_args()
    args.output_dir = os.path.join(args.output_dir, args.job_id, args.trial_id)
    if not os.path.exists(args.output_dir):
        os.makedirs(args.output_dir)
    print("job_id: {}, trial_id: {}".format(args.job_id, args.trial_id))
    return args

def load_data(file_dir, is_train=True):
    """
    :param file_dir:
    :param is_train:
    :return:
    """
    if is_train:
        image_path = file_dir + '/train-images-idx3-ubyte.gz'
        label_path = file_dir + '/train-labels-idx1-ubyte.gz'
    else:
        image_path = file_dir + '/t10k-images-idx3-ubyte.gz'
        label_path = file_dir + '/t10k-labels-idx1-ubyte.gz'
    with open(image_path.replace('.gz', ''), 'wb') as out_f, gzip.GzipFile(image_path) as zip_f:
        out_f.write(zip_f.read())
        # os.unlink(image_path)
    with open(label_path.replace('.gz', ''), 'wb') as out_f, gzip.GzipFile(label_path) as zip_f:
        out_f.write(zip_f.read())
        # os.unlink(label_path)
    with open(label_path[:-3], 'rb') as lbpath:
        magic, n = struct.unpack('>II', lbpath.read(8))
        labels = np.fromfile(lbpath, dtype=np.uint8)
    with open(image_path[:-3], 'rb') as imgpath:
```

```
        magic, num, rows, cols = struct.unpack('>IIII', imgpath.read(16))
        images = np.fromfile(imgpath, dtype=np.uint8).reshape(len(labels), 784)
    return images, labels

def reader_creator(file_dir, is_train=True, buffer_size=100, data_sampling_scale=1):
    """
    :param file_dir:
    :param is_train:
    :param buffer_size:
    :return:
    """
    images, labels = load_data(file_dir, is_train)
    if is_train:
        np.random.seed(0)
        sample_data_num = int(data_sampling_scale * len(images))
        idx = np.arange(len(images))
        np.random.shuffle(idx)
        images, labels = images[0:sample_data_num], labels[0:sample_data_num]
    def reader():
        """
        :return:
        """
        for num in range(int(len(labels) / buffer_size)):
            for i in range(buffer_size):
                yield images[num * buffer_size + i, :], int(labels[num * buffer_size + i])
    return reader

def reader_load(args):
    """reader_load"""
    # 每次读取训练集中的500个数据并随机打乱，传入batched reader中，batched reader 每次 yield args.batch_size个数据
    train_reader = paddle.batch(
        paddle.reader.shuffle(
            reader_creator(args.train_dir, is_train=True, buffer_size=100,
                           data_sampling_scale=args.data_sampling_scale), buf_size=500),
        batch_size=args.batch_size)
    # 读取测试集的数据，每次 yield 64个数据
    test_reader = paddle.batch(
        reader_creator(args.test_dir, is_train=False, buffer_size=100), batch_size=args.batch_size)
    return train_reader, test_reader

def softmax_regression():
    """
    定义softmax分类器：
        一个以softmax为激活函数的全连接层
    Return:
        predict_image -- 分类的结果
    """
    # 输入的原始图像数据，大小为28*28*1
    img = fluid.layers.data(name='img', shape=[1, 28, 28], dtype='float32')
    # 以softmax为激活函数的全连接层，输出层的大小必须为数字的个数10
    predict = fluid.layers.fc(
        input=img, size=10, act='softmax')
    return predict

def multilayer_perceptron():
    """
    定义多层感知机分类器：
        含有两个隐藏层（全连接层）的多层感知器
        其中前两个隐藏层的激活函数采用 ReLU，输出层的激活函数用 Softmax
    Return:
        predict_image -- 分类的结果
    """
    # 输入的原始图像数据，大小为28*28*1
    img = fluid.layers.data(name='img', shape=[1, 28, 28], dtype='float32')
    # 第一个全连接层，激活函数为ReLU
    hidden = fluid.layers.fc(input=img, size=200, act='relu')
    # 第二个全连接层，激活函数为ReLU
    hidden = fluid.layers.fc(input=hidden, size=200, act='relu')
    # 以softmax为激活函数的全连接输出层，输出层的大小必须为数字的个数10
    prediction = fluid.layers.fc(input=hidden, size=10, act='softmax')
```

```
    return prediction

def convolutional_neural_network():
    """
    定义卷积神经网络分类器：
        输入的二维图像，经过两个卷积-池化层，使用以softmax为激活函数的全连接层作为输出层
    Return:
        predict -- 分类的结果
    """
    # 输入的原始图像数据，大小为28*28*1
    img = fluid.layers.data(name='img', shape=[1, 28, 28], dtype='float32')
    # 第一个卷积-池化层
    # 使用20个5*5的滤波器，池化大小为2，池化步长为2，激活函数为Relu
    conv_pool_1 = fluid.nets.simple_img_conv_pool(
        input=img,
        filter_size=5,
        num_filters=20,
        pool_size=2,
        pool_stride=2,
        act="relu")
    conv_pool_1 = fluid.layers.batch_norm(conv_pool_1)
    # 第二个卷积-池化层
    # 使用20个5*5的滤波器，池化大小为2，池化步长为2，激活函数为Relu
    conv_pool_2 = fluid.nets.simple_img_conv_pool(
        input=conv_pool_1,
        filter_size=5,
        num_filters=50,
        pool_size=2,
        pool_stride=2,
        act="relu")
    # 以softmax为激活函数的全连接输出层，输出层的大小必须为数字的个数10
    prediction = fluid.layers.fc(input=conv_pool_2, size=10, act='softmax')
    return prediction

def train_program():
    """
    配置train_program
    Return:
        predict -- 分类的结果
        avg_cost -- 平均损失
        acc -- 分类的准确率
    """
    paddle.enable_static()
    # 标签层，名称为label,对应输入图片的类别标签
    label = fluid.layers.data(name='label', shape=[1], dtype='int64')
    # predict = softmax_regression() # 取消注释将使用 Softmax回归
    # predict = multilayer_perceptron() # 取消注释将使用 多层感知器
    predict = convolutional_neural_network() # 取消注释将使用 LeNet5卷积神经网络
    # 使用类交叉熵函数计算predict和label之间的损失函数
    cost = fluid.layers.cross_entropy(input=predict, label=label)
    # 计算平均损失
    avg_cost = fluid.layers.mean(cost)
    # 计算分类准确率
    acc = fluid.layers.accuracy(input=predict, label=label)
    return predict, [avg_cost, acc]

def optimizer_program():
    """
    :return:
    """
    return fluid.optimizer.Adam(learning_rate=0.001)

def event_handler(pass_id, batch_id, cost):
    """event_handler"""
    # 打印训练的中间结果，训练轮次，batch数，损失函数
    print("Pass %d, Batch %d, Cost %f" % (pass_id, batch_id, cost))

def train_test(train_test_program,
               train_test_feed, train_test_reader, executor, fetch_list):
    """train_test"""
```

```
    # 将分类准确率存储在acc_set中
    acc_set = []
    # 将平均损失存储在avg_loss_set中
    avg_loss_set = []
    # 将测试 reader yield 出的每一个数据传入网络中进行训练
    for test_data in train_test_reader():
        avg_loss_np, acc_np = executor.run(
            program=train_test_program,
            feed=train_test_feed.feed(test_data),
            fetch_list=fetch_list)
        acc_set.append(float(acc_np))
        avg_loss_set.append(float(avg_loss_np))
    # 获得测试数据上的准确率和损失值
    acc_val_mean = np.array(acc_set).mean()
    avg_loss_val_mean = np.array(avg_loss_set).mean()
    # 返回平均损失值，平均准确率
    return avg_loss_val_mean, acc_val_mean

class Model():
    def __init__(self, args, train_reader, test_reader):
        self.args = args
        self.create_model()
        self.train_reader = train_reader
        self.test_reader = test_reader

    def create_model(self):
        """create_model"""
        # 该模型运行在单个CPU上
        self.place = fluid.CPUPlace()
        # 调用train_program 获取预测值，损失值
        self.prediction, [self.avg_loss, self.acc] = train_program()
        # 输入的原始图像数据，大小为28*28*1
        img = fluid.layers.data(name='img', shape=[1, 28, 28], dtype='float32')
        # 标签层，名称为label,对应输入图片的类别标签
        label = fluid.layers.data(name='label', shape=[1], dtype='int64')
        # 告知网络传入的数据分为两部分，第一部分是img值，第二部分是label值
        self.feeder = fluid.DataFeeder(feed_list=[img, label], place=self.place)
        # 选择Adam优化器
        optimizer = fluid.optimizer.Adam(learning_rate=self.args.lr)
        optimizer.minimize(self.avg_loss)

    def run_train(self):
        PASS_NUM = self.args.epoch
        epochs = [epoch_id for epoch_id in range(PASS_NUM)]

        self.exe = fluid.Executor(self.place)
        self.exe.run(fluid.default_startup_program())
        main_program = fluid.default_main_program()
        step = 0
        for epoch_id in epochs:
            print("Epoch %d:" % (epoch_id))
            for step_id, data in enumerate(self.train_reader()):
                metrics = self.exe.run(main_program,
                                       feed=self.feeder.feed(data),
                                       fetch_list=[self.avg_loss, self.acc])
                if step % 100 == 0:  # 每训练100次 更新一次图片
                    event_handler(step, epoch_id, metrics[0])
                step += 1

    def save_model(self):
        """save_model"""
        # 将模型参数存储在名为 save_dirname 的文件中
        save_dirname = self.args.output_dir
        fluid.io.save_inference_model(save_dirname,
                                      ["img"], [self.prediction], self.exe,
                                      model_filename='model',
                                      params_filename='params')

    def evaluate(self):
        """evaluate"""
```

```
        test_program = fluid.default_main_program().clone(for_test=True)
        avg_loss_val, acc_val = train_test(train_test_program=test_program,
                                           train_test_reader=self.test_reader,
                                           train_test_feed=self.feeder,
                                           executor=self.exe,
                                           fetch_list=[self.avg_loss, self.acc])
        print("accuracy: %f" % acc_val)
        return acc_val

def report_final(args, metric):
    """report_final_result"""
    # 结果上报sdk
    amaas_tools = AMaasTools(args.job_id, args.trial_id)
    metric_dict = {args.metric: metric}
    for i in range(3):
        flag, ret_msg = amaas_tools.report_final_result(metric=metric_dict,
                                                        export_model_path=args.output_dir,
                                                        checkpoint_path="")
        print("End Report, metric:{}, ret_msg:{}".format(metric, ret_msg))
        if flag:
            break
        time.sleep(1)
    assert flag, "Report final result to manager failed! Please check whether manager'address or manager'status " \
                 "is ok! "

def main():
    """main"""
    # 获取参数
    args = parse_arg()
    # 加载数据集
    train_reader, test_reader = reader_load(args)
    # 模型定义
    model = Model(args, train_reader, test_reader)
    # 模型训练
    model.run_train()
    # 模型保存
    model.save_model()
    # 模型评估
    acc = model.evaluate()
    # 上报结果
    report_final(args, metric=acc)

if __name__ == '__main__':
    main()
```

示例代码对应的yaml配置如下，请保持格式一致

pwo_search_demo_for_paddle.yml示例内容

```
 **搜索算法参数**
search_strategy:
  algo: PARTICLE_SEARCH #搜索策略：粒子群算法
  params:
    population_num: 8 #种群个体数量 | [1,10] int类型
    round: 10 #迭代轮数 | [5,50] int类型
    inertia_weight: 0.5 # 惯性权重 | (0，1] float类型
    global_acceleration: 1.5 #全局加速度 | (0,4] float类型
    local_acceleration: 1.5 #个体加速度 | (0,4] float类型

**单次训练时数据的采样比例，单位%**
data_sampling_scale: 100 # | （0，100] int类型

**评价指标参数**
metrics:
  name: acc #评价指标 | 任意字符串 str类型
  goal: MAXIMIZE #最大值/最小值 | str类型 MAXIMIZE or MINIMIZE 必须为这两个之一（也即支持大写）
  expected_value: 100 #早停标准值，评价指标超过该值则结束整个超参搜索，单位% | 无限制 int类型

**搜索参数空间**
search_space:
  batch_size:
    htype: choice
    value: [64, 128, 256, 512]
  lr:
    htype: loguniform
    value: [0.0001, 0.1]
  epoch:
    htype: choice
    value: [1, 5, 10]
```

XGBoost 1.3.1代码规范

XGBoost 1.3.1代码规范 基于XGBoost 1.3.1框架的结构化数据的回归问题。

如下所示是其超参搜索任务中一个超参数组合的训练代码，代码会通过argparse模块接受在平台中填写的信息，请保持一致。另外该框架支持发布保存模型为pickle和joblib格式，并且在发布至模型仓库时需要选择相应的模型文件。

xgboost1.3.1_autosearch.py示例代码

```
 **-*- coding:utf-8 -*-**
""" xgboost train demo """
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.metrics import mean_squared_error
import numpy as np
import os
import time
import argparse
from rudder_autosearch.sdk.amaas_tools import AMaasTools

def parse_arg():
    """parse arguments"""
    parser = argparse.ArgumentParser(description='xgboost boston Example')
    parser.add_argument('--train_dir', type=str, default='./train_data',
                        help='input data dir for training (default: ./train_data)')
    parser.add_argument('--test_dir', type=str, default='./test_data',
                        help='input data dir for test (default: ./test_data)')
    parser.add_argument('--output_dir', type=str, default='./output',
                        help='output dir for auto_search job (default: ./output)')
    parser.add_argument('--job_id', type=str, default="job-1234",
                        help='auto_search job id (default: "job-1234")')
    parser.add_argument('--trial_id', type=str, default="0-0",
                        help='auto_search id of a single trial (default: "0-0")')
    parser.add_argument('--metric', type=str, default="mse",
                        help='evaluation metric of the model')
    parser.add_argument('--data_sampling_scale', type=float, default=1.0,
                        help='sampling ratio of the dataset for auto_search (default: 1.0)')
```

```
    parser.add_argument('--max_depth', type=int, default=6,
                        help='maximum depth of the tree (default: 6)')
    parser.add_argument('--gamma', type=float, default=0.1,
                        help='minimum loss reduction required for further splitting (default: 0.1)')
    parser.add_argument('--eta', type=float, default=0.1,
                        help='learning rate (default: 0.1)')
    parser.add_argument('--num_round', type=int, default=10,
                        help='number of trees (default: 10)')
    args = parser.parse_args()
    args.output_dir = os.path.join(args.output_dir, args.job_id, args.trial_id)
    if not os.path.exists(args.output_dir):
        os.makedirs(args.output_dir)
    print("job_id: {}, trial_id: {}".format(args.job_id, args.trial_id))
    return args

def load_data(data_sampling_scale):
    """ load data """
    boston = datasets.load_boston()
    X, Y = boston.data, boston.target
    # 切分，测试训练2,8分
    x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
    train_data = np.concatenate([x_train, y_train.reshape([-1, 1])], axis=1)
    np.random.seed(0)
    np.random.shuffle(train_data)
    train_data = train_data[0:int(data_sampling_scale * len(train_data))]
    x_train, y_train = train_data[:, 0:-1], train_data[:, -1]
    return (x_train, x_test), (y_train, y_test)

def save_model(model, output_dir):
    """ save model with pickle format """
    import pickle
    with open(output_dir + '/clf.pickle', 'wb') as f:
        pickle.dump(model, f)

def save_model_joblib(model, output_dir):
    """ save model with joblib format """
    import joblib
    joblib.dump(model, output_dir + '/clf.pkl')

def evaluate(model, x_test, y_test):
    """evaluate"""
    # 回归mean_squared_error指标
    deval = xgb.DMatrix(x_test)
    predict = model.predict(deval)
    mse = mean_squared_error(y_test, predict)
    print("mean_squared_error: %f" % mse)
    return mse

def report_final(args, metric):
    """report_final_result"""
    # 结果上报sdk
    amaas_tools = AMaasTools(args.job_id, args.trial_id)
    metric_dict = {args.metric: metric}
    for i in range(3):
        flag, ret_msg = amaas_tools.report_final_result(metric=metric_dict,
                                                        export_model_path=args.output_dir,
                                                        checkpoint_path="")
        print("End Report, metric:{}, ret_msg:{}".format(metric, ret_msg))
        if flag:
            break
        time.sleep(1)
    assert flag, "Report final result to manager failed! Please check whether manager'address or manager'status " \
                 "is ok! "

def main():
    """ main """
    # 获取参数
    args = parse_arg()
    # 加载数据集
    (x_train, x_test), (y_train, y_test) = load_data(args.data_sampling_scale)
```

```
    dtrain = xgb.DMatrix(x_train, label=y_train)
    # 模型参数定义
    param = {"gamma": args.gamma, 'max_depth': args.max_depth,
             'eta': args.eta, 'objective': 'reg:squarederror'}
    # 模型训练
    model = xgb.train(param, dtrain, args.num_round)
    # 模型保存
    save_model_joblib(model, args.output_dir)
    # 模型评估
    mse = evaluate(model, x_test, y_test)
    # 上报结果
    report_final(args, metric=mse)

if __name__ == '__main__':
    main()
```

示例代码对应的yaml配置如下，请保持格式一致

cmaes_search_demo.yml示例内容

```
  **搜索算法参数**
search_strategy:
  algo: CMAES_SEARCH #搜索策略：进化-cmaes算法
  params:
    population_num: 8 #种群个体数量 | [1,10] int类型
    round: 10 #迭代轮数 | [5,50] int类型
    step_size: 1.0 # 学习步长 | （0，10] float类型

**单次训练时数据的采样比例，单位%**
data_sampling_scale: 100  # | （0，100] int类型

**评价指标参数**
metrics:
  name: mse #评价指标 | 任意字符串 str类型
  goal: MINIMIZE #最大值/最小值 | str类型 MAXIMIZE or MINIMIZE 必须为这两个之一（也即支持大写）
  expected_value: 10 #早停标准值，评价指标超过该值则结束整个超参搜索，单位% | 无限制 int类型

**搜索参数空间**
search_space:
  max_depth:
    htype: randint
    value: [3, 10]
  num_round:
    htype: randint
    value: [1, 8]
  gamma:
    htype: uniform
    value: [0.1, 1]
  eta:
    htype: loguniform
    value: [0.01, 1]
```

训练作业API

本文将介绍自定义作业下的训练作业的API，您如果是初次使用相关产品，可以参考相关指南。

鉴权机制

在使用API前，您需要完成鉴权操作，可以参考鉴权认证机制中的介绍。

接口详细介绍

平台开放了5个API供用户调用：

- 训练作业-创建
- 训练作业-停止
- 训练作业-删除

- 训练作业-作业列表
- 训练作业-作业详情

【训练作业-创建】

接口：`https://aip.baidubce.com/rpc/2.0/easydl/pro/remotetrain/create`

method: `post`

请求参数：

参数	是否必选	类型	可选值	说明
name	是	string	-	作业名称，1-50个字符，不能有表情字符
description	是	string	-	作业描述， 1-500个字符，不能有表情字符
train_config	是	map		训练配置项
+ env_type	是	string	CCR：百度CCR镜像环境	训练环境类型
+output_UrlSource	是	string	BOS：存储来源是bos；PFS：存储来源是PFS	
+ output_path	是	string	-	训练产出输出路径
+code_UrlSource	是	string	BOS：存储来源是bos；PFS：存储来源是PFS	
+ code_path	否	string	-	启动代码路径，可在自定义镜像中定义训练代码，通过启动命令来使用镜像中的训练代码
+ start_cmd	是	string	-	启动命令
+ image	否	string	-	镜像，自定义镜像下必选，eg:/ns-project/paddle
+ tag	否	string	-	tag，自定义镜像下必选
+ repository_id	否	string	-	BML平台绑定的仓库的ID，自定义镜像下必选
+ distribute_strategy	否	string	Horovod；PaddleFleet; PytorchDDP	分布式策略，自定义镜像下必选
dataset_config	是	map		数据集配置项
+trainset_UrlSource	否	string	BOS：存储来源是bos；PFS：存储来源是PFS	
+ trainset_path	否	string	-	训练数据集，可在自定义镜像中定义
+evalset_UrlSource	否	string	BOS：存储来源是bos；PFS：存储来源是PFS	
+ evalset_path	否	string	-	评估数据集，可在自定义镜像中定义

resource_config	是	map		资源池配置项
+ resource_type	是	string	PUBLIC：公共资源池；USER:用户资源池	资源池类型，默认PUBLIC
+ node_count	是	int	-	节点数,公共资源池支持的节点数范围为1-4
+ resource_name	是	string	资源池名称，公共资源池：GPU_V100、GPU_P4、CPU_4C_16G、CPU_16C_64G。租户资源池请填写在BML平台注册的资源池名称	资源池名称
+ max_duration	是	int	1-168	最大运行时间，单位小时
+ gpu_type	否	string	-	加速卡
+ gpu_num	否	int	-	加速卡数量，单位张
+ cpu_num	否	int	-	cpu数量，单位核
+ memory	否	int	-	内存，单位G
+ rdma	否	bool	-	是否开启RDMA
prjId	是	string	-	项目ID (新增)

返回：

参数	类型	说明
result	int	作业ID
log_id	int	日志ID

【训练作业-停止】

接口：`https://aip.baidubce.com/rpc/2.0/easydl/pro/remotetrain/stop`

method: `post`

参数：

参数	是否必选	类型	参数位置	可选值	说明
id	是	int	RequestBody	-	作业ID
prjId	是	string	RequestBody	-	项目ID(新增)

返回：

参数	类型	说明
log_id	int	日志ID
result	bool	是否停止成功

【训练作业-删除】

接口：`https://aip.baidubce.com/rpc/2.0/easydl/pro/remotetrain/delete`

method: `post`

参数：

参数	是否必选	类型	参数位置	可选值	说明
id	是	int	RequestBody	-	作业ID
prjId	是	string	RequestBody	-	项目ID(新增)

返回：

参数	类型	说明
log_id	int	日志ID
result	bool	是否删除成功

【训练作业-作业列表】

接口：`https://aip.baidubce.com/rpc/2.0/easydl/pro/remotetrain/list`

method: `post`

参数：

参数	是否必选	类型	参数位置	可选值	说明
start	是	int	RequestBody	-	分页offset值，缺损值为0
num	是	int	RequestBody	-	分页limit值，缺损值为20
prjId	是	string	RequestBody	-	项目ID(新增)

返回：

参数	类型	说明
total	int	-
id	int	作业ID
name	string	作业名称
description	string	作业描述
train_status	string	训练状态：UNTRAIN（编辑中未训练的），SUCCESSED（训练完成），RUNNING（训练中），STOPPED（训练终止），FAILED（训练失败），PENDING（排队中）
run_time	int	训练时间，单位分钟
resource_type	string	资源池类型
resource_name	string	资源池名称
framework	string	框架
create_time	int	创建时间
log_id	int	日志ID

【训练作业-作业详情】

接口：`https://aip.baidubce.com/rpc/2.0/easydl/pro/remotetrain/detail`

method: `post`

参数：

参数	是否必选	类型	参数位置	可选值	说明
id	是	int	RequestBody	-	作业ID
prjId	是	string	RequsetBody	-	项目ID(新增)

返回：

参数	类型	说明
id	int	-
name	string	作业名称
description	string	作业描述
train_status	string	训练状态：UNTRAIN（编辑中未训练的），SUCCESSED（训练完成），RUNNING（训练中），STOPPED（训练终止），FAILED（训练失败），PENDING（排队中）
run_time	int	训练时间，单位分钟
create_time	int	创建时间
train_config	map	训练配置
+ code_path	string	启动代码路径
+ start_cmd	string	启动命令
+ output_path	string	输出路径
+ framework	string	训练配置：框架
+ env_type	string	训练配置项：训练环境类型。DEFAULT：平台预置环境；CCR：百度CCR镜像环境
+ image	string	训练配置项：镜像
+ tag	string	训练配置项：镜像tag
+ distribute_strategy	string	训练配置项：分布式策略，Horover；PaddleFleet; PytorchDDP
+ repository_id	string	训练配置项：镜像对应在BML平台绑定的仓库的ID
+ trainset_path	string	数据集配置项：训练数据集
+ evalset_path	string	数据集配置项：评估数据集
+ evalset_path	string	数据集配置项：评估数据集
resource_config	map	资源池配置项
+ node_count	int	节点数
+ resource_name	string	资源池名称
+ max_duration	int	最大运行时间，单位小时
+ gpu_type	string	加速卡
+ gpu_num	int	加速卡数量，单位张
+ cpu_num	string	cpu数量，单位核
+ memory	int	内存，单位G
+ rdma	bool	是否开启RDMA
dataset_config	map	数据集配置项
+ trainset_path	string	训练数据集地址
+ evalset_path	string	测试数据集地址
log_id	int	日志ID

错误码说明

错误码	错误信息
406000	internal server error
406001	param[%s] invalid
406008	[%s] quota exceeded
406012	job does not exist
406013	job stop failed
406014	job can not delete
406015	job name cannot be empty
406016	the job name is too long
406017	job description cannot be empty
406018	job description is too long
406019	code file address error
406020	startup command error
406021	output path error
406022	training dataset address error
406023	test dataset address error
406024	illegal language version
406025	illegal language version
406026	job name already exists
406027	job status exception
406028	resource type error
406029	there is no remaining free quota, please activate payment or recharge
406030	missing image parameters
406031	missing tag parameters
406032	missing distributed strategy parameter
406033	image registry is incorrect
406034	training environment error
406035	the %s quantity has reached the upper limit
406036	the emoji character is not supported
406037	no operation permission
406038	parameter too long
406100	resource pool does not exist
406101	node over limit
406102	cpu over limit
406103	memory over limit
406104	gpu over limit
406105	

可视化建模

概述

可视化建模通过拖拉拽和拼接组件的方式，形成建模流程。用户配置组件参数后，即可训练模型。

平台提供可视化的实验开发环境，开发人员和业务人员根据场景和业务需求能够在交互式画布上直观地连接数据处理、特征工程，算法，模型预测和模型评估等组件，基于无代码方式实现人工智能模型开发。可视化建模在降低模型开发门槛的同时提升了建模的效率。内置数百个成熟的机器学习算法，支持多种算法框架，覆盖了机器学习和深度学习场景，满足用户不同程度的需求。

基本概念

- 画布：可视化建模模式下开发的所有的算法均在画布上组建完成，用户可以自由的在画布上“绘制”属于自己的算法之“画”；
- 组件：可视化建模模式为用户提供了一系列的组件，其中包含了数据集、数据处理、算法等不同用途的组件，用户可以通过拖拉拽的方式，将相关的组件拖到画布上，并用箭头联系起来；
- 箭头：箭头代表着两端组件的逻辑顺序以及数据流向关系，用户需要将各个组件通过箭头连接起来，才能完成一个完整的算法任务配置；
- 实验管理：可视化建模任务均采用实验管理模块进行记录和管理，分为运行和模型；
- 运行：每一次提交可视化建模任务，将在实验管理中发生一条运行记录。

使用流程

- 第一步：通过拖拉拽将组件和箭头联系起来形成一个算法任务结构；
- 第二步：提交并运行；
- 第三步：发布对应的模型到模型仓库进行纳管；
- 第四步：将相应的模型发布为API服务；
- 第五步：调用API服务进行预测。

快速入门

本文以信用卡诈骗场景为例，帮助您快速构建可视化建模实验。

数据准备

本例所采用的Credit card fraud（https://www.kaggle.com/mlg-ulb/creditcardfraud） 数据集包含 2013 年九月欧洲的信用卡持有者的交易记录，交易次数共有 284807 次，其中的 492 次为诈骗交易。该数据集多用于金融领域，建模的目的是检测信用卡欺诈行为。

1. 进入“数据集 > 项目数据集“菜单，新建creditcard数据集，上传数据。



2. 数据集上传成功后，可点进数据集名称，查看数据表详情。数据表中包含V1-V28特征列（经PCA处理）、Time列（表示数据集中每个交易和第一个交易之间经过的秒数）、Amount列（表示交易金额）和 Class列（在发生欺诈时其值为1，否则为0）。

创建实验

进入“模型开发 > 可视化建模”页面，新建实验。



完善个人信息。



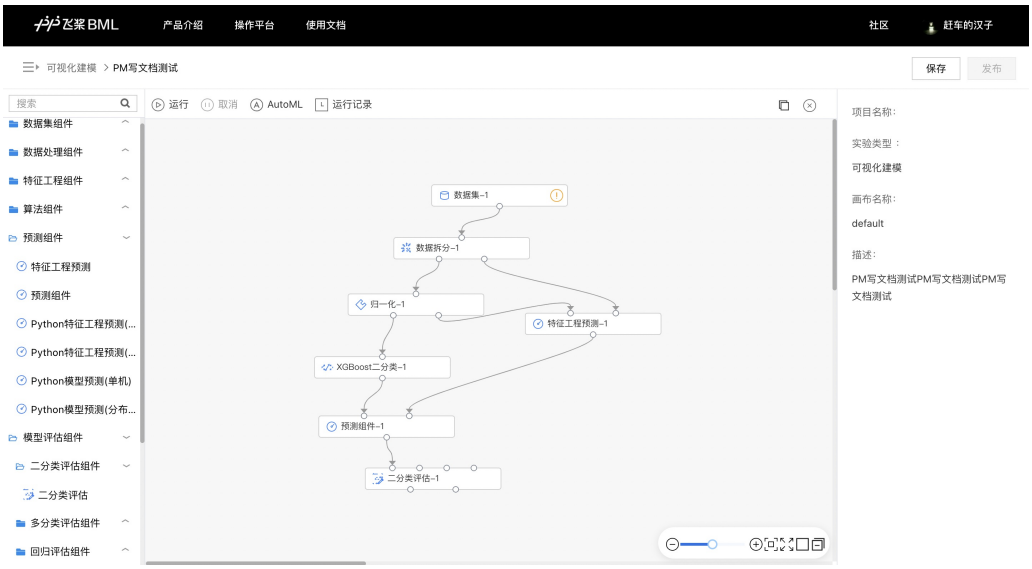
填写项目信息。



完成创建，并生成一条记录。

训练模型

1. 进入实验详情，从左侧组件面板拖拽下列组件，拼接实验。



各算子组件的配置详情如下：

数据集

拖拽“数据集”组件到画布中。在右侧的“字段设置”配置区，选择creditcard数据集。



数据拆分

拖拽“数据拆分”组件，将数据集拆分成为训练集和测试集两个部分。字段设置保持系统默认配置；资源配置选择当前可用的资源池，并配置各类资源。


数据拆分参数配置
* 拆分方式 必选
Rows
* 拆分比例 必选
0.80
* 随机拆分 必选
随机种子 可选
0
* 分层拆分 必选
否
Spark运行参数
* Driver运行环境 必选
CPU4核、16GB内存 ￥0.025/分...
CPU16核、64GB内存 ￥0.092/分...
* Executor运行环境 必选
2
CPU4核、16GB内存 ￥0.025/分...
CPU16核、64GB内存 ￥0.092/分...

归一化

拖拽“归一化”组件。在“参数设置“的“归一化方法”中选择“MAXMIN”，在“字段设置”的“归一化列”中选择“Amount(double)"，将交易金额归一化处理。



XGBoost二分类

拖拽“XGBoost二分类“到画布中，在“参数配置”的“评估指标”中选择“aucpr"（PR曲线下的面积），其他可保持默认配置或根据训练情况修改。
在“字段设置”的“特征列”中选择v1-v28列和Amount列，“标签列”中选择“Class(int)"。



特征工程预测

拖拽“特征工程预测”组件，对测试集做相应处理。

预测组件

拖拽“预测组件”，用于模型预测。

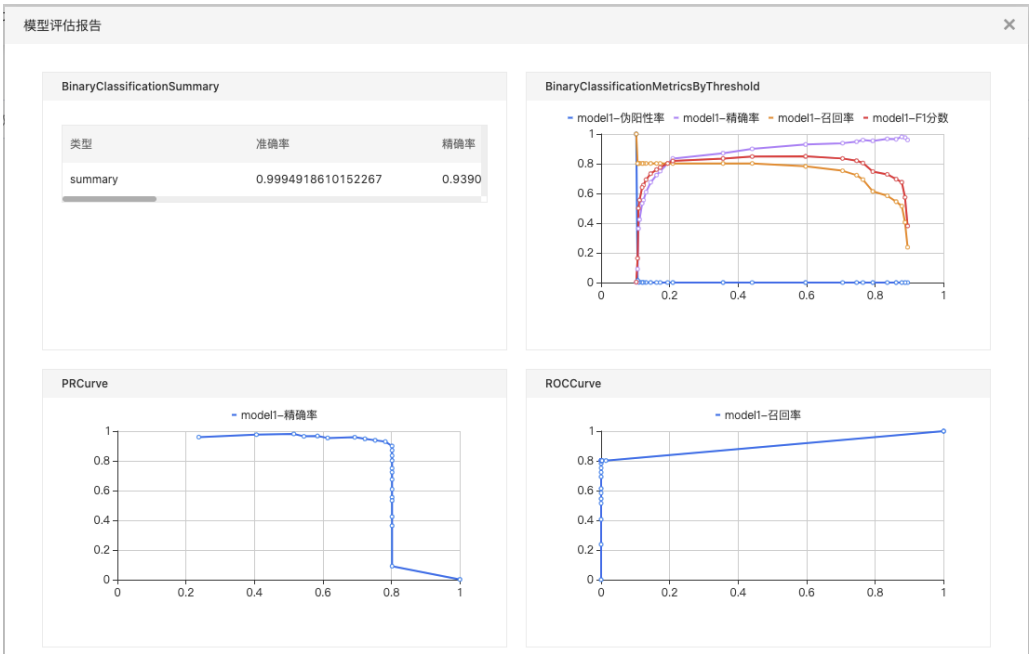
二分类评估

拖拽“二分类评估“组件到画布中，查看模型训练效果。在“字段设置”的“第一个输入真实标签列中选择“Class(int)"。



组建参数配置完成后，点击页面上方的”开始训练“按钮。

训练完成后，右击“二分类评估”组件，选择”查看评估报告“。



发布模型到模型中心

模型中心是AI模型的集中纳管、评估、优化转换处理场所，提供模型纳管、评估、压缩等功能。

1. 模型训练完成后，选择画布右上方的“发布”按钮，将模型发布到模型中心，方便后续部署为API服务并进行调用。



2. 发布模型后，进入“模型中心”菜单，在“模型列表”页面找到相应的creditcard模型。点进模型，当模型状态变为“就绪”时，表示模型发布成功。

导出模型文件

平台支持您在模型中心中导出模型包，跨平台使用。

1. 进入模型详情页，点击页面上方的“导出模型文件”按钮。



2. 在导出页面中，设置模型包名称，并选择存储卷，点击“确定”。



3. 进入任务列表，可查看导出任务的状态，并点击“详情”，跳转至存储页面。

4. 进入存储卷，下载模型包。



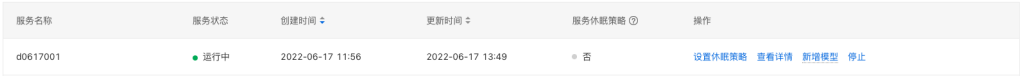
发布模型为预测服务

如果您部署了预测服务，可将模型发布为服务，方便API调用。

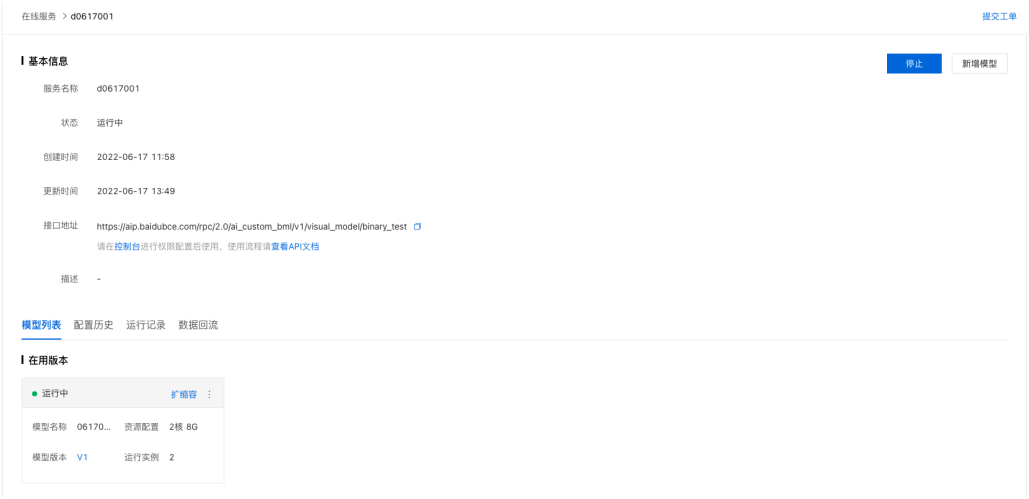
1. 进入模型管理-版本列表页，点击“部署-在线服务”。

在弹出的创建页面中，设置服务名称、接口地址、模型名称及版本、资源等参数。点击“确定”，将模型发布为服务。

2. 进入“公有云部署 > 在线服务“菜单，当服务的状态变为”运行中“时，表示服务发布成功。



点进服务详情，可查看服务地址，并监控服务的模型列表、配置历史、运行记录等信息。



关于预测服务的详细内容，请参考“预测服务”章节。

画布操作说明

概述

概述

您可在可视化建模面板中执行以下操作：

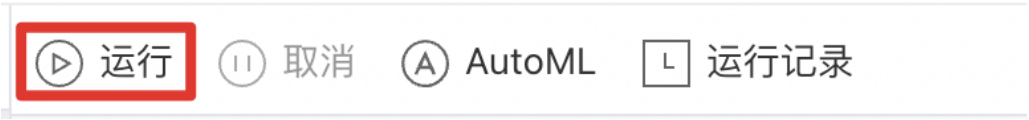


- 运行
- AutoML
- 运行记录

运行

运行

您在画布中配置好算子组件以及参数后，需点击画布上方的“运行”按钮，一键训练模型。



AutoML(自动调参)

AutoML(自动调参)

为了帮助模型达到更精准的效果，平台支持自动调参。支持自动调参的组件有XGBoost二分类、XGBoost多分类、XGBoost回归等等。

1. 点击“AutoML"按钮。



2. 在弹出的“自动调参”页面中，选择对应的算法组件。

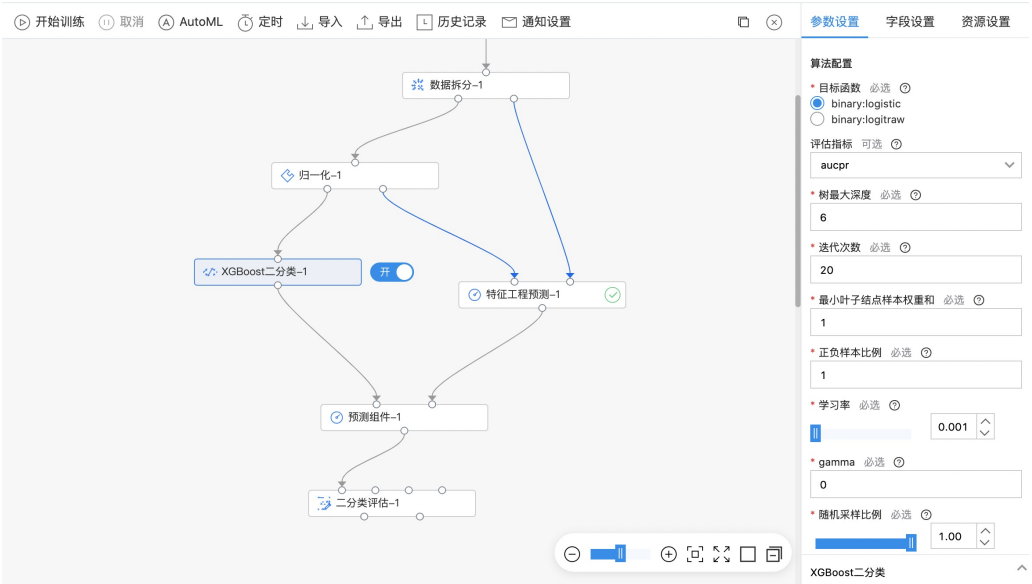
3. 在调参配置中，设置数据拆分比例、网格拆分数、参数范围和调参方式等。



4. 设置调参模型的评估标准。

5. 点击“确定”，完成自动调参。

   回到可视化建模画布，该算法组件右侧会有一个开关，默认为开。“开”表示该组件按照自动化调参方式运行，“关”则代表按照右侧的自定义参数设置进行。



组件菜单

基本操作

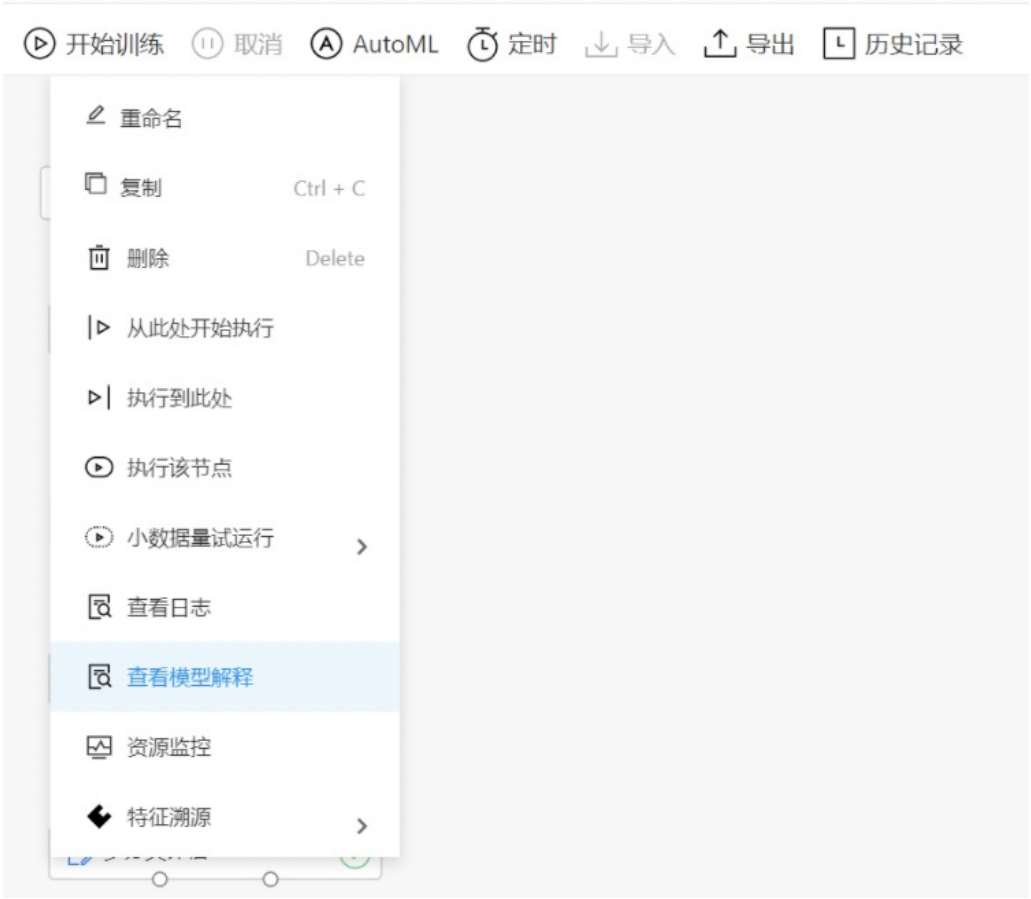
右键画布中的组件后，会出现工具菜单。您可执行以下操作。不同组件节点的右键菜单栏有所不同。

操作名	说明
删除	删除组件。
从此处开始执行	模型运行流程会从选择的节点开始执行，之前的流程不会执行。
执行到此处	模型运行流程会从开头运行到该节点处停止。
执行该结点	模型运行流程只运行该节点。
查看数据	数据集，数据处理、特征工程组件和预测组件等运行成功后，可执行该操作。 其中： 右击”预测组件“，执行“查看数据“操作后： 若是分类任务，数据表中的“result*label"列是内置的预测结果列。针对二分类任务，probability是负标签的预测概率，positive表示正标签的预测概率。针对多分类任务，probability*{类别标签}是各个类别的预测概率。 若是回归任务，数据表中的“result_score"列是内置的预测结果列。
小数据量试运行	从数据集中随机抽取小数据量运行模型流程。
查看日志	运行完毕后，支持查看日志。
导出数据	导出运行组件处理后的数据至数据集。
查看评估报告	模型评估类组件运行完毕后，点击可查看可视化评估报告。
查看分析报告	统计分析类组件运行完毕后，点击可查看可视化分析报告。
查看python绘制图形	自定义组件可通过Python代码实现绘图，点击后可查看所绘制的图形。
资源监控	可查看组件粒度的资源使用情况。
获取debug代码	python组件可查看并复制代码，方便在notebook中调试代码，避免在可视化建模中调试代码浪费大量时间，通常直接将代码复制到notebook中即可；其他组件可查看输入和输出数据文件路径的相关参数和kubectl命令，kubectl命令主要在研发排查问题时使用。 注意：如果使用notebook调试代码，镜像中必须包含代码所依赖的包。

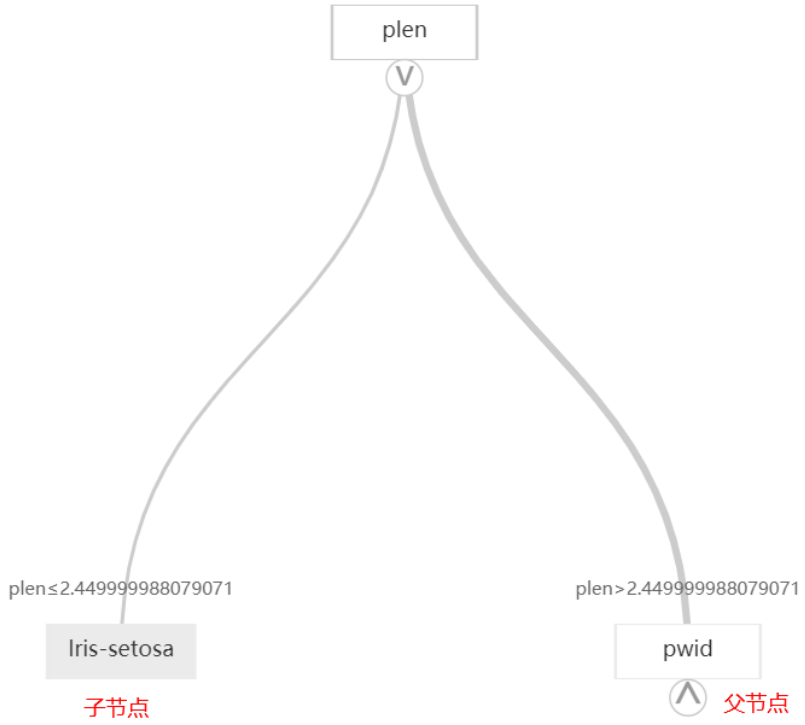
查看模型可解释性

对于DecisionTree二分类、DecisionTree多分类和DecisionTree回归等决策树相关算子，支持**在算子运行成功后**，查看模型训练过程，了解决策树的形成过程。

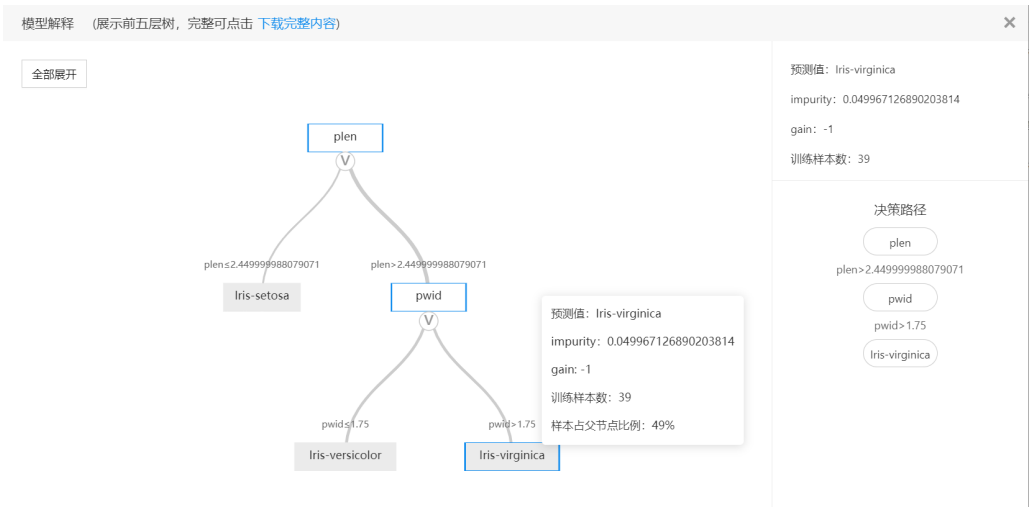
实验运行成功后，鼠标右键点击相应的算子组件，如“DecisionTree多分类”多分类，选择“查看模型解释”，即可在弹窗中可视化展示决策树。



决策树默认展示前两层，其中灰色底色的为子节点，带箭头符号的为父节点（如图）。点击父节点可展示下一层子节点。当前页面弹框中最多可支持展示5层，点击弹框中的“全部展开”按钮将前5层全部展开。



鼠标hover至相应节点后，可展示该节点对应的相关信息及决策路径，如图：



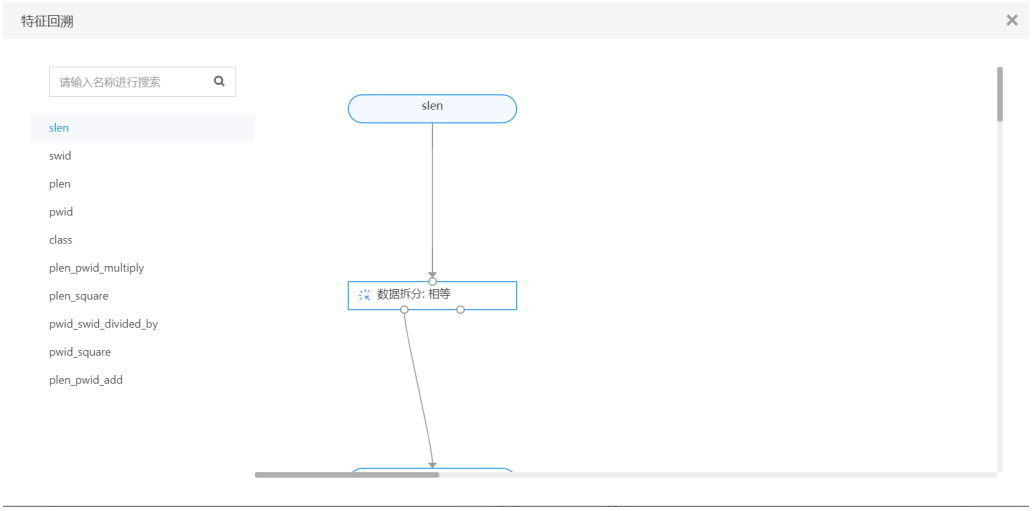
对于5层展示不全的决策树，可点击“下载完整内容”。

查看模型特征溯源

可视化建模提供特征溯源功能，对模型可用特征进行可视化溯源。 如果数据集组件到算法组件的路径中仅包含“数据处理组件（除‘Python数据处理组件’）”或“特征工程组件（除‘Python特征工程组件’）”，则可以生成特征溯源信息。 算法组件运行成功后，点击鼠标右键，选择“特征溯源”查看。



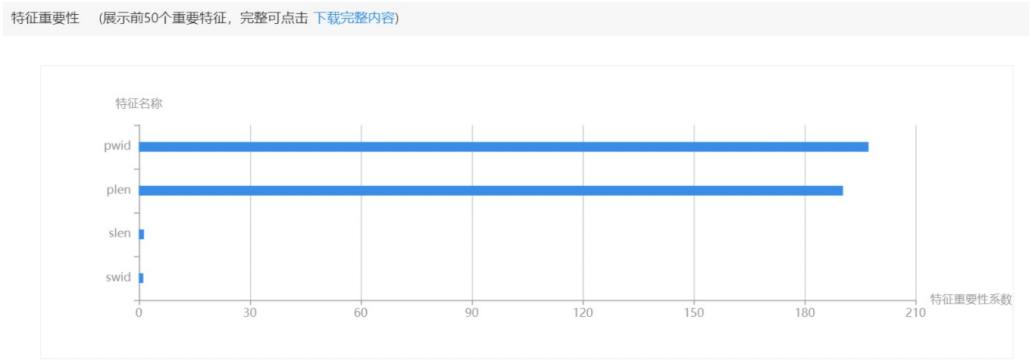
在弹出的界面中选择需要查看的特征名称，查看对应的溯源信息。



查看特征重要性

对于LR二分类、LR多分类、广义线性回归、XGBoost二分类、XGBoost多分类、XGBoost回归等算子组件，支持**在算子运行成功后**，查看其重要特征。

实验运行成功后，鼠标右键点击相应的算子组件，如“XGboost多分类“，选择“查看特征重要性”，即可显示前50个重要的特征。

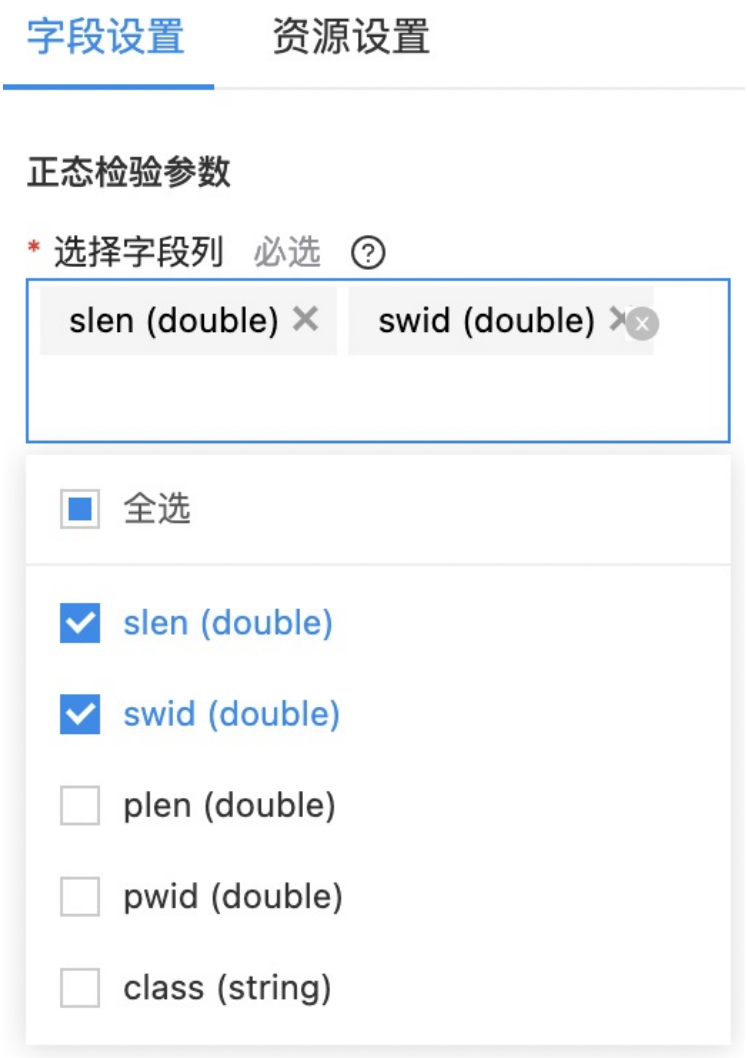


如需查看全部特征的特征重要性指标，可以点击弹框上方的“下载完整内容”，下载完整的特征重要性说明。

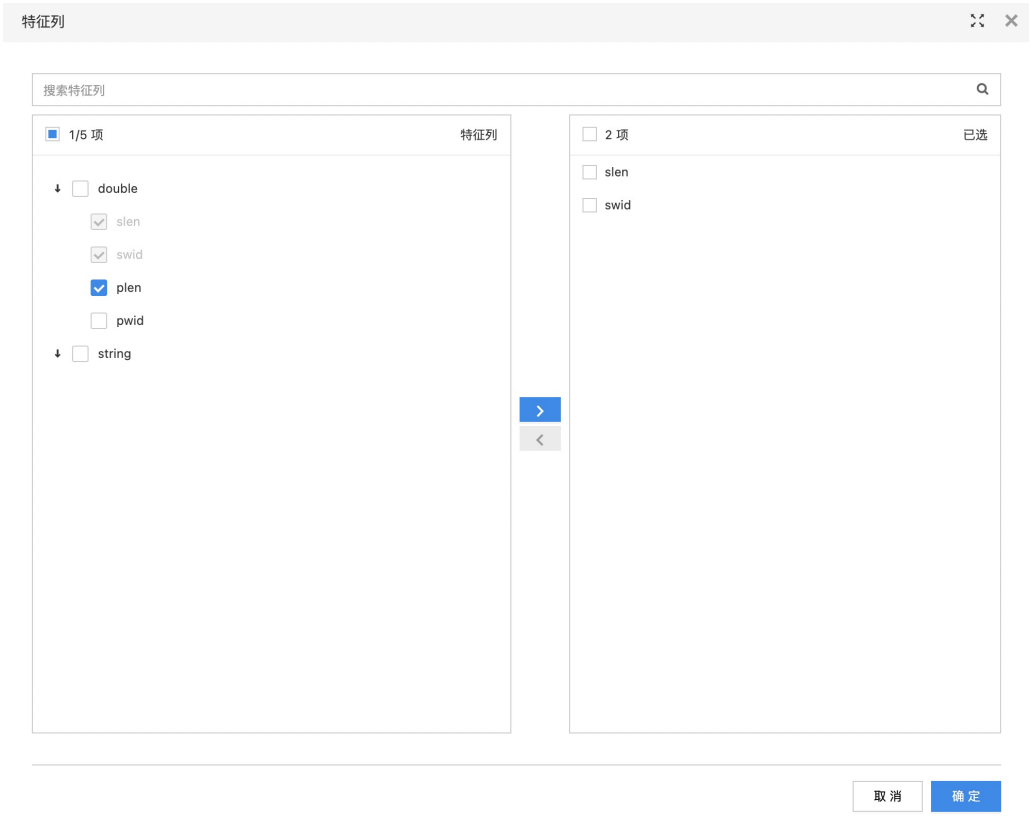
组件列选择

在选择特征列或标签列时，支持单独勾选与批量选择的方式选择数据列，支持搜索查找数据列。

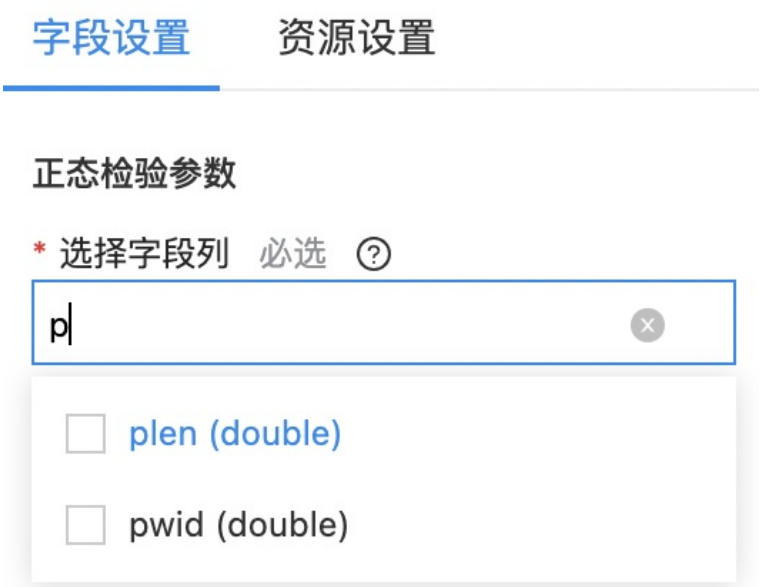
当需要选择的数据列比较少时，可以直接勾选数据列进行选择。



当数据列比较多时，点击【批量操作】后，选择需要的数据列，将数据列从左侧列表移动到右侧列表，点击确定即可，**数据列已经按照字段类型进行了分类**。



如果只需要在大量数据列中找某个数据列，则可以使用搜索查找，直接在搜索框输入字段信息即可。



组件状态

在画布中，各组件有如下状态：

- 运行成功：组件右侧有勾代表该组件运行成功。
- 运行中：旋转动态标识，代表该组件正在运行。
- 等待中：蓝色省略号标识，代表该组件正在等待运行。
- 运行失败：红色x号标识，代表由于某种原因运行失败。

组件资源配置

在画布中，拖拽组件后，需为组件配置对应的资源。

Spark运行参数

Spark任务中包含Driver和Executor，关系为一对多，您需要分别对Driver和Executor进行资源配置。

参数名称	是否必选	参数描述
Driver 运行环境	是	平台提供两种资源套餐可供选择
Executor 运行环境	是	平台提供两种资源套餐可供选择

Python单机算子配置

Python单机算子配置适用于Python单机类的算子。

参数名称	是否必选	参数描述
运行环境	是	平台提供两种资源套餐可供选择

组件说明

数据集组件

数据集组件

数据集

导入用户在EasyData中创建的数据集中的数据。

输入

- 选择平台上的数据集。要求为“**1、数据集数据量不为0；2、数据集状态非智能标注中，非导入中；3、数据集需为2022年7月15日后创建。**”

输出

- 输出是选择的数据集。

数据处理组件

数据处理

SMOTE过采样

SMOTE算法的基本思想就是采用KNN技术对少数类别样本进行分析和模拟，并将人工模拟的新样本添加到数据集中，进而使原始数据中的类别不再严重失衡。

输入

- 输入一个数据集。
- 填写过采样率、过采样依赖的标签列与过采样标签。

输出

- SMOTE过采样后的数据集。

算子参数

参数名称	是否必选	参数描述	默认值
过采样率	是	过采样扩充的样本数占当前该标签样本数的百分比，输入范围：[1，inf)	100

字段参数

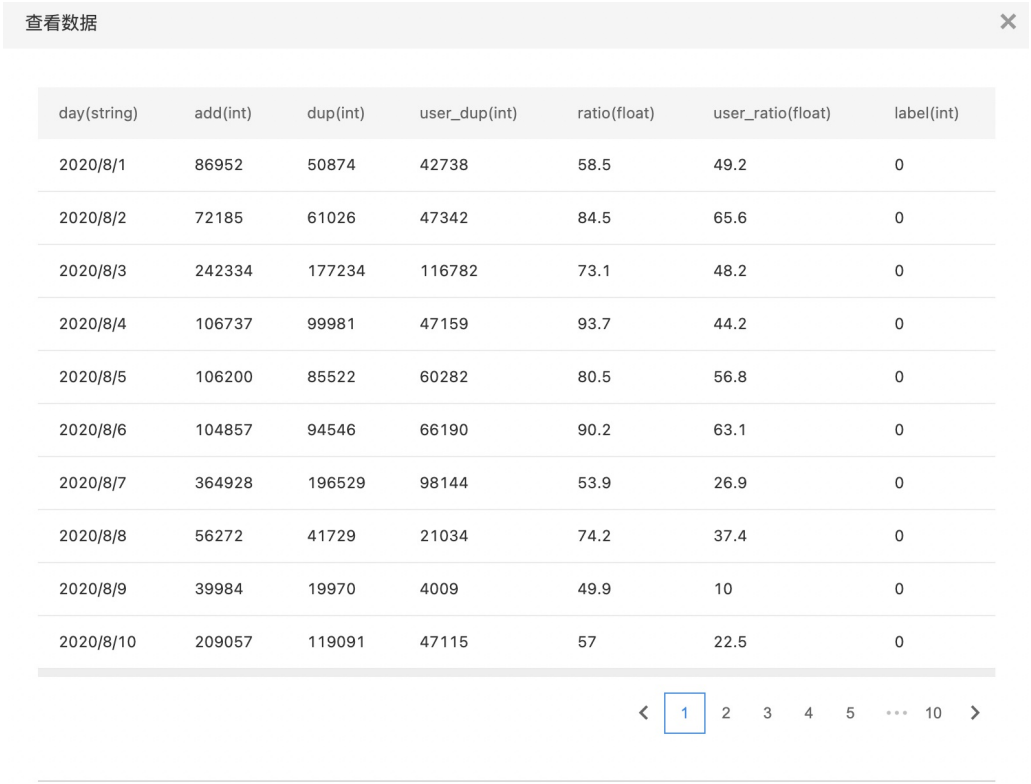
参数名称	是否必选	参数描述	默认值
标签列	是	SMOTE过采样依据的标签列，即用户希望平衡的样本标签	无
过采样标签	是	要进行SMOTE过采样的标签，即用户希望扩充样本的类别	无

计算逻辑

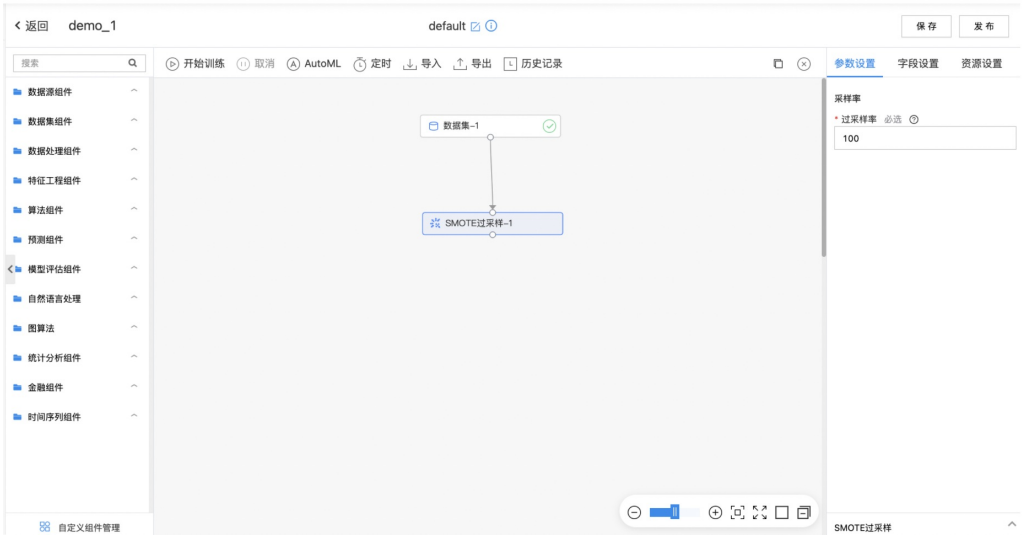
假设当前数据集存在10个样本类别为0，100个样本类别为1，过采样率输入50，标签列选择数据集对应的类别标签，过采样标签输入0，最终输出数据集为原始的10个类别为0的样本，100个类别为1的样本，还有过采样扩充的5个类别为0的样本，其中【扩充样本数=该标签样本数*过采样率】。

使用示例

1. 查看原始数据，此数据集中包含110个样本，其中10个样本类别为0，100个样本类别为1。



2. 构建算子结构，配置参数，完成训练。



3. 运行完成后查看扩充数据，最新数据集包含120个样本，其中20个样本类别为0，100个样本类别为1。

查看数据

day(string)	add(int)	dup(int)	user_dup(int)	ratio(double)	user_ratio(double)	label(i
2020/8/1	86952	50874	42738	58.5	49.20000076293945	0
2020/8/2	72185	61026	47342	84.5	65.5999984741211	0
2020/8/3	242334	177234	116782	73.0999984741211	48.20000076293945	0
2020/8/4	106737	99981	47159	93.69999694824219	44.20000076293945	0
2020/8/5	106200	85522	60282	80.5	56.79999923706055	0
2020/8/6	104857	94546	66190	90.19999694824219	63.099998474121094	0
2020/8/7	364928	196529	98144	53.900001525878906	26.899999618530273	0
2020/8/8	56272	41729	21034	74.19999694824219	37.400001525878906	0
2020/8/9	39984	19970	4009	49.900001525878906	10	0
2020/8/10	209057	119091	47115	57	22.5	0

< 1 2 3 4 5 >

简单采样

简单采样提供了三种采样方式，即随机采样、分层采样、权重采样，采样量计算方式可以选择比例计算与数量计算。

输入

- 输入一个数据集，根据需要选择不同的采样方式，选择响应的键列与采样数计算方式。

输出

- 简单采样后的数据集。

字段参数

参数名称	是否必选	参数描述	默认值
采样方式	是	选择采样的方式：RandomSampling - 随机采样、StratifiedSampling - 分层采样、WeightedSampling - 权重采样	无
分层键列	是	选泽用于分层采样的键列，不能是数组类型，采样结果的类别比例分层键列的分层比例一致	无
权重值列	是	选择权重采样的权重值列，必须是数值类型，样本对应的该列值即被采样的概率，权重越大被采样的概率越大	无
采样量计算方式	是	采样量的计算方式：Nmuber - 数量方式、Ratio - 比例方式	无
采样比例	是	采样的比例，输入范围：[0.01，0.99]	0.80
采样个数	是	采样的个数，输入范围：[1，inf)	无
随机种子	否	采样过程是随机的，采样结果与随机种子输入的值相关，若其他参数不变，随机种子输入一样的值，那么采样结果也是相同的	0
是否放回采样	是	采样过程中是否放回样本	关闭

计算逻辑

随机采样：系统随机从数据集中采集样本，随机种子的输入值不同导致采样结果不同。

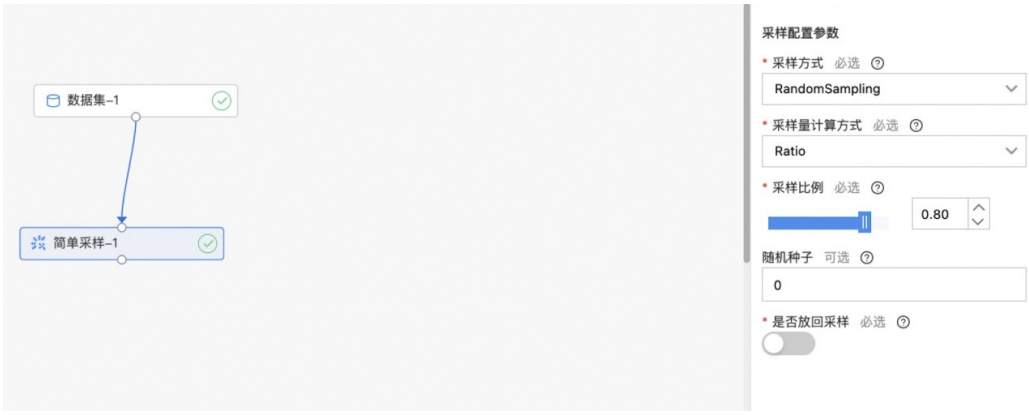
分层采样：选择分层键列，假设分层键列为性别，其中男性与女性的比例为6:4，那么采样结果的样本比例也为6:4。

权重采样：选择权重值列，假设权重值列为班级，样本A的班级序号为2，样本B的班级序号为1，则样本A被采样的概率为样本B的2倍。

采样数：最终的采样数依赖于采样量计算方式，假设原始数据集样本数为100，如果选择数量方式，则最终数据集的采样数量与输入数量一致，如果选择比例方式，比例为0.8，则最终数据集的采样数量80。

使用示例

如下图所示，构建算子结构，配置采样条件，运行后得到采样结果。



欠采样

为解决样本不平衡问题，欠采样对多数类别样本进行分析，并去除部分多数类别样本，进而使原始数据中的类别不再严重失衡。

输入

- 输入一个数据集，填写采样率、欠采样依赖的标签列与欠采样标签。

输出

- 欠采样后的数据集。

算子参数

参数名称	是否必选	参数描述	默认值
采样率	是	欠采样后该标签样本数占原该标签样本数的百分比，输入范围：[0.01，0.99]。	0.5

字段参数

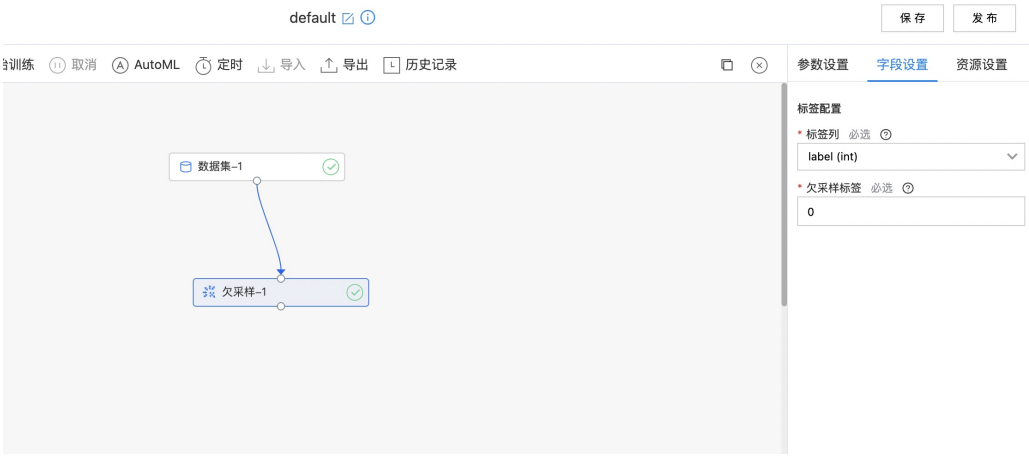
参数名称	是否必选	参数描述	默认值
标签列	是	欠采样依据的标签列，即用户希望平衡的样本标。	无
欠采样标签	是	要进行欠采样的标签，即用户希望缩减样本的类别。	无

计算逻辑

假设当前数据集存在10个样本类别为0，100个样本类别为1，欠采样率输入0.5，标签列选择数据集对应的类别标签，过采样标签输入1，最终输出数据集为原始的10个类别为0的样本，50个类别为1的样本。

使用示例

如下图所示，构建算子结构，配置需要欠采样的标签，运行后得到采样结果。



列选择

选择需要保留的数据列组合成为一个新的数据集作为输出。

输入

- 上游数据集组件中选择的数据集，选择需要保留的数据列。

输出

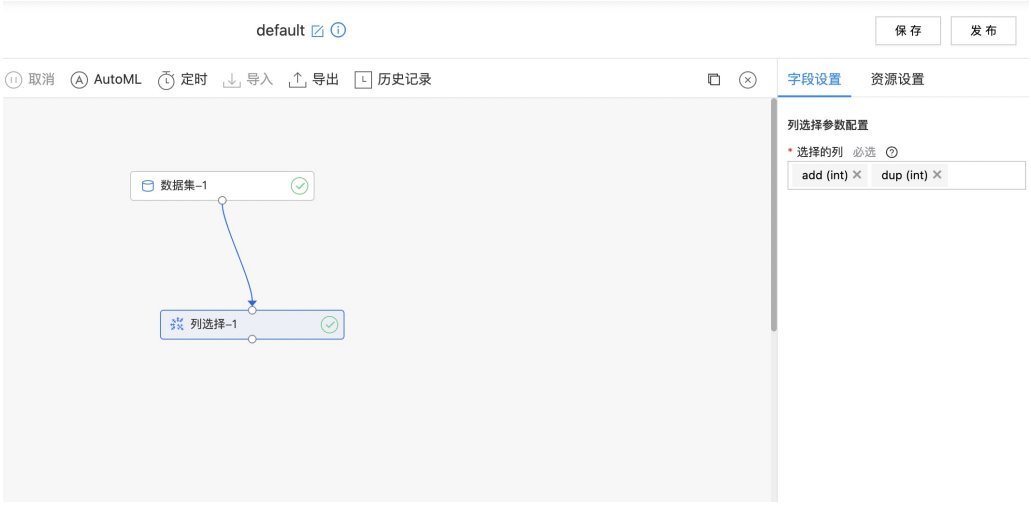
- 输出数据集的列名称顺序与选择的列顺序一致；可用于修改列的顺序。

字段参数

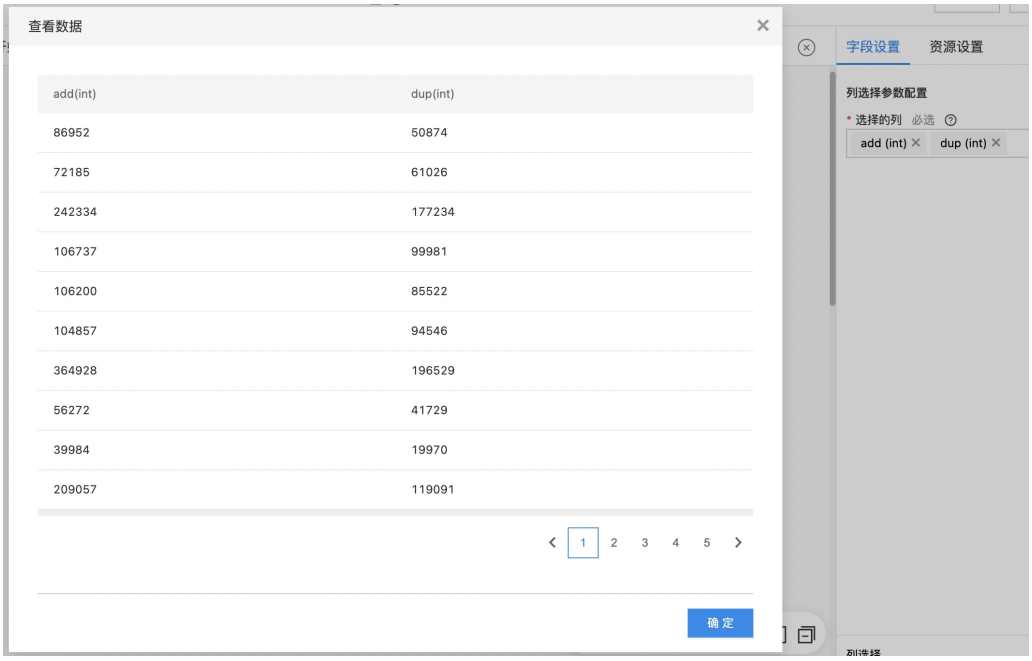
参数名称	是否必选	参数描述	默认值
选择的列	是	选择需要保留的列，可多选	无

使用示例

1. 如下图所示，构建算子结构，选择需要的数据列。



2. 查看列选择结果。



数据类型转换

选择需要进行类型转换的列，经过数据类型转换，得到目标类型的列，并替换原始数据集对应列。

输入

- 输入一个数据集，选择需要转换类型的列。

输出

- 输出一个数据集，需要转换数据类型的列已经完成转换并替换原始数据对应列。

算子参数

参数名称	是否必选	参数描述	默认值
输出类型	是	目标转换类型，目前支持整型，双精度浮点型，字符串之间的相互合法转换，以及字符串合法转换成时间戳、日期	字符串

字段参数

参数名称	是否必选	参数描述	默认值
输入列	是	选择需要进行类型转换的列	无

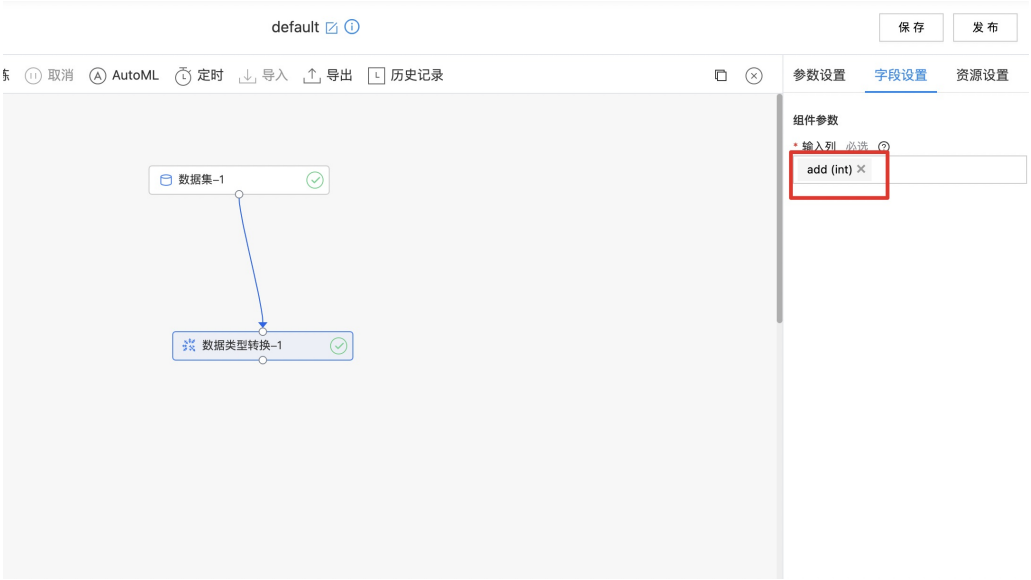
特殊格式说明

时间戳字符串支持格式，时间单位：yyyy(年)|MM(月)|dd(日)|HH(时[0,23])|mm(分)|ss(秒)，不同的时间单位可以任意顺序组合且之间的连接是特殊符号即可（例如/、-、#、@等），例如yyyy-MM-dd HH:mm:ss（默认）。

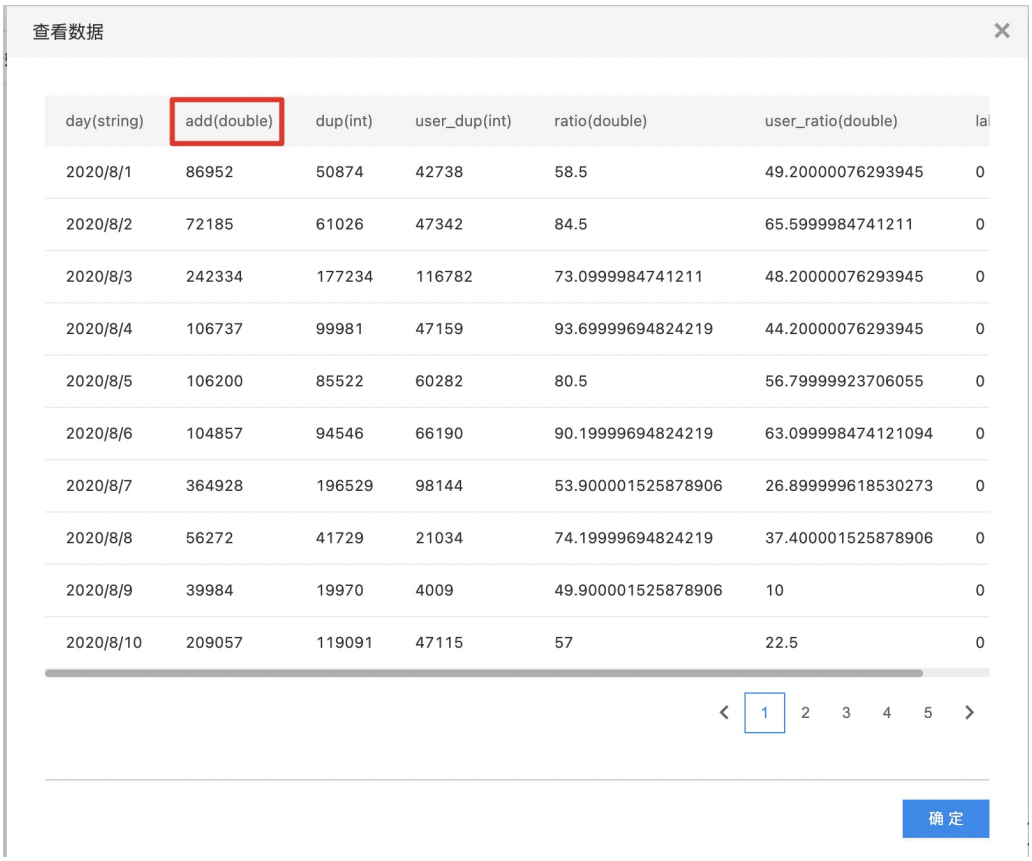
日期字符串支持格式，时间单位：yyyy(年)|MM(月)|dd(日)，不同的时间单位可以任意顺序组合且之间的连接是特殊符号即可（例如/、-、#、@等），例如yyyy-MM-dd（默认）。

使用示例

1. 如下图所示，构建算子结构，选择需要转换数据类型的列。



2. 查看数据类型转换结果。



Join组件

输入两个数据集，用户选择两个数据集用于Join的Key键列和指定的Join方式进行数据连接。

输入

- 输入两个数据集，用户分别选择左侧与右侧的数据集连接键列(单选)，再分别选择左侧和右侧的数据集输出列(多选)，最后选择Join连接方式。

输出

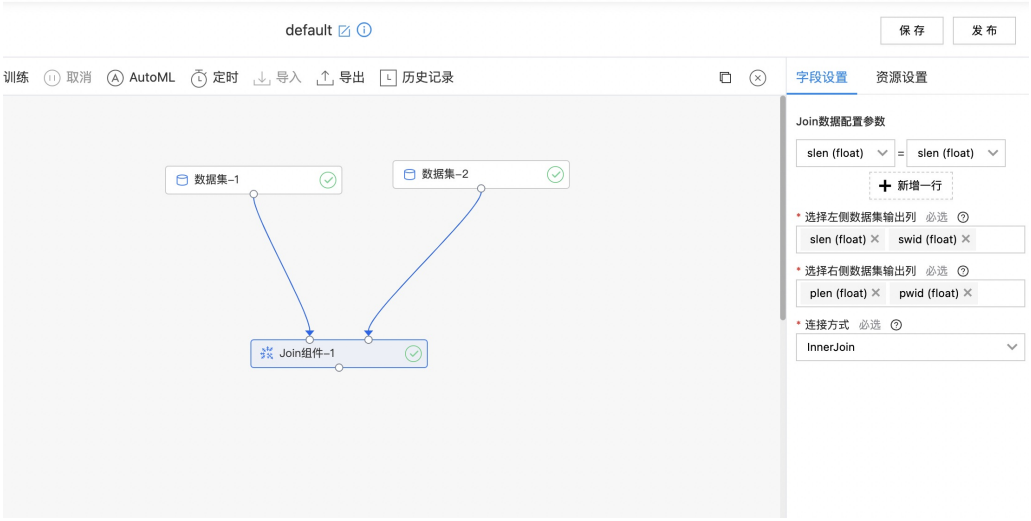
- 输出Join操作后的结果数据集。

字段参数

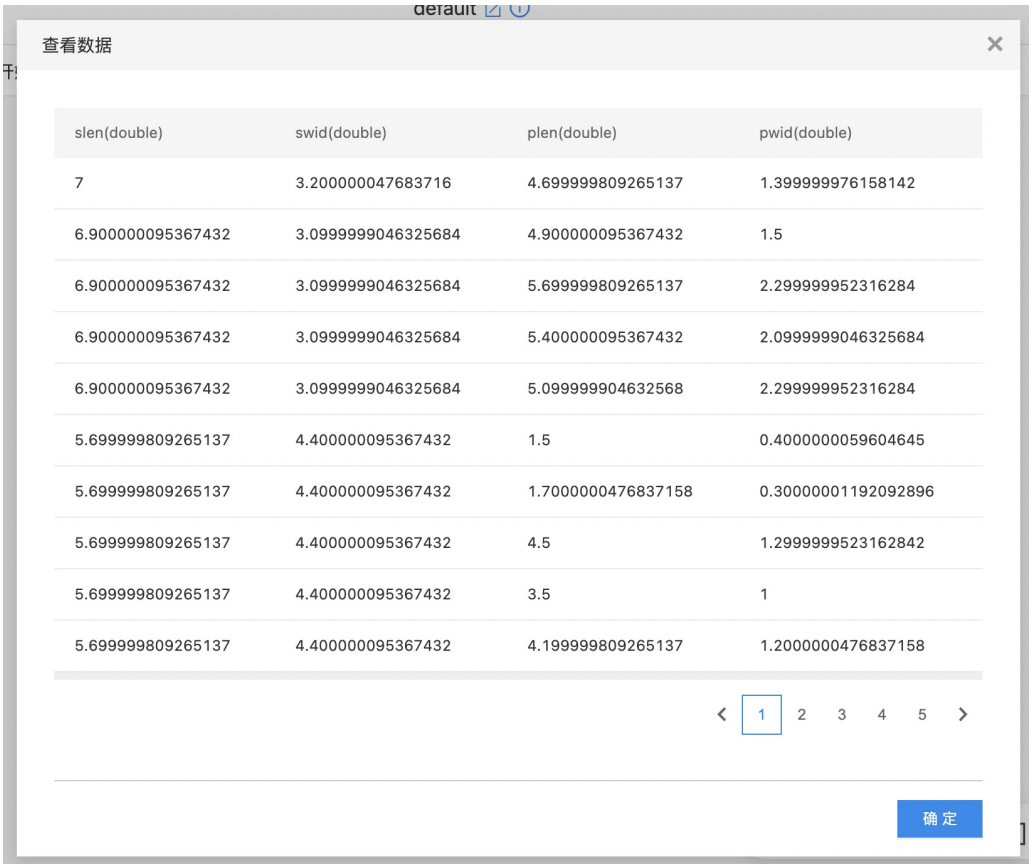
参数名称	是否必选	参数描述	默认值
左侧数据集连接键列	是	选择左侧的数据集连接键列，支持多键列	无
右侧数据集连接键列	是	选择右侧的数据集连接键列，支持多键列	无
选择左侧数据集输出列	是	选择左侧数据集需要的输出列(多选)，选择的输出列名称不能重复	无
选择右侧数据集输出列	是	选择右侧数据集需要的输出列(多选)，选择的输出列名称不能重复，当与左侧数据集列名称重复时，列名称默认会加前缀right_	无
连接方式	是	两个输入数据集的Join连接方式，支持FullOuterJoin - 全连接、InnerJoin - 内连接、LeftOuterJoin - 左连接、RightOuterJoin - 右连接	InnerJoin

使用示例

1. 如下图所示，构建算子结构，选择需要join处理的列。



2. 查看结果数据集。



数据拆分

数据拆分组件可以根据拆分条件，将数据集拆分为两个数据集。

输入

- 输入一个数据集，选择数据拆分方式、拆分比例等参数。

输出

- 输出拆分后的两个数据集。

字段参数

参数名称	是否必选	参数描述	默认值
拆分方式	是	选择数据集的拆分方式名，支持：Rows - 行拆分、Times - 时间戳拆分、Date - 日期拆分	Rows
拆分比例	是	选择数据集的拆分比例，输入范围[0.01，0.99]	0.80
随机拆分	是	是否随机拆分开关	开启
随机种子	否	随机种子初始值	0
分层拆分	是	是否选择分层拆分	否
用于分层拆分的列名称	是	选择分层拆分后，拆分后的两个数据集内样本类别比例与拆分键列的分类比例一致	无
时间列	是	时间拆分方式下，选择用于拆分的时间列列名，要求是时间戳类型，格式: 2006-01-01 00:00:00	无
日期列	是	用于拆分的日期列列名，要求是date类型，格式: 2006-01-01	无
时间阈值	是	时间戳阈值，拆分后小于此时间戳的为一组数据，大于等于此时间戳的为后一组数据，格式: 2006-01-01 00:00:00	2006-01-01 00:00:00
时间阈值格式	是	上述时间戳阈值的格式，按照 java.text.SimpleDateFormat 的标准填写	yyyy-MM-dd

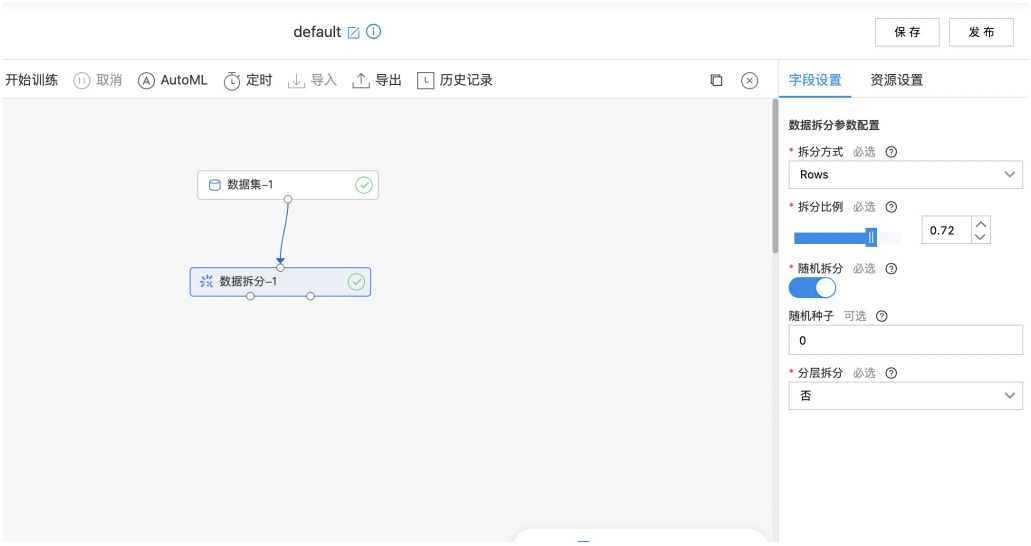
计算逻辑

拆分比例：假设数据集A样本数为100，拆分比例输入0.6，则拆分后的数据集B含样本数60，数据集C含样本数40。

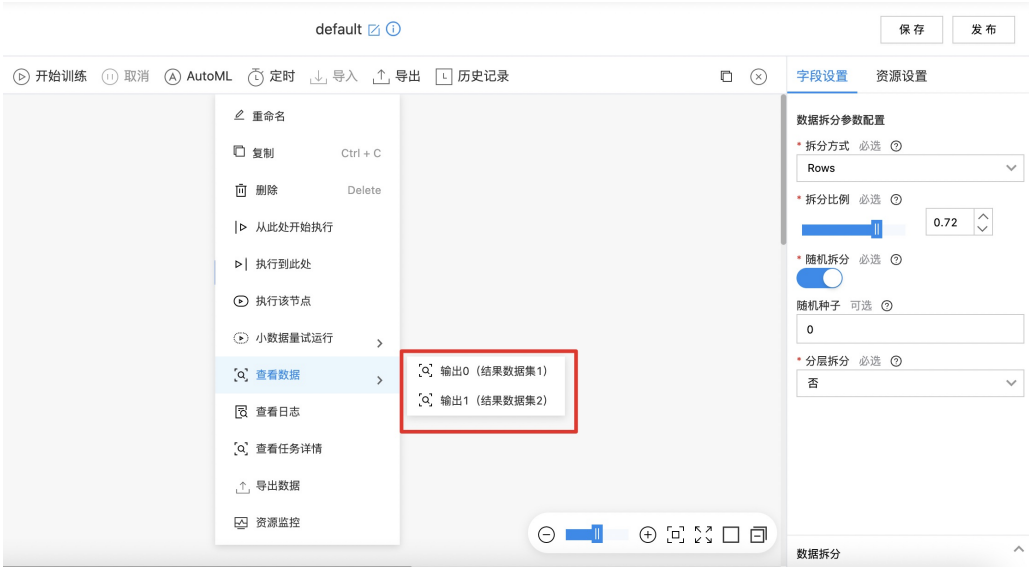
分层拆分：假设数据集A样本数为100，其中40个样本为类别0，60个样本为类别1，拆分比例输入0.6，选择分层拆分列为性别，该列的类别比例男:女为5:5，最终的拆分结果为数据集B包含60个样本，其中30个样本类别为0，30个样本类别为1，其余样本被拆分至数据集C。

使用示例

1. 如下图所示，构建算子结构，配置拆分方式。



2. 拆分为两个数据集，右键可查看结果。



Union组件

Union组件将两个数据集通过Union操作合并成一个数据集，要求两个数据集的列名、列数和schema一致。

输入

- 可输入四个数据集，用户分别选择对应数据集需要输出的列。

输出

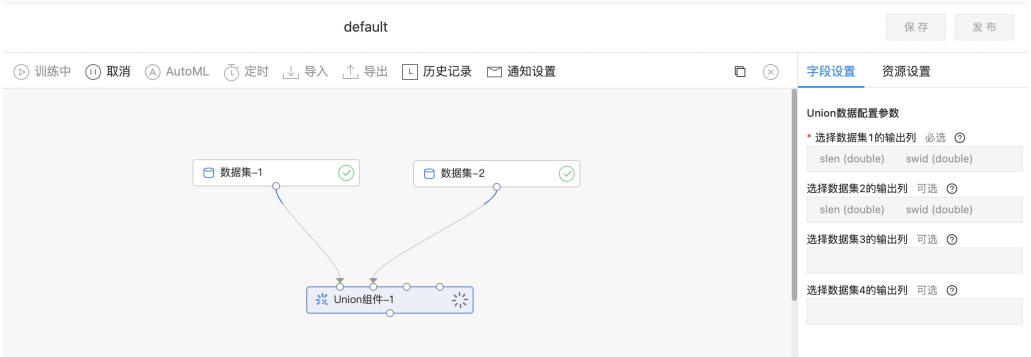
- 输出Union操作后的结果数据集。

字段参数

参数名称	是否必选	参数描述	默认值
选择左侧数据集输出列	是	选择左侧数据集输出列名称，选择的输出列名称不能重复	无
选择右侧数据集输出列	是	选择右侧数据集输出列名称，选择的输出列名称不能重复	无

使用示例

如下图所示，构建算子结构，选择需要输出的列。



SQL脚本

SQL脚本组件支持最多输入四个数据集，利用自定义SQL语句对数据集进行查询操作，查询结果作为最终输出数据。

输入

- 输入1~4个数据集，数据表名默认映射为data1、data2、data3、data4，表名也可自定义，编辑SQL语句查询数据表。

输出

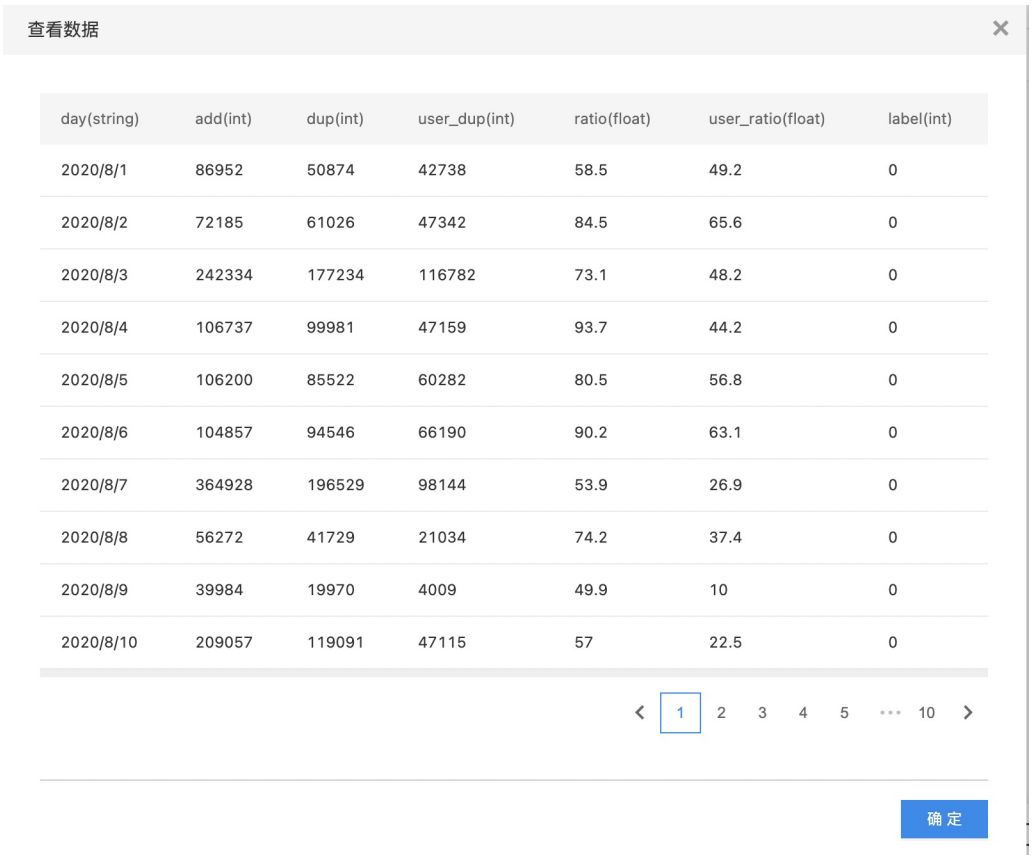
- 根据SQL语句查询结果输出数据集。

字段参数

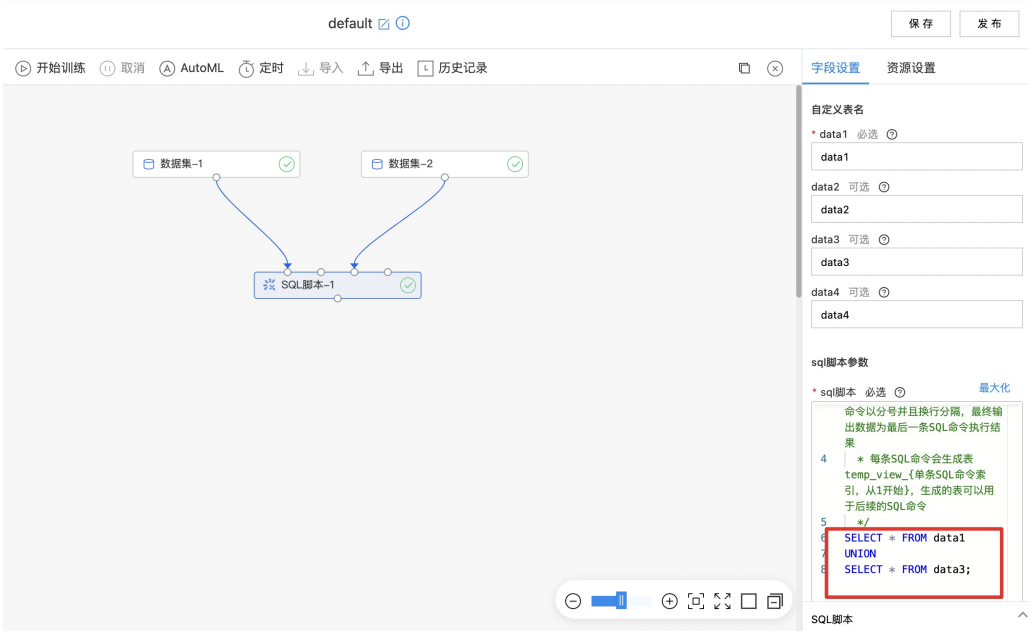
参数名称	是否必选	参数描述	默认值
data1	是	输入表名1，由字母、数字和下划线组成，第一个字符必须是字母	data1
data2	否	输入表名2，由字母、数字和下划线组成，第一个字符必须是字母	data2
data3	否	输入表名3，由字母、数字和下划线组成，第一个字符必须是字母	data3
data4	否	输入表名4，由字母、数字和下划线组成，第一个字符必须是字母	data4
sql脚本	是	输入数据表的名字已自动映射为 data1 data2 data3 data4，也可以自定义数据表名；支持多条SQL命令，每条SQL命令以分号并且换行分隔，最终输出数据为最后一条SQL命令执行结果；每条SQL命令会生成表 temp*view*{单条SQL命令索引，从1开始}，生成的表可以用于后续的SQL命令	

使用示例

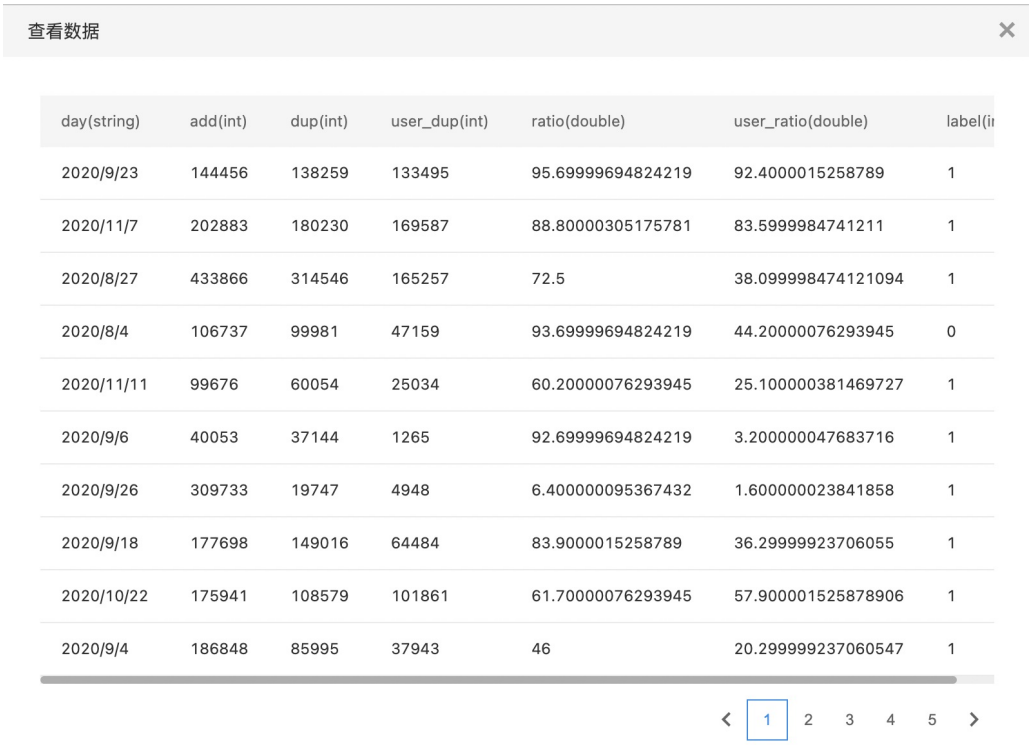
1. 拖入两个数据集组件：数据集-1、数据集-2



2. 拖入sql脚本组件，将数据集-1和数据集-2分别连接至组件的第一输入节点和第三输入节点，编写sql脚本并运行算子节点



3. 算子运行完成后查看输出数据



day(string)	add(int)	dup(int)	user_dup(int)	ratio(double)	user_ratio(double)	label(i
2020/9/23	144456	138259	133495	95.69999694824219	92.4000015258789	1
2020/11/7	202883	180230	169587	88.80000305175781	83.5999984741211	1
2020/8/27	433866	314546	165257	72.5	38.099998474121094	1
2020/8/4	106737	99981	47159	93.69999694824219	44.20000076293945	0
2020/11/11	99676	60054	25034	60.20000076293945	25.100000381469727	1
2020/9/6	40053	37144	1265	92.69999694824219	3.200000047683716	1
2020/9/26	309733	19747	4948	6.400000095367432	1.600000023841858	1
2020/9/18	177698	149016	64484	83.9000015258789	36.29999923706055	1
2020/10/22	175941	108579	101861	61.70000076293945	57.900001525878906	1
2020/9/4	186848	85995	37943	46	20.299999237060547	1

特征探索工具

Featuretools 使用 DFS（深度特征合成）进行自动化特征工程。可以结合原始数据以及对数据的理解为机器学习和预测建模构建有意义的特征，它通过时间和关系数据集自动创建特征。

输入

- 输入最多四个数据集。用于构建时间以及关系数据集（实体集）。

输出

- 输出是一个数据集，包含由DFS自动衍生的特征，可以选择与某个（非空）输入数据集进行合并输出。

算子参数

参数名称	是否必选	参数描述	默认值
输入实体1	是	加载第1个输入（列）数据作为一个实体，并添加至实体集	无
索引设置模式	是	提供两种模式：生成模式、选择模式，选择模式需要选择索引列	无
时间索引列	否	输入实体1的时间索引列，需要存在于第1个输入列中，且需要是时间戳或日期类型	无
实体1_1	否	通过输入实体1创建实体1_1，并添加至实体集；输入实体1为其基础实体	无
索引列	是	输入实体1的索引列，需要存在于第1个输入列中，且需要是字符串或整数类型，且列中值唯一	无
附加变量	否	要从基础实体中删除并移至实体1_1的变量列表。需要包含于基础实体的数据框中；不能包含实体1_1及其基础实体的索引列；需要包含实体1_1的时间索引列；保留对应于索引列的数据	无
实体1_2	否	通过输入实体1或其已创建的实体1_1创建实体1_2，并添加至实体集	无
基础实体	是	实体1_2的基础实体，需要是输入实体1或其已创建的实体1_1	无
目标实体	是	要进行预测的实体，需要是实体集中已存在的实体	无
输出方式	是	提供5种输出方式：特征矩阵、第1个数据集左连接特征矩阵、第2个数据集左连接特征矩阵、第3个数据集左连接特征矩阵、第4个数据集左连接特征矩阵。如果选择左连接方式输出，则需要对应输入数据集存在	无

算子参数说明

特征探索工具组件最多可以输入4个数据集，上述表格仅是1个数据集的参数信息，与其余3个数据集的参数信息一致，不再赘述。

字段参数

参数名称	是否必选	参数描述	默认值
第1个输入列	是	深度特征合成使用的输入列，用于加载第1个输入数据作为一个实体，需要是非数组类型	无
第2个输入列	否	深度特征合成使用的输入列，用于加载第1个输入数据作为一个实体，需要是非数组类型	无
第3个输入列	否	深度特征合成使用的输入列，用于加载第1个输入数据作为一个实体，需要是非数组类型	无
第4个输入列	否	深度特征合成使用的输入列，用于加载第1个输入数据作为一个实体，需要是非数组类型	无

使用示例

首先对操作流程加以概述：

1. 首先创建包含**实体**（及关系）的**实体集**（实体：单个数据集对象；实体集：多个关联的数据集对象）

   1)先创建一个空的实体集 - 后端已实现（无需关心）

   2)然后往实体集中添加实体：有两种添加实体的方式 - 需前端配置参数 - 直接加载单个数据集为一个实体：输入实体 - 通过已有实体创建一个新的实体：输入实体创建实体、输入实体创建的实体创建实体（包含实体间的一个关系，关系是由创建过程 中新实体索引列指定，创建的新实体为父实体，原实体为子实体，父实体数据与子实体数据针对索引列是一对多的关系）

3)添加关系：关联两个实体（父实体与子实体通过关联列构成数据一对多的关系） - 需前端配置参数

2. DFS使用包含实体（及关系）的实体集：DFS会通过实体集及要进行预测的实体自动构建特征

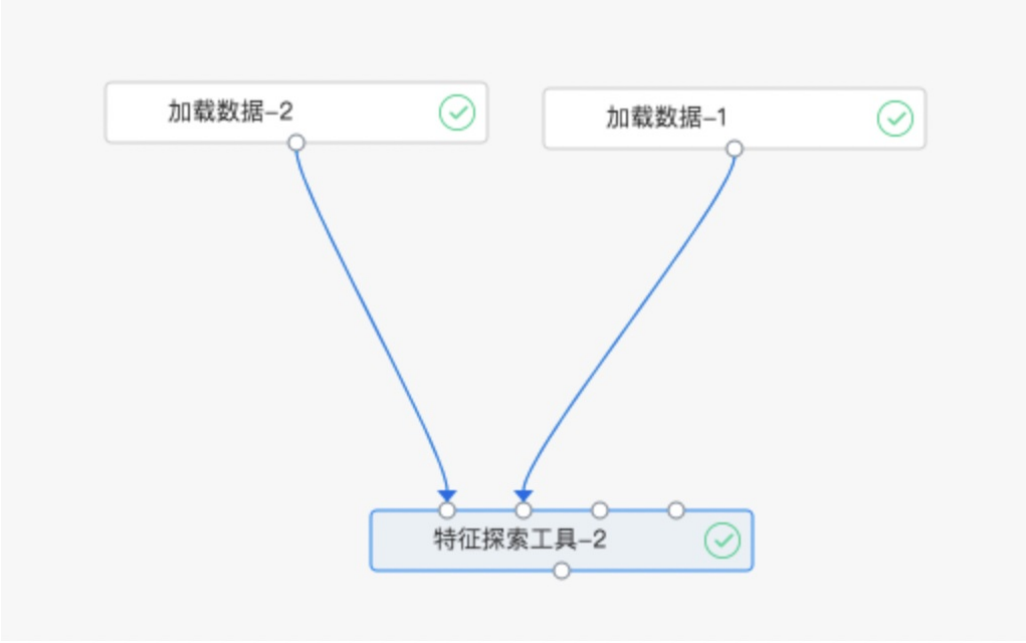
1)选择实体集入口：目标实体 - 需前端输入实体集中已存在的实体（名称）

2)输出方式：DFS自动构建的特征或其连接输入特征 - 需前端选择

组件说明：最多支持4个输入数据集（用于创建输入实体）- 最多创建4个输入实体，每个输入实体最多关联创建2个实体，最多添加4个实体间关系

下面给出详细的配置图例：

1. 算子组件连接图：



2. 参数配置图：


参数设置
字段设置
资源设置
组件配置
* 第1个输入列 必选
transaction_id (bigint)
session_id (bigint)
transaction_time (timestamp)
product_id (bigint)
amount (double)
customer_id (bigint)
device (string)
session_start (timestamp)
zip_code (string)
join_date (timestamp)
date_of_birth (timestamp)
第2个输入列 可选
product_id (bigint)
brand (string)
第3个输入列 可选
第4个输入列 可选


参数设置
字段设置
资源设置
第1个输入参数配置
* 输入实体1 必选
input_entity_1
* 输入实体1索引设置模式 必选
选择模式
* 索引列 必选
transaction_id (bigint)
时间索引列 可选
transaction_time (timestamp)
实体1_1 可选
entity_1_1
* 索引列 必选
session_id (bigint)
时间索引列 可选
session_start (timestamp)
附加变量 可选
device (string)
customer_id (bigint)
zip_code (string)
session_start (timestamp)
join_date (timestamp)


参数设置
字段设置
资源设置
实体1_2 可选
entity_1_2
* 基础实体 必选
entity_1_1
* 索引列 必选
customer_id (bigint)
时间索引列 可选
join_date (timestamp)
附加变量 可选
zip_code (string)
join_date (timestamp)
第2个输入参数配置
输入实体2 可选
input_entity_2
* 输入实体2索引设置模式 必选
选择模式
* 索引列 必选
product_id (bigint)



Pyml脚本

使用Pyml脚本组件处理数据。

输入

- 支持输入四个数据集，需要编写处理数据的Python代码。

输出

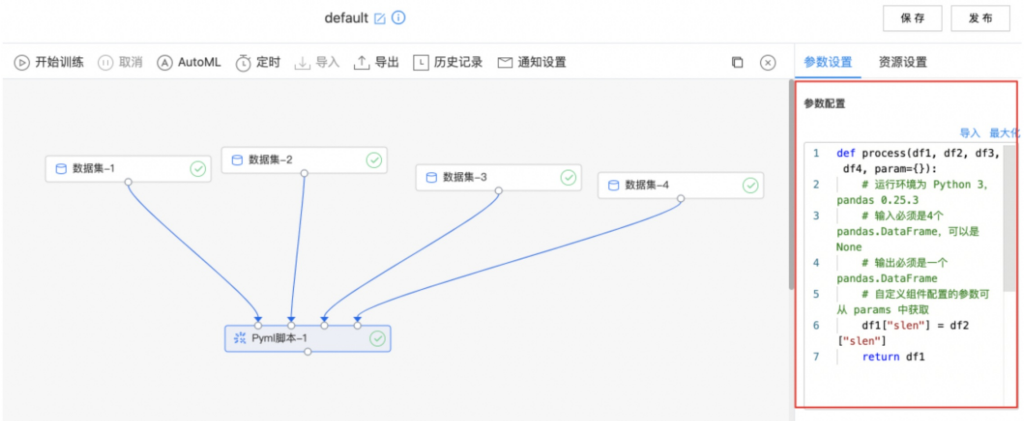
- 处理后的数据集。

算子参数

参数名称	是否必选	参数描述	默认值
Python代码编辑窗口	是	在编辑窗口中编辑处理数据脚本	无

使用示例

1. 编写处理数据的Python代码，等待算子运行成功。



2. 查看数据处理结果。

查看数据

slen(double)	swid(double)	plen(double)	pwid(double)	class(string)
5.099999904632568	3.5	1.399999976158142	0.20000000298023224	Iris-setosa
4.900000095367432	3	1.399999976158142	0.20000000298023224	Iris-setosa
4.699999809265137	3.200000047683716	1.2999999523162842	0.20000000298023224	Iris-setosa
4.599999904632568	3.0999999046325684	1.5	0.20000000298023224	Iris-setosa
5	3.5999999046325684	1.399999976158142	0.20000000298023224	Iris-setosa
5.400000095367432	3.9000000953674316	1.7000000476837158	0.4000000059604645	Iris-setosa
4.599999904632568	3.4000000953674316	1.399999976158142	0.30000001192092896	Iris-setosa
5	3.4000000953674316	1.5	0.20000000298023224	Iris-setosa
4.400000095367432	2.9000000953674316	1.399999976158142	0.20000000298023224	Iris-setosa
4.900000095367432	3.0999999046325684	1.5	0.10000000149011612	Iris-setosa

1 2 3 4 5

确定

PySpark导出数据

使用PySpark脚本导出数据。

输入

- 输入一个数据集，需要编写导出数据的Python代码。

输出

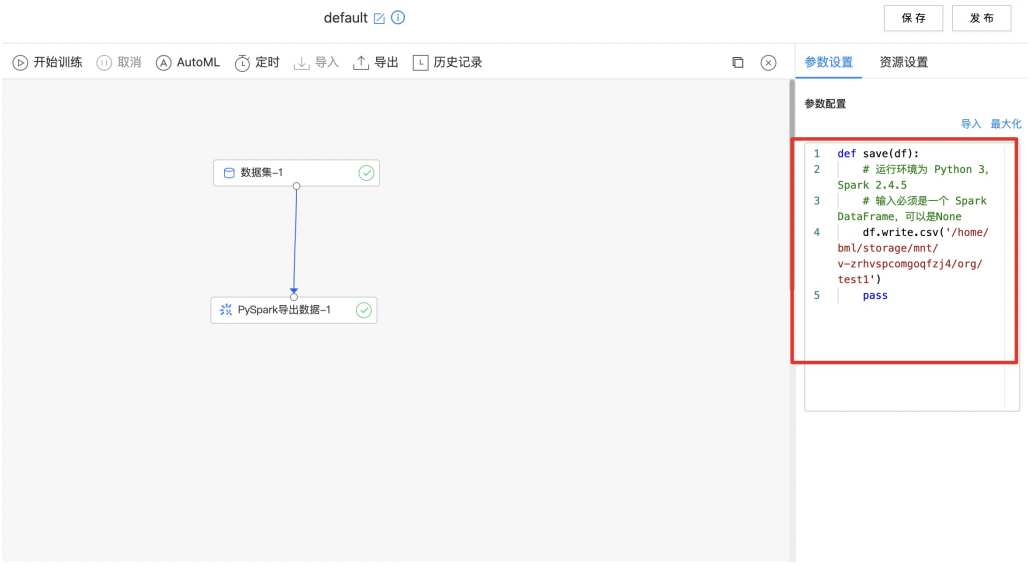
- 导出数据集。

算子参数

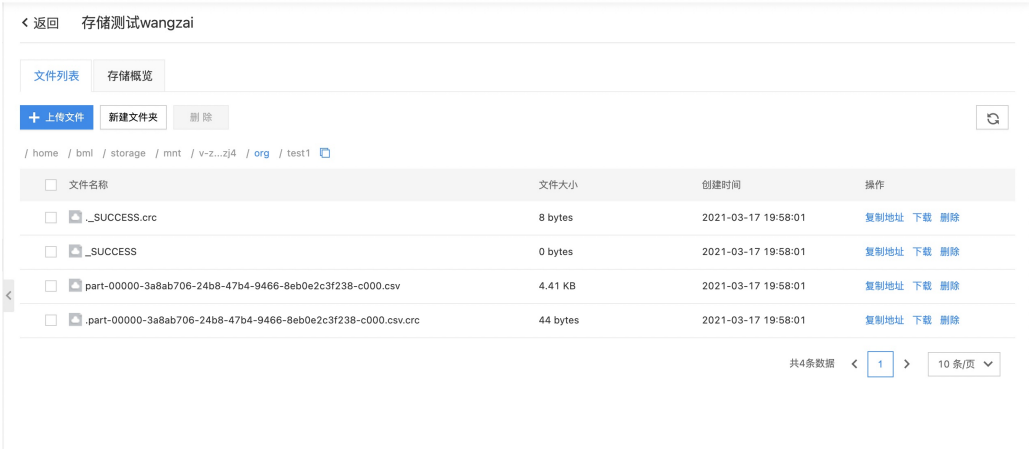
参数名称	是否必选	参数描述	默认值
Python代码编辑窗口	是	在编辑窗口中编辑导出数据脚本	无

使用示例

1. 编写数据导出Python代码，等待算子运行成功。



2. 到相应路径查看导出的数据文件



PySpark脚本

使用PySpark脚本导出数据。

输入

- 最多输入四个数据表。

输出

- 输出一个数据表。

算子参数

参数名称	是否必选	参数描述	默认值
Python代码编辑窗口	是	在编辑窗口中编辑数据处理脚本	无

使用示例

1. 拖入数据集组件，选择数据集

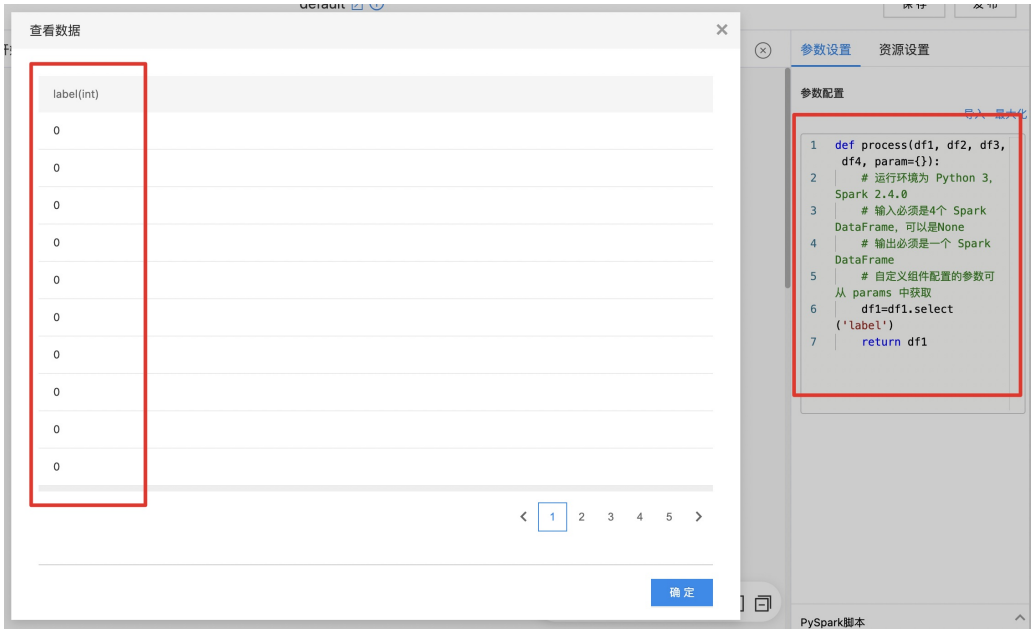
查看数据

day(string)	add(int)	dup(int)	user_dup(int)	ratio(float)	user_ratio(float)	time(int)	label(int)
2020/8/1	86952	50874	42738	58.5	49.2	175500	0
2020/8/2	72185	61026	47342	84.5	65.6	175500	0
2020/8/3	242334	177234	116782	73.1	48.2	175500	0
2020/8/4	106737	99981	47159	93.7	44.2	175500	0
2020/8/5	106200	85522	60282	80.5	56.8	175500	0
2020/8/6	104857	94546	66190	90.2	63.1	175500	0
2020/8/7	364928	196529	98144	53.9	26.9	175500	0
2020/8/8	56272	41729	21034	74.2	37.4	175500	0
2020/8/9	39984	19970	4009	49.9	10	175500	0
2020/8/10	209057	119091	47115	57	22.5	175500	0

1 2 3 4 5 ••• 10

确 定

2. 拖入PySpark脚本组件，连接数据集组件后，在代码编辑窗口编写代码，运行查看结果



特征工程组件

特征工程

特征自动选择

使用特征选择方法，筛选出特征重要性前N的特征。如输入的特征数不足N时，则保留全部特征，目前支持三种特征选择方法：Gini增益、信息增益、信息值。

输入

- 输入一个数据集，选择需要筛选的特征列并设置挑选的特征数量与连续特征的分区数。

输出

- 输出筛选后的数据集与特征权值表。

算子参数

参数名称	是否必选	参数描述	默认值
特征选择方法	是	使用的特征选择方法；选择信息值时，标签列的唯一值数量必须为2，即二分类。	信息值
挑选TopN特征	是	根据特征重要性，保留TopN特征。如输入特征数不足TopN值，则保留全部特征 范围：[1, 2147483647]。	5
连续特征分区数	是	对连续特征进行分位数离散化的分区数 范围：[1, 2147483647]。	10

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	选择字段使用过滤法进行过滤，只能是数值或字符串类型。	无
标签列	是	标签列，标签列可以为字符串或整型。选择信息值时，标签列的唯一值数量必须为2，即二分类。	无

计算逻辑

- 使用特征选择方法，筛选出特征重要性前N的特征。如输入的特征数不足N时，则保留所有输入特征。
- 选择的特征列，只能是数值或字符串类型。
- 选择的数据标签列的值数量必须为2，即二分类。
- 对于连续特征可通过设置分区数可以进行离散化分箱操作。
- 当前支持的特征选择方法有：

1.Gini增益：根据Gini增益的值大小选择TopN个特征，其中：

Gini系数为：

$$Gini(D) = 1 - \sum_{i=1}^{n} p_i^2$$

Gini增益为：

$$\Delta Gini(X) = Gini(D) - Gini_X(D)$$，其中D表示全体样本，p_i表示每种类别出现的概率

2.信息增益：根据信息熵的值大小选择TopN个特征，其中：

信息熵为：

$$H(D) = -\sum_{i=1}^{n} p_i log p_i$$

信息增益为：

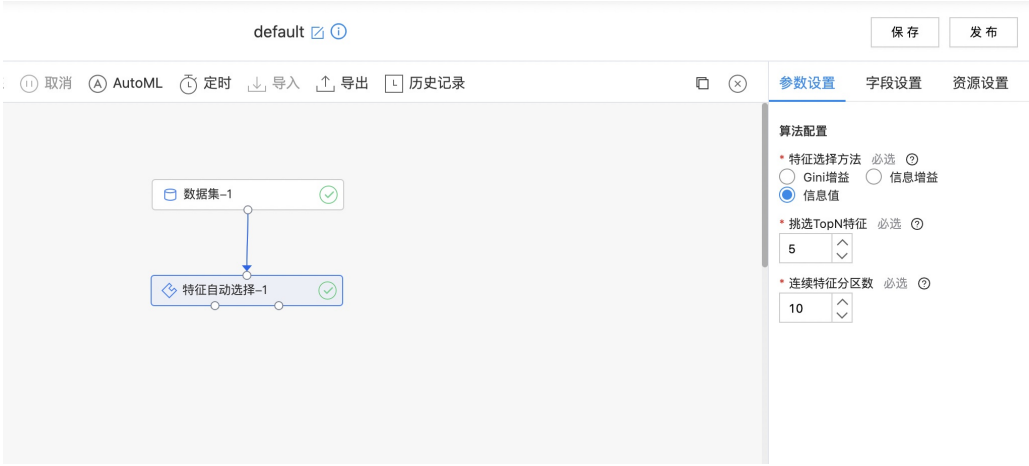
$$\Delta H(X) = H(D) - H(D|X)$$

3.信息值(依赖离散化结果)
根据信息的值大小选择TopN个特征，其中：

- 正样本占比：p0 = count0/total0
- 负样本占比：p1 = count1/total1
- 信息值：iv = sum((p0i-p1i)log(p0i/p1i), i=1...n)

使用示例

如下图所示，构建算子结构，配置特征选择方法。



特征自动衍生(spark)

依据数据集中的数值和字符串(枚举)类特征，自动衍生出大量新的特征，衍生方式包括数学运算衍生、笛卡尔积衍生和二值化衍生，衍生结果根据特征重要性排序，取 Top N 重要的衍生特征。

输入

- 输入是一个数据集，选择特征选择方法与数量，需要指定特征列（数值或字符串类型）和标签列（字符串或整型）。

输出

- 输出数据集和转化模型，其中数据集添加了指定数量的衍生特征。

算子参数

参数名称	是否必选	参数描述	默认值
特征选择方法	是	筛选衍生特征时使用的特征选择方法，支持Gini增益、信息增益。	
衍生特征数	是	根据特征重要性，保留部分衍生特征 范围：[1, 2147483647]。	
离散变量最大唯一值	是	以特征交叉的方式衍生离散变量时，允许的最大唯一值个数 范围：[16, 512]。	128

字段参数

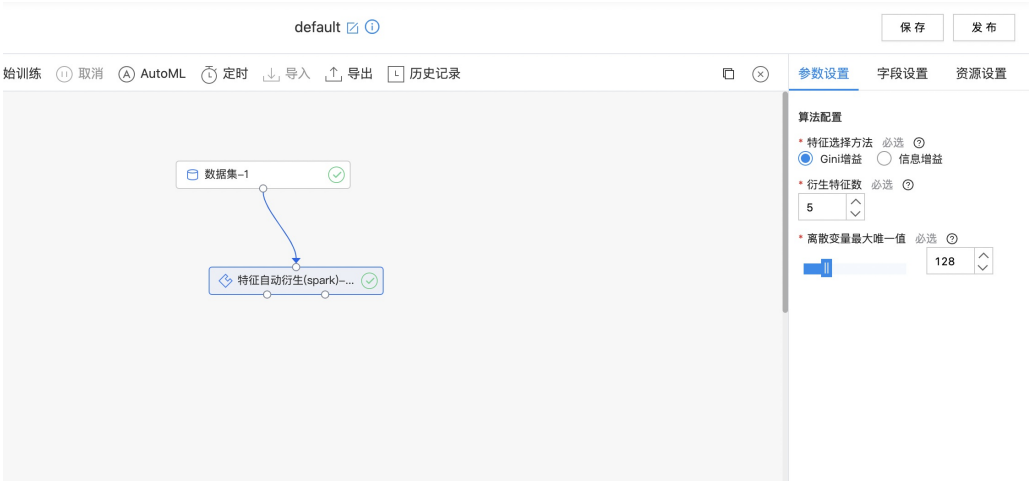
参数名称	是否必选	参数描述	默认值
特征列	是	要进行衍生的特征列，只能是数值或字符串类型。	无
标签列	是	用来筛选衍生特征，标签列可以为字符串或整型。	无

计算逻辑

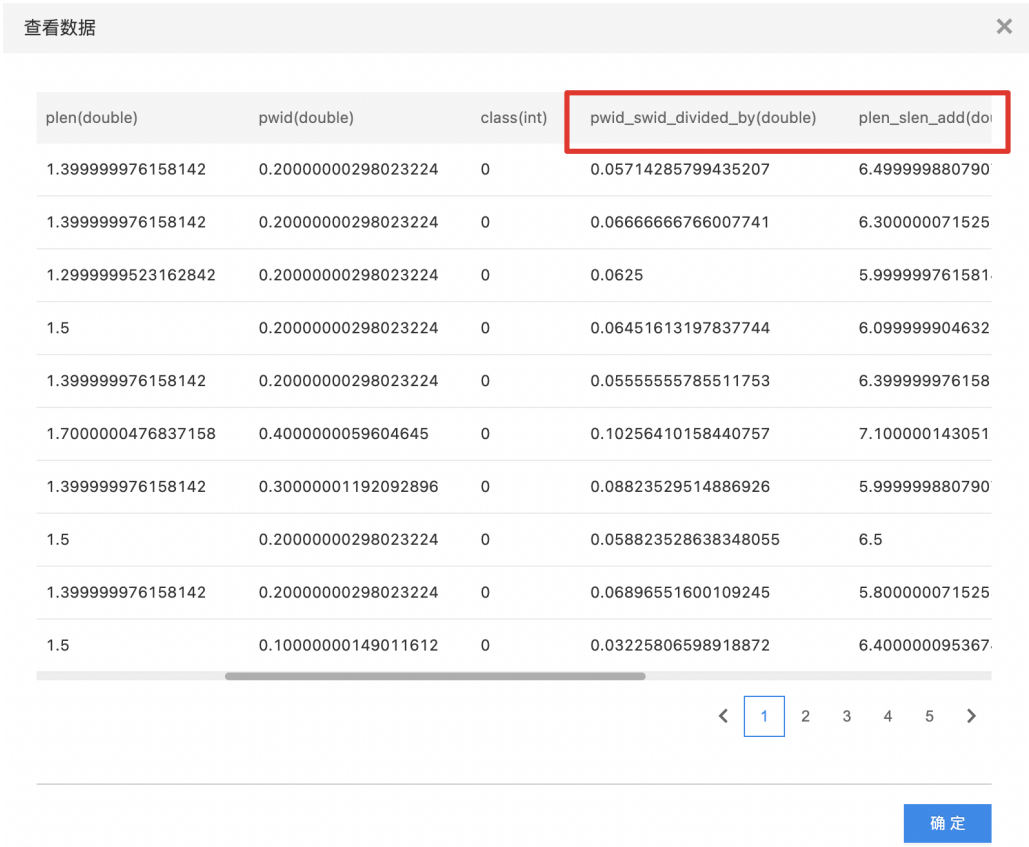
- 依据数据集中的数值和枚举类特征，自动衍生出大量新的特征。
- 将特征分为“数值”和“枚举”两大类，每一类两两成对组合进行特征衍生。
- 数值类做加减乘除的四则运算衍生，枚举类做特征交叉（笛卡尔积，可限制做笛卡尔积后的特征唯一值）。
- 将衍生得到的特征和原始的枚举特征一起，使用特征自动选择组件做重要性排序，取TopN重要的特征，并将这些特征中的枚举特征做onehot二值化处理。
- 当前支持的特征选择方法：Gini增益(GiniGain)、信息增益(InformationGain)，计算公式见特征自动选择组件。

使用示例

1. 如下图所示，构建算子结构，选择特征衍生的参考列，配置衍生方法。



2. 查看特征衍生结果。



特征交叉

特征交叉需要选择至少两个特征列，通过笛卡尔积运算衍生新的特征列。

输入

- 输入是一个数据集，指定需要特征交叉的列集合。

输出

- 第一个输出是结果数据集。
- 第二个输出是特征交叉变换(Transformation)，后续可以使用这个变换操作其它数据集。

字段参数

参数名称	是否必选	参数描述	默认值
输入列	是	需要特征交叉的列集合，输入两列或三列非列表类型列	无

计算逻辑

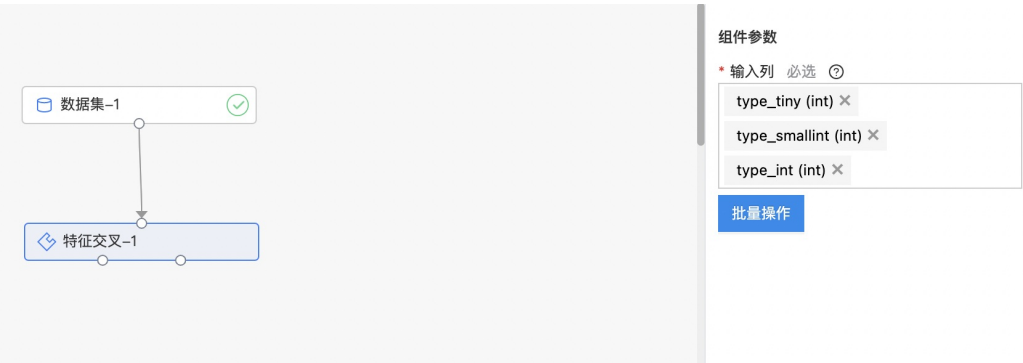
- 选择表中的两列或三列非列表类型的列进行交叉。

- 多个特征做笛卡尔积：

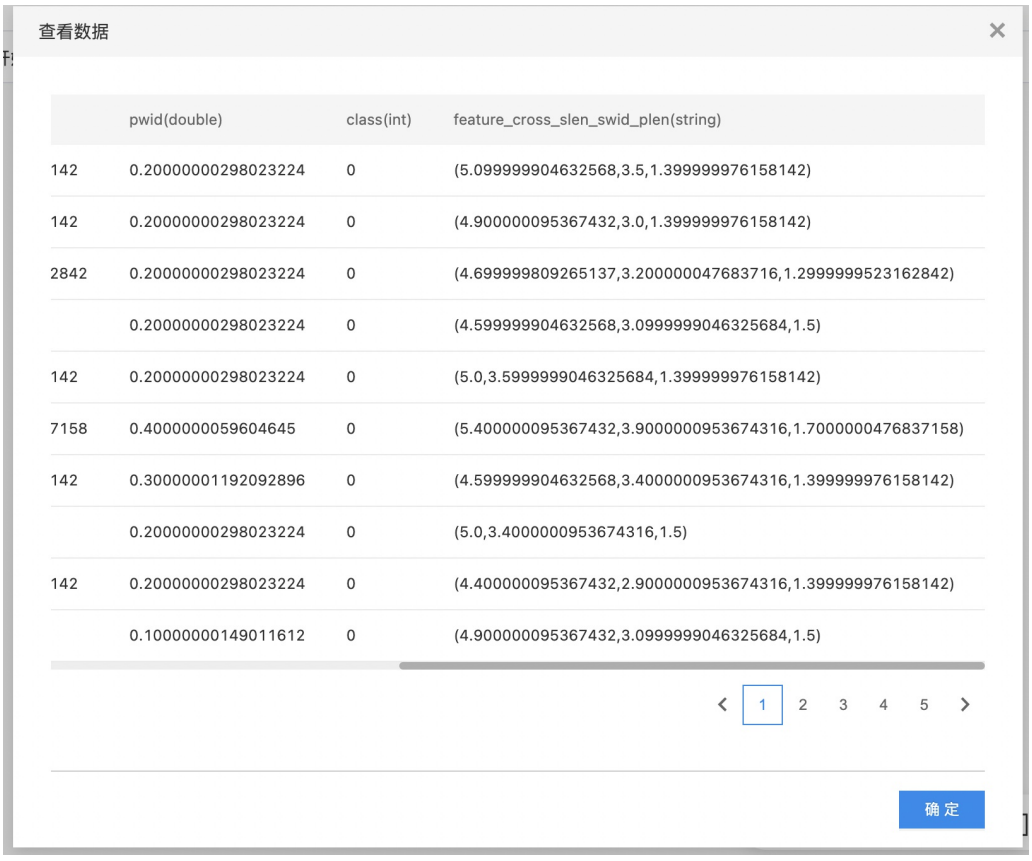
在输入数据集的基础上添加了"feature*cross_特征1_特征2*.._特征n"一列，如“feature_cross_sex_age_work”，列的类型是String，内容是输入特征的笛卡尔积，如 "(男,24,工人)"。

使用示例

1. 如下图所示，构建算子结构，选择需要特征交叉的列。



2. 查看特征交叉结果。



多项式扩展

多项式扩展需要选择要做多项式扩展的列，至少选择2列，至多选择5列（防止衍生特征维度爆炸），选择的列的类型需要是数值类型，用户还可以选择是否保留原始列。

输入

- 一个数据集，指定需要做多项式扩展的列集合。

输出

- 第一个输出是结果数据集。
- 第二个输出是多项式扩展变换(Transformation)，后续可以使用这个变换操作其它数据集。

算子参数

参数名称	是否必选	参数描述	默认值
多项式次数	是	设置多项式次数，对于m个特征的n次多项式特征衍生，得到的特征个数与m的n次方是同阶的，为了防止衍生特征过多/维度爆炸，需控制多项式次数 范围：[2, 8]	2

字段参数

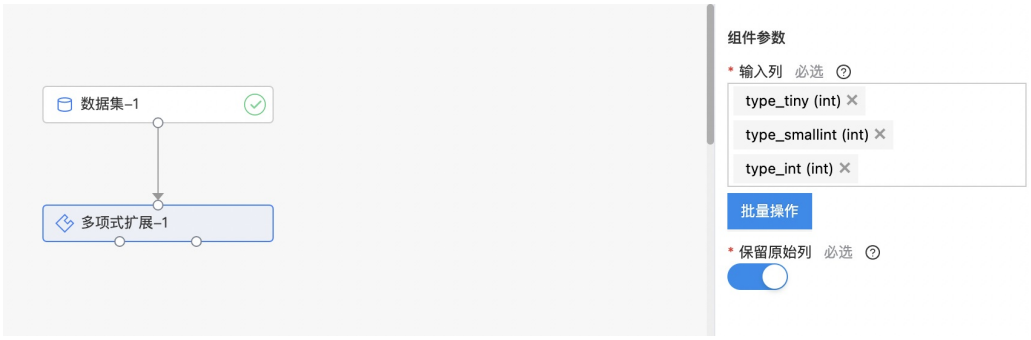
参数名称	是否必选	参数描述	默认值
输入列	是	需要多项式特征衍生的列集合，要求是数值类型；为了防止衍生特征过多，需控制输入列个数范围：[2, 5]	无
保留原始列	是	是否保留原始列，保留后输出数据集包含原始数据列	开启

计算逻辑

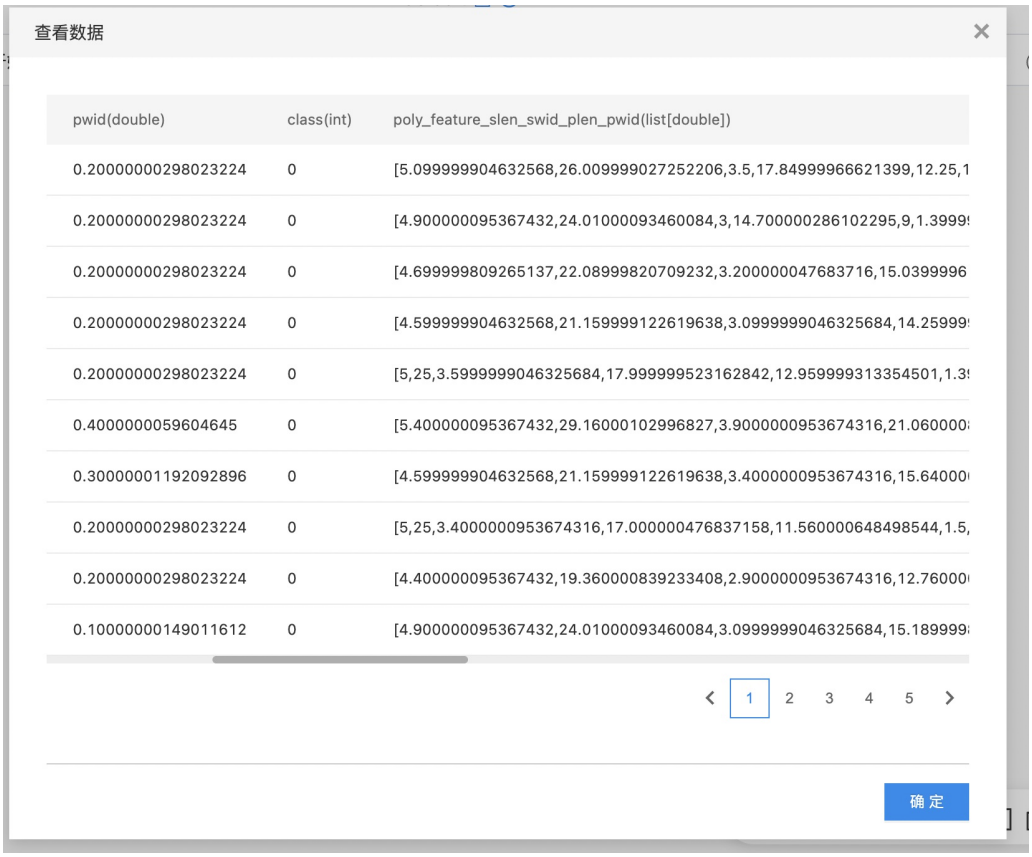
- 多项式扩展是指在多项式空间中执行特征展开。
- 以一个二维特征向量（x，y）为例，将其进行二次多项式扩展，则扩展后的特征向量为（x, x*x, y, x*y, y*y）。
- 输出结果在输入数据集的基础上添加了"poly*feature*原始列名(下划线分隔)"列，列的类型是vector,如果用户选择不保留原始列，输出数据集中会把原始列删掉。

使用示例

1. 如下图所示，构建算子结构，选择需要进行多项式扩展的列，配置扩展参数。



2. 查看多项式扩展结果。



时间戳特征衍生

时间戳特征衍生是指将时间戳或者日期类型特征列数据进行拆分处理，得到年、月、日等特征。

输入

- 输入是一个数据集，需要选择要做时间戳特征衍生的列，可以选择一列或多列，选择的列的类型需要是时间戳类型或日期类型。

输出

- 第一个输出是结果数据集。
- 第二个输出是时间戳特征衍生变换(Transformation)，后续可以使用这个变换操作其它数据集。

字段参数

参数名称	是否必选	参数描述	默认值
输入列	是	时间戳类型或日期类型列，可以输入多列	无

计算逻辑

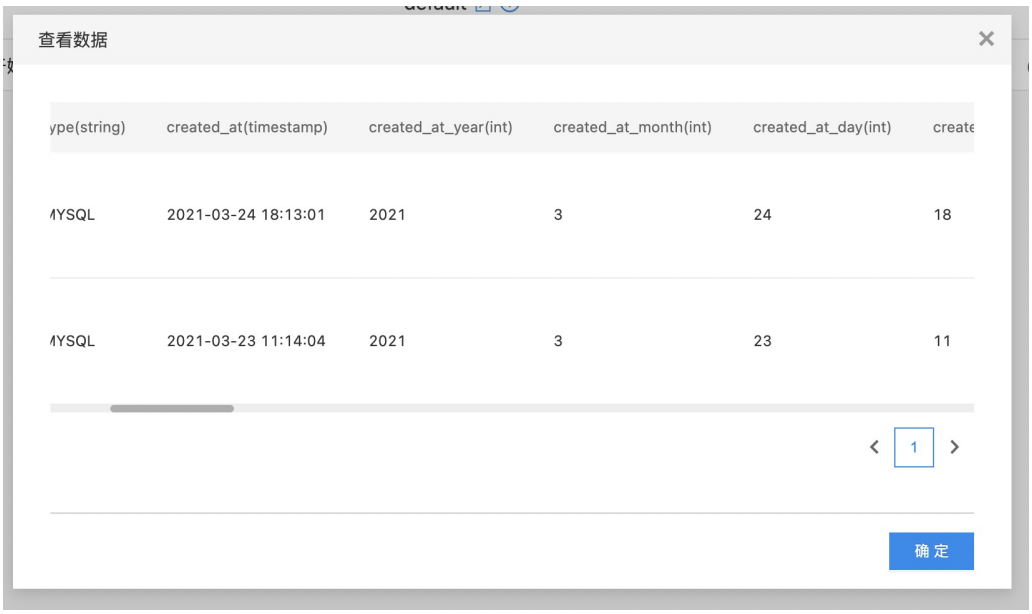
- 时间戳特征衍生是指将时间戳或者日期类型特征列数据进行拆分处理，得到年、月、日等特征。
  - **衍生的特征**
    - 时间戳类型：年、月、日、时、分、季度等等。
    - 日期类型：年、月、日、季度等等。

使用示例

1. 如下图所示，构建算子结构，选择需要特征衍生的时间戳列。



2. 查看时间戳衍生结果。



特征异常平滑

为了给数据集降噪，提升模型预测效果，特征异常平滑组件提供了三种特征平滑方式：Zscore平滑、百分位平滑、阈值平滑。

输入

- 输入一个数据集，选择要做平滑的列，选择的列需要是数值类型。

输出

- 第一个输出是结果数据集。
- 第二个输出是模型，连接特征工程预测组件进行预测。

算子参数

参数名称	是否必选	参数描述	默认值
特征异常平滑方式	是	特征异常平滑方式：Zscore平滑、百分位平滑、阈值平滑。	Zscore平滑
Zscore平滑方式中的阈值	是	设置阈值(theta)和数据计算出标准差(std)均值(mean)，将小于(-theta*std+mean)的值替换为(-theta*std+mean)和将大于(theta*std+mean)的值替换为(theta*std+mean)范围：[1.0, inf)。	3
最小百分位数位置	是	最小百分位数的位置，将小于最小百分位数的值替换为最小百分位数，最小百分位数的位置应小于等于最大百分位数的位置，如设置为50则表示为中位数，假设p为最小百分位数的位置对应的值为pv，则将小于pv的值设置为pv 范围：[0.0, 100.0]	0
最大百分位数位置	是	最大百分位数的位置，将大于最大百分位数的值替换为最大百分位数，最大百分位数的位置应大于等于最小百分位数的位置，如设置为50则表示为中位数，假设p为最大百分位数的位置对应的值为pv，则将大于pv的值设置为pv 范围：[0.0, 100.0]	100
最小阈值	是	将小于最小阈值的值替换为最小阈值，最小阈值应大约等于最大阈值。	-10000
最大阈值	是	将大于最大阈值的值替换为最大阈值，最大阈值应大于等于最小阈值。	10000
保留原始列	是	是否保留原始列。	开启

字段参数

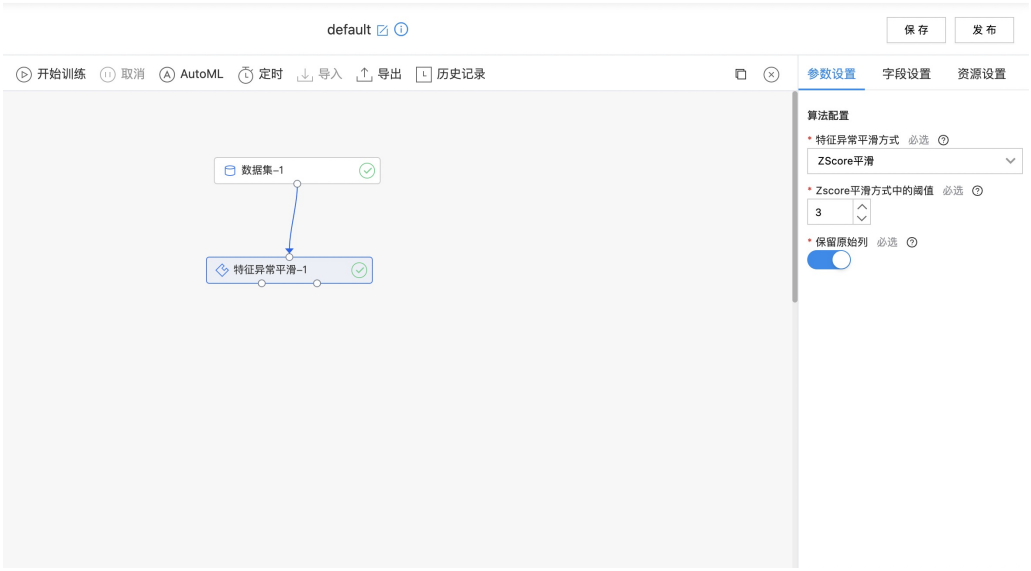
参数名称	是否必选	参数描述	默认值
输入列	是	选择字段进行特征异常平滑，需要是数值类型。	无

计算逻辑

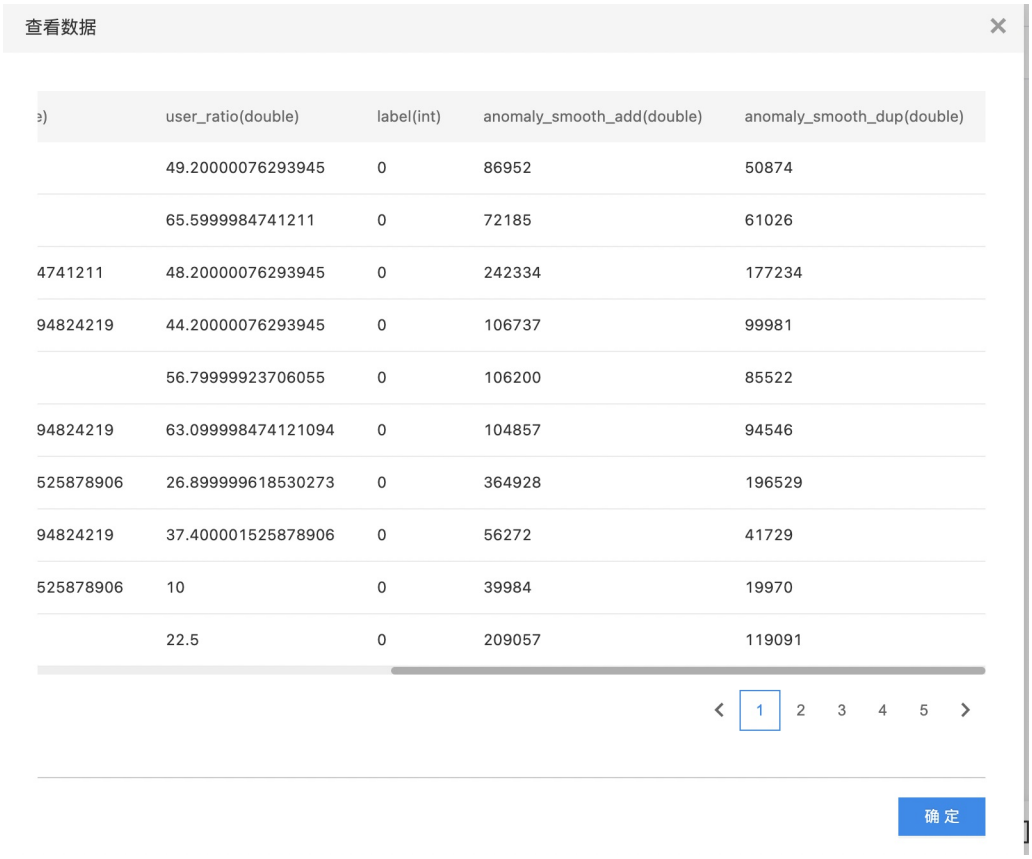
- 特征异常平滑是指对输入的特征使用异常判断方式，对判断为异常的数据进行处理。
- 当前实现的异常平滑方式有ZScore平滑、阈值平滑和百分位平滑:
  - ZScore平滑：利用设置的阈值(theta)和数据计算出的标准差(std)均值(mean)，将小于(-theta*std+mean)的值替换为(-theta*std+mean)和将大于(theta*std+mean)的值替换为(theta*std+mean)。
  - 阈值平滑：利用设置的最小阈值和最大阈值，将小于最小阈值的值替换为最小阈值，将大于最大阈值的值替换为最大阈值。
  - 百分位平滑：利用设置的最小百分位和最大百分位，将小于最小百分位数的值替换为最小百分位数，将大于最大百分位数的值替换为最大百分位数。

使用示例

1. 如下图所示，构建算子结构，选择需要特征异常平滑的列，配置平滑方式。



2. 查看特征异常平滑结果。



分箱

分箱组件可选择多列指定统一的规则对Double,Vector,Integer,String进行分箱，目前支持等频、等宽和自定义区间三种分箱方法。

输入

- 输入一个数据集，选择分箱方式与对应的分箱策略。

输出

- 第一个输出为结果数据集。
- 第二个输出是模型，连接特征工程预测组件进行预测。

算子参数

标签列	是	分箱的依据	无
参数名称	是否必选	参数描述	默认值
分箱方式	是	分箱方式，目前自定义区间仅对数值型列有效。不同分箱方式对数值类型列分箱方式不同；对字符串类型列分箱方式相同，都是将字符串映射成数字 参数可选值: EQUALWIDTH：等宽分箱 DECISIONTREE：决策树分箱 CUSTOMEDGES：自定义区间 QUANTILES：等频分位数分箱 CHIMERGE：卡方分箱	等频分位数分箱
分箱个数	是	指定输入列连续特征的默认分箱个数 范围：[2, inf)。	10
自定义分箱区间	是	连续特征自定义分箱区间，每条区间以英文分号隔开。每列的区间类似[col:]-inf,2.0,3.0,inf指定。第一条自定义区间为对所有选定列的默认规则。vector以(列名_i)表示vector的第i列	无
决策树最大深度	是	指定输入列连续特征的决策树模型最大深度，假设最大深度为k，那么对应的最大分箱数为pow(2,k) 范围：[1, inf)	5
自定义分箱个数	否	自定义分箱个数，对具体某一列的分箱个数进行指定。以英文分号间隔，每条规则以":"间隔，例如a:3;b:10。	无
是否保存原始列	是	是否保存原始列，为true时，处理过的列重命名 "bin_"前缀	关闭

字段参数

参数名称	是否必选	参数描述	默认值
输入列	是	分箱列，可以是数值、字符串或数值数组类型；依据标签列分箱方法：卡方分箱要求标签列是数值类型，决策树分箱要求标签列是字符串或数值类型	无

计算逻辑

对表中的列，进行分箱处理，将数据映射到对应的箱中，所选的列应是数值、字符串或数值数组类型

等宽分箱:

- 指定默认分箱个数后，按照列中的最小值到最大值之间，将数据N等分，数据分到每个箱中，和等频分箱相比每个箱中的数据量不固定。
- 自定义分箱个数，对具体某一列的分箱个数进行指定。以;间隔，每条规则以:间隔，例如`a:3;b:10`。

决策树分箱：

- 决策树分箱是一种自上而下的分箱方法，通过计算信息增益，我们可以对单个自变量的划分生成决策树。决策树的分裂节点就是每个箱的边界。最终得到的叶子节点就是每一个箱。

自定义区间

- 自定义区间分箱，目前仅对数据值列有效
- 连续特征自定义分箱区间，每条区间以;隔开。每列的区间类似`[col:]-inf,2.0,3.0,inf`指定。第一条自定义区间为对所有选定列的默认规则。例子`1.0,2.0;a_1:3.0, 4.0`即定义了[1.0,2.0]和[3.0,4.0]两个分箱区间

等频分位数分箱:

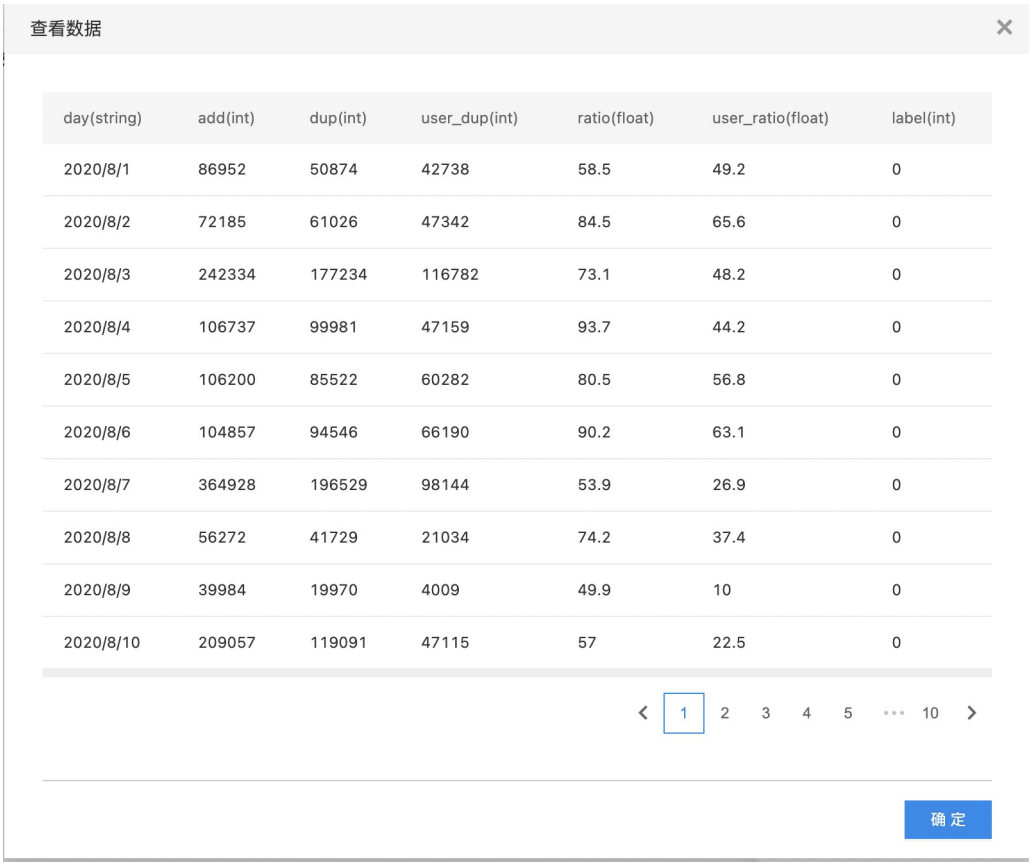
- 指定默认分箱个数后，将数据分箱到每个箱中并且每个箱中的数据量大致相等
- 自定义分箱个数，对具体某一列的分箱个数进行指定。以;间隔，每条规则以:间隔，例如`a:3;b:10`

卡方分箱：

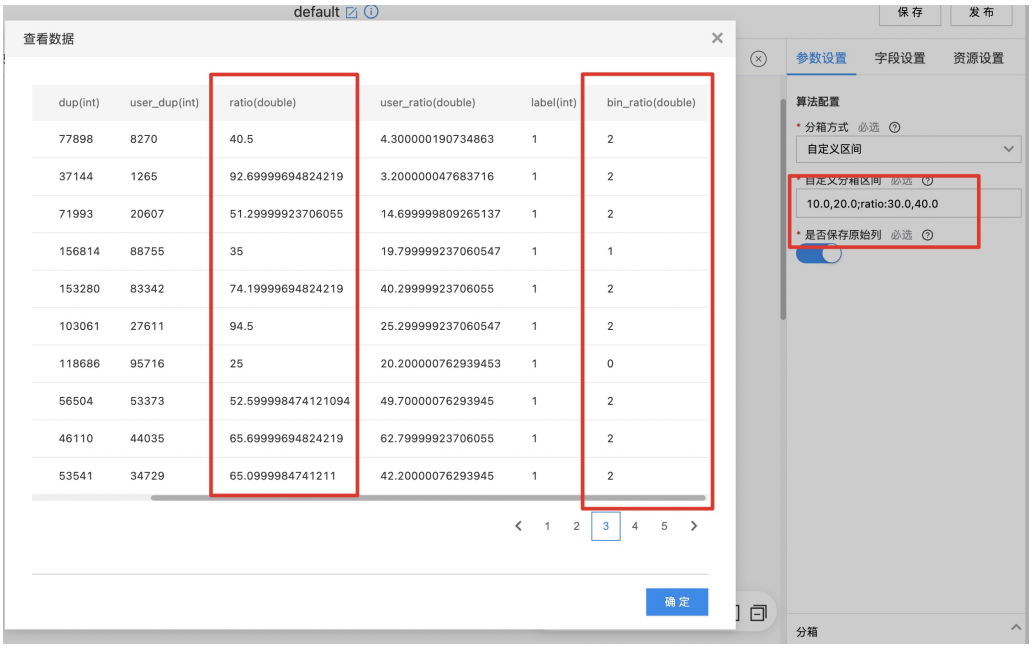
- 卡方分箱是一种自底向上的(即基于合并的)数据离散化方法。 先算出单变量每个取值的卡方值，再把卡方值相近的取值合并到一个箱里。

使用示例

1. 拖入数据集组件：



2. 设置自定义分箱区间，该示例针对ratio列进行分箱，可以从图例中看出分为了0，1，2三个分箱。



特征哈希

特征哈希将一组分类或数值特征投影到指定维度的特征向量中（通常比原始特征空间维度小得多），在特征维度过大的时候，容易产生特征稀疏的现象，导致模型收敛速度过慢、模型参数爆炸等问题，特征哈希用低维度向量表示高维度特征可以有效的缓解上述问题。

输入

- 输入一个数据集，需要选择要做特征哈希的列。输入列的类型需要是数值、字符串或布尔类型。
- 可以设置要视为分类特征的数值列（可选），用户还可以选择是否保留原始列。

输出

- 第一个输出是结果数据集，在输入数据集的基础上添加了"feature*hash*原始列名(下划线分隔)"列，列的类型是spark vector。如果用户选择不保留原始列，输出数据集中会把原始列删掉。

- 第二个输出是特征哈希变换（Transformation），后续可以使用这个变换操作其它数据集。

算子参数

参数名称	是否必选	参数描述	默认值
输出向量维度	是	特征数量。应该大于0（默认值=262144）。由于是使用哈希值的简单模来确定向量索引，因此建议设置此参数的值为2的幂；否则，这些特征将不会均匀地映射到向量索引。 范围：[1, inf)	262144

字段参数

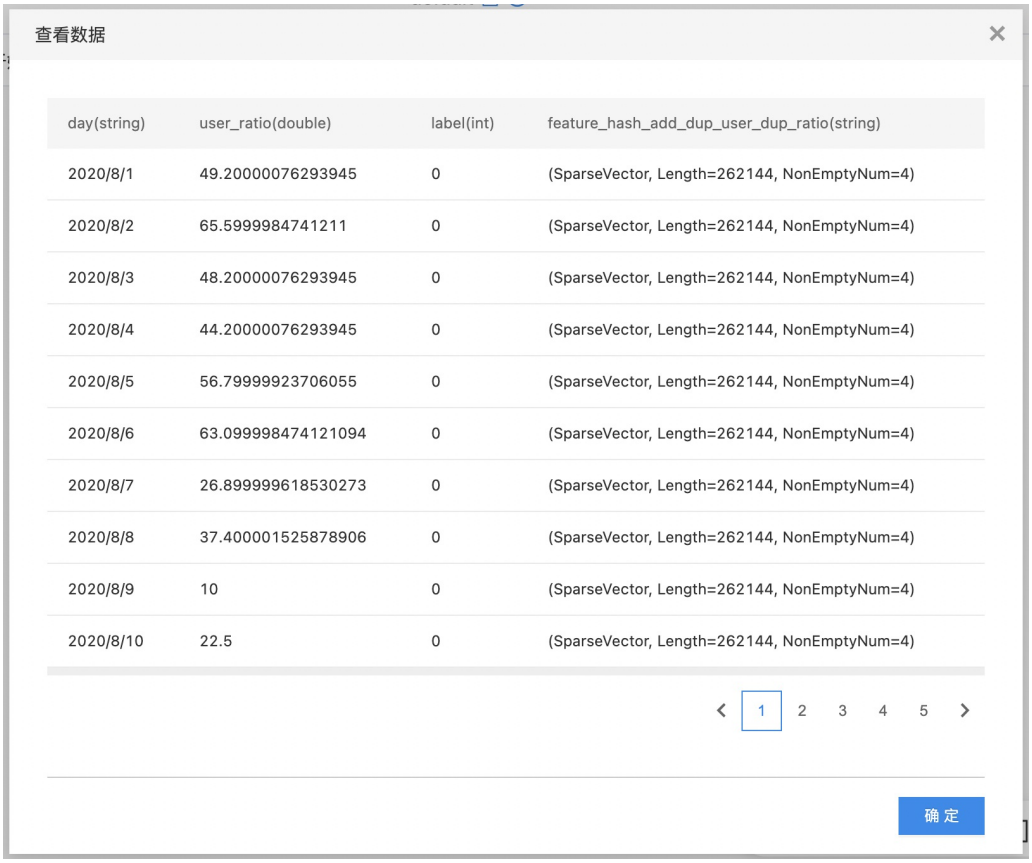
参数名称	是否必选	参数描述	默认值
输入列	是	需要特征哈希的列集合，要求不是数组类型。	无
要视为分类特征的数值列	否	默认情况下，只有字符串列和布尔列被视为分类特征，可以使用此参数显式指定要视为分类特征的数值列。注意，相关列也必须在 ' 输入列 ' 中设置。	无
保留原始列	是	是否保留原始列。	关闭

使用示例

1. 如下图所示，构建算子结构，选择需要进行特征哈希的列，配置输出向量维度。



2. 查看特征哈希结果。



OneHot(哑编码)

OneHot(哑编码)支持独热码和哑编码两种编码方式，两种编码方式都是为了帮助模型更好的处理属性数据问题，在一定程度上也起到扩充特征的作用。独热码从n个变量推论出所有类别，特征数据直观但存在特征冗余的情况；相对于独热码，哑编码从n-1个变量推论出n个类别，数据不够直观但不冗余，通常线性模型有截距项的时候推荐使用哑编码的方式处理特殊。

输入

- 输入一个数据集，需要选择要做onehot的列，可以选择一列或多列；选择的列的类型需要是String或数值类型，用户还可以选择是否保留原始列。

输出

- 第一个输出是结果数据集，在输入数据集的基础上添加了"onehot_原始列名"列，列的类型是Vector，内容是onehot的SparseVector。如果用户选择不保留原始列，输出数据集中会把原始列删掉。
- 第二个输出是Onehot变换（Transformation），后续可以使用这个变换操作其它数据集。

算子参数

参数名称	是否必选	参数描述	默认值
编码方式	是	哑编码会将训练集中未出现的类以零向量表示；独热编码会将每一类编码成一个特征维度。	哑编码

字段参数

参数名称	是否必选	参数描述	默认值
输入列	是	需要onehot的列集合，要求不是数组类型	无
保留原始列	是	是否保留原始列	关闭

计算逻辑

假设当前有一个数据集(人口学历统计)，其中四个样本示例如下：

姓名	学历
小明	小学
小红	初中
小刚	高中
小亮	大学

使用独热码结果如下：

姓名	小学	初中	高中	大学
小明	1	0	0	0
小红	0	1	0	0
小刚	0	0	1	0
小亮	0	0	0	1

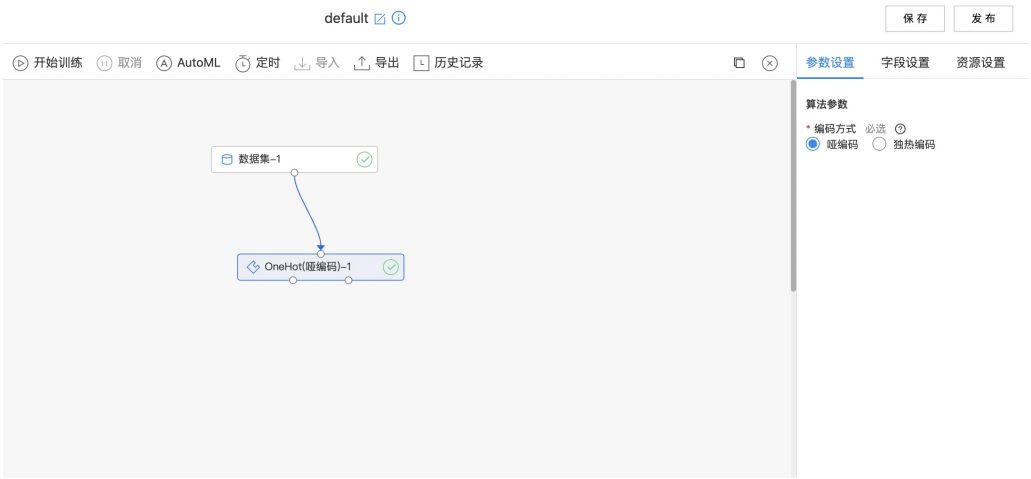
使用哑编码结果如下：

姓名	小学	初中	高中
小明	1	0	0
小红	0	1	0
小刚	0	0	1
小亮	0	0	0

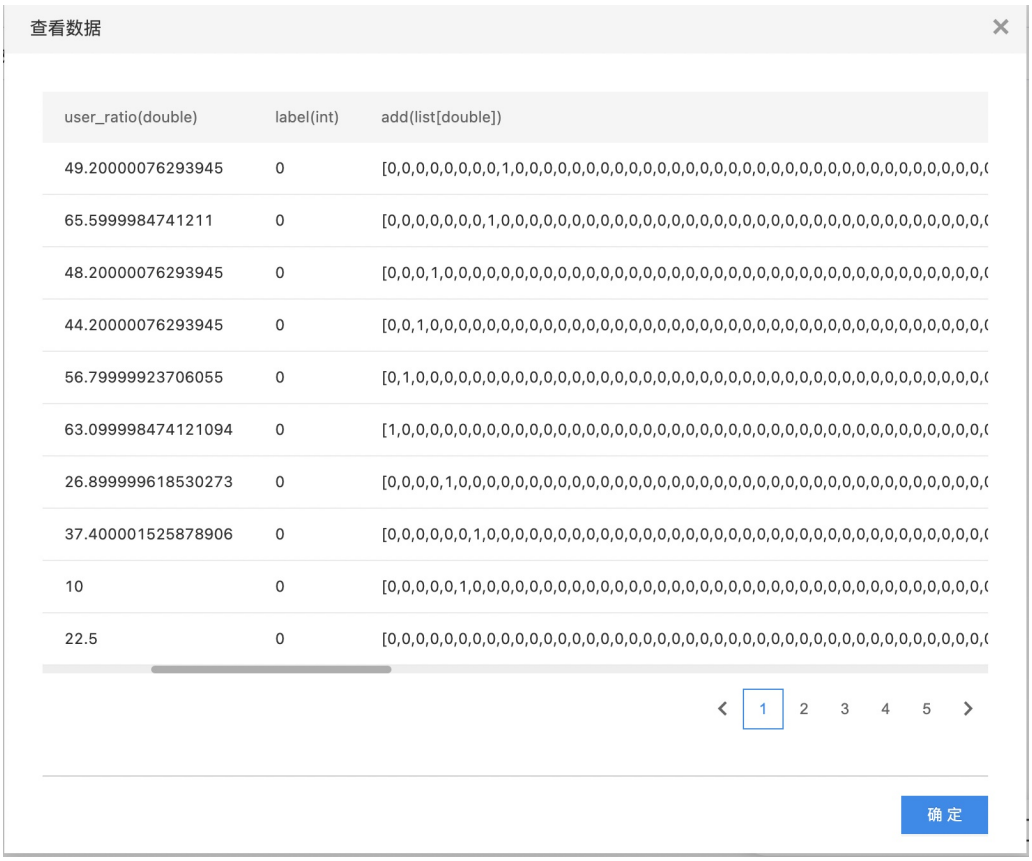
根据小亮与小明、小红、小刚三人的特征值不同可以推断出小亮的学历为大学。

使用示例

1. 如下图所示，构建算子结构，选择需要OneHot编码的列，选择编码方式。



2. 查看OneHot编码结果。



异常值处理

异常值处理组件选择列处理,需与选择的填充规则匹配其中浮点型或数值数组型支持NULL,NAN,NULLORNAN,CUSTOMVALUE;整型支持NULL,CUSTOMVALUE；字符串型支持NULL,NULLOREMPTY,CUSTOMVALUE。

输入

- 输入一个数据表，用户可选择多列指定统一的规则对异常值进行处理，也可以按格式输入多行自定义规则，但列名不允许重复。

输出

- 第一个输出为结果数据集。
- 第二个输出是模型，连接特征工程预测组件进行预测。

算子参数

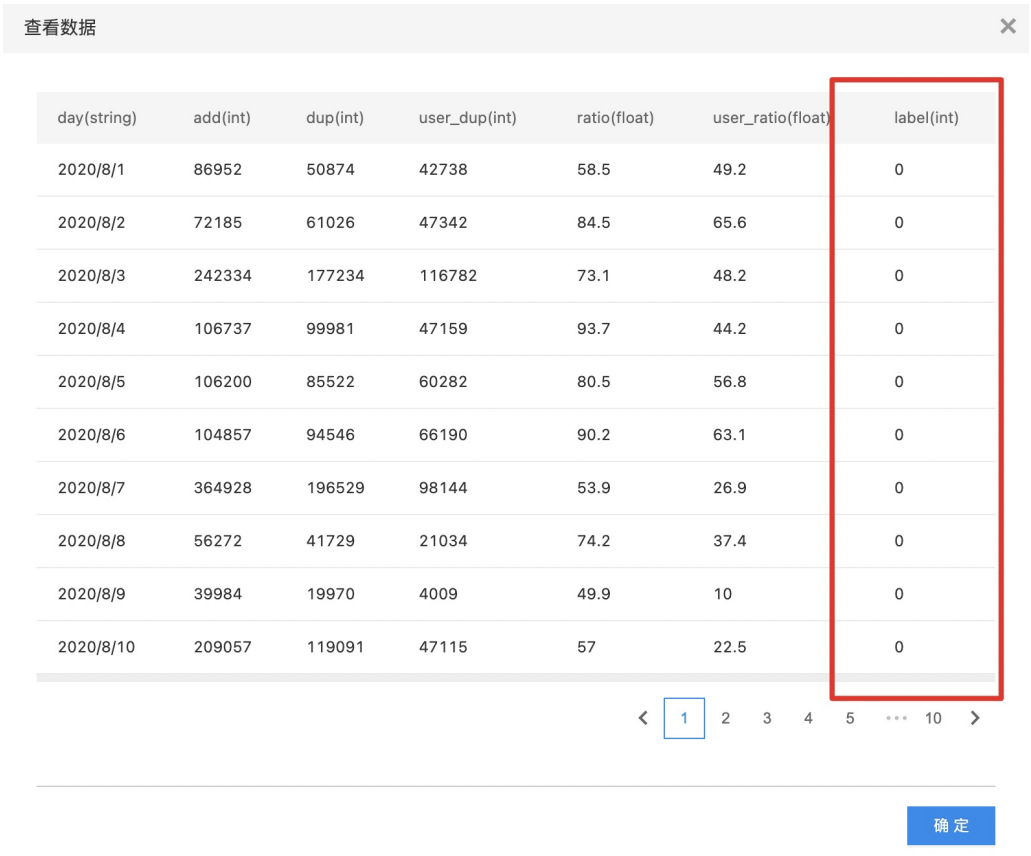
参数名称	是否必选	参数描述	默认值
异常值类型	是	异常值类型： CUSTOMVALUE(String和数值型) EMPTY(String) NAN(double,数值数组型) NULL(数值型,数值数组型和String) NULLOREMPTY(String) NULLORNAN(double,数值数组型)	NULL(数值型,数值数组型和String)
自定义异常值	是	自定义异常值,仅支持String和数值型	无
填充类型	是	使用的填充类型： CUSTOMVALUE(String和数值型) MAX(数值型和数值数组型) MEAN(均值，数值型和数值数组型) MEDIAN(中位数,数值型) MIN(数值型和数值数组型) MODE(众数,String和数值型) QUANTILE(数值型) RANDOM(数值型)	MIN(数值型和数值数组型)
自定义异常值处理规则	否	自定义异常值处理规则,输入的列名与输入列不能重复。以英文分号间隔，每条规则含5列以","间隔,包含col(列名),outlierType(异常值类型),customOutlier(自定义异常值),fillerMethod(填充类型),customFillValue(自定义填充值)。	无

字段参数

参数名称	是否必选	参数描述	默认值
输入列	是	选择字段进行异常值处理,需与选择的填充规则匹配。浮点型或数值数组型支持NULL,NAN,NULLORNAN,CUSTOMVALUE；整型支持NULL,CUSTOMVALUE；字符串型支持NULL,NULLOREMPTY,CUSTOMVALUE	无

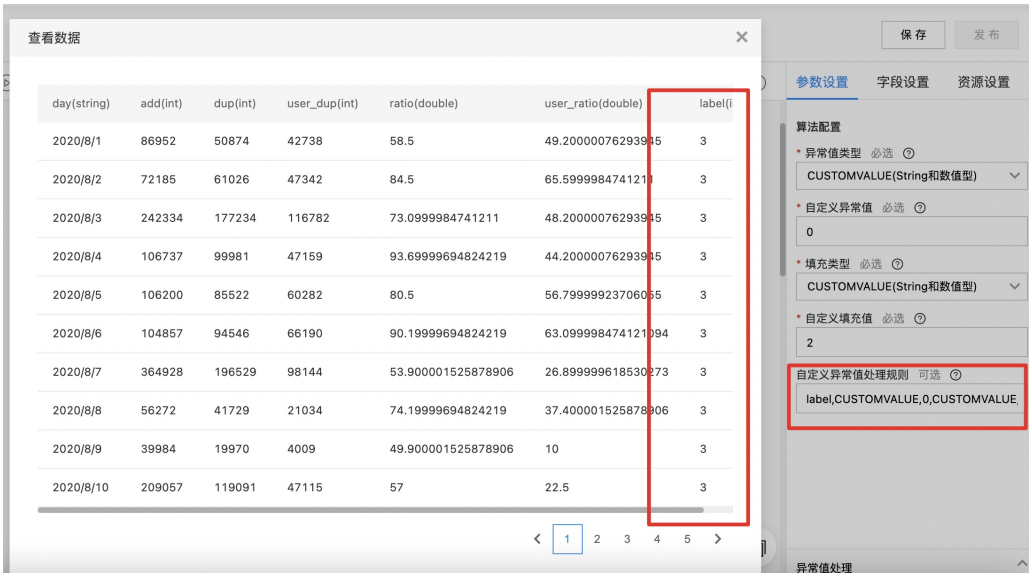
使用示例

1. 拖入数据集组件，本示例以**自定义**处理label列为样例：



2. 输入自定义规则参数：**label**(列名),**CUSTOMVALUE**(异常值类型),**0**(自定义异常值),**CUSTOMVALUE**(填充类型),**3**(自定义填充值);

从图例中可知，自定义异常值0已经被替换为3。



PCA主成分分析

PCA主成分分析法是一种常用的一种降维手段，它主要是基于方差来提取最有价值的信息。通过正交变换将一组可能存在相关性的变量转换为一组线性不相关的变量，转换后的这组变量叫主成分。设法将原来变量重新组合成一组新的互相无关的几个综合变量，同时根据实际需要从中可以取出几个较少的综合变量尽可能多地反映原来变量。

输入

- 输入一个数据集，用户选择要做PCA的列（必须是int或double或vector类型），输入PCA结果的列数，并且可以选择PCA之前是否中心化、标准化，可以选择是否保留原始列。

输出

- 第一个输出是这个数据集的pca结果。
- 第二个输出是可以对其它数据集应用的PCA模型。

算子参数

参数名称	是否必选	参数描述	默认值
结果维度	是	PCA结果的维度 范围：[1, inf)	无
预处理方式	是	主成分分析前的预处理，"中心化"指将每个特征的均值调整成0，"标准化"在中心化的基础上将每个特征的标准差化成1	无

字段参数

参数名称	是否必选	参数描述	默认值
输入列	是	需要PCA的列集合，需要是数值或数值数组类型	无
保留原始数据	是	是否保留原始列	开启

计算逻辑

- 通过正交变换将一组可能存在相关性的变量转换为一组线性不相关的变量，转换后的这组变量叫主成分。设法将原来变量重新组合成一组新的互相无关的几个综合变量，同时根据实际需要从中可以取出几个较少的综合变量尽可能多地反映原来变量的信息的统计方法叫做主成分分析或称主分量分析，也是数学上用来降维的一种方法。
- 选择多列，进行PCA主成分分析，选择的列应是数值或数值数组类型。
- 指定输出结果的维度。
- PCA主成分分析预处理方式有：
    1. 无

2. 中心化：将每个特征的均值调整成0。

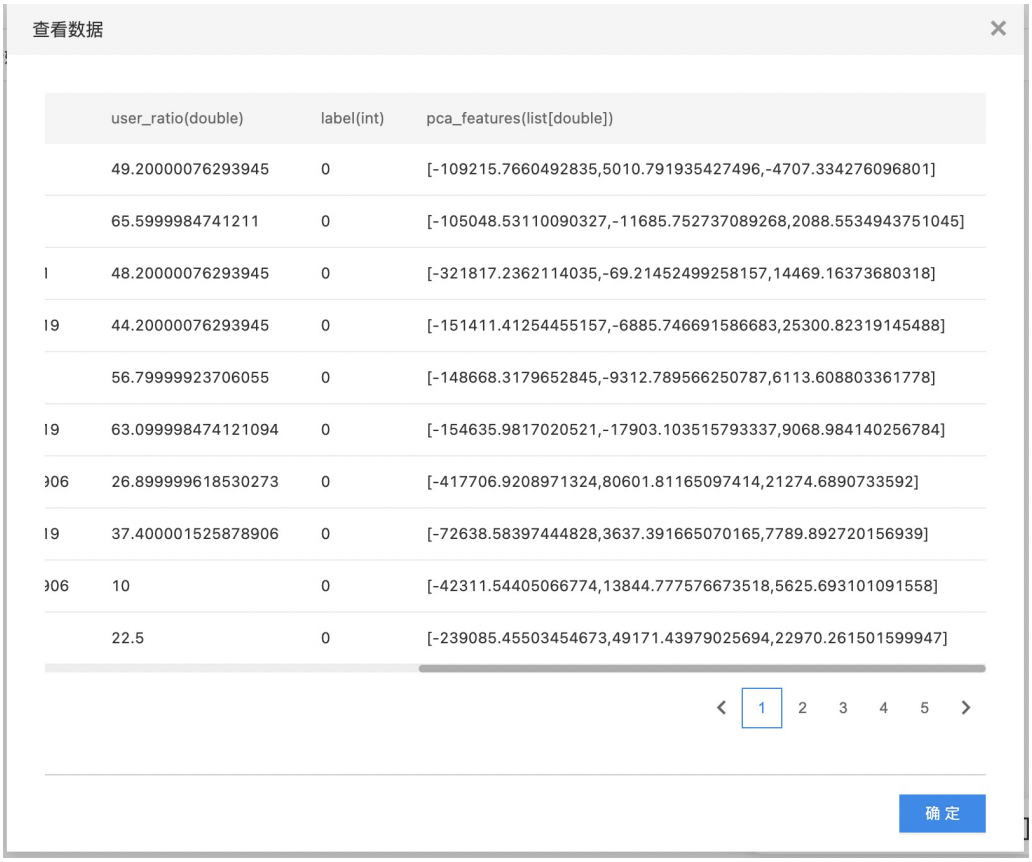
3. 标准化：在中心化的基础上将每个特征的标准差化成1。

使用示例

1. 如下图所示，构建算子结构，选择需要进行PCA主成分分析的列，填写结果维度。



2. 查看PCA主成分分析的结果。



数值替换

对数据表中的一列，根据自定义的规则进行数值的替换，将符合规则的数值替换成新的数值。

输入

- 输入一个数据集和一条格式的替换指令。

输出

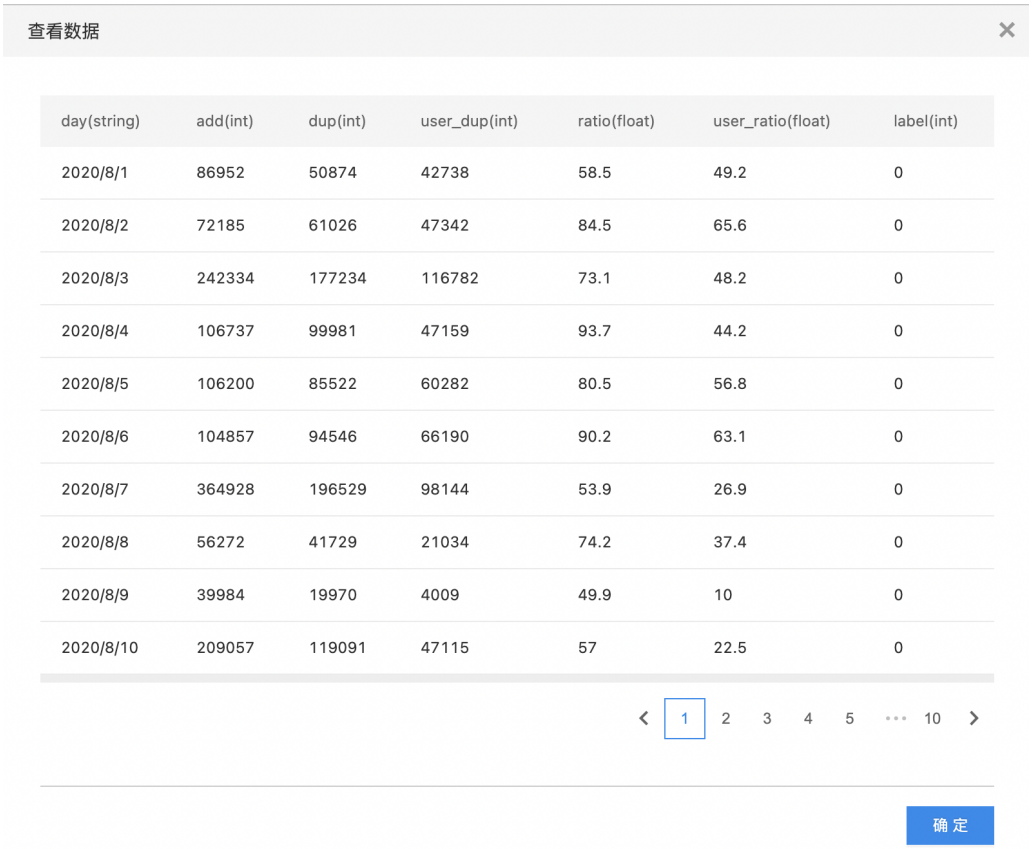
- 第一个输出是数值替换后的数据集。
- 第二个输出是数值替换模型。

算子参数

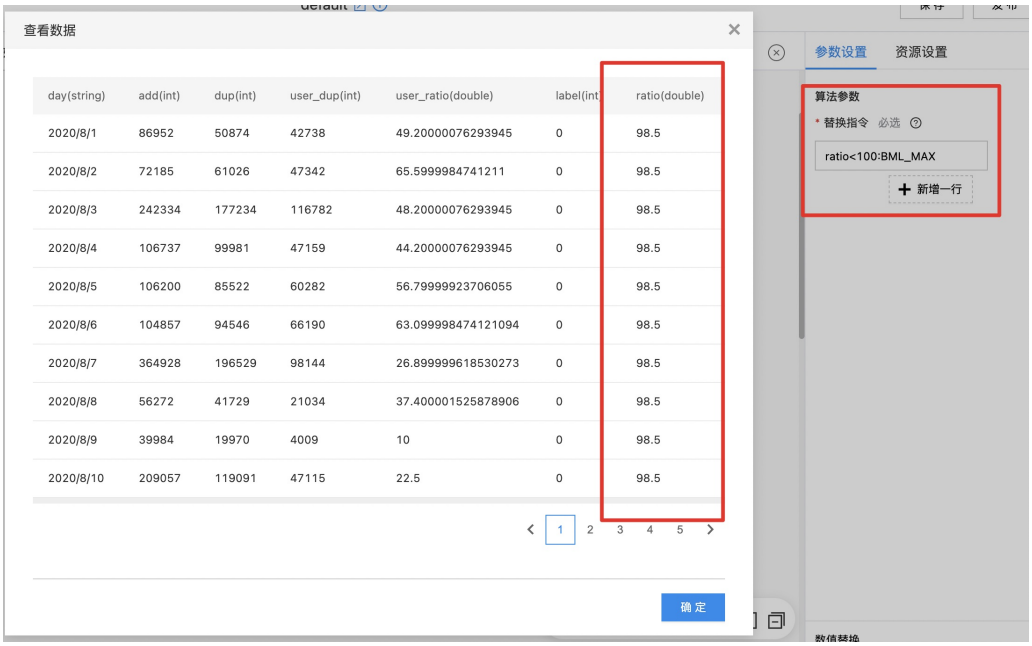
参数名称	是否必选	参数描述	默认值
替换指令	是	替换指令的每一条格式：列名[><=]值:替换值。如amount<10:9代表了把amount列中所有小于10的值转换成9。转换列支持数值和字符串类型；关系符号支持<, >, =, <=, >= 五种；列名、值、替换值只允许字母、数字和.-_三个特殊符号；替换值除了正常的值，还包括BML_MIN, BML_MAX, BML_MEAN三个特殊值，代表该列替换前的最小值、最大值和平均值。	无

使用示例

1. 拖入数据集组件选择数据集



2. 拖入数值替换组件，连接数据集组件后，输入替换指令【ratio<100:BML_MAX】，即令ratio小于100的数值替换为该列最大的数值



归一化

对数据表中的一列或者多列，进行归一化处理，所选的列应是数值型或数值数组类型。

输入

- 输入一个数据集，选定需要归一化的列，选定归一化方法，选择是否保留原列。

输出

- 第一个输出是归一化后的数据集。
- 第二个输出是归一化的模型。

算子参数

参数名称	是否必选	参数描述	默认值
归一化方法	是	归一化方法,当前支持: 1.maxmin:0-1标准化 2.zscore:标准差标准化 3.logistic:使用logistic处理所选数据，范围(0,1) 4.lognormal:使用对数正态分布概率密度函数处理所选数据，范围(0,1) 5.tanH:使用tanH处理所选数据，范围(-1,1) 6.softmax:使用softmax处理所选数据，范围(0,1)	MAXMIN
是否保留原始列	是	是否保存原始列,状态为开启时，处理过的列重命名 "normalized_"前缀	关闭

字段参数

参数名称	是否必选	参数描述	默认值
归一化列	是	需要归一化的列集合，需要是数值或数值数组类型	

计算逻辑

1.MAXMIN：

$$Z = \frac{X - min(x)}{max(x) - min(x)}$$

其中Z为归一化后输出数据，X为输入数据，$min(x)$为对应数据列的最小值，$max(x)$为对应数据列的最大值

2.ZSCORE：

$$Z = \frac{X - mean(x)}{stedv(x)}$$

其中Z为归一化后输出数据，X为输入数据，$mean(x)$为对应数据列的平均值，$stedv(x)$为对应数据列的标准差

3.LOGISTIC:

$$Z = \frac{1}{1 + exp(-x)}$$

其中Z为归一化后输出数据，X为输入数据

4.LOGNORMAL:

$$Z = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(x-\mu)^2}{2\sigma^2}), x > 0$$

即正态分布公式，其中Z为归一化后输出数据，X为输入数据，μ为平均数，σ为标准差

5.TANH:

$$Z = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

其中Z为归一化后输出数据，X为输入数据

6.SOFTMAX:

$$softmax(x)_i = \frac{exp(x_i)}{\sum_j exp(x_j)}$$

示例

假设当前有一列特特征值分别为[1，5，3]，对应的softmax归一化输出值如下:

$$softmax(1) = \frac{e^1}{e^1 + e^5 + e^3} = 0.016$$

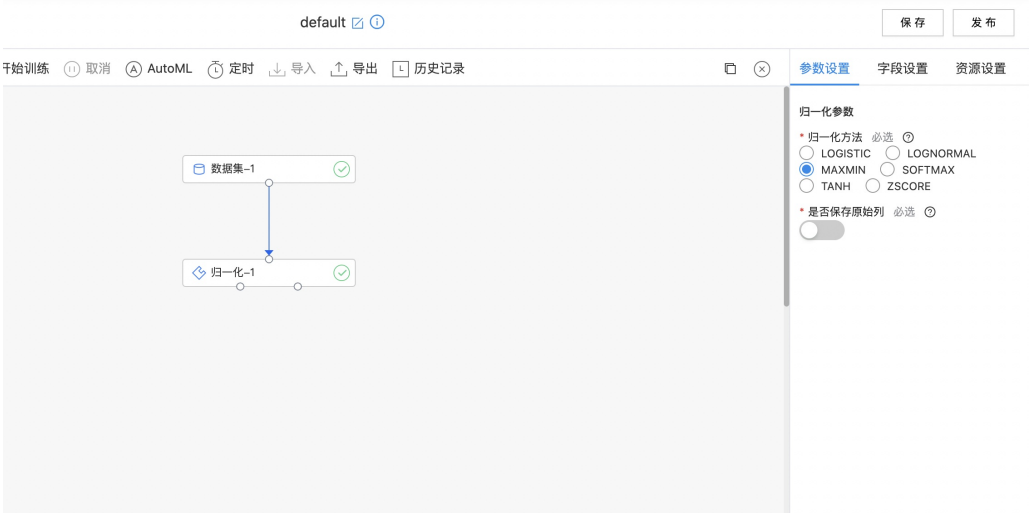
$$softmax(5) = \frac{e^5}{e^1 + e^5 + e^3} = 0.867$$

$$softmax(3) = \frac{e^3}{e^1 + e^5 + e^3} = 0.117$$

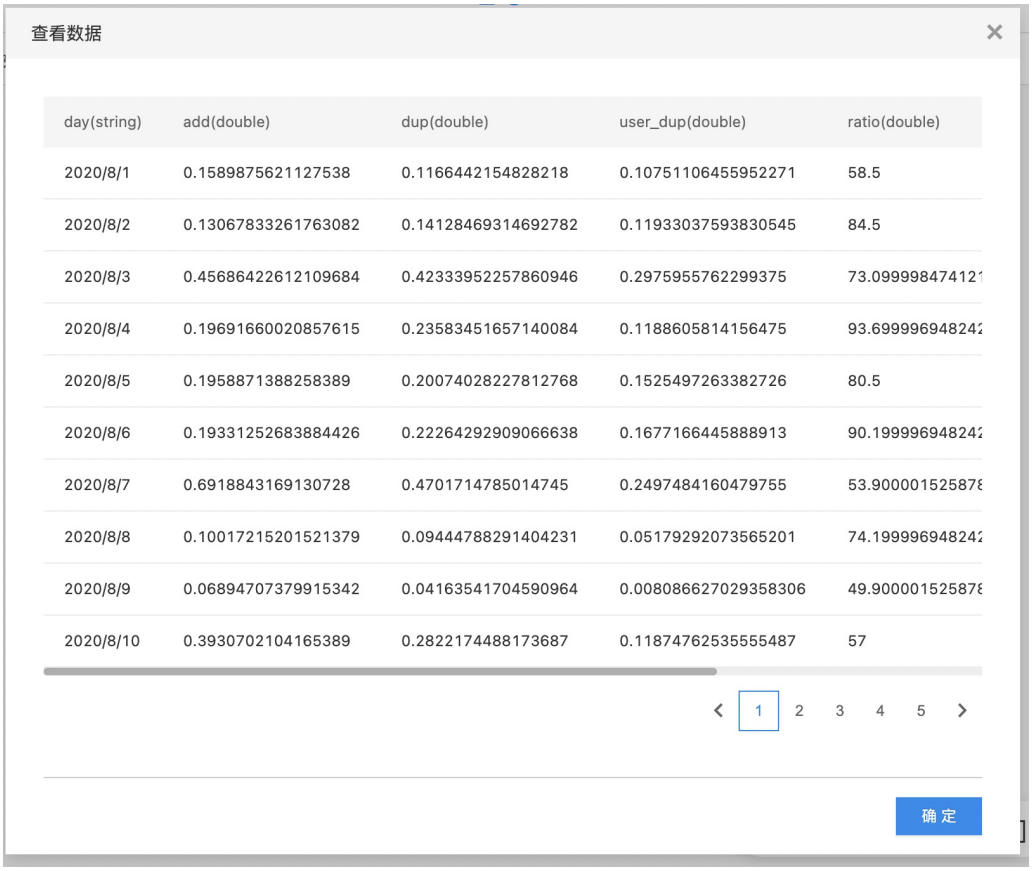
$$softmax(1) + softmax(5) + softmax(3) = 1$$

使用示例

1. 如下图所示，构建算子结构，选择需要归一化的列，选择归一化方法。



2. 查看归一化的结果。



直方映射

直方图编码，主要针对类别型特征与类别型标签的一种特征编码方式。

输入

- 输入一个数据集，选择需要直方编码的列集合与标签列。

输出

- 第一个输出是结果数据集。
- 第二个输出是直方映射变换（Transformation），后续可以使用这个变换操作其它数据集。

字段参数

参数名称	是否必选	参数描述	默认值
输入列	是	需要直方编码的列集合，要求是整型、长整型、字符串，唯一值不超过500。	无
标签列	是	标签列只能数值类型	无
保留原始列	是	是否保留原始列	

计算逻辑

假设类别特征F={A，A，B，B，B，C，C}，对应的二分类标签为{0，1，0，1，1，0，0}，则计算类别特征F中对应的类别的编码值的方法如下：

以A为例，类别特征F的值为A的样本有两个，这两个样本的标签分别为{0，1}，则A被直方图编码为：

$$\frac{(0+1)}{2}=\frac{1}{2}$$

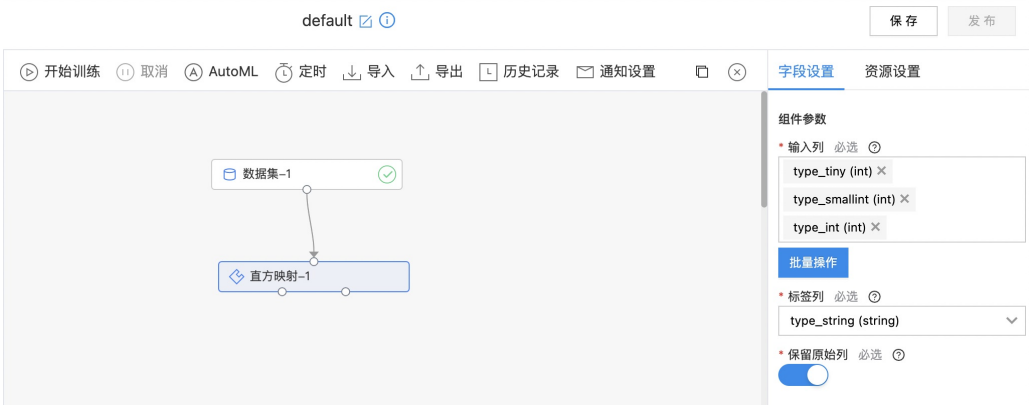
同理，对B进行类别编码，F值为B的一共3个样本，其中一个样本标签为0，两个样本标签为1，所以B被编码为：

$$\frac{(0+1+1)}{3}=\frac{2}{3}$$

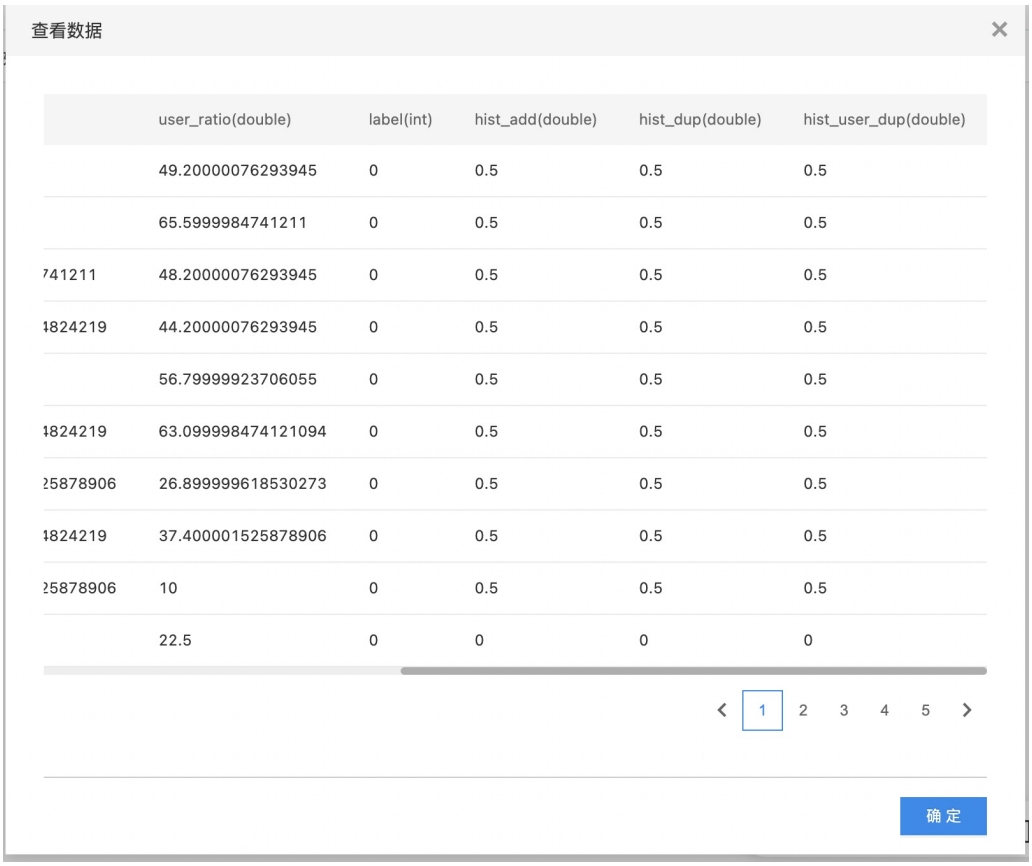
对于C来说，一共两个样本，并且两个样本标签均为0，则编码为0,可以看出编码结果为对应标签值想加取平均数。

使用示例

1. 如下图所示，构建算子结构，选择需要直方映射的列。



2. 查看直方映射的结果。



woe编码

WOE是一种有监督的编码方式,将预测类别的集中度的属性作为编码的数值，编码仅仅针对于二分类问题。

输入

- 输入一个数据集，需要选择要做woe编码的列，可以选择一列或多列。

输出

- 第一个输出是结果数据集，在输入数据集的基础上添加了"woe_选择的列名"列，列的类型是double。
- 第二个输出是woe编码变换（Transformation），后续可以使用这个变换操作其它数据集；若测试集的对应特征列存在值未在训练集中出现，则对应的值woe编码为null。

字段参数

参数名称	是否必选	参数描述	默认值
输入列	是	需要woe编码的列集合，要求是整型、长整型、字符串。	无
标签列	是	标签列只能是0-1变量；1表示响应；要求是数值类型、字符串。	无

计算逻辑

woe计算公式：

$$woe = \sum log \frac{G_i / G_{total}}{B_i / B_{total}}$$

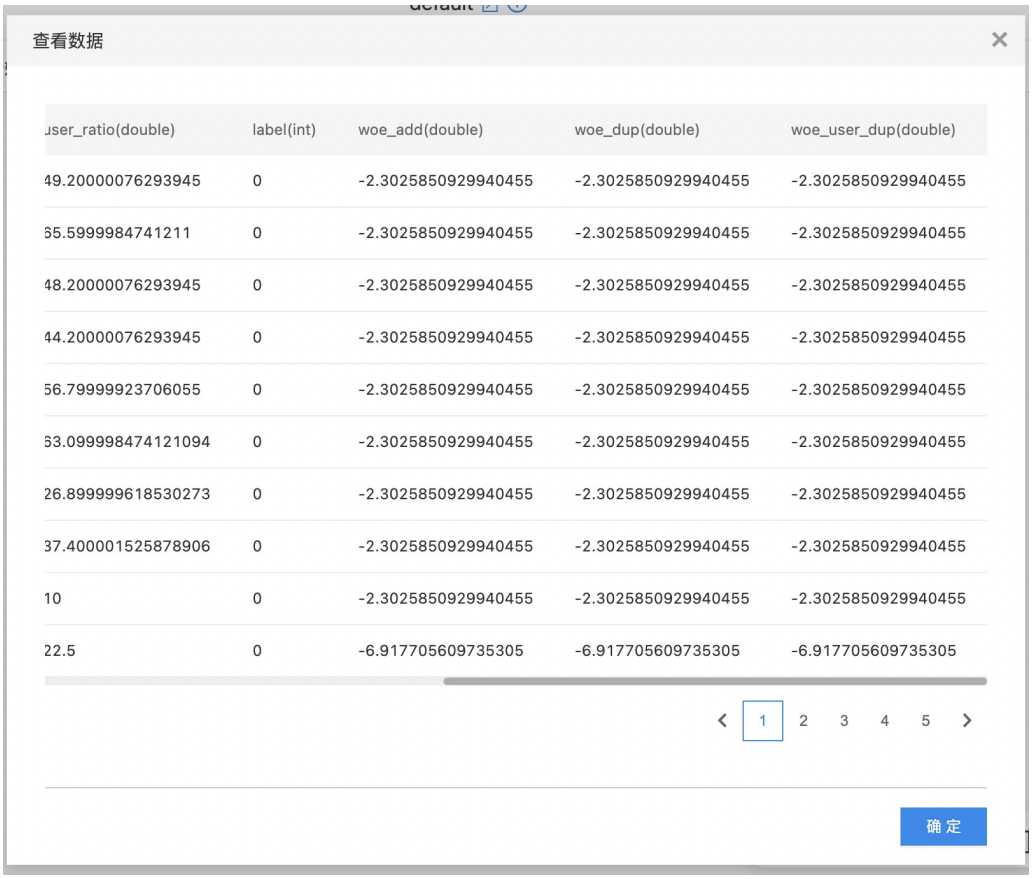
其中G和B为样本的类别。

使用示例

1. 如下图所示，构建算子结构，选择需要进行woe编码的列。



2. 查看woe编码结果。



嵌入式特征选择

嵌入式特征选择是在学习器训练过程中自动进行特征选择，特征选择算法本身作为组成部分嵌入到学习算法里。

输入

- 输入一个数据集，选择要做嵌入式特征选择的列，选择特征选择的方法与条件。

输出

- 第一个输出是结果数据集，在输入数据集的基础上去除不重要的特征列。
- 第二个输出是嵌入式特征选择变换（Transformation），后续可以使用这个变换操作其它数据集。

算子参数

参数名称	是否必选	参数描述	默认值
监督模式	是	根据监督任务类型，给出估计器类型（分类/回归）	分类
分类估计器	是	选择分类算法： GBDT分类 逻辑回归(内置交叉验证)+L1正则 随机森林分类 SVM分类	逻辑回归(内置交叉验证)+L1正则
回归估计器	是	选择回归算法： GBDT回归 LASSO(内置交叉验证) 随机森林回归 SVM回归	LASSO(内置交叉验证)
交叉验证折数	是	交叉验证折数，必须不小于2。对于分类任务，交叉验证折数不能大于每个类别的样本个数；若不能确定交叉验证折数是否大于每个类别的样本个数，建议将交叉验证折数设置小一点， 范围：[2, 20]。	5
特征阈值选择模式	是	用于特征选择的阈值输入模式。特征重要性不小于阈值，保留对应特征，否则丢弃。有三种模式：自动选择、自定义字符串、自定义浮点数。对于自动选择模式，如果估计器的参数惩罚被显示或隐式地设置为L1(例如Lasso)，则使用的阈值为1e-5；否则，默认使用“mean”	自动选择
自定义浮点数	是	阈值浮点数格式输入	0.25
自定义字符串	是	阈值字符串格式输入。既可以字符串”mean”、”median”，分别边表示所有特征重要性的平均数、中位数，也可以是浮点数和mean或median相乘对应的字符串，例如”1.25*mean”,表示mean或median乘以一个缩放因子	mean

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	嵌入式特征选择使用的特征列，要求必须是数值类型。	无
标签列	是	嵌入式特征选择目标列，分类任务：要求是整数或字符串类型，回归任务：需要是数值类型。	无

计算逻辑

嵌入式特征选择分为基于惩罚项与基于树模型两大类，线性模型的损失函数增加惩罚项进行稀疏处理，即重要性小的特征系数迭代为0，进而达到特征选择的目的，进而达到特征选择的目的；树模型根据子节点纯度判断特征选择的优劣，达到特征选择的目的，用户可以根据自身需求自行选择具体的线性模型或树模型。

使用示例

如下图所示，构建算子结构，选择需要进行特征选择的列，配置选择方法。



FastICA

独立成分分析（Independent Component Analysis，ICA）是一种从多维数据中寻找其具有内在独立性和非高斯性因子的数据处理方法。ICA可以在未知源信号及混合矩阵信息的情况下分离出相互统计独立的源信号，FastICA是基于负熵的快速独立分量分析算法。

独立分量分析将多变量信号分解为独立性最强的加性子组件。 它通过Fast ICA算法在 scikit-learn 中实现。 ICA 通常不用于降低维度，而是用于分离叠加信号。 由于 ICA 模型不包括噪声项，因此要使模型正确，必须使用白化(whitening)。这可以在内部使用 whiten 参数或手动使用 PCA 的一种变体，ICA 通常用于分离混合信号（称为**盲源分离**的问题）。

输入

- 输入一个数据集，需要选择数据集的特征列。

输出

- 第一个输出是结果数据集。
- 第二个输出是FastICA模型，需要连接Python特征工程预测组件，进行预测，如果选择不保留原始列，输出数据集中删去原始列。

算子参数

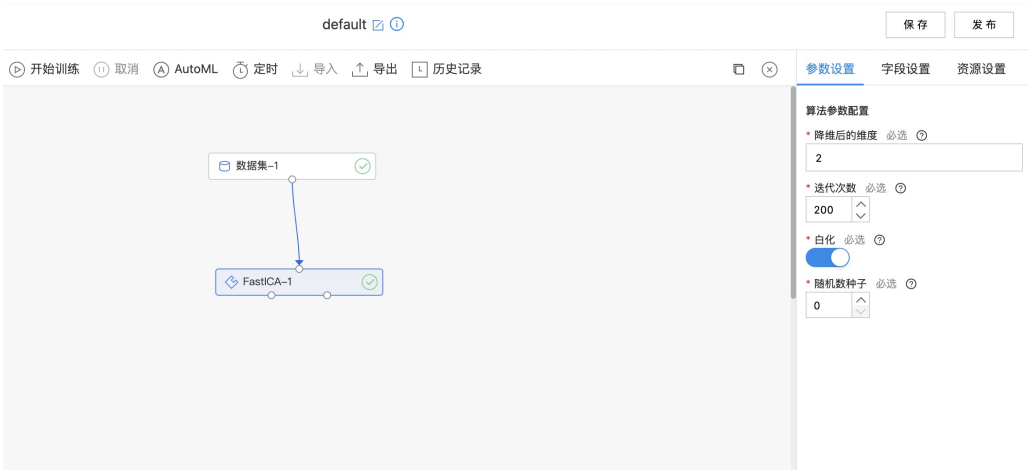
参数名称	是否必选	参数描述	默认值
降维后的维度	是	降维后的维度，该值应大于0且小于等于选择的特征个数；当出现infs or NaNs错误时请将请尝试调小该值，或者调整随机种子和开启白化。	2
迭代次数	是	最大迭代次数 范围：[1, inf)。	200
白化	是	白化是一种重要的预处理过程，其目的是为了降低输入数据的冗余性，使得经过白化处理的输入数据特征之间相关性较低, 所有特征具有相同的方差。	开启
随机数种子	是	随机数种子是为了保证每次随机的结果都是一样的 范围：[0, inf)。	0

字段参数

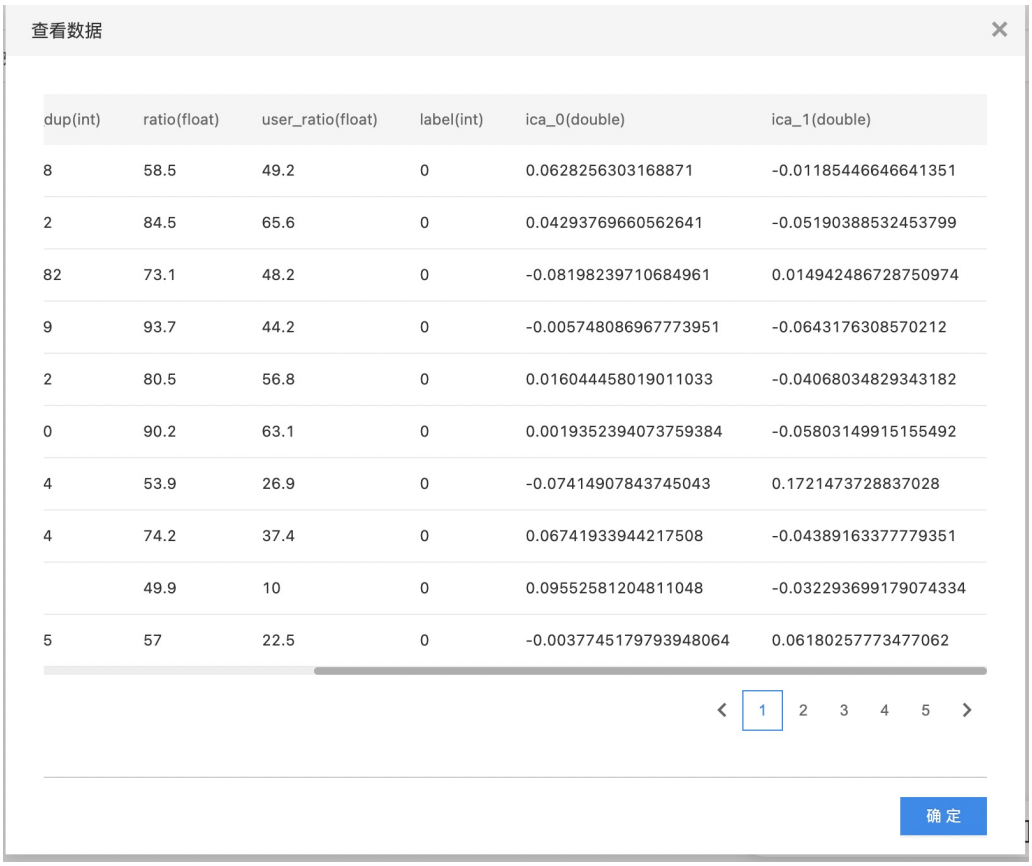
参数名称	是否必选	参数描述	默认值
特征列	是	ICA使用的特征列，最少选择两列，要求必须是数值类型	无
保留原始列	是	是否保留原始列	开启

使用示例

1. 如下图所示，构建算子结构，选择特征列，配置算法参数。



2. 查看FastICA结果。



LDA

线性判别分析（LDA）是一种经典的线性学习方法，在二分类问题上因为最早由 Fisher 提出，亦称“Fisher 判别分析”。 LDA 的思想非常朴素：给定训练样本集，设法将样本投影到一条直线上，使得同类样本的投影点尽可能接近、异类样本的投影点尽可能远离；在对新样本进行分类时，将其投影到同样的这条直线上，再根据投影点的位置来确定新样本的类别。

输入

- 输入一个数据集，数据集的特征列double/int，标记列是Double或Int类型。

输出

- 第一个输出是结果数据集。
- 第二个输出是Python预测模型，需要连接Python特征工程预测组件进行预测。

算子参数

参数名称	是否必选	参数描述	默认值
最优化算法	是	求解最优化问题的算法： svd eigen	svd
降维后的维度	是	降维后的维度，该值应小于标签列类别数-1，降维后的维度为该值和标签列类别数-1之间的最小值	2

字段参数

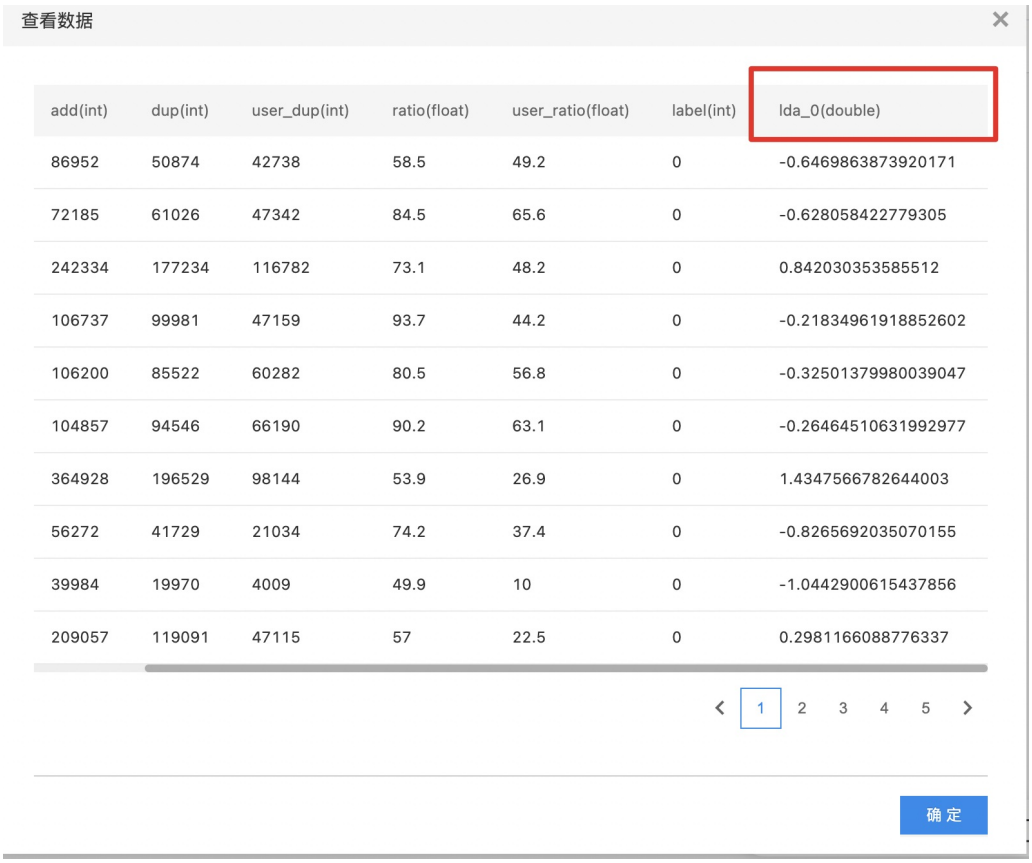
参数名称	是否必选	参数描述	默认值
特征列	是	LDA使用的特征列，最少选择两列，要求必须是数值类型	无
标签列	是	LDA目标列，要求是整数或字符串类型	无

使用示例

1.如下图所示，构建算子结构，选择需要进行LDA处理的列。



2.查看LDA的结果。



OrdinalEncoder(单机)

OrdinalEncoder的功能是将类别特征(categorical features)变为有序数值特征(ordinal integers)。

输入

- 输入一个数据集，指定输入列。

输出

- 第一个输出是结果数据集，模型在原数据集的基础上添加了k列（设k是输入列的个数），类型为整数，名称是对应输入列加上前缀"ordinalencoder_"，如果选择不保留原始列，输出数据集中或删去原始列。
- 第二个输出是OrdinalEncoder(单机)模型。

字段参数

参数名称	是否必选	参数描述	默认值
输入列	是	OrdinalEncoder使用的特征列，要求数值、字符类型	无
保留原始列	是	是否保留原始列	开启

计算逻辑

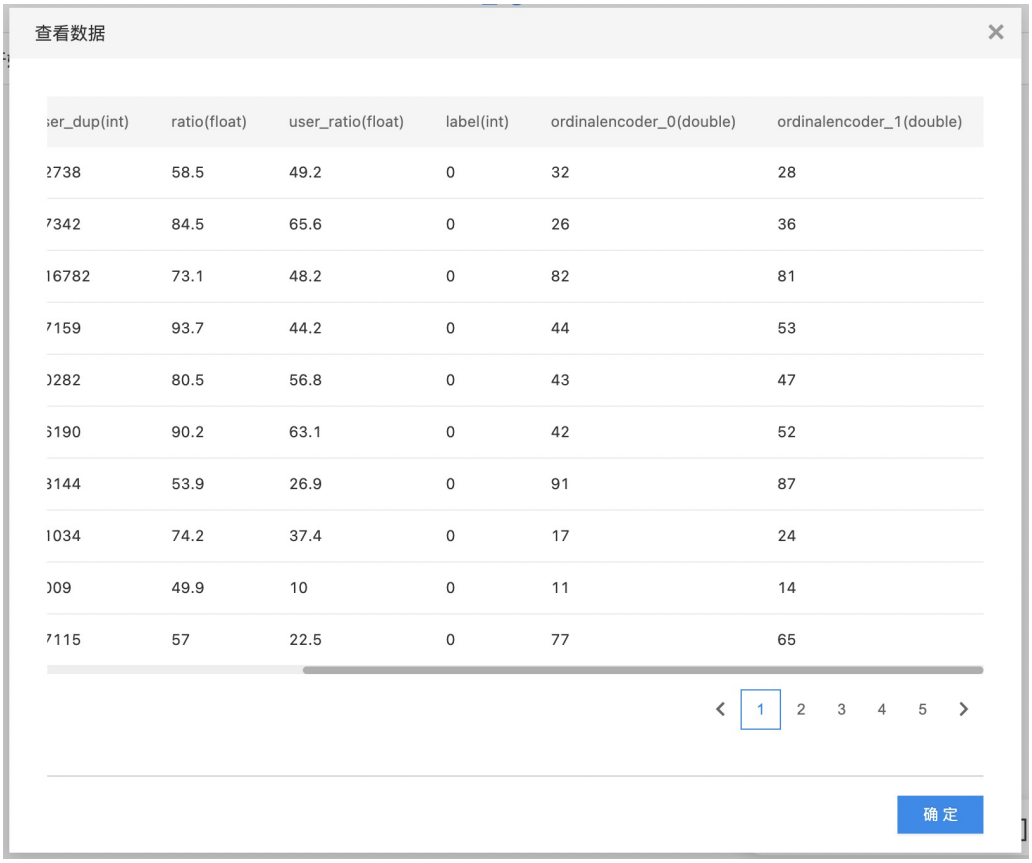
假设输入X = [['Male', 1], ['Female', 3], ['Female', 2]]，这里输入数据中含有两列，说明有两类特征，第一列取值范围为：['Female'，'Male']，按照顺序编码为[0,1]，第二列取值范围为：[1,2,3],编码结果为[0,1,2]。按照这个思路，['Female', 3], ['Male', 1]的顺序编码分别为[0,2]和[1,0]

使用示例

1. 如下图所示，构建算子结构，选择特征列。



2. 查看OrdinalEncoder(单机)的结果。



尺度变换

数值型特征的数据转换，目前包括 对数，指数，box-cox。

输入

- 输入一个数据集，用户需要选择要做尺度变换的列，选择的列的类型需要是数值类型。

输出

- 第一个输出是结果数据集，在输入数据集的基础上添加名为"变换模式前缀_选择列"的特征列，变换模式前缀：对数（log_base）/指数（exp_base）/box-cox（box_cox）。
- 第二个输出是尺度变换变换（Transformation），后续可以使用这个变换操作其它数据集。

算子参数

参数名称	是否必选	参数描述	默认值
尺度变换模式	是	支持变换：对数/指数/box-cox	对数
指数基底	否	支持以2/e为底	e
对数基底	否	支持以2/e/10为底	e
参数lmbda选择模式	否	支持两种选择模式：自动选择（默认）、自定义标量。自动选择：找到最大化对数似然函数的lambda，自定义标量：自定义参数lmbda值	自动选择
自定义标量	否	整数或浮点数格式输入	0

字段参数

参数名称	是否必选	参数描述	默认值
尺度变换列	是	尺度变换的列集合，要求是数值类型	无

计算逻辑

box-cox计算公式：

$$y(\lambda) = \begin{cases} \frac{y_i^{\lambda}-1}{\lambda}, \text{如果} \lambda \neq 0 \\ ln(y_i), \text{如果} \lambda = 0 \end{cases}$$

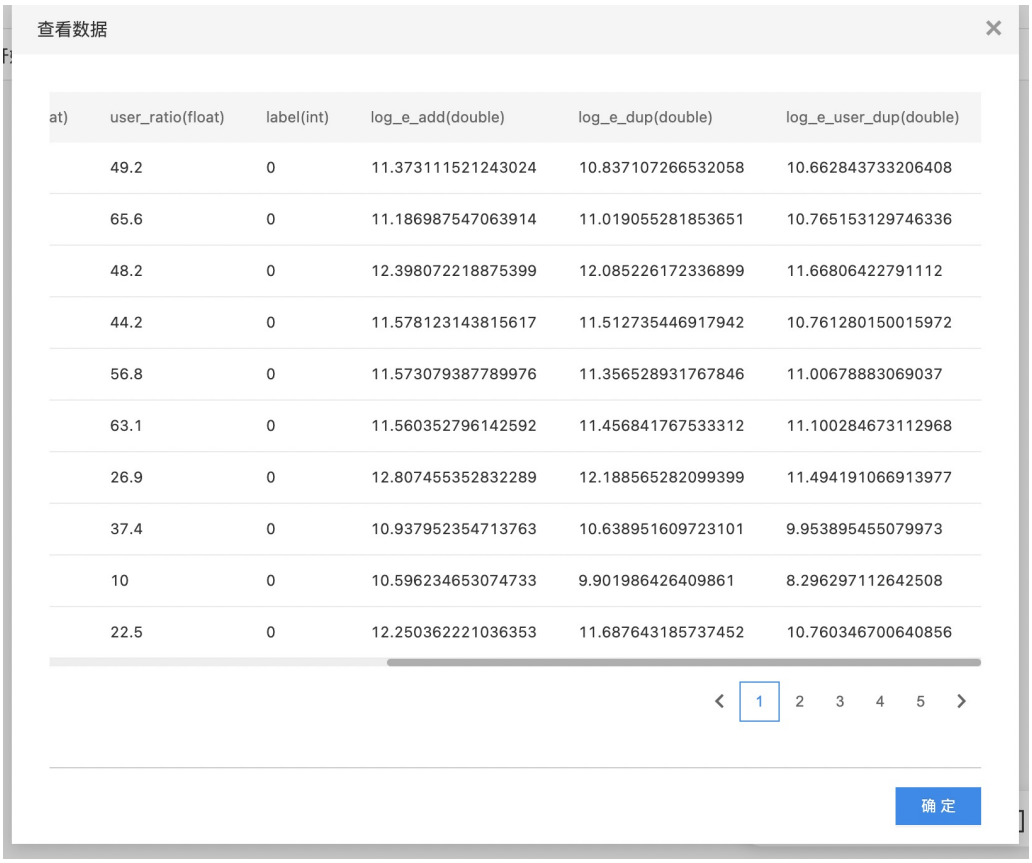
其中λ即为算子参数中的lmbda值。

使用示例

1. 如下图所示，构建算子结构，选择需要尺度变换的列，选择变换方式。



2. 查看尺度变换的结果。



包装式特征选择

包装式特征选择基于贪婪搜索算法，因为它们会评估特征的所有可能组合，并为特定的机器学习算法选择能产生最佳结果的组合，例如用户需要训练一个随机森林的模型，算子参数选择随机森林，那么算子输出的特征集即为最优的特征选择结果。这种方法的缺点是测试所有可能的特征组合可能在计算上非常昂贵，尤其是在特征集非常大的情况下。

输入

- 输入一个数据集，选择要做包装式特征选择的列，选择特征选择的方法与条件。

输出

- 第一个输出是结果数据集，在输入数据集的基础上消除不重要的特征列。
- 第二个输出是包装式特征选择变换（Transformation），后续可以使用这个变换操作其它数据集。

算子参数

参数名称	是否必选	参数描述	默认值
监督模式	是	根据监督任务类型，给出估计器类型（分类/回归）	分类
分类估计器	是	选择分裂算法： GBDT分类 逻辑回归 随机森林分类	逻辑回归
回归估计器	是	选择回归算法： GBDT回归 线性回归 随机森林回归	线性回归
特征排除步长	是	若值不小于1，表示每次迭代排除（值取整）特征个数；若值(0.0，1.0)之间，表示每次迭代按比例排除特征。 范围：[0.1, inf)	1
交叉验证折数	是	交叉验证折数，必须不小于2。对于分类任务，交叉验证折数不能大于每个类别的样本个数；若不能确定交叉验证折数是否大于每个类别的样本个数，建议将交叉验证折数设置小一点。 范围：[2, 20]	5

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	包装式特征选择使用的特征列，要求必须是数值类型	无
标签列	是	包装式特征选择目标列。分类任务：要求是整数或字符串类型，回归任务：需要是数值类型	无

使用示例

1. 如下图所示，构建算子结构，选择需要进行特征选择的列，配置选择方式。



自定义Python特征工程(单机)

自定义编写Python代码进行特征工程处理。

输入

- 输入一个数据集，需要编写Python代码处理特征。

输出

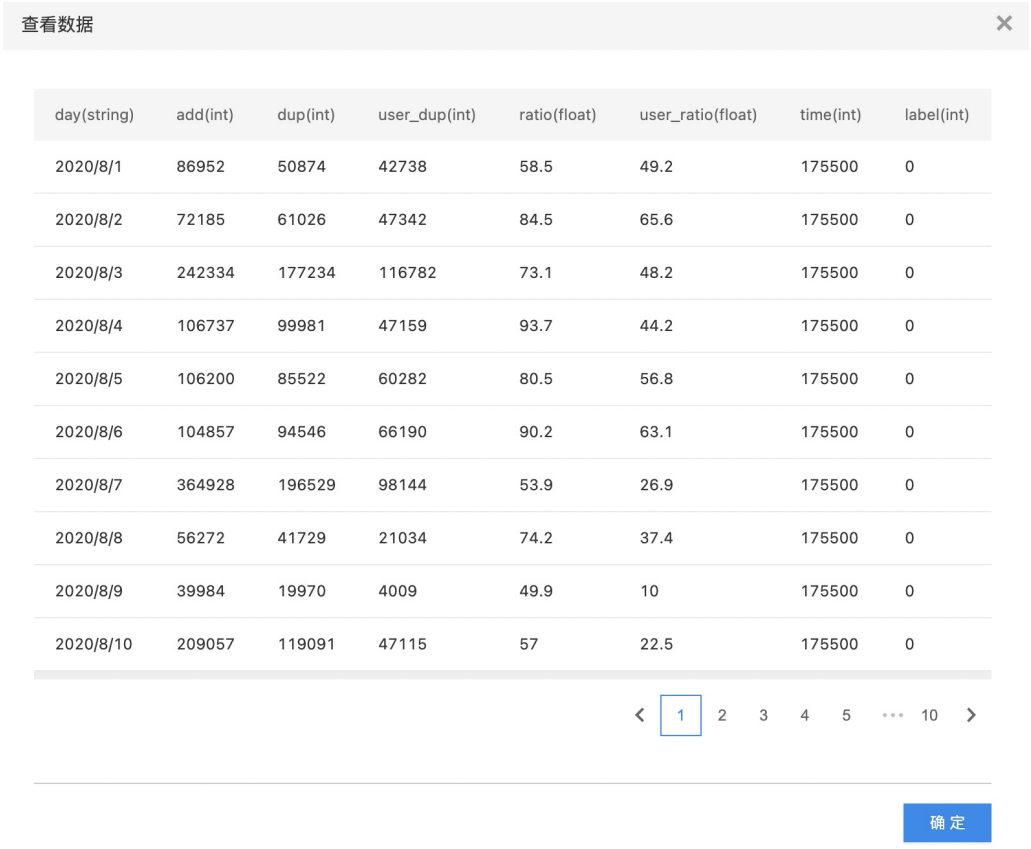
- 第一个输出是结果数据集。
- 第二个输出是自定义 python 特征工程变换(单机)模型，后续可以使用这个变换操作其它数据集。

算子参数

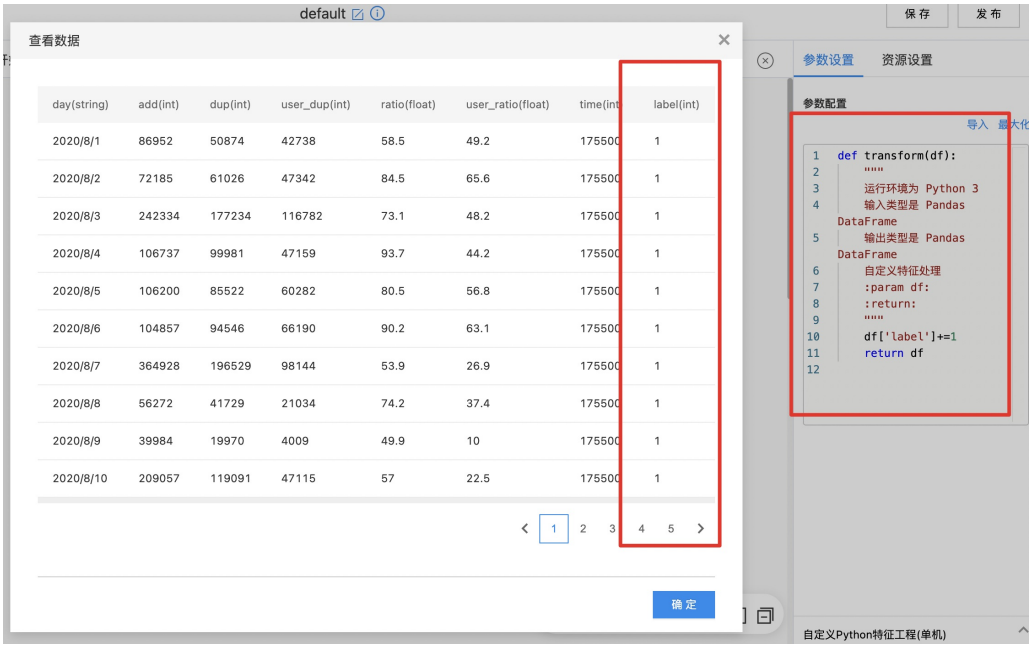
参数名称	是否必选	参数描述	默认值
Python代码编辑窗口	是	在编辑窗口中编辑数据处理脚本	无

使用示例

1. 拖入数据集组件，选择数据集



2. 拖入自定义Python特征工程(单机)组件，连接数据集组件后，编写Python代码，运行查看结果



自定义Python特征工程(分布式)

自定义编写Python代码进行特征工程处理，支持多机运行（分布式）。

输入

- 输入一个数据集，需要编写Python代码处理特征。

输出

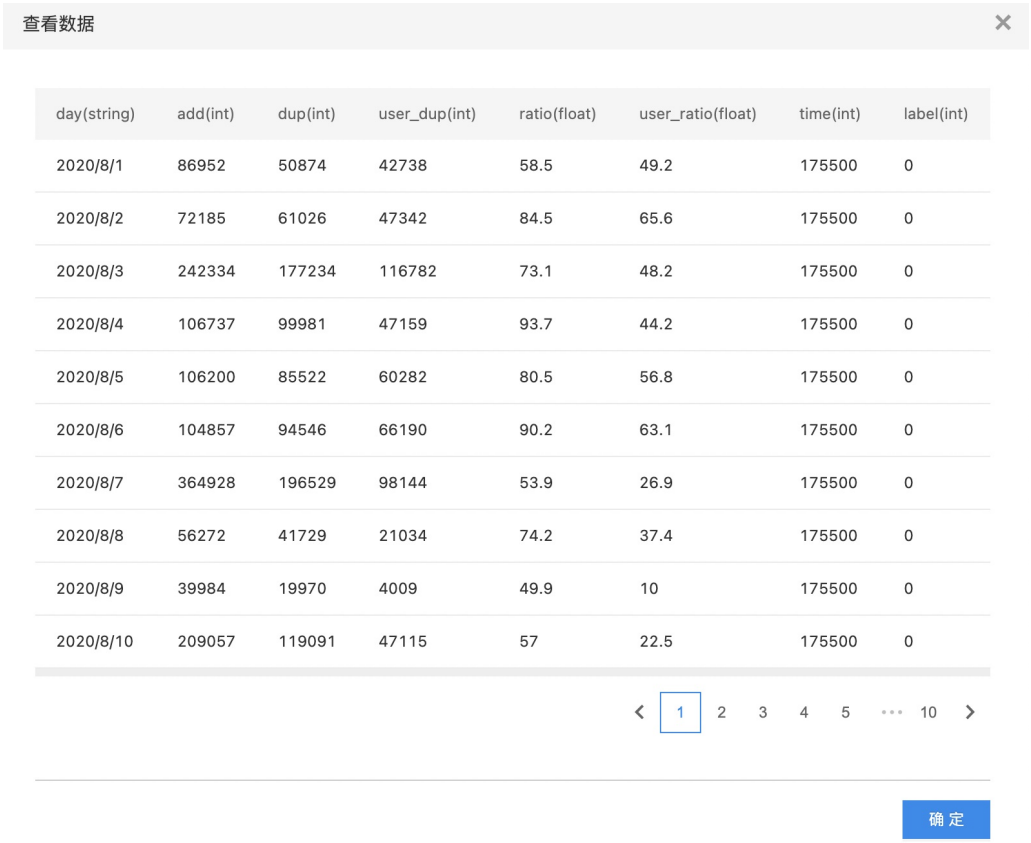
- 第一个输出是结果数据集。
- 第二个输出是自定义 python 特征工程变换(单机)模型，后续可以使用这个变换操作其它数据集。

算子参数

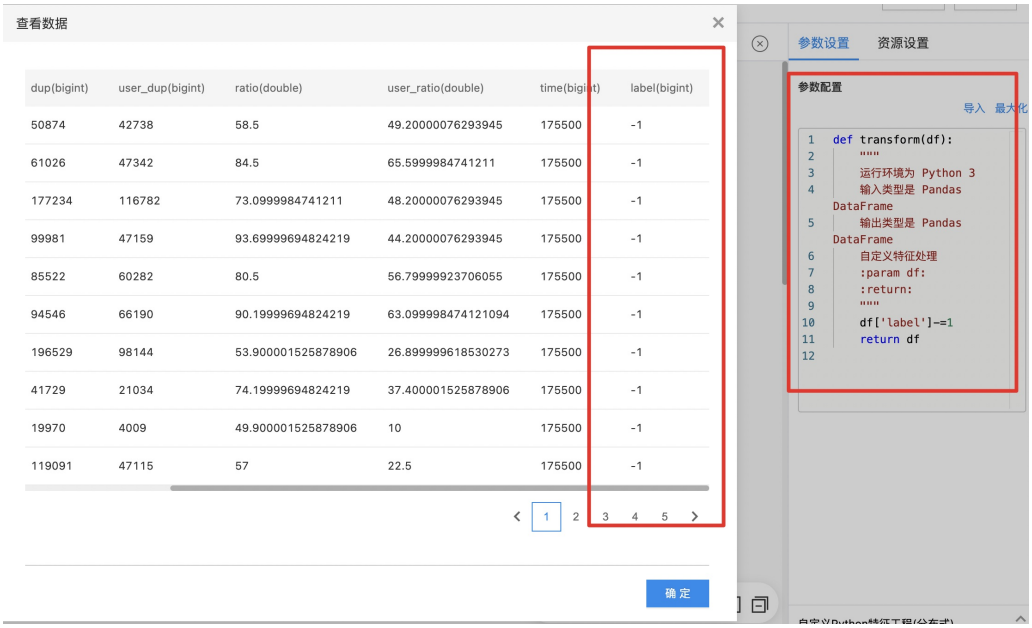
参数名称	是否必选	参数描述	默认值
Python代码编辑窗口	是	在编辑窗口中编辑数据处理脚本	无

使用示例

1. 拖入数据集组件，选择数据集



2. 拖入自定义Python特征工程(分布式)组件，连接数据集组件后，编写Python代码，运行查看结果



分类算法

分类算法

DNN二分类

深度神经网络（DNN）是多层功能神经元模型，每层神经元与下一层神经元全互连，神经元之间不存在同层连接，也不存在跨层连接，其中输入层神经元接受外界输入，隐层与输出层神经元对信号进行加工，最终结果由输出层神经元输出，换言之，输入层神经元仅是接受输入，不进行函数处理，隐层与输出层包含功能神经元。神经网络的学习过程，就是根据训练数据来调整神经元之间的连接权以及每个功能神经元的阈值，换言

之，神经网络学到的东西，蕴涵在连接权与阈值中。 神经元模型：神经元接受到来自 n 个其他神经元传递过来的输入信号，这些输入信号通过带权重的连接进行传递，神经元接受到的总输入值将与神经元的阈值进行比较，然后通过激活函数处理以产生神经元的输出。 学习算法：误差逆传播算法， DNN 二分类：输出层神经元完成二分类任务。

输入

- 输入一个数据集，数据集的特征列必须是Double或Int，标记列必须是String或Int类型（如果列中的unique值超过了两种，运行算子时会报错）。

输出

- 输出DNN二分类模型。

算子参数

参数名称	是否必选	参数描述	默认值
隐藏层节点数	是	隐藏层节点数，数组中必须为正整数 范围：[1, inf)。	10
数据块大小	否	用于设置在矩阵中堆叠输入数据的块大小以加速计算 范围：[1, 1000]。	128
优化方法	是	优化方法支持 gd(minibatch gradient descent) 或 l-bfgs。	l-bfgs
迭代步长	是	每次优化的迭代步长，仅适用于gd 范围：[1.0E-12, inf)。	0.03
最大迭代次数	是	最大迭代次数，默认迭代100次 范围：[0, inf)。	100
收敛容差	是	迭代时的收敛容差 范围：[0.0, inf)。	1e-6
随机种子	是	随机种子	1

字段参数

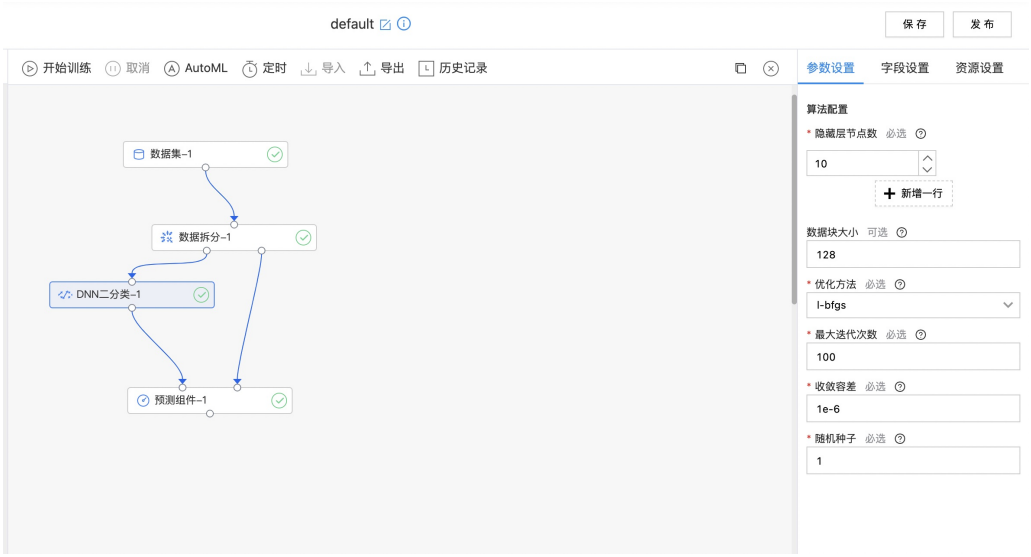
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列。要求必须是数值或数值数组类型。	无
标签列	是	真实标签列。要求不是数组类型，唯一值是2。	无
正样本标签值	否	二分类的正样本的标签值，应在标签列中存在。	无

AutoML参数

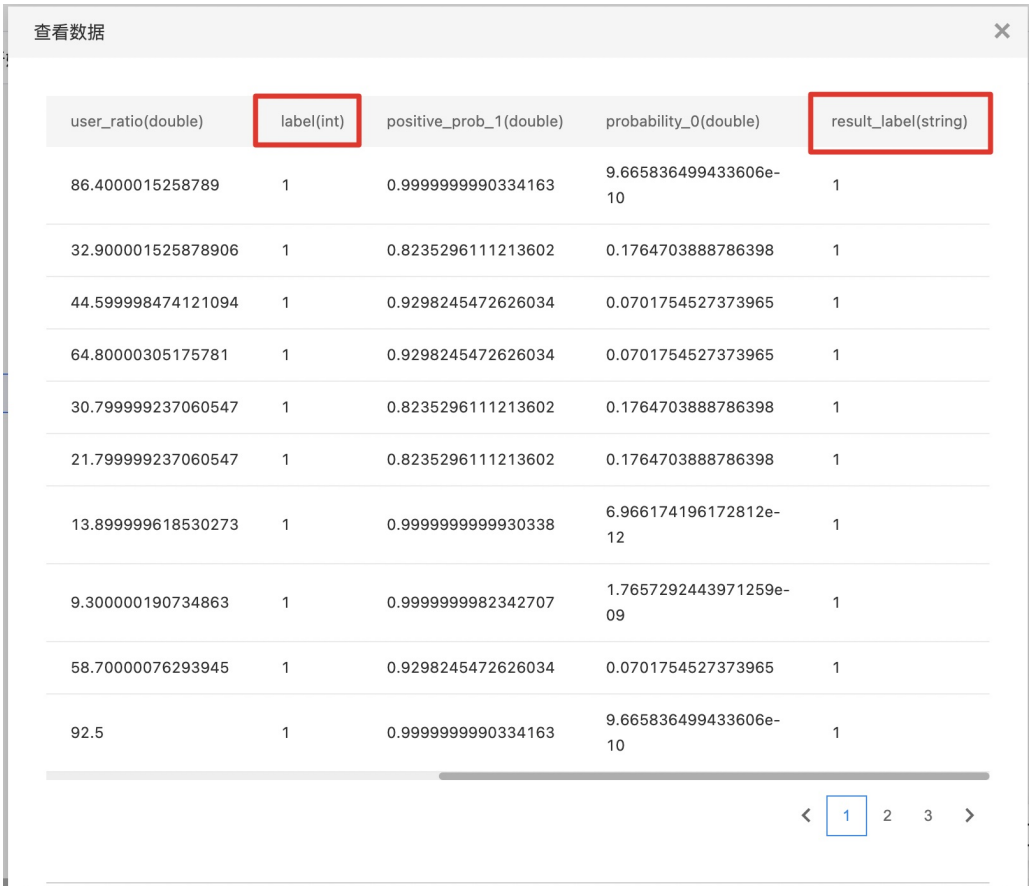
参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]。	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]。	5
参数总组数	是	选择参数总组数，范围:[2,10]。	5
最大迭代次数	是	最大迭代次数，默认迭代100次 范围：[0, inf)。	100
评估标准	是	选择评估标准： 准确率 pr曲线下面积 roc曲线下面积 F1分数 精确率 召回率	roc曲线下面积
保存模型数量	是	保存模型的数量。	1

使用示例

1. 构建算子结构，配置模型参数，完成训练。



2. 查看预测结果。



DNN多分类

深度神经网络（DNN）多分类是基于多层功能神经元模型的多分类模型，其中输出层神经元完成多分类任务。

输入

- 输入一个数据集，数据集的特征列必须是Double或Int，标记列必须是String或Int类型（列中的unique值超过两种）。

输出

- 输出DNN多分类模型，支持查看Top10分类的概率。

算子参数

参数名称	是否必选	参数描述	默认值
隐藏层节点数	是	隐藏层节点数，数组中必须为正整数 范围：[1, inf)。	10
数据块大小	否	用于设置在矩阵中堆叠输入数据的块大小以加速计算 范围：[1, 1000]。	128
优化方法	是	优化方法支持 gd(minibatch gradient descent) 或 l-bfgs。	l-bfgs
迭代步长	是	每次优化的迭代步长，仅适用于gd 范围：[1.0E-12, inf)。	0.03
最大迭代次数	是	最大迭代次数，默认迭代100次 范围：[0, inf)。	100
收敛容差	是	迭代时的收敛容差 范围：[0.0, inf)。	1e-6
随机种子	是	随机种子	1

字段参数

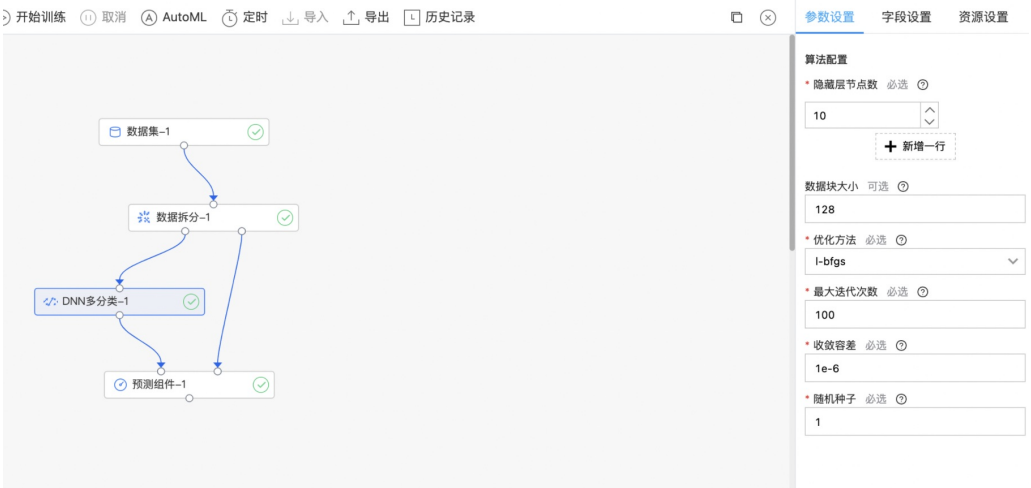
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	真实标签列。当数据量或标签数量过大时，需增加内存。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]。	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]。	5
参数总组数	是	选择参数总组数，范围:[2,10]。	5
最大迭代次数	是	最大迭代次数，默认迭代100次 范围：[0, inf)。	100
评估标准	是	选择评估标准： 准确率 加权F1分数 加权精确率 加权召回率	准确率
保存模型数量	是	保存模型的数量。	1

使用示例

构建算子结构，配置参数，完成训练。



DecisionTree二分类

决策树（DecisionTree）是一类常见的机器学习方法。它从给定训练数据集学得一个模型用以对新示例进行分类。二分类的任务，可看作对“当前样本属于正类吗？”这个问题的“决策”或“判定”过程。决策树是基于树结构来进行决策的。 决策树学习的关键是如何选择最优划分属性。一般而言，随着划分过程不断进行，决策树分支结点的“纯度”越来越高。

输入

- 输入一个数据集，数据集的特征列double/int，标记列是String或Int类型，标记列唯一值是2。

输出

- 输出DecisionTree二分类模型。

算子参数

参数名称	是否必选	参数描述	默认值
树最大深度	是	决策树最大深度 范围：[0, 30]。	5
连续特征分箱数	是	决策树连续特征分箱数 范围：[2, inf)。	32
分裂指标	是	分裂指标： gini-基尼系数 entropy-熵 计算公式可查看【特征自动选择】组件	gini
最小信息增益	是	决策树分裂时最小信息增益 范围：[0.0, inf)。	0
节点最小样本数	是	决策树节点最小样本数，当拆分节点后子节点中样本数量小于最小样本数时，该节点不进行拆分 范围：[1, inf)。	5
随机种子	否	随机种子	5

字段参数

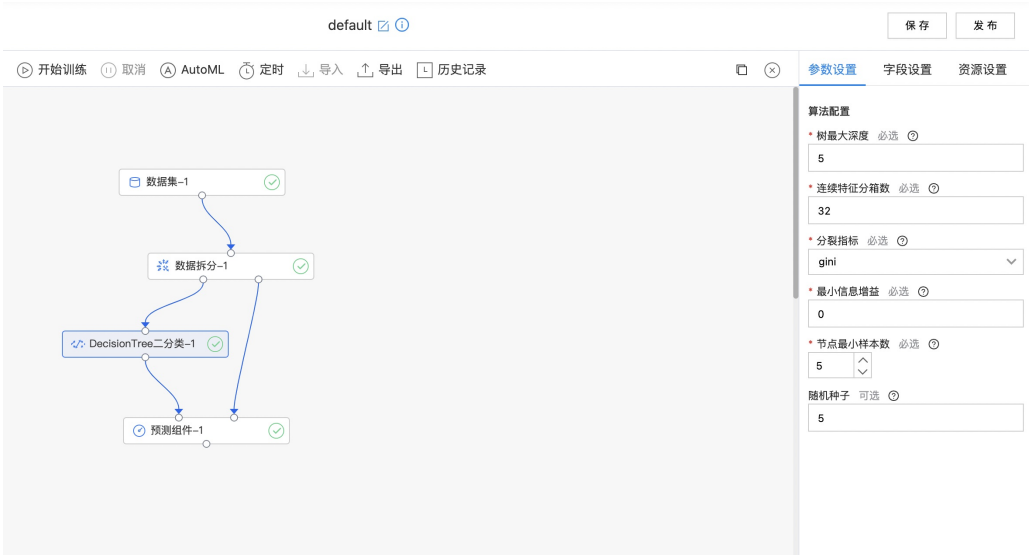
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	真实标签列。要求不是数组类型，唯一值是2。	无
正样本标签值	否	二分类的正样本的标签值，应在标签列中存在。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]。	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]。	5
参数总组数	是	选择参数总组数，范围:[2,10]。	5
树最大深度	是	决策树最大深度 范围：[0, 30]。	5
连续特征分箱数	是	决策树连续特征分箱数 范围：[2, inf)。	32
节点最小样本数	是	决策树节点最小样本数，当拆分节点后子节点中样本数量小于最小样本数时，该节点不进行拆分 范围：[1, inf)。	5
最小信息增益	是	决策树分裂时最小信息增益 范围：[0.0, inf)。	0
评估标准	是	选择评估标准： 准确率 pr曲线下面积 roc曲线下面积 F1分数 精确率 召回率	roc曲线下面积
保存模型数量	是	保存模型的数量。	1

使用示例

构建算子结构，配置参数，完成训练。



DecisionTree多分类

决策树（DecisionTree）多分类是基于树结构对于多分类任务进行决策，原理和划分选择与二分类一样。

输入

- 输入一个数据集，数据集的特征列double/int，标记列是String或Int类型，标记列唯一值>=3。

输出

- 输出DecisionTree多分类模型，支持查看Top10分类的概率。

算子参数

参数名称	是否必选	参数描述	默认值
树最大深度	是	决策树最大深度 范围：[0, 30]	5
连续特征分箱数	是	决策树连续特征分箱数 范围：[2, inf)	32
分裂指标	是	分裂指标： gini entropy	gini
最小信息增益	是	决策树分裂时最小信息增益 范围：[0.0, inf)	0
节点最小样本数	是	决策树节点最小样本数，当拆分节点后子节点中样本数量小于最小样本数时，该节点不进行拆分 范围：[1, inf)	5
随机种子	否	随机种子	5

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求不是数组类型。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]	5
参数总组数	是	选择参数总组数，范围:[2,10]	5
树最大深度	是	决策树最大深度 范围：[0, 30]	5
连续特征分箱数	是	决策树连续特征分箱数 范围：[2, inf)	32
节点最小样本数	是	决策树节点最小样本数，当拆分节点后子节点中样本数量小于最小样本数时，该节点不进行拆分 范围：[1, inf)	5
最小信息增益	是	决策树分裂时最小信息增益 范围：[0.0, inf)	0
评估标准	是	选择评估标准： 准确率 加权F1分数 加权精确率 加权召回率	准确率
保存模型数量	是	保存模型的数量	1

使用示例

构建算子结构，配置参数，完成训练。



逻辑回归二分类

经典逻辑回归是一个二分类算法。它通过对数几率函数（logistic function）将线性回归模型产生的预测值转化为一个接近 0 或 1 的值。组件支持稠密数据格式。逻辑回归实现分类的思想：将每条样本进行打分，然后设置一个阈值，达到这个阈值的，分为一个类别，没有达到这个阈值的，分为另外一个类别。对于阈值，划分为哪个类别都可以，但是，要保证阈值划分的一致性。

输入

- 输入一个数据集，数据集的特征列必须是Double或Int，标记列必须是String或Int类型（如果列中的unique值超过了两种，运行算子时会报错）。

输出

- 输出逻辑回归二分类模型。

算子参数

参数名称	是否必选	参数描述	默认值
alpha	是	正则项的选择，alpha=0代表L2正则，alpha=1代表L1正则，取中间值代表二者的线性组合 范围：[0.0, 1.0]	0.00
lambda	是	正则项的系数，0代表没有正则，越大代表正则强度越大 范围：[0.0, 2.0]	1
随机种子	否	随机种子，用于保证多次训练结果相同	无
最大迭代轮数	否	最大迭代轮数 范围：[1, 1000]	20
目标收敛阈值	否	计算方式(old_val - new_val)/old_val。两次迭代之间的这个值小于阈值则停止训练。默认值1e-6 范围：[1.0E-15, 0.999999999999999]	1e-6
是否交叉验证	是	是否进行交叉验证	否
交叉验证份数	是	交叉验证的份数 范围：[2, 20]	5
交叉验证划分方式	是	交叉验证每份的划分方式。选择分层划分的话，是按照分类目标列分层划分。目前支持：随机划分、分层划分、取余划分。	随机划分

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列。要求必须是数值或数值数组类型	无
标签列	是	真实标签列。要求不是数组类型，唯一值是2	无
正样本标签值	否	二分类的正样本的标签值，应在标签列中存在	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]	5
参数总组数	是	选择参数总组数，范围:[2,10]	5
alpha	是	正则项的选择，alpha=0代表L2正则，alpha=1代表L1正则，取中间值代表二者的线性组合 范围：[0.0, 1.0]	0
lambda	是	正则项的系数，0代表没有正则，越大代表正则强度越大 范围：[0.0, 2.0]	1
最大迭代轮数	是	最大迭代轮数 范围：[1, 1000]	20
目标收敛阈值	是	计算方式(old_val - new_val)/old_val。两次迭代之间的这个值小于阈值则停止训练。默认值1e-6 范围：[1.0E-15, 0.999999999999999]	1e-6
评估标准	是	选择评估标准： 准确率 pr曲线下面积 roc曲线下面积 F1分数 精确率 召回率	roc曲线下面积
保存模型数量	是	保存模型的数量	1

计算逻辑

逻辑回归二分类模型(Sigmoid函数)：

$$y = \frac{1}{1 + e^{-(\omega^T x + b)}}$$

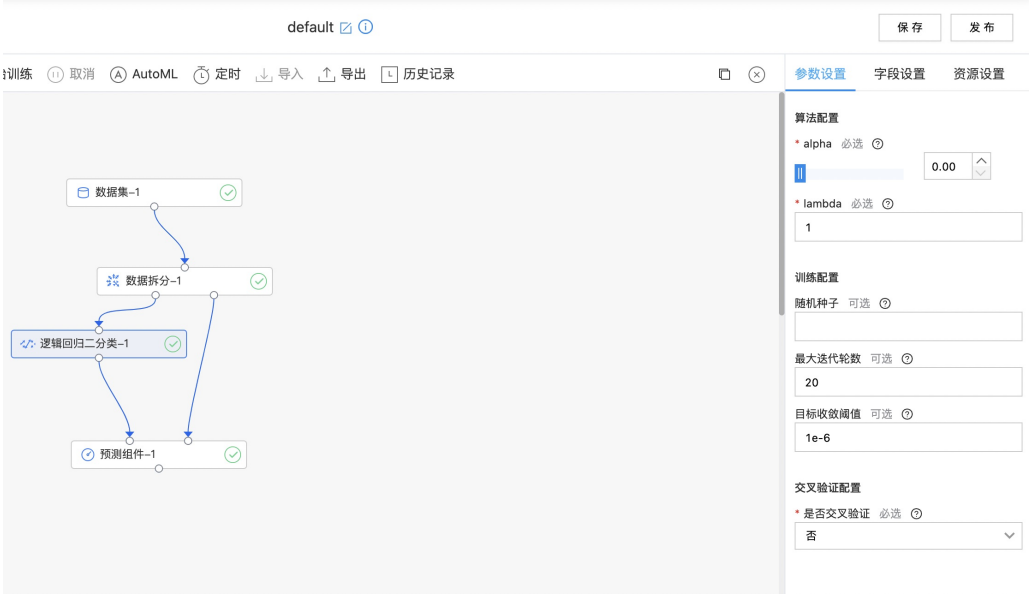
其中x为输入样本，y为分类概率

L1正则是指权值向量中各个元素的绝对值之和。

L2正则是指权值向量中各个元素的平方和然后再求平方根（可以看到Ridge回归的L2正则化项有平方符号）。

使用示例

构建如下图算子结构，配置算子参数后开始训练。



逻辑回归多分类(softmax)

经典逻辑回归的推广，是一个多分类算法。它通过 softmax 函数将线性回归模型产生的预测值转化为多个接近 0 或 1 的值。组件支持稠密数据格式。

输入

- 输入一个数据集，数据集的特征列必须是Double或Int。

输出

- 输出逻辑回归多分类模型，支持查看Top10分类的概率。

算子参数

参数名称	是否必选	参数描述	默认值
alpha	是	正则项的选择，alpha=0代表L2正则，alpha=1代表L1正则，取中间值代表二者的线性组合 范围：[0.0, 1.0]	0.00
lambda	是	正则项的系数，0代表没有正则，越大代表正则强度越大 范围：[0.0, 2.0]	1
随机种子	否	随机种子，用于保证多次训练结果相同	无
最大迭代轮数	否	最大迭代轮数 范围：[1, 1000]	20
目标收敛阈值	否	计算方式(old_val - new_val)/old_val。两次迭代之间的这个值小于阈值则停止训练。默认值1e-6 范围：[1.0E-15, 0.999999999999999]	1e-6
是否交叉验证	是	是否进行交叉验证	否
交叉份数	是	交叉验证的份数 范围：[2, 20]	5
交叉验证划分方式	是	交叉验证每份的划分方式。选择分层划分的话，是按照分类目标列分层划分	随机划分

字段参数

参数名称	是否必选	参数描述	默认值
标签列	是	预测使用的特征列，要求必须是数值或数值数组类型	无
特征列	是	交叉验证每份的划分方式。选择分层划分的话，是按照分类目标列分层划分。目前支持：随机划分、分层划分、取余划分。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]	5
参数总组数	是	选择参数总组数，范围:[2,10]	5
alpha	是	正则项的选择，alpha=0代表L2正则，alpha=1代表L1正则，取中间值代表二者的线性组合 范围：[0.0, 1.0]	0
lambda	是	正则项的系数，0代表没有正则，越大代表正则强度越大 范围：[0.0, 2.0]	1
最大迭代轮数	是	最大迭代轮数 范围：[1, 1000]	20
目标收敛阈值	是	计算方式(old_val - new_val)/old_val。两次迭代之间的这个值小于阈值则停止训练。默认值1e-6 范围：[1.0E-15, 0.999999999999999]	1e-6
评估标准	是	选择评估标准： 准确率 加权F1分数 加权精确率 加权召回率	准确率
保存模型数量	是	保存模型的数量	1

计算逻辑

逻辑回归多分类（k 类）模型：

$$P(y=j|x)=\frac{e^{x^Tw_j}}{\sum_{i=1}^{k}e^{x^Tw_i}}$$

使用示例

构建算子结构，配置参数，完成训练。



朴素贝叶斯二分类

朴素贝叶斯是一种简单的概率分类器，它基于贝叶斯定理应用特征之间的强独立假设（属性条件独立性假设）。朴素贝叶斯二分类对应于二分类任务。 朴素贝叶斯的一个优点是它只需要少量的训练数据来估计分类所需的参数。

输入

- 输入一个数据集，数据集的特征列必须是Double或Int，标记列必须是String或Int类型（如果列中的unique值超过了两种，运行算子时会报错）。

输出

- 输出朴素贝叶斯二分类模型。

算子参数

参数名称	是否必选	参数描述	默认值
模型	是	贝叶斯分类模型。如果使用伯努利模型，需要保证每个特征列都是0-1变量；如果使用多项式模型，需要保证每个特征列都是非负值。	多项式模型
平滑参数	是	平滑参数越大模型越平滑，不容易过拟合 范围：[0.0, inf)	1.0

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列。要求必须是数值或数值数组类型。	无
标签列	是	真实标签列。要求不是数组类型，唯一值是2。	无
正样本标签值	否	二分类的正样本的标签值，应在标签列中存在。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]	5
参数总组数	是	选择参数总组数，范围:[2,10]	5
平滑参数	是	平滑参数越大模型越平滑，不容易过拟合 范围：[0.0, inf	1.0
评估标准	是	选择评估标准： 准确率 pr曲线下面积 roc曲线下面积 F1分数 精确率 召回率	roc曲线下面积
保存模型数量	是	保存模型的数量	1

计算逻辑

朴素贝叶斯分类器：

$$y = argmax P(y_k) \prod_{i=1}^{n} P(x_i|y_k), y_k \in \mathcal{Y}$$

其中，y_k表示类别，$\mathcal{Y}$表示类别集合，x表示样本数据，x_i为对应样本的第i个属性值，参数估计使用极大似然法

多项式模型：

当特征是离散的时候，使用多项式模型。多项式模型在计算先验概率和条件概率时，会做一些平滑处理。如果不做平滑，当某一维特征的值没在训练样本中出现过时，会导致条件概率值为0，从而导致后验概率为0，加上平滑就可以克服这个问题。

$$P(y_k) = \frac{N_{yk} + \alpha}{N + k\alpha}$$，其中N是总的样本个数，k是总的类别个数，N_{yk}是类别为y_k的样本个数，α是平滑值。

$$P(x_i|y_k) = \frac{N_{yk,x_i} + \alpha}{N_{yk} + n\alpha}$$，其中N_{yk}是类别为y_k的样本个数，n是特征的维数，N_{yk,x_i}是类别为y_k的样本中，第i维特征的值是x_i的样本个数，α是平滑值。

伯努利模型：

与多项式模型一样，伯努利模型适用于离散特征的情况，所不同的是，伯努利模型中每个特征的取值只能是1和0(以文本分类为例，某个单词在文档中出现过，则其特征值为1，否则为0)。

伯努利模型中，条件概率的计算方式是：

$$\text{当特征值}x_i\text{为}1\text{时，}P(x_i|y_k) = P(x_i = 1|y_k)$$

$$\text{当特征值}x_i\text{为}0\text{时，}P(x_i|y_k) = 1 - P(x_i = 1|y_k)$$

使用示例

构建算子结构，配置参数，完成训练。



朴素贝叶斯多分类

朴素贝叶斯多分类是基于贝叶斯定理以及属性条件独立性假设得到的多分类模型，原理与二分类一样。

输入

- 输入一个数据集，数据集的特征列必须是Double或Int。

输出

- 输出朴素贝叶斯多分类模型，支持查看Top10分类的概率。

算子参数

参数名称	是否必选	参数描述	默认值
模型	是	贝叶斯分类模型。如果使用伯努利模型，需要保证每个特征列都是0-1变量；如果使用多项式模型，需要保证每个特征列都是非负值。	多项式模型
平滑参数	是	平滑参数越大模型越平滑，不容易过拟合 范围：[0.0, inf)	1.0

字段参数

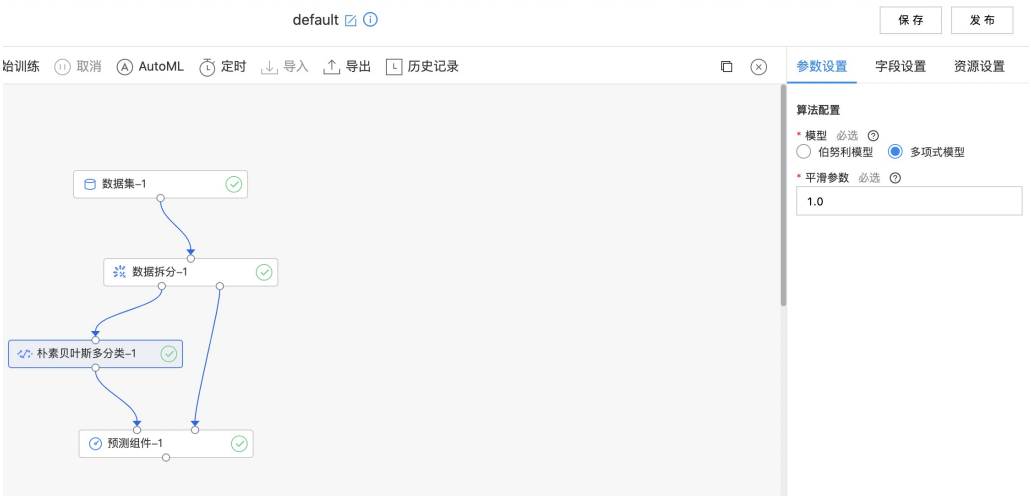
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型	无
标签列	是	预测目标列，要求不是数组类型。当数据量或标签数量过大时，需增加内存。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]	5
参数总组数	是	选择参数总组数，范围:[2,10]	5
平滑参数	是	平滑参数越大模型越平滑，不容易过拟合 范围：[0.0, inf	1.0
评估标准	是	选择评估标准： 准确率 加权F1分数 加权精确率 加权召回率	准确率
保存模型数量	是	保存模型的数量	1

使用示例

构建算子结构，配置参数，完成训练。



随机森林二分类

随机森林（RF）是 Bagging(并行式集成学习方法) 的一个扩展变体。RF 在以决策树为基学习器构建 Bagging 集成的基础上，进一步在决策树的训练过程中引入了随机属性选择。具体来说，传统决策树在选择划分属性时是在当前结点的属性集合（假定有 d 个属性）中选择一个最优属性；而在 RF 中，对基决策树的每个结点，先从该结点的属性集合中随机选择一个包含 k 个属性的子集，然后再从这个子集中选择一个最优属性用于划分。随机森林二分类对应于二分类任务。基学习器为 DecisionTree 二分类。结合策略是使用投票法。 随机森林简单、容易实现、计算开销小，它在很多现实任务中展现出强大的性能。

输入

- 输入一个数据集，数据集的特征列double/int，标记列是String或Int类型，标记列唯一值是2。

输出

- 输出随机森林二分类模型。

算子参数

参数名称	是否必选	参数描述	默认值
特征采样策略	是	每次分裂子树时，考虑多少个特征，支持以下策略： 所有特征 自动选择 总特证数以二为底的对数 定值 三分之一的特征 总特征数的一定比例 总特征数的平方根	自动选择
特征采样数	是	每次分裂子树时，考虑多少个特征 范围：[1, inf)	10
信息增益计算	是	分裂时计算子节点信息增益的策略，目前支持：信息熵、基尼不纯度。	基尼不纯度
树的最大深度	是	树的最大深度，范围：[2, 30]	5
最小信息增益	是	每次分裂所需的最小信息增益 范围：[0.0, inf)	0.0
节点最小样本数	是	节点最小的样本数，范围：[1, inf)	1
树的数量	是	子树的数量，范围：[1, inf)	20
随机种子	是	随机种子	1
采样率	是	训练每个子树时，使用的样本占总样本的比例。采样率不宜过小，至少应当保证(训练样本数*采样率>1)，否则训练出的模型在预测时会报错 Can not normalize the 0-vector. 范围：[0.001, 0.999]	0.632

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列。要求必须是数值或数值数组类型。	无
标签列	是	真实标签列。要求不是数组类型，唯一值是2。	无
正样本标签值	否	二分类的正样本的标签值，应在标签列中存在。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]	5
参数总组数	是	选择参数总组数，范围:[2,10]	5
树的最大深度	是	最大迭代次数，默认迭代100次 范围：[0, inf)	5
最小信息增益	是	每次分裂所需的最小信息增益 范围：[0.0, inf)	0.0
节点最小样本数	是	节点最小的样本数，范围：[1, inf)	1
树的数量	是	子树的数量，范围：[1, inf)	20
采样率	是	训练每个子树时，使用的样本占总样本的比例。采样率不宜过小，至少应当保证(训练样本数*采样率>1)，否则训练出的模型在预测时会报错 Can not normalize the 0-vector. 范围：[0.001, 0.999]	0.632
评估标准	是	选择评估标准： 准确率 pr曲线下面积 roc曲线下面积 F1分数 精确率 召回率	roc曲线下面积
保存模型数量	是	保存模型的数量	1

计算逻辑

基尼不纯度：从一个数据集中随机选取子项，度量其被错误的划分到其他组里的概率，也就是一个随机事件变成它的对立事件的概率。

$$I_G(f) = \sum_{i=1}^{m} f_i(1-f_i) = \sum_{i=1}^{m}(f_i - f_i^2) = \sum_{i=1}^{m} f_i - \sum_{i=1}^{m} f_i^2 = 1 - \sum_{i=1}^{m} f_i^2$$

其中f_i为某概率事件发生的概率

可见，基尼不纯度可以作为衡量系统混乱程度的标准；基尼不纯度越小，纯度越高，集合的有序程度越高，分类的效果越好；基尼不纯度为 0 时，表示集合类别一致；在决策树中，比较基尼不纯度的大小可以选择更好的决策条件（子节点）。

使用示例

构建算子结构，配置参数，完成训练。



随机森林多分类

随机森林多分类是基于随机森林且对应于多分类任务的多分类模型。原理与二分类一样，基学习器为 DecisionTree 多分类，结合策略是使用投票法。

输入

- 输入一个数据集，数据集的特征列double/int，标记列是String或Int类型，标记列唯一值>=3。

输出

- 输出随机森林多分类模型。

算子参数

参数名称	是否必选	参数描述	默认值
特征采样策略	是	每次分裂子树时，考虑多少个特征，支持以下策略： 所有特征 自动选择 总特证数以二为底的对数 定值 三分之一的特征 总特征数的一定比例 总特征数的平方根	自动选择
特征采样数	是	每次分裂子树时，考虑多少个特征 范围：[1, inf)	10
信息增益计算	是	分裂时计算子节点信息增益的策略，目前支持：信息熵、基尼不纯度。	基尼不纯度
树的最大深度	是	树的最大深度，范围：[2, 30]	5
最小信息增益	是	每次分裂所需的最小信息增益 范围：[0.0, inf)	0.0
节点最小样本数	是	节点最小的样本数，范围：[1, inf)	1
树的数量	是	子树的数量，范围：[1, inf)	20
随机种子	是	随机种子	1
采样率	是	训练每个子树时，使用的样本占总样本的比例。采样率不宜过小，至少应当保证(训练样本数*采样率>1)，否则训练出的模型在预测时会报错 Can not normalize the 0-vector. 范围：[0.001, 0.999]	0.632

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	是
标签列	是	每次分裂子树时，考虑多少个特征。	是

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]	5
参数总组数	是	选择参数总组数，范围:[2,10]	5
树的最大深度	是	最大迭代次数，默认迭代100次 范围：[0, inf)	5
最小信息增益	是	每次分裂所需的最小信息增益 范围：[0.0, inf)	0.0
节点最小样本数	是	节点最小的样本数，范围：[1, inf)	1
树的数量	是	子树的数量，范围：[1, inf)	20
采样率	是	训练每个子树时，使用的样本占总样本的比例。采样率不宜过小，至少应当保证(训练样本数*采样率>1)，否则训练出的模型在预测时会报错 Can not normalize the 0-vector. 范围：[0.001, 0.999]	0.632
评估标准	是	选择评估标准： 准确率 加权F1分数 加权精确率 加权召回率	准确率
保存模型数量	是	保存模型的数量	1

使用示例

构建算子结构，配置参数，完成训练。



GBDT二分类

梯度提升决策树（Gradient Boosting Decision Tree，GBDT）是一种基于boosting集成学习思想的加法模型，训练时采用前向分布算法进行贪婪的学习，每次迭代都学习一棵CART树来拟合之前 t-1 棵树的预测结果与训练样本真实值的残差。

输入

- 输入一个数据集，数据集的特征列必须是数值或数值数组类型，标记列必须是String或Int类型（如果列中的unique值超过了两种，运行算子时会报错）。

输出

- 输出GBDT二分类模型。

算子参数

参数名称	是否必选	参数描述	默认值
树的最大深度	是	GBDT中树(基学习器)的最大深度 范围：[0, inf)	5
连续特征分箱数	是	连续特征的最大分箱数 范围：[2, inf)	32
节点最小样本数	是	GBDT中树(基学习器)的节点最小样本数，当拆分节点后子节点中样本数量小于最小样本数时，该节点不进行拆分 范围：[1, inf)	1
最小信息增益	是	GBDT中树(基学习器)分裂时最小信息增益 范围：[0.0, inf)	0
训练基学习器选取训练数据集的百分比	是	每次迭代训练基学习器时所使用的训练数据集的百分比 范围：[1.0E-15, 1.0]	1
学习率	是	学习率,用于缩小(shrinking)每个基学习器的贡献 范围：[1.0E-15, 1.0]	0.1
最大迭代轮数	是	当迭代次数大于该数值时，停止迭代 范围：[1, 200]	100
随机种子	否	随机种子	5

字段参数

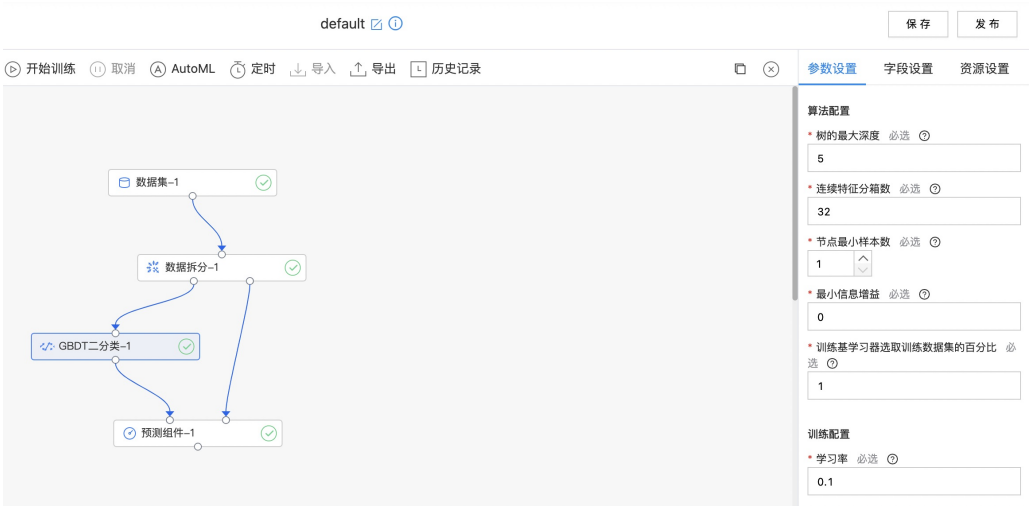
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	真实标签列。要求不是数组类型，唯一值是2。	无
正样本标签值	否	二分类的正样本的标签值，应在标签列中存在	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]	5
参数总组数	是	选择参数总组数，范围:[2,10]	5
树的最大深度	是	GBDT中树(基学习器)的最大深度 范围：[0, inf)	5
连续特征分箱数	是	连续特征的最大分箱数 范围：[2, inf)	32
节点最小样本数	是	GBDT中树(基学习器)的节点最小样本数，当拆分节点后子节点中样本数量小于最小样本数时，该节点不进行拆分 范围：[1, inf)	1
最小信息增益	是	GBDT中树(基学习器)分裂时最小信息增益 范围：[0.0, inf)	0
训练基学习器选取训练数据集的百分比	是	每次迭代训练基学习器时所使用的训练数据集的百分比 范围：[1.0E-15, 1.0]	1
最大迭代轮数	是	当迭代次数大于该数值时，停止迭代 范围：[1, 200]	100
学习率	是	学习率,用于缩小(shrinking)每个基学习器的贡献 范围：[1.0E-15, 1.0]	0.1
评估标准	是	选择评估标准： 准确率 pr曲线下面积 roc曲线下面积 F1分数 精确率 召回率	roc曲线下面积
保存模型数量	是	保存模型的数量	1

使用示例

构建算子结构，配置参数，完成训练。



GBDT多分类

梯度提升决策树（Gradient Boosting Decision Tree，GBDT）是一种基于boosting集成学习思想的加法模型，训练时采用前向分布算法进行贪婪的学习，每次迭代都学习一棵CART树来拟合之前 t-1 棵树的预测结果与训练样本真实值的残差，对于多分类问题来说，将所有树的分数相加，经过 softmax 函数转换，得到预测结果（类别）。

输入

- 输入一个数据集，数据集的特征列必须是Double或Int或数值数组，标记列必须是String或Int类型（如果列中的unique值不足三种，运行算子时会报错）。

输出

- 输出GBDT多分类模型。

算子参数

参数名称	是否必选	参数描述	默认值
树的最大深度	是	GBDT中树(基学习器)的最大深度 范围：[0, inf)。	5
连续特征分箱数	是	续特征的最大分箱数 范围：[2, inf)	32
节点最小样本数	是	GBDT中树(基学习器)的节点最小样本数，当拆分节点后子节点中样本数量小于最小样本数时，该节点不进行拆分 范围：[1, inf)	1
最小信息增益	是	GBDT中树(基学习器)分裂时最小信息增益 范围：[0.0, inf)	0
训练基学习器选取训练数据集的百分比	是	每次迭代训练基学习器时所使用的训练数据集的百分比 范围：[1.0E-15, 1.0]	1
学习率	是	学习率,用于缩小(shrinking)每个基学习器的贡献 范围：[1.0E-15, 1.0]	0.1
最大迭代轮数	是	当迭代次数大于该数值时，停止迭代 范围：[1, 200]	100
随机种子	否	随机种子	5

字段参数

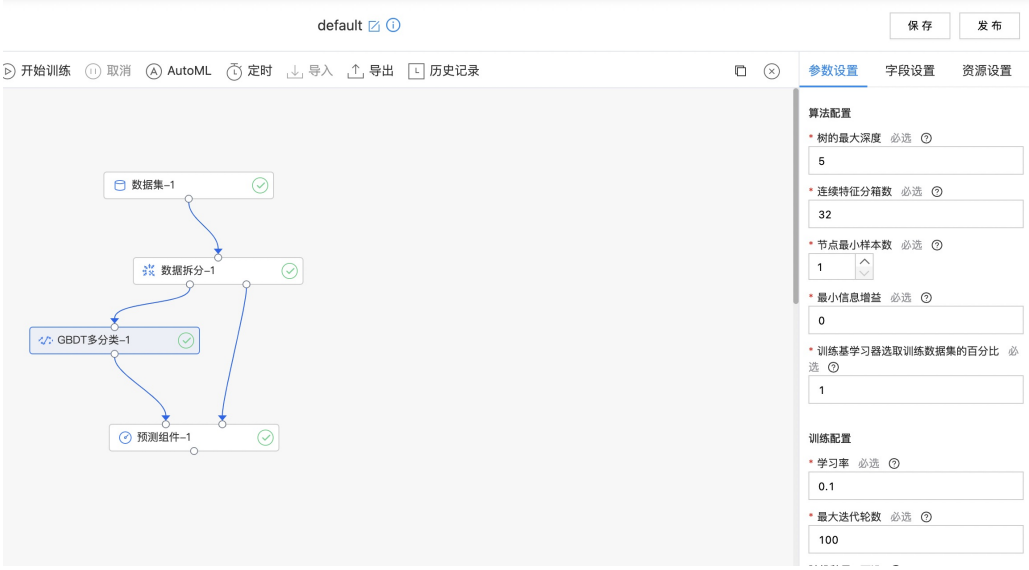
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求不是数组类型。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]	5
参数总组数	是	选择参数总组数，范围:[2,10]	5
树的最大深度	是	GBDT中树(基学习器)的最大深度 范围：[0, inf)	5
连续特征分箱数	是	连续特征的最大分箱数 范围：[2, inf)	32
节点最小样本数	是	GBDT中树(基学习器)的节点最小样本数，当拆分节点后子节点中样本数量小于最小样本数时，该节点不进行拆分 范围：[1, inf)	1
最小信息增益	是	GBDT中树(基学习器)分裂时最小信息增益 范围：[0.0, inf)	0
训练基学习器选取训练数据集的百分比	是	每次迭代训练基学习器时所使用的训练数据集的百分比 范围：[1.0E-15, 1.0]	1
最大迭代轮数	是	当迭代次数大于该数值时，停止迭代 范围：[1, 200]	100
学习率	是	学习率,用于缩小(shrinking)每个基学习器的贡献 范围：[1.0E-15, 1.0]	0.1
评估标准	是	选择评估标准： 准确率 加权F1分数 加权精确率 加权召回率	准确率
保存模型数量	是	保存模型的数量	1

使用示例

构建算子结构，配置参数，完成训练。



SVM二分类

支持向量机（SVM）是基于统计学习理论的一种机器学习方法，通过寻求结构化风险最小来提高学习机泛化能力，实现经验风险和置信范围的最小化，从而达到在统计样本量较少的情况下，亦能获得良好统计规律的目的。 SVM 二分类是线性支持向量机，不是采用核函数方式实现的。

输入

- 输入一个数据集，数据集的特征列必须是Double或Int，标记列必须是String或Int类型（如果列中的unique值超过了两种，运行算子时会报错）。

输出

- 输出SVM二分类模型。

算子参数

参数名称	是否必选	参数描述	默认值
最大迭代次数	是	控制算法的迭代次数 范围：[1, 10000]	100
正则参数	是	正则参数越大模型越平滑，不容易过拟合 范围：[0.0, inf)	0.0

字段参数

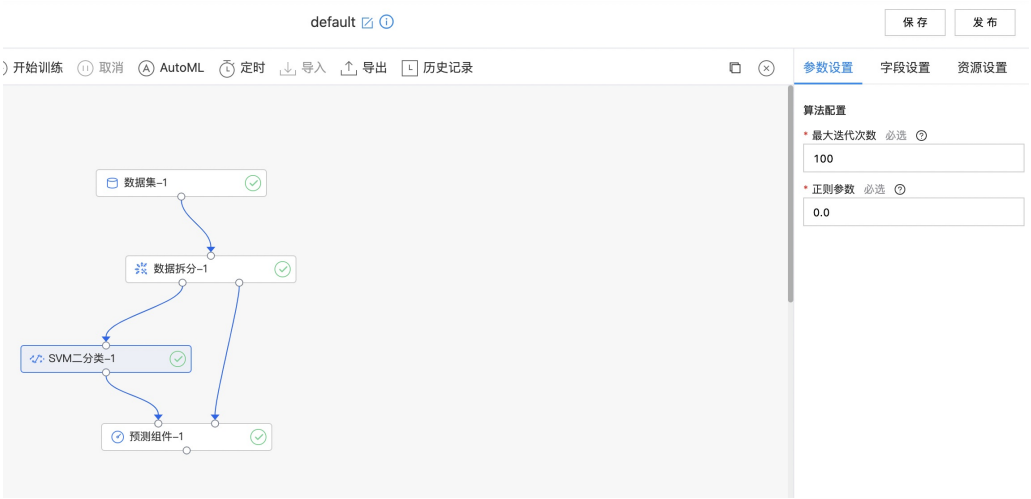
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列。要求必须是数值或数值数组类型	无
标签列	是	真实标签列。要求不是数组类型，唯一值是2。	无
正样本标签值	否	二分类的正样本的标签值，应在标签列中存在。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]	5
参数总组数	是	选择参数总组数，范围:[2,10]	5
正则参数	是	正则参数越大模型越平滑，不容易过拟合 范围：[0.0, inf)	0.0
最大迭代次数	是	控制算法的迭代次数 范围：[1, 10000]	100
评估标准	是	选择评估标准： 准确率 F1分数 精确率 召回率	F1分数
保存模型数量	是	保存模型的数量	1

使用示例

构建算子结构，配置参数，完成训练。



SVM多分类

支持向量机（SVM）多分类是基于线性支持向量机（二分类）以及“一对其余”拆分策略得到的多分类模型。

输入

- 输入一个数据集，数据集的特征列必须是Double或Int，标记列必须是String或Int类型（如果列中的unique值不足三种，运行算子时会报错）。

输出

- 输出SVM多分类模型。

算子参数

参数名称	是否必选	参数描述	默认值
最大迭代次数	是	控制算法的迭代次数 范围：[1, 10000]	100
正则参数	是	正则参数越大模型越平滑，不容易过拟合 范围：[0.0, inf)	0.0

字段参数

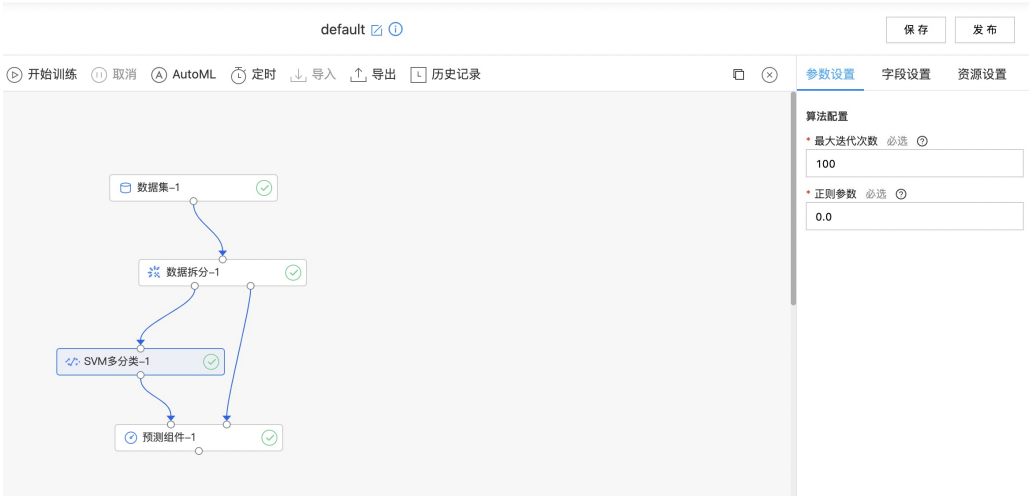
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型	无
标签列	是	预测目标列，要求不是数组类型。当数据量或标签数量过大时，需增加内存。	无

AutoML

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]	5
参数总组数	是	选择参数总组数，范围:[2,10]	5
最大迭代次数	是	控制算法的迭代次数 范围：[1, 10000]	100
正则参数	是	正则参数越大模型越平滑，不容易过拟合 范围：[0.0, inf)	0.0
评估标准	是	选择评估标准： 准确率 加权F1分数 加权精确率 加权召回率	准确率
保存模型数量	是	保存模型的数量	1

使用示例

构建算子结构，配置参数，完成训练。



LR二分类(稀疏)

经典逻辑回归是一个二分类算法，它通过对数几率函数（logistic function）将线性回归模型产生的预测值转化为一个接近 0 或 1 的值，组件支持稀疏数据格式。

输入

- 输入一个数据集，数据集的特征列必须是数值或数值数组或稀疏向量类型，标记列必须是String或Int类型（如果列中的unique值超过了两种，运行算子时会报错）。

输出

- 输出LR二分类(稀疏)模型。

算子参数

参数名称	是否必选	参数描述	默认值
二分类阈值	是	根据二分类阈值将预测结果映射到分类类别上 范围：[0.0, 1.0]	0.5
正则化惩罚参数	是	设置正则化惩罚参数，值越大对模型训练集的拟合系数的惩罚程度越大 范围：[0.0, inf)	0
ElasticNet参数	是	正则项的选择，当值为0代表L2正则，当值为1代表L1正则，取中间值代表二者的线性组合 范围：[0.0, 1.0]	0.00
带截距的回归	是	是否带截距的回归	开启
标准化处理	是	是否在训练前对特征标准化处理	开启
最大迭代轮数	否	当迭代次数大于该数值时，停止迭代 范围：[1, inf)	100
目标收敛阈值	否	计算方式(old_val - new_val)/old_val。两次迭代之间的这个值小于阈值则停止训练 范围：[1.0E-15, 0.999999999999999]	1e-6

字段参数

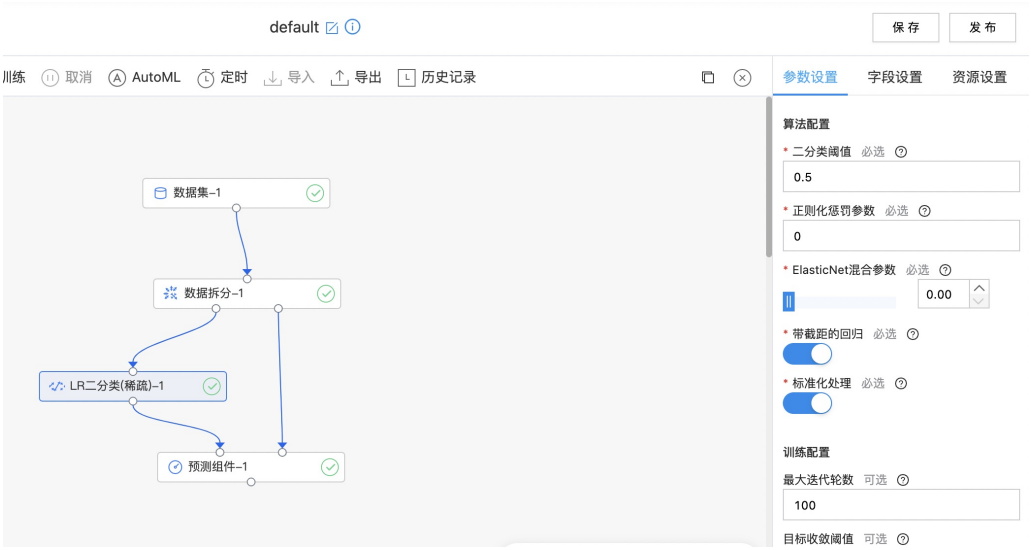
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是稀疏向量类型。	无
标签列	是	真实标签列。要求不是数组类型，唯一值是2。	无

AutoML

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]	5
参数总组数	是	选择参数总组数，范围:[2,10]	5
正则化惩罚参数	是		0
ElasticNet混合参数	是	正则项的选择，当值为0代表L2正则，当值为1代表L1正则，取中间值代表二者的线性组合 范围：[0.0, 1.0]	0
最大迭代轮数	是	当迭代次数大于该数值时，停止迭代 范围：[1, inf)	100
目标收敛阈值	是	计算方式(old_val - new_val)/old_val。两次迭代之间的这个值小于阈值则停止训练 范围：[1.0E-15, 0.999999999999999]	1e-6
评估标准	是	选择评估标准： 准确率 pr曲线下面积 roc曲线下面积 F1分数 精确率 召回率	准确率
保存模型数量	是	保存模型的数量	1

使用示例

构建算子结构，配置参数，完成训练。



LR多分类(稀疏)

经典逻辑回归的推广，是一个多分类算法。它通过 softmax 函数将线性回归模型产生的预测值转化为多个接近 0 或 1 的值，组件支持稀疏数据格式。

输入

- 输入一个数据集，数据集的特征列必须是数值或数值数组或稀疏向量类型，标记列必须是String或Int类型（如果列中的unique值少于两种，运行算子时会报错）。

输出

- 输出LR多分类(稀疏)模型，支持查看Top10分类的概率。

算子参数

参数名称	是否必选	参数描述	默认值
多分类阈值	是	根据多分类阈值以调整预测每个类别的概率，新概率为p/t(其中p为原始概率t是对应类别的阈值)，用":"间隔类别标签值和阈值，如labelItem(类别标签值):threshold(类别对应的阈值)，每个类别一行，阈值应大于等于0且最多只能有一个0	0
正则化惩罚参数	是	设置正则化惩罚参数，值越大对模型训练集的拟合系数的惩罚程度越大 范围：[0.0, inf)	0
ElasticNet参数	是	正则项的选择，当值为0代表L2正则，当值为1代表L1正则，取中间值代表二者的线性组合 范围：[0.0, 1.0]	0.00
带截距的回归	是	是否带截距的回归	开启
标准化处理	是	是否在训练前对特征标准化处理	开启
最大迭代轮数	否	当迭代次数大于该数值时，停止迭代 范围：[1, inf)	100
目标收敛阈值	否	计算方式(old_val - new_val)/old_val。两次迭代之间的这个值小于阈值则停止训练 范围：[1.0E-15, 0.999999999999999]	1e-6

字段参数

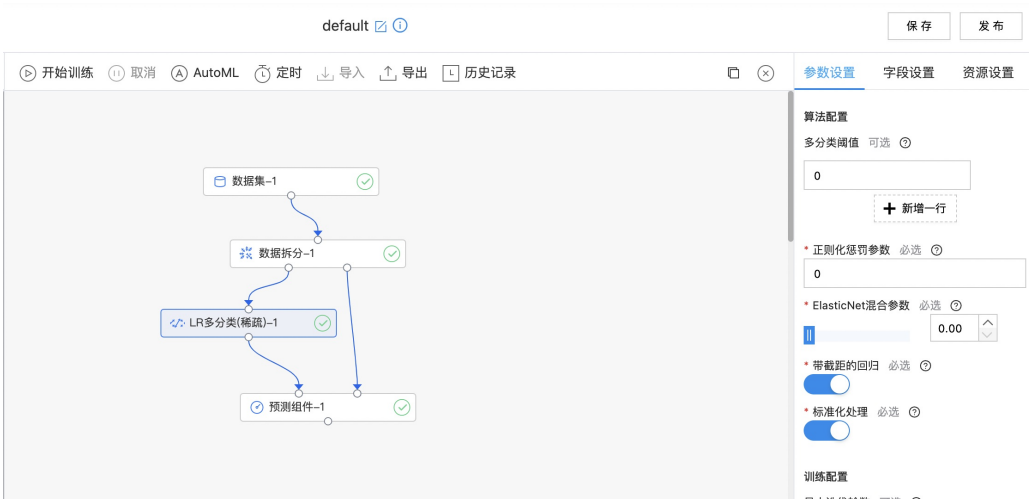
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是稀疏向量类型。	无
标签列	是	真实标签列，要求不是数组类型。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]	5
参数总组数	是	选择参数总组数，范围:[2,10]	5
正则化惩罚参数	是	设置正则化惩罚参数，值越大对模型训练集的拟合系数的惩罚程度越大 范围：[0.0, inf)	0
ElasticNet混合参数	是	正则项的选择，当值为0代表L2正则，当值为1代表L1正则，取中间值代表二者的线性组合 范围：[0.0, 1.0]	0
最大迭代轮数	是	当迭代次数大于该数值时，停止迭代 范围：[1, inf)	100
目标收敛阈值	是	计算方式(old_val - new_val)/old_val。两次迭代之间的这个值小于阈值则停止训练 范围：[1.0E-15, 0.999999999999999]	1e-6
评估标准	是	选择评估标准： 准确率 加权F1分数 加权精确率 加权召回率	准确率
保存模型数量	是	保存模型的数量	1

使用示例

构建算子结构，配置参数，完成训练。



XGBoost二分类

XGBoost是一种提升树模型，它是将多个树模型（CART）集成在一起，形成一个很强的分类器。 对于一个样本，每棵树都会预测出一个结果（分数），对于二分类问题来说，将所有树的分数相加，经过 sigmoid 函数转换，得到预测结果（类别）。

输入

- 输入一个数据集，选择特征列与标签列，特征列要求必须是数值或数值列表类型，标签列要求不是数组类型。

输出

- 输出XGBoost二分类模型。

算子参数

参数名称	是否必选	参数描述	默认值
目标函数	是	待优化的目标函数： binary：logistic binary：logitraw	binary：logistic
评估指标	否	评价指标：auc aucpr error logloss	error
树最大深度	是	xgboost中每棵树的最大深度，树越深通常模型越复杂，更容易过拟合。 范围：[1, inf)	6
迭代次数	是	xgboost迭代次数 范围：[1, 200]	5
最小叶子节点样本权重和	是	叶子结点需要的最小样本权重和 范围：[0.0, inf)	1
正负样本比例	是	正负样本比例 范围：[1.0E-8, inf)	1
学习率	是	学习率 范围：[0.0, 1.0]	0.3
gama	是	节点分裂所需的最小损失函数下降值 范围：[0.0, inf)	0
随机采样比例	是	构造每棵树的所用样本比例（样本采样比例） 范围：[0.01, 1.0]	1.00
随机采样特征比例	是	构造每棵树的所用特征比例 范围：[0.01, 1.0]	1.00
L1正则化系数	是	L1正则化系数 范围：[0.0, inf)	0
L2正则化系数	是	L2正则化系数 范围：[0.0, 2.0]	1
随机种子	是	随机种子	0
UseExternalMemory	是	是否使用ExternalMemory	开启

字段参数

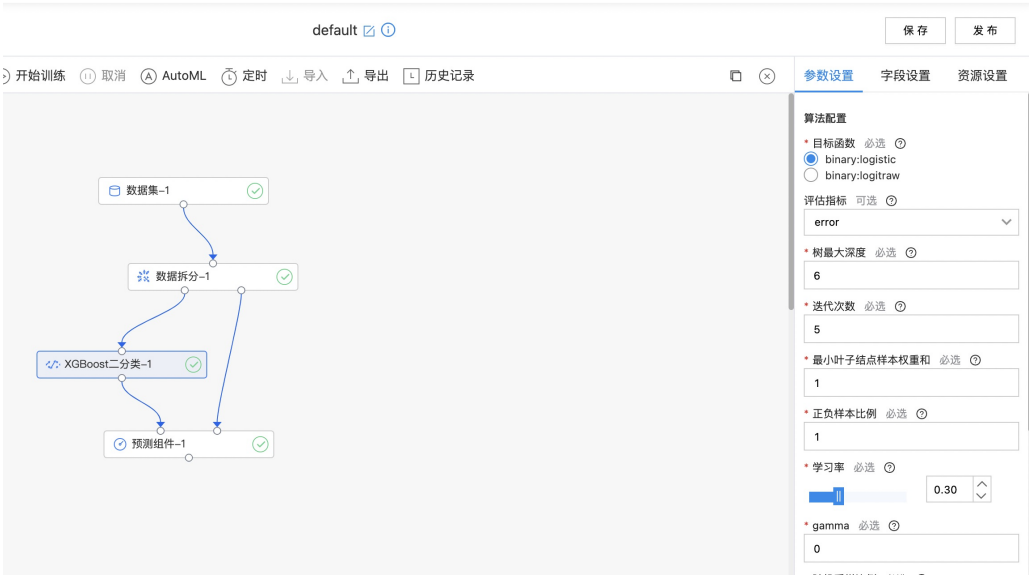
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值列表类型。当数据量过大时，需增加内存。	无
标签列	是	真实标签列，要求是只有两类的枚举类型。	无
正样本标签值	否	二分类的正样本的标签值，必须在标签列中存在。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]	5
参数总组数	是	选择参数总组数，范围:[2,10]	5
树最大深度	是	xgboost中每棵树的最大深度，树越深通常模型越复杂，更容易过拟合。 范围：[1, inf)	6
迭代次数	是	迭代次数	100
最小叶子节点样本权重和	是	叶子结点需要的最小样本权重和 范围：[0.0, inf)	1
学习率	是	学习率 范围：[0.0, 1.0]	0.3
gamma	是	节点分裂所需的最小损失函数下降值 范围：[0.0, inf)	0
L1正则化系数	是	L1正则化系数 范围：[0.0, inf)	0
L2正则化系数	是	L2正则化系数 范围：[0.0, 2.0]	1
评估标准	是	选择评估标准： 准确率 pr曲线下面积 roc曲线下面积 F1分数 精确率 召回率	roc曲线下面积
保存模型数量	是	保存模型的数量	1

使用示例

构建算子结构，配置参数，完成训练，算子支持查看全量特征重要性。



XGBoost多分类

XGBoost 是一种提升树模型，它是将多个树模型（CART）集成在一起，形成一个很强的分类器。 对于一个样本，每棵树都会预测出一个结果（分数），对于多分类问题来说，将所有树的分数相加，经过 softmax 函数转换，得到预测结果（类别）。

输入

- 输入一个数据集，选择特征列与标签列，特征列要求必须是数值或数值列表类型，标签列要求不是数组类型。

输出

- 输出XGBoost多分类模型，支持查看Top10分类的概率。

算子参数

参数名称	是否必选	参数描述	默认值
评估指标	否	评价指标： merror mlogloss	merror
树最大深度	是	xgboost中每棵树的最大深度，树越深通常模型越复杂，更容易过拟合。 范围：[1, inf)	6
迭代次数	是	xgboost迭代次数 范围：[1, 200]	5
最小叶子节点样本权重和	是	叶子结点需要的最小样本权重和 范围：[0.0, inf)	1
正负样本比例	是	正负样本比例 范围：[1.0E-8, inf)	1
学习率	是	学习率 范围：[0.0, 1.0]	0.3
gama	是	节点分裂所需的最小损失函数下降值 范围：[0.0, inf)	0
随机采样比例	是	构造每棵树的所用样本比例（样本采样比例） 范围：[0.01, 1.0]	1.00
随机采样特征比例	是	构造每棵树的所用特征比例 范围：[0.01, 1.0]	1.00
L1正则化系数	是	L1正则化系数 范围：[0.0, inf)	0
L2正则化系数	是	L2正则化系数 范围：[0.0, 2.0]	1
随机种子	是	随机种子	0
UseExternalMemory	是	是否使用ExternalMemory	开启

字段参数

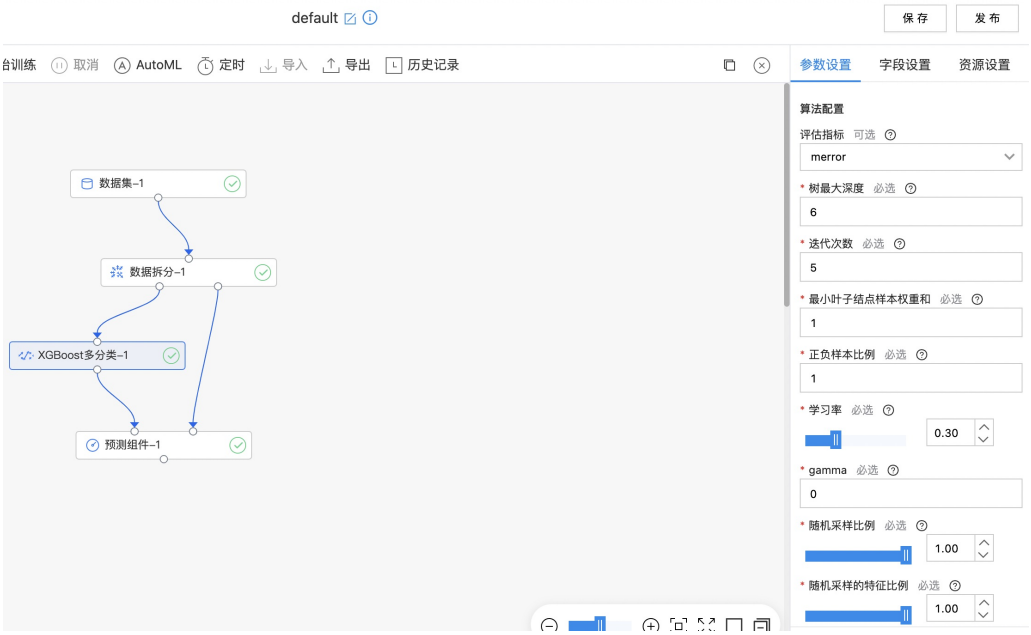
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求不是数组类型。当数据量或标签数量过大时，需增加内存。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]	5
参数总组数	是	选择参数总组数，范围:[2,10]	5
树最大深度	是	xgboost中每棵树的最大深度，树越深通常模型越复杂，更容易过拟合。 范围：[1, inf)	6
迭代次数	是	迭代次数	100
最小叶子节点样本权重和	是	叶子结点需要的最小样本权重和 范围：[0.0, inf)	1
学习率	是	学习率 范围：[0.0, 1.0]	0.3
gamma	是	节点分裂所需的最小损失函数下降值 范围：[0.0, inf)	0
L1正则化系数	是	L1正则化系数 范围：[0.0, inf)	0
L2正则化系数	是	L2正则化系数 范围：[0.0, 2.0]	1
评估标准	是	选择评估标准： 准确率 加权F1分数 加权精确率 加权召回率	roc曲线下面积
保存模型数量	是	保存模型的数量	1

使用示例

构建算子结构，配置参数，完成训练，算子支持查看全量特征重要性。



回归算法

回归算法

DecisionTree回归

决策树（DecisionTree）回归是基于回归树的决策树模型。一个回归树对应着输入空间（特征空间）的一个划分以及在划分单元上的输出值。输入空间的划分是通过遍历所有输入变量，找到最优的切分变量和最优的切分点，将输入空间划分为两部分，然后重复这个操作得到；划分单元上的输出值是该单元内所有样本点的均值。 划分度量使用均方误差的方法。

输入

- 输入一个数据集，数据集的特征列是数值或数组类型，标记列是数值类型。

输出

- 输出DecisionTree回归模型。

算子参数

参数名称	是否必选	参数描述	默认值
树最大深度	是	决策树最大深度 范围：[0, 30]。	5
连续特征分箱	是	决策树连续特征分箱数 范围：[2, inf)。	32
最小信息增益	是	决策树分裂时最小信息增益 范围：[0.0, inf)。	0
节点最小样本数	是	决策树节点最小样本数，当拆分节点后子节点中样本数量小于最小样本数时，该节点不进行拆分 范围：[1, inf)。	5
随机种子	否	随机种子	5

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求是数值类型。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]。	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]。	5
参数总组数	是	选择参数总组数，范围:[2,10]。	5
树最大深度	是	决策树最大深度 范围：[0, 30]。	5
连续特征分箱数	是	决策树连续特征分箱数 范围：[2, inf)。	32
节点最小样本数	是	决策树节点最小样本数，当拆分节点后子节点中样本数量小于最小样本数时，该节点不进行拆分 范围：[1, inf)。	5
最小信息增益	是	决策树分裂时最小信息增益 范围：[0.0, inf)。	0
评估标准	是	选择评估标准： 平均绝对误差 平均绝对百分误差 均方误差 判定系数 均方根误差	均方误差
保存模型数量	是	保存模型的数量	1

评估方法

下列公式中：

$h(x_i)$代表样本预测值，y_i代表样本真实值，y代表平均值

平均绝对误差：它表示预测值和观测值之间绝对误差的平均值。

$$MAE(X,h)=\frac{1}{m}\sum_{i=1}^{m}|h(x_i)-y_i|$$

平均绝对百分误差：它是一个百分比值，因此比其他统计量更容易理解。例如，如果 MAPE 为 5，则表示预测结果较真实结果平均偏离5%。

$$MAPE(X,h)=\sum_{i=1}^{m}|\frac{h(x_i)-y_i}{h(x_i)}|\times\frac{100}{n}$$

均方误差：表示预测数据和原始数据对应点误差的平方和的均值

$$MSE(X,h)=\frac{1}{m}\sum_{i=1}^{m}(h(x_i)-y_i)^2$$

判定系数：用于估计回归方程是否很好的拟合了样本的数据，判定系数为估计的回归方程提供了一个拟合优度的度量。

$$R^2=\frac{SSR}{SST}$$

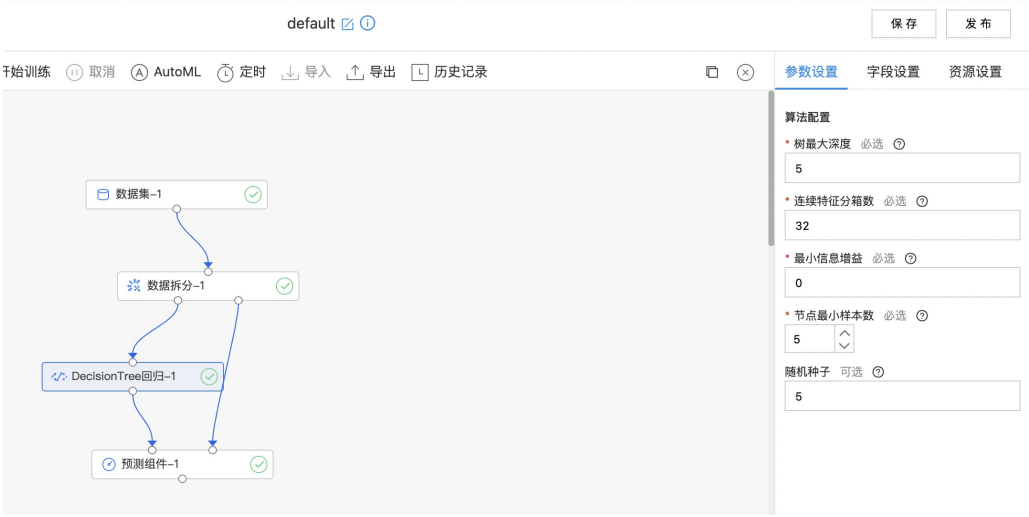
其中$SST=\sum_{i=1}^{m}(h(x_i)-y_i)^2, SSR=\sum_{i=1}^{m}(y_i-y)^2$

均方根误差：表示预测值和观测值之间差异（称为残差）的样本标准差。均方根误差为了说明样本的离散程度。做非线性拟合时,RMSE越小越好。

$$RMSE(X,h)=\sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(x_i)-y_i)^2}$$

使用示例

1.构建算子结构，配置参数，完成训练。



GBDT回归

GBDT是一种迭代的决策树算法，该算法由多棵决策树组成，所有树的结论累加起来做最终答案。它在被提出之初就和SVM一起被认为是泛化能力较强的算法。GBDT中的树是回归树（不是分类树），GBDT用来做回归预测，调整后也可以用于分类。

输入

- 输入一个数据集，数据集的特征列是数值或数组类型，标记列是数值类型。

输出

- 输出GBDT回归模型。

算子参数

参数名称	是否必选	参数描述	默认值
损失函数	是	最小化损失函数，支持 "squared" (L2) 和 "absolute" (L1)。	squared
树的最大深度	是	GBDT中树(基学习器)的最大深度 范围：[0, inf)。	5
连续特征分箱数	是	连续特征的最大分箱数 范围：[2, inf)。	32
节点最小样本数	是	GBDT中树(基学习器)的节点最小样本数，当拆分节点后子节点中样本数量小于最小样本数时，该节点不进行拆分 范围：[1, inf)。	1
最小信息增益	是	GBDT中树(基学习器)分裂时最小信息增益 范围：[0.0, inf)	0
训练集学习器选取训练数据集的百分比	是	每次迭代训练基学习器时所使用的训练数据集的百分比 范围：[1.0E-15, 1.0]。	1
学习率	是	学习率,用于缩小(shrinking)每个基学习器的贡献 范围：[1.0E-15, 1.0]。	0.1
最大迭代轮数	是	当迭代次数大于该数值时，停止迭代 范围：[1, 200]。	100
随机种子	否	随机种子	5

字段参数

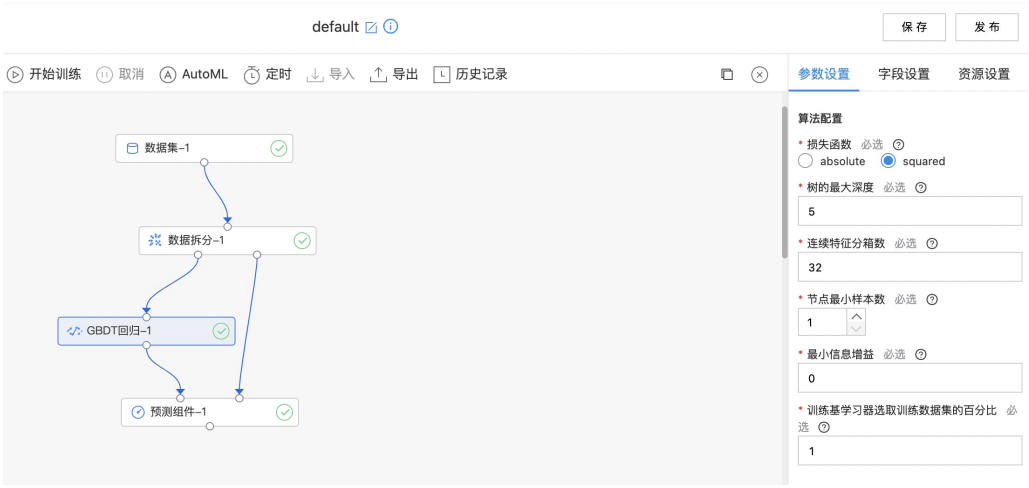
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求是数值类型。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]。	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]。	5
参数总组数	是	选择参数总组数，范围:[2,10]。	5
树的最大深度		GBDT中树(基学习器)的最大深度 范围：[0, inf)。	5
连续特征分箱数		连续特征的最大分箱数 范围：[2, inf)。	32
节点最小样本数		GBDT中树(基学习器)的节点最小样本数，当拆分节点后子节点中样本数量小于最小样本数时，该节点不进行拆分 范围：[1, inf)。	1
最小信息增益		GBDT中树(基学习器)分裂时最小信息增益 范围：[0.0, inf)	0
训练集学习器选取训练数据集的百分比		每次迭代训练基学习器时所使用的训练数据集的百分比 范围：[1.0E-15, 1.0]。	1
学习率		学习率,用于缩小(shrinking)每个基学习器的贡献 范围：[1.0E-15, 1.0]。	0.1
最大迭代轮数		当迭代次数大于该数值时，停止迭代 范围：[1, 200]。	100
评估标准	是	选择评估标准： 平均绝对误差 平均绝对百分误差 均方误差 判定系数 均方根误差	均方误差
保存模型数量	是	保存模型的数量	1

使用示例

构建算子结构，配置参数，完成训练。



广义线性回归

广义线性模型 (generalized linear model) 是在普通线性模型的基础上推广而得出的应用范围更广，更具实用性的回归模型。

输入

- 输入一个数据集，数据集的特征列是数值或数组类型，标记列是数值类型。

输出

- 输出广义线性回归模型，误差分布类型为twrrdie时不支持pmml导出。

算子参数

参数名称	是否必选	参数描述	默认值
误差分布类型	是	误差分布类型，只有误差分布类型为tweedie时支持标签列含负数(不支持导出为Pmml模型)，分布类型为binomial时标签列的值应在[0,1]之间，目前支持分布类型： binomial gamma gaussian tweedie poisson	gaussian
连接函数	是	连接函数，描述线性预测器和分布函数均值之间的关系，目前支持：logit cloglog probit identity inverse log sqrt	identity
方差函数中的幂	是	方差函数中的幂表示方差与分布均值之间的关系，支持的值为0和[1,inf),当值为0、1或2分别对应Gaussian, Poisson or Gamma Family 范围：[0.0, inf)	0.0
连接参数	否	当值为0, 1, -1 或 0.5时，分别对应Log, Identity, Inverse 或 Sqrt 连接	无
截距	是	是否开启截距	开启
正则化参数	是	L2正则化的正则化参数 范围：[0.0, inf)	0.0
最大迭代轮数	是	当迭代次数大于该数值时，停止迭代 范围：[1, 200]	25
目标收敛阈值	是	两次迭代之间的这个值小于阈值则停止训练 范围：[0.0, inf)	1e-6

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求是数值类型，只有误差分布类型为tweedie时支持标签列含负数。	无

计算逻辑

广义线性模型可以定义为：

$$Y = f(w_0 x_{i0} + w_2 x_{i2} + \dots w_n x_{in})$$

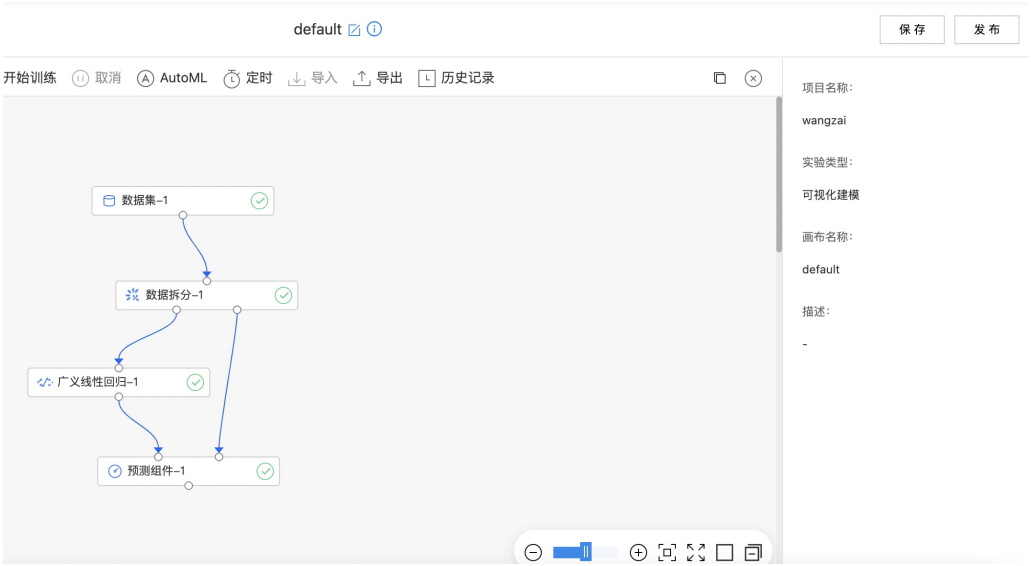
其中：

Y 为响应变量，$f(.)$ 函数称为 $linkfunction$，意思是通过该函数将 Y 和 $w_0 x_{i0} + w_2 x_{i2} + \dots w_n x_{in}$ 联结起来

响应变量的分布推广至指数分散族 (exponential dispersion family)：比如泊松分布、二项分布、伽玛分布、高斯分布等。

使用示例

构建算子结构，配置参数，完成训练。



线性回归

线性回归是分析因变量和多个自变量之间线性关系的模型。它试图学得一个线性模型以尽可能准确地预测实值输出标记。

输入

- 输入一个数据集，数据集的特征列是数值或数组类型，标记列是数值类型。

输出

- 输出线性回归模型。

算子参数

参数名称	是否必选	参数描述	默认值
alpha	是	正则项的选择，alpha=0代表L2正则，alpha=1代表L1正则，取中间值代表二者的线性组合 范围：[0.0, 1.0]。	0.00
lambda	是	正则项的系数，0代表没有正则，越大代表正则强度越大 范围：[0.0, 2.0]	1
随机种子	否	随机种子	无
最大迭代轮数	否	最大迭代轮数 范围：[1, 1000]	20
目标收敛阈值	否	计算方式(old_val - new_val)/old_val。两次迭代之间的这个值小于阈值则停止训练。默认值1e-6 范围：[1.0E-15, 0.999999999999999]	1e-6
是否交叉验证	是	是否进行交叉验证	否
交叉分数	是	交叉验证的份数 范围：[2, 20]	5
交叉验证划分方式	是	交叉验证每份的划分方式。选择分层划分的话，是按照分类目标列分层划分。目前支持：取余划分和随机划分。	随机划分

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求是数值类型。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]。	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]。	5
参数总组数	是	选择参数总组数，范围:[2,10]。	5
alpha	是	正则项的选择，alpha=0代表L2正则，alpha=1代表L1正则，取中间值代表二者的线性组合 范围：[0.0, 1.0]。	0
lambda	是	正则项的系数，0代表没有正则，越大代表正则强度越大 范围：[0.0, 2.0]。	1
最大迭代轮数	是	最大迭代轮数 范围：[1, 1000] 。	20
目标收敛阈值	是	计算方式(old_val - new_val)/old_val。两次迭代之间的这个值小于阈值则停止训练。默认值1e-6 范围：[1.0E-15, 0.999999999999999]。	1e-6
评估标准	是	选择评估标准： 平均绝对误差 平均绝对百分误差 均方误差 判定系数 均方根误差	均方误差
保存模型数量	是	保存模型的数量	1

计算逻辑

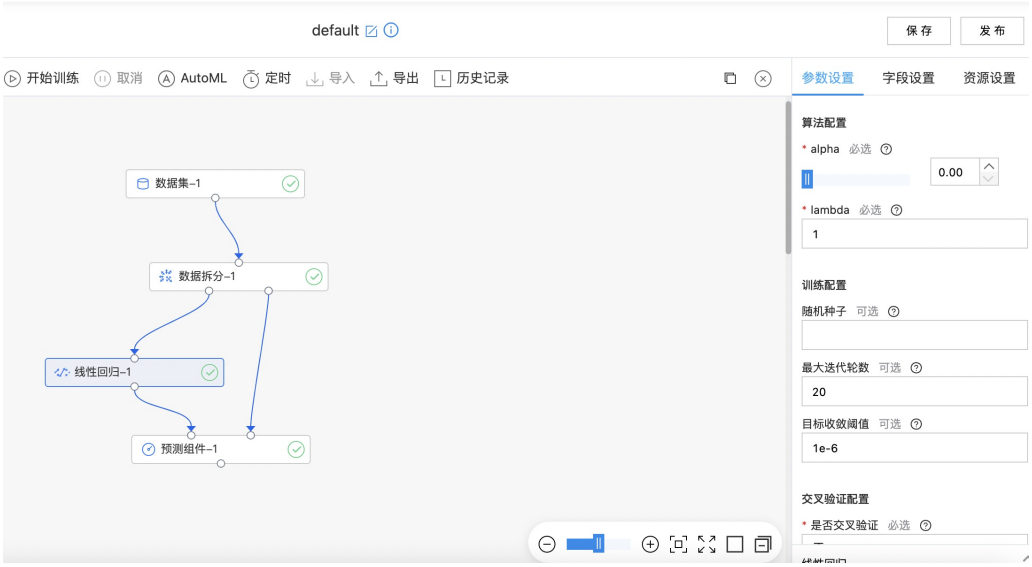
线性回归模型：

$$f(x_i) = \omega x_i + b, \text{ 其中} f(x_i) \text{是模型在} x_i \text{的预测值}$$

损失函数通过最小化训练样本到模型的距离，进而迭代出最优模型参数。

使用示例

构建算子结构，配置参数，完成训练。



随机森林回归

随机森林回归是基于随机森林且对应于回归任务的回归模型。原理与分类一致。基学习器为 DecisionTree 回归。结合策略是使用平均法。

输入

- 输入一个数据集，数据集的特征列是数值或数组类型，标记列是数值类型。

输出

- 输出随机森林回归模型。

算子参数

参数名称	是否必选	参数描述	默认值
特征采样策略	是	选择特征采样： 所有特征 自动选择 总特征树以二为底的对数 定值 三分之一的特征 总特征数的一定比例 总特证数的平方根	自动选择
特征采样数	是	每次分裂子树时，考虑多少个特征 范围：[1, inf)。	10
比例	是	每次分裂子树时，考虑的特征数占总特征数的比例 范围：[0.001, 0.999]。	0.632
树的最大深度	是	子树的最大深度，范围：[2, 30]。	5
最小信息增益	是	每次分裂所需的最小信息增益 范围：[0.0, inf)。	0.0
节点最小样本数	是	随机森林中树的节点最小样本数，当拆分节点后子节点中样本数量小于最小样本数时，该节点不进行拆分，范围：[1, inf)。	1
树的数量	是	树的数量，范围：[1, inf)。	20
随机种子	是	随机种子	1
采样率	是	训练每个子树时，使用的样本占总样本的比例。采样率不宜过小，至少应当保证(训练样本数*采样率>1)，否则训练出的模型在预测时会报错 Can not normalize the 0-vector. 范围：[0.001, 0.999]。	0.632

字段参数

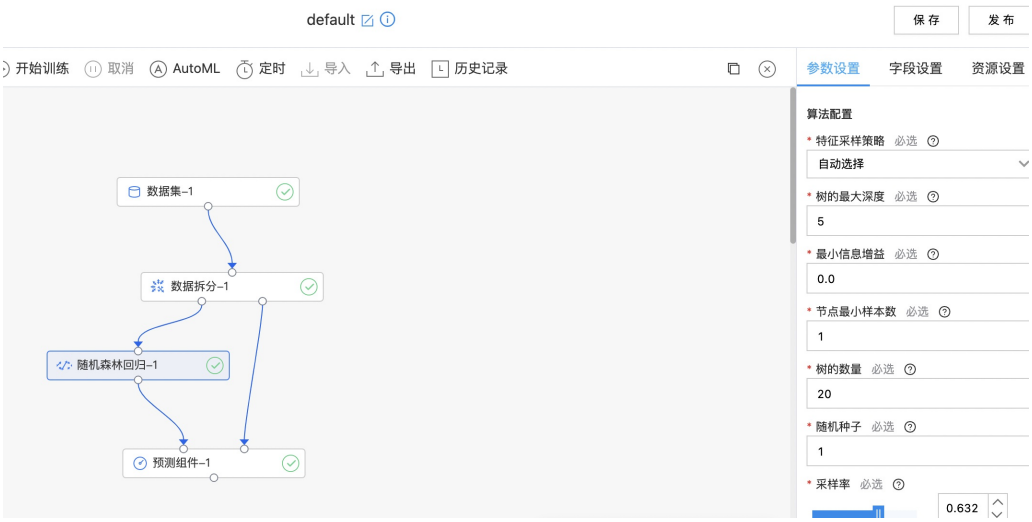
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求是数值类型。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]。	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]。	5
参数总组数	是	选择参数总组数，范围:[2,10]。	5
树的最大深度	是	子树的最大深度，范围：[2, 30]。	5
最小信息增益	是	每次分裂所需的最小信息增益 范围：[0.0, inf)。	0.0
节点最小样本数	是	随机森林中树的节点最小样本数，当拆分节点后子节点中样本数量小于最小样本数时，该节点不进行拆分，范围：[1, inf)。	1
树的数量	是	树的数量，范围：[1, inf)。	20
采样率	是	训练每个子树时，使用的样本占总样本的比例。采样率不宜过小，至少应当保证(训练样本数*采样率>1)，否则训练出的模型在预测时会报错 Can not normalize the 0-vector. 范围：[0.001, 0.999]。	0.632
评估标准	是	选择评估标准： 平均绝对误差 平均绝对百分误差 均方误差 判定系数 均方根误差	均方误差
保存模型数量	是	保存模型的数量。	1

使用示例

构建算子结构，配置参数，完成训练。



岭回归

岭回归可以被看作为一种改良后的最小二乘法，它通过向损失中添加L2正则项有效防止模型出现过拟合，且有助于解决非满秩条件下求逆困难的问题，从而提升模型的解释能力。

输入

- 输入一个数据集，数据集的特征列是数值或数组类型，标记列是数值类型。

输出

- 输出岭回归模型。

算子参数

参数名称	是否必选	参数描述	默认值
lambda	是	正则项的系数，0代表没有正则，越大代表正则强度越大 范围：[0.0, 2.0]。	1
随机种子	否	随机种子，用于保证多次训练结果相同。	无
最大迭代轮数	否	最大迭代轮数 范围：[1, 1000]。	20
目标收敛阈值	否	计算方式(old_val - new_val)/old_val。两次迭代之间的这个值小于阈值则停止训练。默认值1e-6 范围：[1.0E-15, 0.999999999999999]。	1e-6
是否交叉验证	是	是否进行交叉验证	否
交叉份数	是	交叉验证的份数 范围：[2, 20]。	5
交叉验证划分方式	是	交叉验证每份的划分方式。选择分层划分的话，是按照分类目标列分层划分。目前支持取余划分和随机划分。	随机划分

字段参数

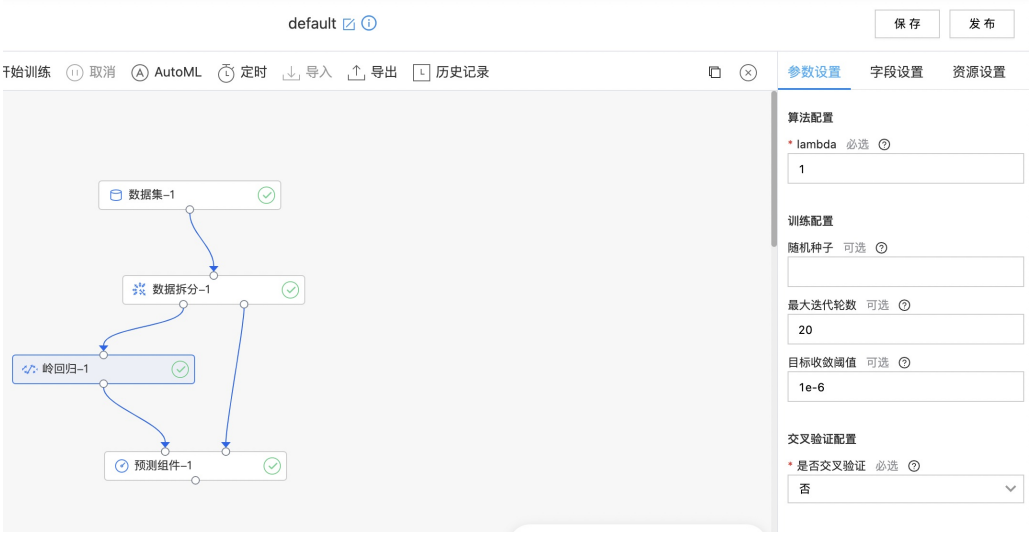
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求是数值类型。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]。	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]。	5
参数总组数	是	选择参数总组数，范围:[2,10]。	5
lambda	是	正则项的系数，0代表没有正则，越大代表正则强度越大 范围：[0.0, 2.0]。	1
最大迭代轮数	是	最大迭代轮数 范围：[1, 1000]。	20
目标收敛阈值	是	计算方式(old_val - new_val)/old_val。两次迭代之间的这个值小于阈值则停止训练。默认值1e-6 范围：[1.0E-15, 0.999999999999999]。	1e-6
评估标准	是	选择评估标准： 平均绝对误差 平均绝对百分误差 均方误差 判定系数 均方根误差	均方误差
保存模型数量	是	保存模型的数量。	1

使用示例

构建算子结构，配置参数，完成训练。



XGBoost回归

XGBoost 是一种迭代的决策树算法，它是由多棵决策树组成，所有树的结论累加起来做最终答案。XGBoost 几乎可用于所有回归问题(线性/非线性)。

输入

- 输入一个数据集，数据集的特征列是数值或数组类型，标记列是数值类型。

输出

- 输出XGBoost回归模型。

算子参数

参数名称	是否必选	参数描述	默认值
目标函数	是	待优化的目标函数，如果选择reg:logistic，标签列只能在0到1之间。目前支持： reg:gamma reg:logistic reg:squarederror reg:tweedie	reg:squarederror
评估指标	是	选择评价指标： rmse mae	rmse
树最大深度	是	xgboost中每棵树的最大深度，树越深通常模型越复杂，更容易过拟合。 范围：[1, inf)	6
迭代次数	是	xgboost迭代次数 范围：[1, 200]。	5
最小叶子节点样本权重和	是	叶子结点需要的最小样本权重和 范围：[0.0, inf)。	1
正负样本比例	是	正负样本比例 范围：[1.0E-8, inf)。	1
学习率	是	学习率 范围：[0.0, 1.0]。	0.30
gamma	是	节点分裂所需的最小损失函数下降值 范围：[0.0, inf)。	0
随机采样比例	是	构造每棵树的所用样本比例（样本采样比例） 范围：[0.01, 1.0]。	1.00
随机采样的特正比例	是	构造每棵树的所用特征比例 范围：[0.01, 1.0]。	1.00
L1正则化系数	是	L1正则化系数 范围：[0.0, inf)。	0
L2正则化系数	是	L2正则化系数 范围：[0.0, inf)。	1
随机种子	是	随机种子	0
UseExternalMemory	是	是否使用ExternalMemory	开启

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。当数据量过大时，需增加内存。	无
标签列	是	预测标签列，要求是数值类型。	无

AutoML参数

参数名称	是否必选	参数描述	默认值
数据拆分比例	是	选择数据拆分比例，范围：[0.1，0.9]。	0.8
调参方式	是	选择调参方式： GridSearch RandomSearch	GridSearch
网格拆分数	是	选择网格拆分数，范围:[2,10]。	5
参数总组数	是	选择参数总组数，范围:[2,10]。	5
树最大深度	是	xgboost中每棵树的最大深度，树越深通常模型越复杂，更容易过拟合。 范围：[1, inf)	6
迭代次数	是	xgboost迭代次数 范围：[1, 200]。	5
最小叶子节点样本权重和	是	叶子结点需要的最小样本权重和 范围：[0.0, inf)。	1
学习率	是	学习率 范围：[0.0, 1.0]。	0.3
gamma	是	节点分裂所需的最小损失函数下降值 范围：[0.0, inf)。	0
L1正则化系数	是	L1正则化系数 范围：[0.0, inf)。	0
L2正则化系数	是	L2正则化系数 范围：[0.0, inf)。	1
评估标准	是	选择评估标准： 平均绝对误差 平均绝对百分误差 均方误差 判定系数 均方根误差	均方误差
保存模型数量	是	保存模型的数量。	1

使用示例

构建算子结构，配置参数，完成训练，算子支持查看全量特征重要性。



异常检测算法

异常检测算法

IsolationForest

在高维数据集中实现异常值检测的一种有效方法是使用随机森林。隔离森林（IsolationForest）通过随机选择特征然后随机选择所选特征的最大值和最小值之间的分割值来隔离观测。 由于递归划分可以由树形结构表示，因此隔离样本所需的分割次数等同于从根节点到终止节点的路径长度。 在这样的随机树的森林中取平均的路径长度作为决策量度。 随机划分能为异常观测产生明显的较短路径。 因此，当随机树的森林共同为特

定样本产生较短的路径长度时，这些样本就很有可能是异常观测。

输入

- 输入一个数据集，选择需要进行异常检测的特征列。

输出

- 输出IsolationForest模型。

算子参数

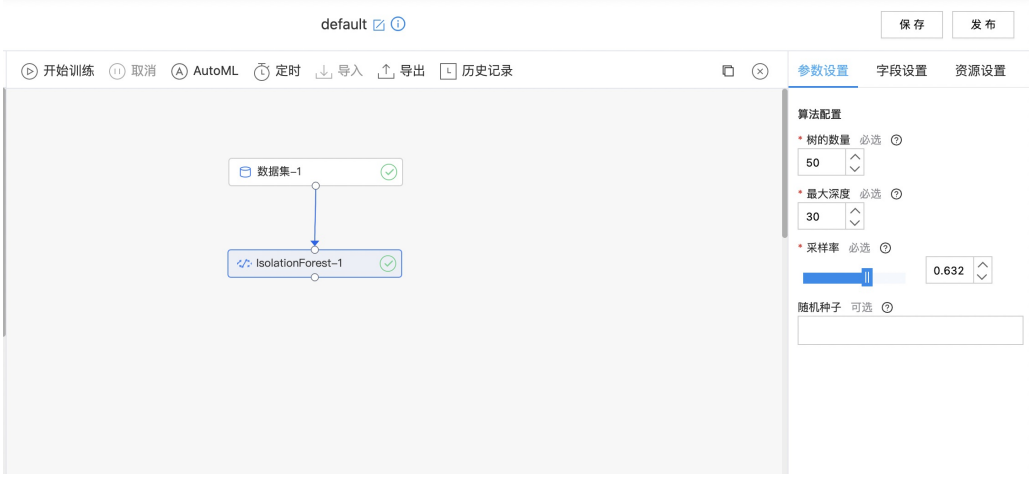
参数名称	是否必选	参数描述	默认值
树的数量	是	模型中树的数量。数量较大时建议增加内存 范围：[1, 1000]。	50
最大深度	是	每棵树最大的深度 范围：[1, 50]。	30
采样率	是	计算每棵树时从训练数据集随机采样的比率 范围：[0.001, 1.0]。	0.632
随机种子	否	随机种子，用于保证多次训练结果相同。	无

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	支持数值或数值数组类型。	无

使用示例

如下图所示，构建算子结构，选择特征列，配置检测条件参数。



聚类算法

聚类算法

高斯混合模型聚类

高斯混合模型（Gaussian Mixture Model）通常简称GMM，高斯混合模型是由多个高斯分布的结合组成的概率分布模型，是一种业界广泛使用的聚类算法，该方法使用了高斯分布作为参数模型，并使用了期望最大（Expectation Maximization，简称EM）算法进行训练。

输入

- 输入一个数据集，选择需要聚类的特征列，特征列只支持数值或数值列表类型。

输出

- 输出高斯混合聚类模型。

算子参数

参数名称	是否必选	参数描述	默认值
聚类数	是	聚类数 范围：[2, 500]。	2
最大迭代次数	是	最大迭代次数 范围：[1, 10000]。	100
收敛容差	否	当小于该值时，停止迭代 范围：[0.0, inf)。	0.01
随机种子	否	随机种子，用于保证多次训练结果相同。	-1

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	支持数值或数值列表类型。	无

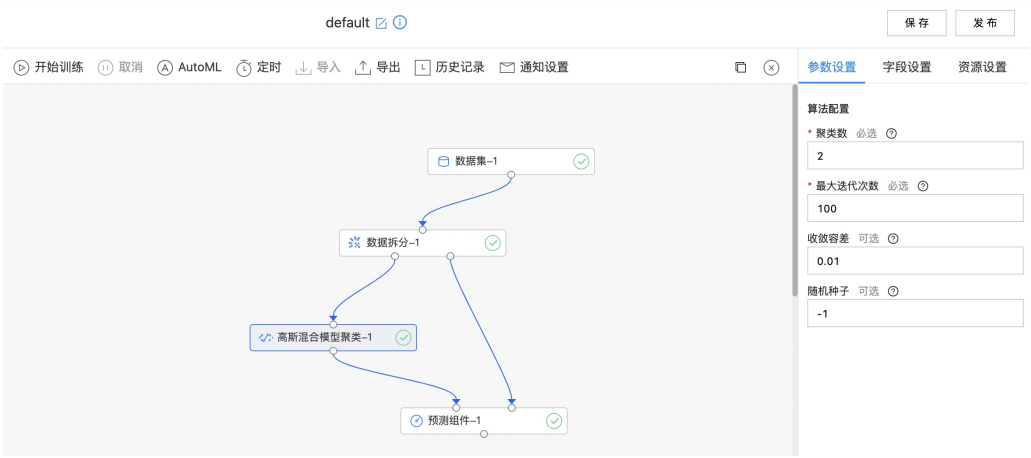
计算逻辑

高斯混合模型：

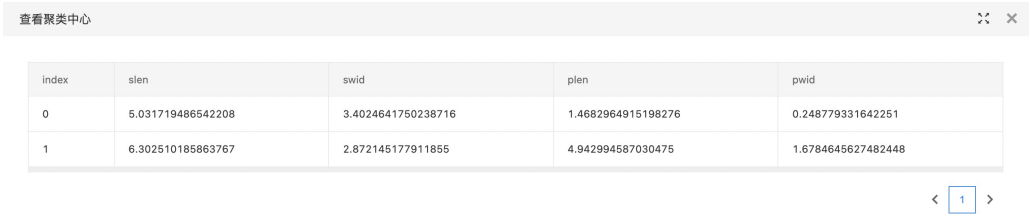
$$p(x) = \sum_{k=1}^{K} \pi_k p(x|k)$$

其中K为模型的个数，π_k为第k个高斯的权重，则为第k个高斯的概率密度函数

使用示例



查看聚类中心。



查看聚类结果。

查看数据

slen(double)	swid(double)	plen(double)	pwid(double)	class(string)
4.300000190734863	3	1.100000023841858	0.10000000149011612	Iris-setosa
4.400000095367432	3.200000047683716	1.2999999523162842	0.20000000298023224	Iris-setosa
4.599999904632568	3.0999999046325684	1.5	0.20000000298023224	Iris-setosa
5	2	3.5	1	Iris-versicolor
5	2.299999952316284	3.299999952316284	1	Iris-versicolor
5	3.299999952316284	1.399999976158142	0.20000000298023224	Iris-setosa
5	3.4000000953674316	1.5	0.20000000298023224	Iris-setosa
5	3.5999999046325684	1.399999976158142	0.20000000298023224	Iris-setosa
5.099999904632568	3.799999952316284	1.5	0.30000001192092896	Iris-setosa
5.099999904632568	3.799999952316284	1.899999976158142	0.4000000059604645	Iris-setosa

KMeans聚类

KMeans 聚类是一种得到最广泛使用的聚类算法，把 n 个对象分为 k 个簇，使簇内具有较高的相似度。相似度根据一个簇中对象的平均值来计算。 算法首先随机地选择 k 个对象，每个对象初始地代表了一个簇的平均值或中心。对剩余的每个对象根据其与各个簇中心的距离，将它赋给最近的簇，然后重新计算每个簇的平均值。这个过程不断重复，直到准则函数收敛。 它假设对象属性来自于空间向量，并且目标是使各个群组内部的均方误差总和最小。

输入

- 输入一个数据集，选择需要聚类的特征列，支持数值或数值列表类型。

输出

- 输出KMeans聚类。

算子参数

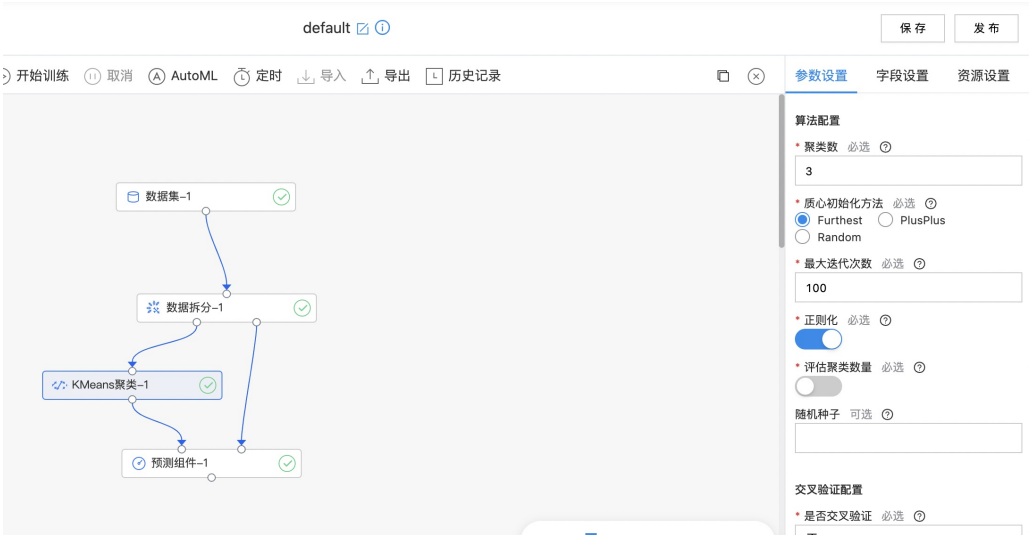
参数名称	是否必选	参数描述	默认值
聚类数	是	聚类数 范围：[2, 500].	2
质心初始化方法	是	质心初始化方法： Furthest PlusPlus Random	Furthest
最大迭代次数	是	最大迭代次数 范围：[1, 10000]	100
正则化	是	是否进行正则化处理	开启
评估聚类数量	是	开启后，算法会从1到设置的聚类数依次评估合适的聚类数	关闭
随机种子	否	随机种子，用于保证多次训练结果相同	无
是否交叉验证	是	是否进行交叉验证	否
交叉份数	是	交叉验证的份数 范围：[2, 20]	2
交叉验证划分方式	是	交叉验证每份的划分方式，目前支持取余划分和随机划分	随机划分

字段参数

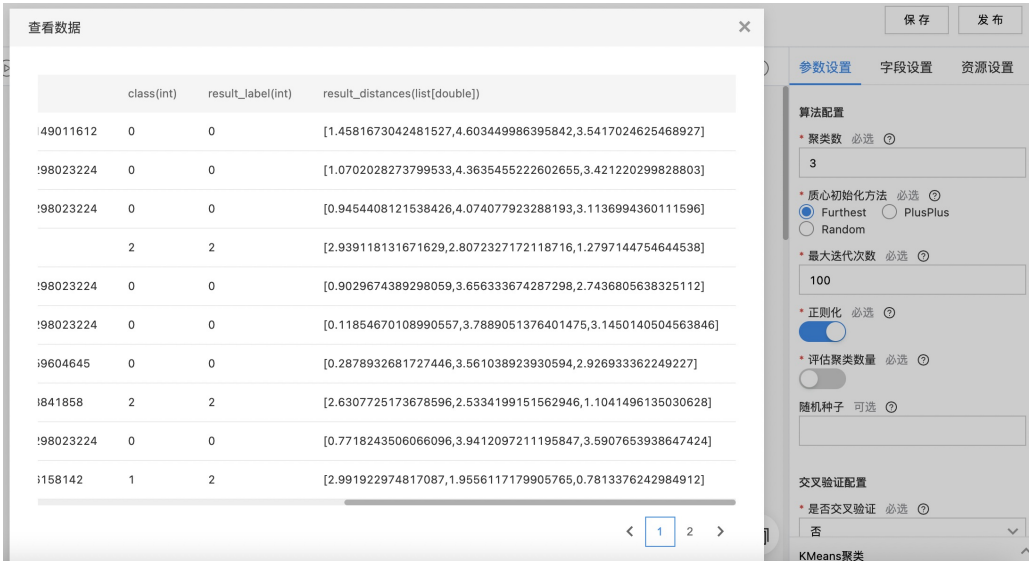
参数名称	是否必选	参数描述	默认值
特征列	是	支持数值或数值列表类型	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看聚类结果。



Python算法组件

Python算法组件

注意：Python算法组件，需连接Python预测组件。

AP聚类

AP 算法的基本思想是将全部样本看作网络的节点，通过网络中各条边的消息传递计算出各样本的聚类中心。聚类过程中，共有两种消息在各节点间传递，分别是吸引度(responsibility)和归属度(availability)。AP 算法通过迭代过程不断更新每一个点的吸引度和归属度值，直到产生m个高质量的 Exemplar（类似于质心），同时将其余的数据点分配到相应的聚类中。

输入

- 输入一个数据集，数据集的特征列必须是数值类型；数据点的参考度列(可选)用于刻画每个点的偏好，需要是数值类型。

输出

- 输出AP聚类模型。聚类模型预测结果为在原始列上加上两列：result_label，代表聚类index，从0开始；result_distances，代表样本到各个cluster的距离。

算子参数

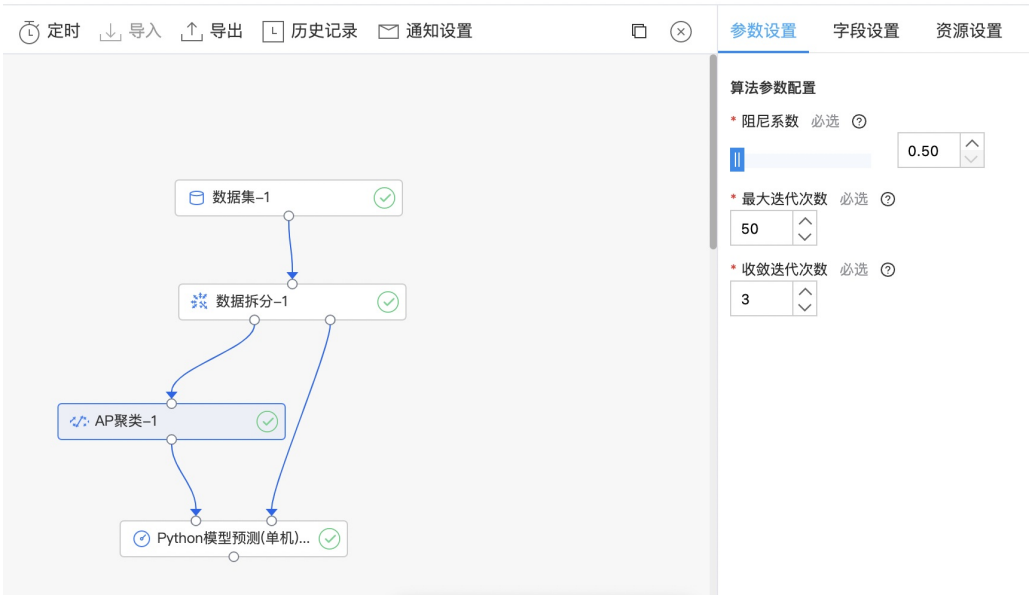
参数名称	是否必选	参数描述	默认值
阻尼系数	是	为了避免在更新时出现数值振荡，默认值：0.5 范围：[0.5, 0.99]	0.5
最大迭代次数	是	控制算法的迭代次数，默认值：50。此算法时间复杂度较高，为O(NNlogN)，其中N为样本数，不适合用于大数据（N>5000）聚类 范围：[1, inf)	50
收敛迭代次数	是	迭代次数，估计簇的数量没有变化，停止收敛，默认值：3 范围：[1, inf)	3

字段参数

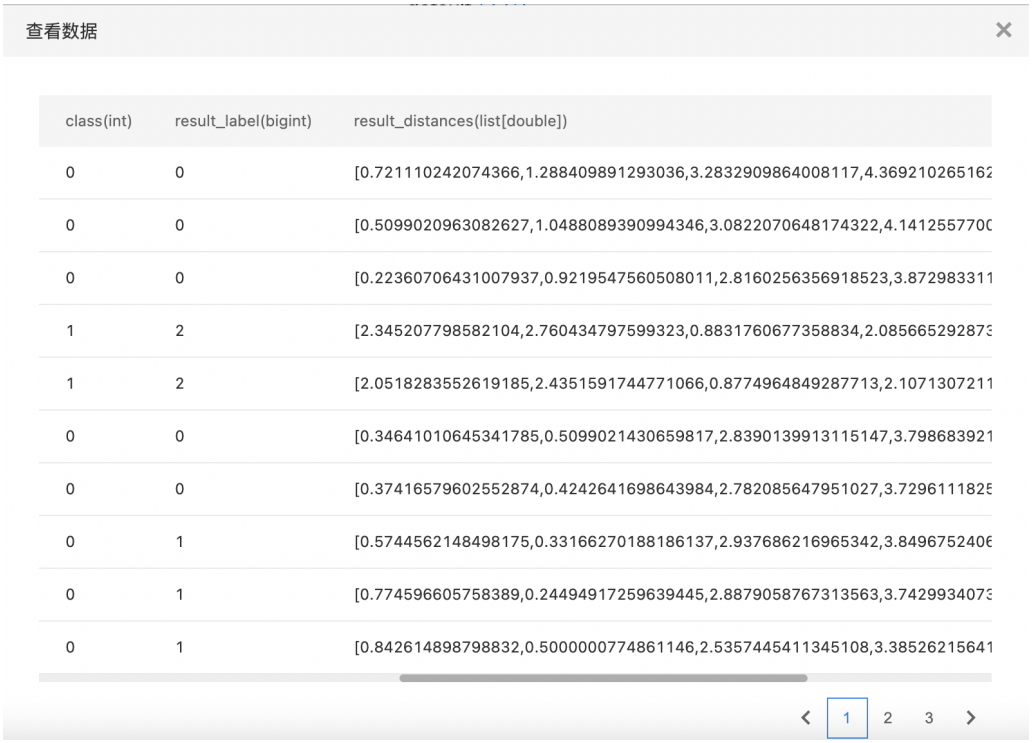
参数名称	是否必选	参数描述	默认值
特征列	是	支持数值或数值数组类型	无
数据点的参考度列	否	每个点的偏好-具有较大偏好值的点更有可能被选择为聚类中心。 聚类中心的个数（即簇的数量）受输入偏好值的影响。 如果偏好值未作为参数传递，则将其设置为输入相似度的中位数	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看训练结果。



BML Neural Network

BML Neural Network组件可以进行可视化的神经网络开发，搭建网络方式与可视化建模一致，同时需要编辑代码自定义算子的fit、save、 load、transform 方法。

输入

- 输入一个数据集，需要搭建网络结构并且编写算子的fit、save、 load、transform 方法。

输出

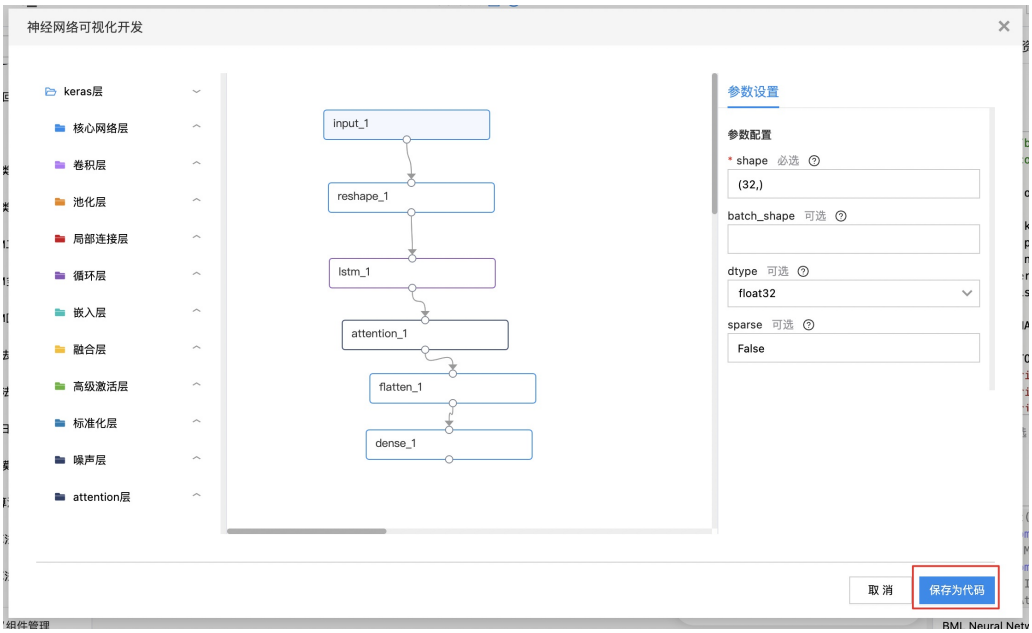
- 输出神经网络模型。

算子参数

参数名称	是否必选	参数描述	默认值
Python代码编辑窗口	是	使用Python开发神经网络	
网络定义	是	可在可视化神经网络编辑界面进行编辑	

使用示例

选中“BML Neural Network”组件后，点击右侧“可视化开发”按钮，进入神经网络可视化开发界面。神经网络可视化开发同可视化建模操作体验类似，选择相应的组件进行串联后，点击组件可进行参数设置。待全部设置完毕后，可点击“保存为代码”。



Catboost二分类(单机)

CatBoost是一种基于对称决策树（oblivious trees）为基学习器实现的参数较少、支持类别型变量和高准确性的GBDT框架，主要解决的痛点是高效合理地处理类别型特征，这一点从它的名字中可以看出来，CatBoost是由Categorical和Boosting组成。此外，CatBoost还解决了梯度偏差（Gradient Bias）以及预测偏移（Prediction shift）的问题，从而减少过拟合的发生，进而提高算法的准确性和泛化能力。

输入

- 输入一个数据集，特征列是数值或数值数组类型，或者类别类型（字符串类型），二分类任务标签列需要是整数或字符串类型，标签唯一值为2。

输出

- 输出Catboost二分类(单机)模型。

算子参数

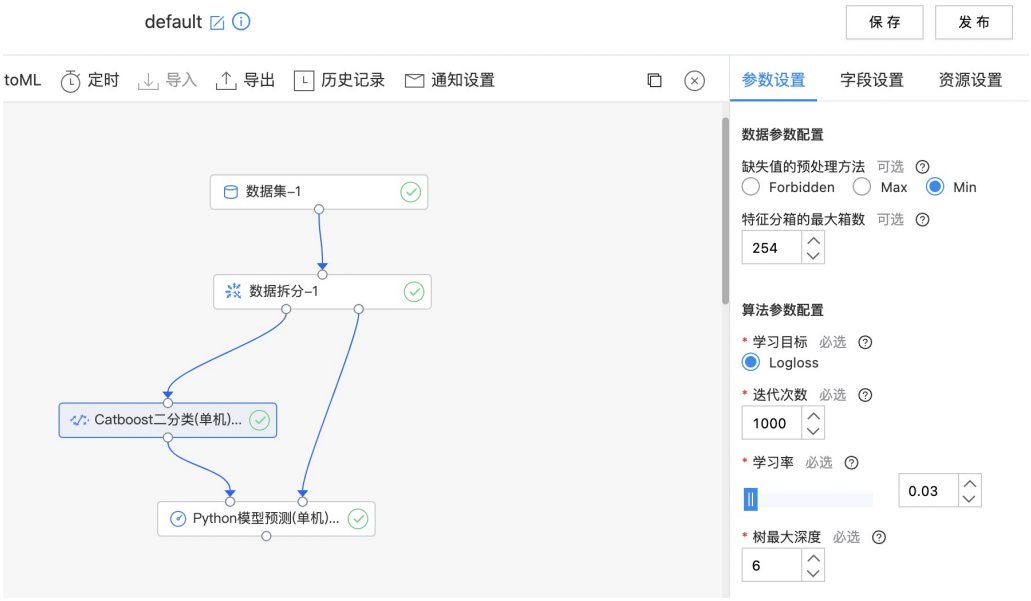
参数名称	是否必选	参数描述	默认值
缺失值的预处理方法	否	三种缺失值处理方法，Forbidden: 不支持缺失值、Min: 缺失值赋值为最小值、Max: 缺失值赋值为最大值，默认用最小值补缺失值，类别特征不支持空值和填充	Min
特征分箱的最大箱数	否	特征值分箱的最大箱数，较少的分箱数可能会降低训练的准确性，但可能会提高总体训练效果（处理过拟合） 范围：[1, 65535]	254
学习目标	是	二分类任务的学习目标	Logloss
迭代次数	是	提升迭代次数，也是可以建立的最大树数 范围：[1, inf)	1000
学习率	是	步长收缩率 范围：[0.01, 1.0]	0.03
树最大深度	是	一个树的最大深度，建议在4-10之间，如出现不明原因的失败，请尝试调小该值或调大内存 范围：[1, 16]。	6
L2正则化系数	是	代价函数的L2正则化项的系数，允许任何正值 范围：[0.0, inf)。	3
样本采样随机种子	否	样本采样随机种子 范围：[0, inf)。	0
采样类型	否	目前三种类型：Bayesian、Bernoulli、MVS。	MVS
特征列子采样比例	否	特征列子采样比例 范围：[0.01, 1.0]。	1.00
Bagging强度	否	Bagging强度，范围：[0.01, 1.0]。	1.00
提升模式	否	提升模式，目前两种类型：Ordered、Plain	Plain
计算叶子节点值的方法	否	计算叶子节点值的方法，目前支持Gradient和Newton。	Gradient
是否采用输入数据的顺序	否	是否采用输入数据的顺序	关闭
随机子空间	否	随机子空间	1.00
线程数	否	默认-1表示按CPU核数。要获得最佳速度，请将其设置为实际CPU内核数，而不是线程数（大多数CPU使用超线程为每个CPU内核生成2个线程）。如果数据集很小，请不要将其设置得太大 范围：[0, inf)	0

字段参数

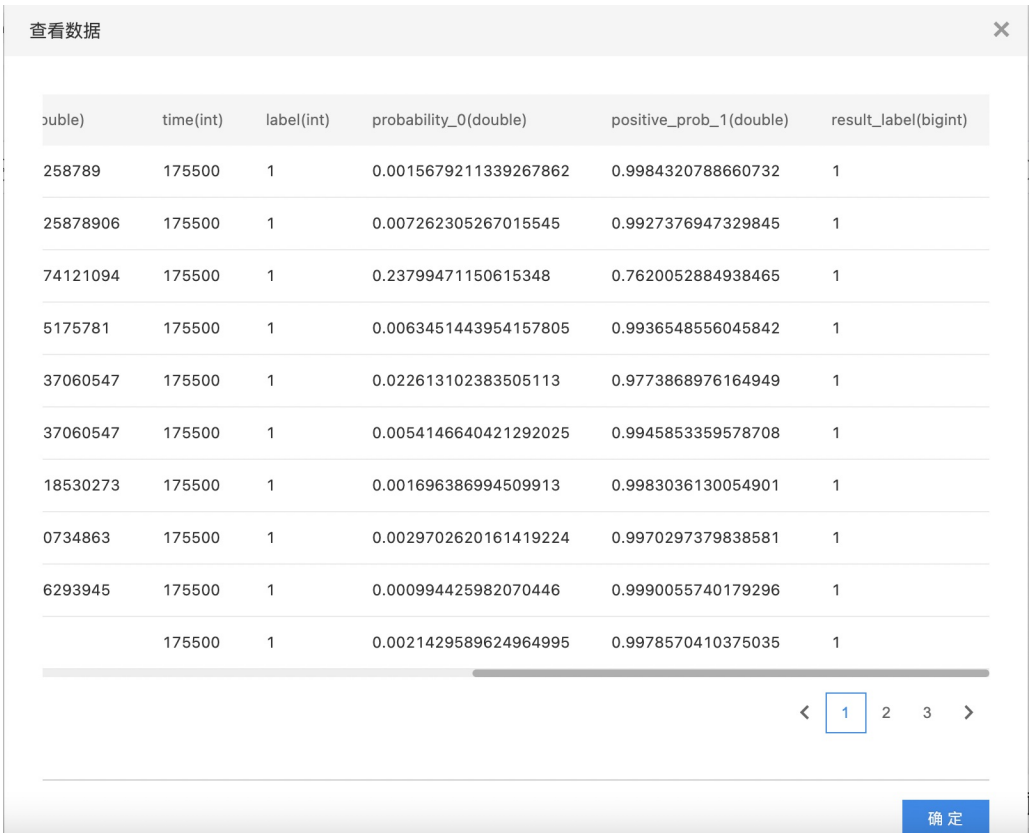
参数名称	是否必选	参数描述	默认值
特征列	是	训练和预测使用的特征列，可以是数值、数值数组、类别类型；当包含类别特征时，需要对类别特征进行标识，且类别特征不能包含空值。	无
标签列	是	预测目标列，要求是整数或字符串类型，唯一值是2。	无
正样本标签值	否	二分类的正样本的标签值，应在标签列中存在。	无
类别特征标示	否	当特征列(不包括Label列)包含【类别】特征时，需要将类别特征单独标示出来，仅支持int或者string类型，不支持空值，选择的列需要是特征列的子集。	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看训练结果。



Catboost多分类(单机)

Catboost多分类与二分类算法原理一致，运用sotfmax和ova(一对多)的方式实现多分类。

输入

- 输入一个数据集，特征列是数值或数值数组类型，或者类别类型（字符串类型），多分类任务标签列需要是整数或字符串类型，标签唯一值大于2。

输出

- 输出Catboost多分类(单机)模型，支持查看Top10分类的概率。

算子参数

参数名称	是否必选	参数描述	默认值
缺失值的预处理方法	否	三种缺失值处理方法，Forbidden: 不支持缺失值、Min: 缺失值赋值为最小值、Max: 缺失值赋值为最大值，默认用最小值补缺失值，类别特征不支持空值和填充	Min
特征分箱的最大箱数	否	特征值分箱的最大箱数，较少的分箱数可能会降低训练的准确性，但可能会提高总体训练效果（处理过拟合） 范围：[1, 65535]	254
学习目标	是	多分类任务的学习目标，如果出现内存溢出或不明原因的失败，请尝试调大内存或者减少树的深度。目前支持MultiClassoftmax)和MultiClassOneVsALL(ova)。	MultiClass(softmax)
迭代次数	是	提升迭代次数，也是可以建立的最大树数 范围：[1, inf)	1000
学习率	是	步长收缩率 范围：[0.01, 1.0]	0.03
树最大深度	是	一个树的最大深度，建议在4-10之间，如出现不明原因的失败，请尝试调小该值或调大内存 范围：[1, 16]。	6
L2正则化系数	是	代价函数的L2正则化项的系数，允许任何正值 范围：[0.0, inf)。	3
样本采样随机种子	否	样本采样随机种子 范围：[0, inf)。	0
采样类型	否	目前三种类型：Bayesian、Bernoulli、MVS。	MVS
特征列子采样比例	否	特征列子采样比例 范围：[0.01, 1.0]。	1.00
Bagging强度	否	Bagging强度，范围：[0.01, 1.0]。	1.00
提升模式	否	提升模式，目前两种类型：Ordered、Plain	Plain
计算叶子节点值的方法	否	计算叶子节点值的方法，目前支持Gradient和Newton。	Gradient
是否采用输入数据的顺序	否	是否采用输入数据的顺序	关闭
随机子空间	否	随机子空间	1.00
线程数	否	默认-1表示按CPU核数。要获得最佳速度，请将其设置为实际CPU内核数，而不是线程数（大多数CPU使用超线程为每个CPU内核生成2个线程）。如果数据集很小，请不要将其设置得太大 范围：[0, inf)	0

字段参数

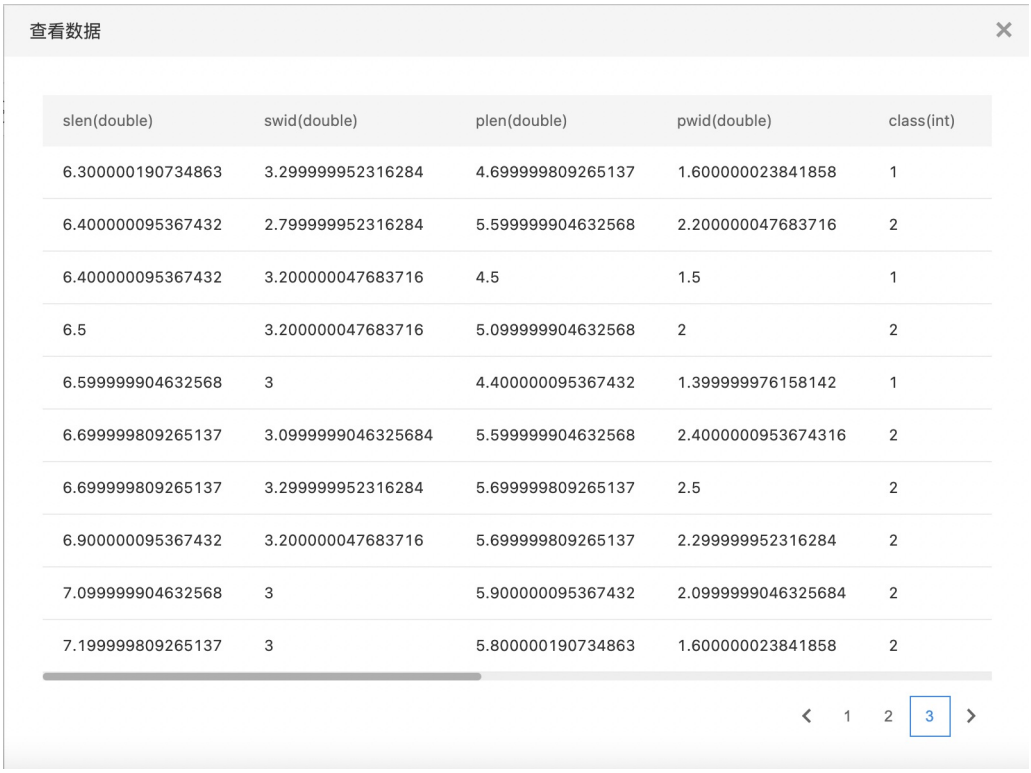
参数名称	是否必选	参数描述	默认值
特征列	是	训练和预测使用的特征列，可以是数值、数值数组、类别类型；当包含类别特征时，需要对类别特征进行标识，且类别特征不能包含空值。	无
标签列	是	预测目标列，要求是整数或字符串类型。	无
类别特征标示	否	当特征列(不包括Label列)包含【类别】特征时，需要将类别特征单独标示出来，仅支持int或者string类型，不支持空值，选择的列需要是特征列的子集。	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看训练结果。



slen(double)	swid(double)	plen(double)	pwid(double)	class(int)
6.300000190734863	3.299999952316284	4.699999809265137	1.600000023841858	1
6.400000095367432	2.799999952316284	5.599999904632568	2.200000047683716	2
6.400000095367432	3.200000047683716	4.5	1.5	1
6.5	3.200000047683716	5.099999904632568	2	2
6.599999904632568	3	4.400000095367432	1.399999976158142	1
6.699999809265137	3.0999999046325684	5.599999904632568	2.4000000953674316	2
6.699999809265137	3.299999952316284	5.699999809265137	2.5	2
6.900000095367432	3.200000047683716	5.699999809265137	2.299999952316284	2
7.099999904632568	3	5.900000095367432	2.0999999046325684	2
7.199999809265137	3	5.800000190734863	1.600000023841858	2

Catboost回归(单机)

Catboost支持回归任务，不同于分类任务，回归任务的学习目标使用RMSE、MAE、Poisson、Tweedie等。

输入

- 输入一个数据集，特征列是数值或数值数组类型，或者类别类型（字符串类型），标签列需要是数值类型，对于回归任务目标函数gamma，要求标签列是正的，对于回归任务目标函数tweedie，要求标签列是非负的。

输出

- 输出Catboost回归(单机)模型。

算子参数

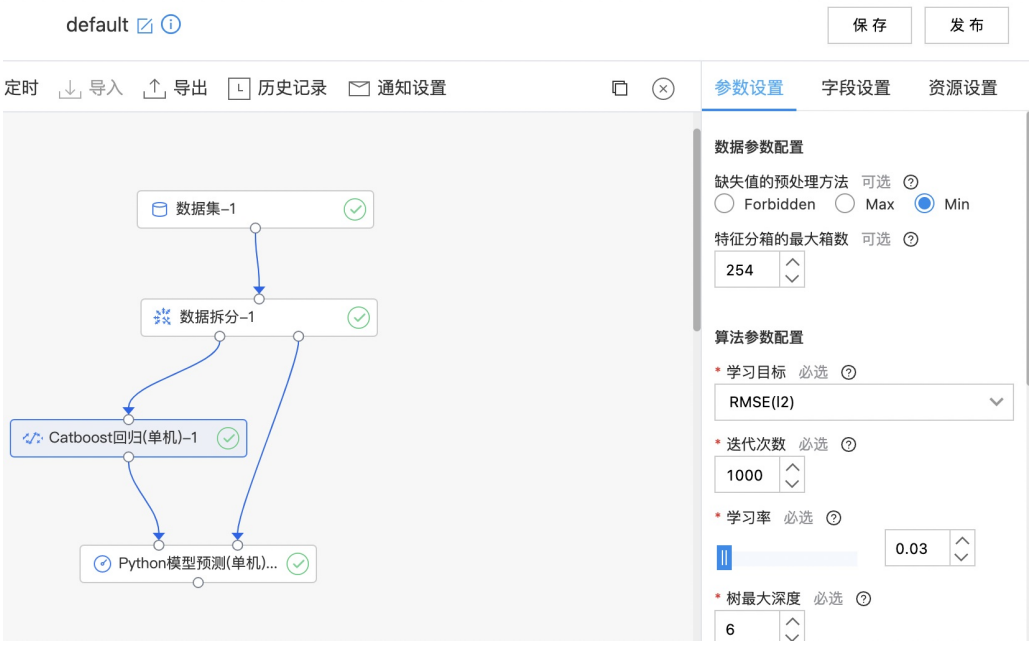
参数名称	是否必选	参数描述	默认值
缺失值的预处理方法	否	三种缺失值处理方法，Forbidden: 不支持缺失值、Min: 缺失值赋值为最小值、Max: 缺失值赋值为最大值，默认用最小值补缺失值，类别特征不支持空值和填充	Min
特征分箱的最大箱数	否	特征值分箱的最大箱数，较少的分箱数可能会降低训练的准确性，但可能会提高总体训练效果（处理过拟合） 范围：[1, 65535]	254
学习目标	是	回归任务的学习目标： RMSE(l2) MAE(l1) Poisson Tweedie	RMSE(l2)
迭代次数	是	提升迭代次数，也是可以建立的最大树数 范围：[1, inf)	1000
学习率	是	步长收缩率 范围：[0.01, 1.0]	0.03
树最大深度	是	一个树的最大深度，建议在4-10之间，如出现不明原因的失败，请尝试调小该值或调大内存 范围：[1, 16]。	6
L2正则化系数	是	代价函数的L2正则化项的系数，允许任何正值 范围：[0.0, inf)。	3
样本采样随机种子	否	样本采样随机种子 范围：[0, inf)。	0
采样类型	否	目前三种类型：Bayesian、Bernoulli、MVS。	MVS
特征列子采样比例	否	特征列子采样比例 范围：[0.01, 1.0]。	1.00
Bagging强度	否	Bagging强度，范围：[0.01, 1.0]。	1.00
提升模式	否	提升模式，目前两种类型：Ordered、Plain	Plain
计算叶子节点值的方法	否	计算叶子节点值的方法，目前支持Gradient和Newton。	Gradient
是否采用输入数据的顺序	否	是否采用输入数据的顺序	关闭
随机子空间	否	随机子空间	1.00
线程数	否	默认-1表示按CPU核数。要获得最佳速度，请将其设置为实际CPU内核数，而不是线程数（大多数CPU使用超线程为每个CPU内核生成2个线程）。如果数据集很小，请不要将其设置得太大 范围：[0, inf)	0

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	训练和预测使用的特征列，可以是数值、数值数组、类别类型；当包含类别特征时，需要对类别特征进行标识，且类别特征不能包含空值。	无
标签列	是	预测目标列，要求是数值类型。	无
类别特征标示	否	当特征列(不包括Label列)包含【类别】特征时，需要将类别特征单独标示出来，仅支持int或者string类型，不支持空值，选择的列需要是特征列的子集。	无

使用示例

构建算子结构，配置参数，完成训练。



DBSCAN

DBSCAN（Density-Based Spatial Clustering of Applications with Noise，具有噪声的基于密度的聚类方法）是一种很典型的密度聚类算法，和 K-Means，BIRCH 这些一般只适用于凸样本集的聚类相比，DBSCAN 既可以适用于凸样本集，也可以适用于非凸样本集。这类密度聚类算法一般假定类别可以通过样本分布的紧密程度决定。同一类别的样本，他们之间的紧密相连的，也就是说，在该类别任意样本周围不远处一定有同类别的样本存在。通过将紧密相连的样本划为一类，这样就得到了一个聚类类别。通过将所有各组紧密相连的样本划为各个不同的类别，则我们就得到了最终的所有聚类类别结果。

输入

- 输入一个数据集，数据集的特征列必须是数值或数值数组类型。

输出

- 输出DBSCAN模型。

算子参数

参数名称	是否必选	参数描述	默认值
邻域距离阈值	是	一个样本在另一个样本的邻域内的最大距离。这不是簇中点之间距离的最大界限。这是最重要的DBSCAN参数，可以根据数据集和距离函数进行适当选择。 范围：[0.01, inf)	0.5
邻域样本数阈值	是	将某个点视为核心点的邻域中的样本数（包括点本身）。 范围：[2, inf)	5
距离度量	是	提供两种距离度量方式：欧式距离（euclidean）和曼哈顿距离（manhattan）。	欧式距离

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	支持数值或数值数组类型。	无

计算逻辑

欧式距离也称欧几里得距离，是最常见的距离度量，衡量的是多维空间中两个点之间的绝对距离。

$$dist(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

曼哈顿距离中的距离计算公式比欧氏距离的计算公式看起来简洁很多，只需要把两个点坐标的 x 坐标相减取绝对值，y 坐标相减取绝对值，再加和。

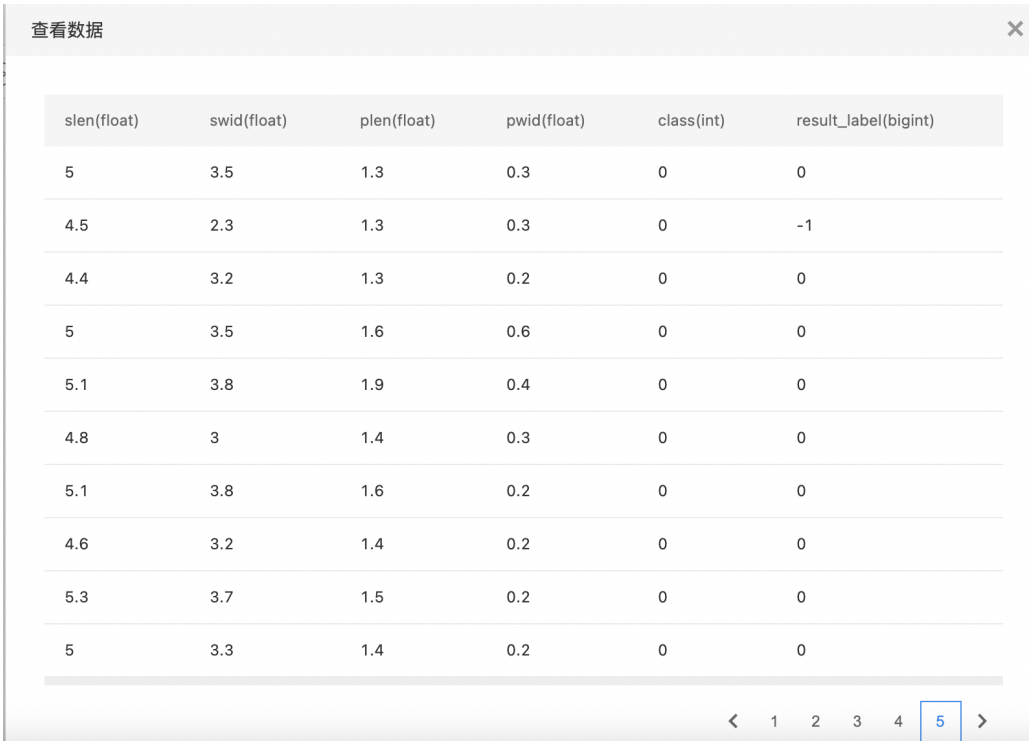
$$dist(X,Y) = \sum_{i=1}^{n}|x_i - y_i|$$

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看输出结果。



slen(float)	swid(float)	plen(float)	pwid(float)	class(int)	result_label(bigint)
5	3.5	1.3	0.3	0	0
4.5	2.3	1.3	0.3	0	-1
4.4	3.2	1.3	0.2	0	0
5	3.5	1.6	0.6	0	0
5.1	3.8	1.9	0.4	0	0
4.8	3	1.4	0.3	0	0
5.1	3.8	1.6	0.2	0	0
4.6	3.2	1.4	0.2	0	0
5.3	3.7	1.5	0.2	0	0
5	3.3	1.4	0.2	0	0

1 2 3 4 5

KNN二分类

k 近邻（KNN）是一种常用的监督学习方法，其工作机制非常简单：给定测试样本，基于某种距离度量找出训练集中与其最靠近的 k 个训练样本，然后基于这 k 个“邻居”的信息来进行预测。通常，在分类任务中可使用“投票法”，即选择这 k 个样本中出现最多的类别标记作为预测结果。 k 近邻没有显式的学习过程。事实上，它是懒惰学习（lazy learning）的著名代表，此类学习技术在训练阶段仅仅是把样本保存起来，训练时间开销为零，待收到测试样本后再进行处理。 KNN 二分类：利用 k 近邻学习方法完成二分类任务。

输入

- 输入一个数据集，数据集的特征列必须是数值或数值数组类型，标签列必须是整数类型或字符串类型（如果列中的unique值超过了两种，运行算子时会报错）。

输出

- 输出KNN二分类模型。

算子参数

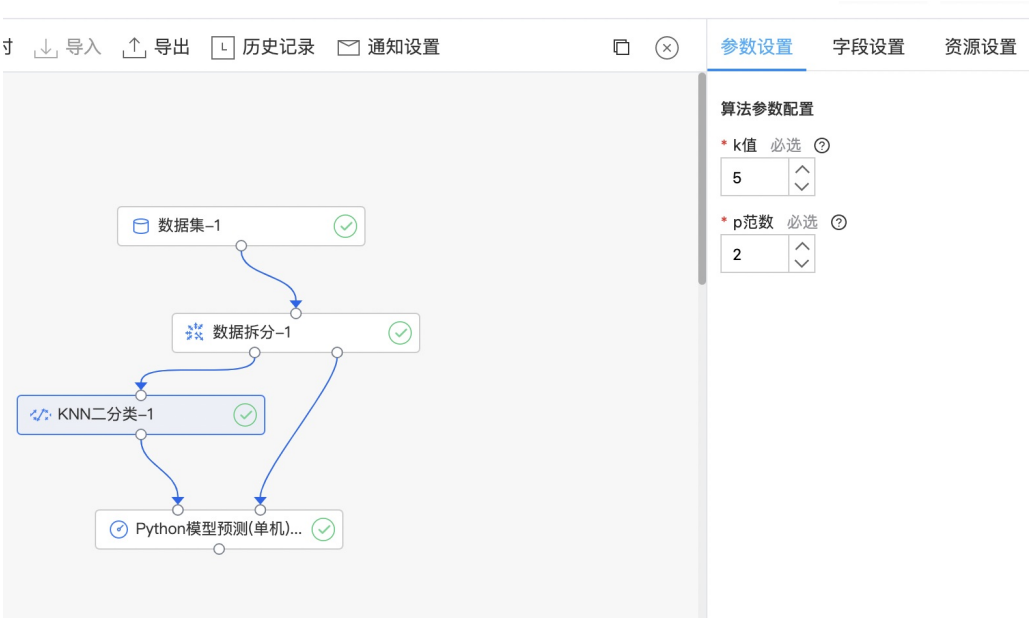
参数名称	是否必选	参数描述	默认值
k值	是	k值选择，要求不大于样本个数 范围：[1, 100]	5
p范数	是	Lp距离度量，常用p=1,2 范围：[1, 10]	2

字段参数

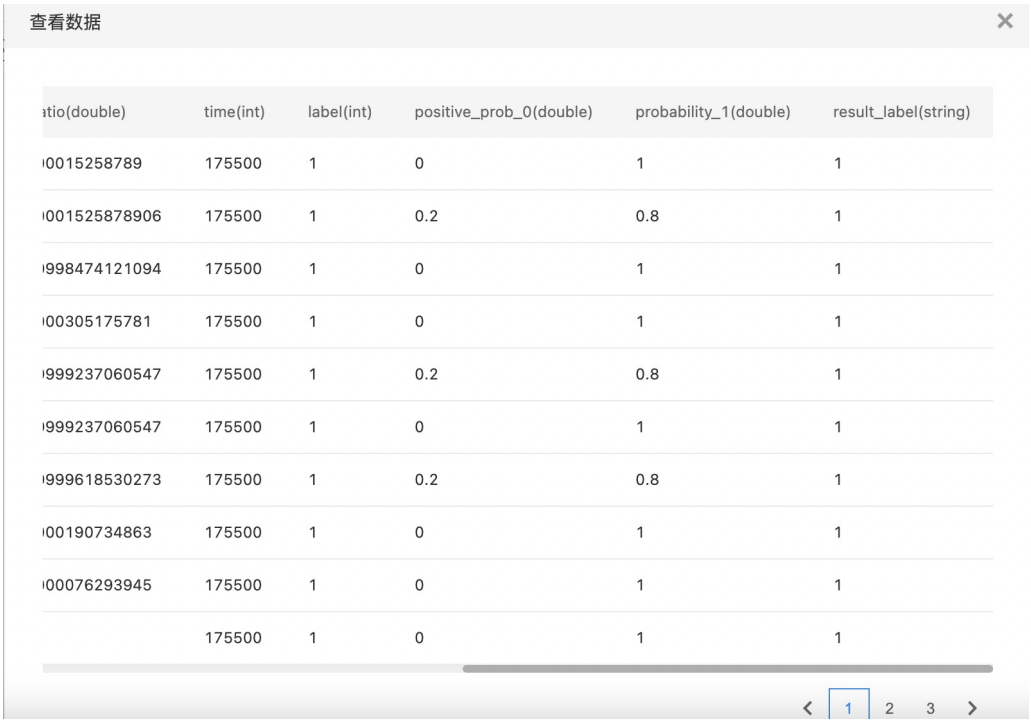
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求是整数或字符串类型，唯一值是2。	无
正样本标签值	否	二分类的正样本的标签值，应在标签列中存在。	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看训练结果。



KNN多分类

KNN多分类任务即已分类样本中含两个以上的类别，待分类样本计算距离最近的K个样本，根据"投票法"确定样本分类。

输入

- 输入一个数据集，数据集的特征列必须是数值或数值数组类型，标签列必须是整数类型或字符串类型（如果列中的unique值不足三种，运行算子时会报错）。

输出

- 输出KNN多分类模型，支持查看Top10分类的概率。

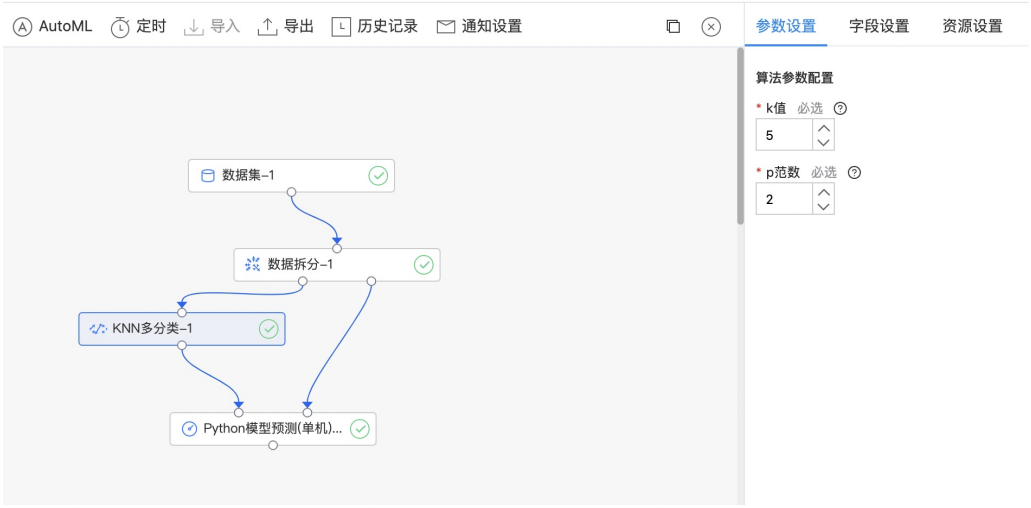
算子参数

参数名称	是否必选	参数描述	默认值
k值	是	k值选择，要求不大于样本个数 范围：[1, 100]	5
p范数	是	Lp距离度量，常用p=1,2 范围：[1, 10]	2

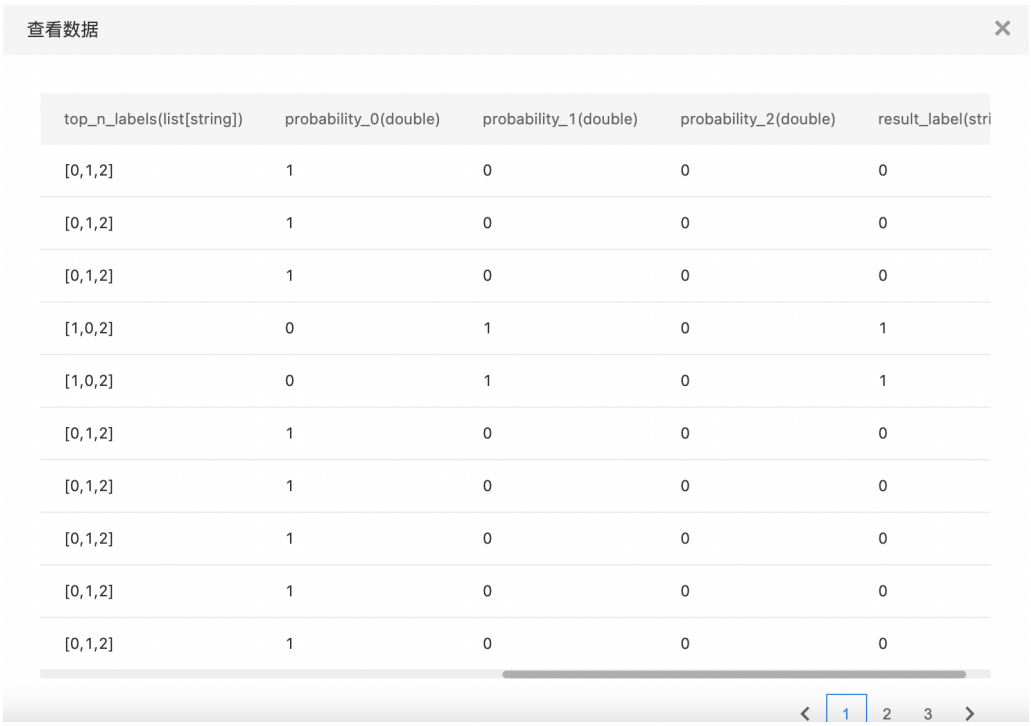
字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求是整数或字符串类型。	无

1. 构建算子结构，配置参数，完成训练。



2. 查看训练结果。



LightGBM二分类(单机)

提升树是利用加模型与前向分布算法实现学习的优化过程，它有一些高效实现，如XGBoost，GBDT等。其中GBDT采用负梯度作为划分的指标

（信息增益），XGBoost则利用到二阶导数。他们共同的不足是，计算信息增益需要扫描所有样本，从而找到最优划分点。在面对大量数据或者特征维度很高时，他们的效率和扩展性很难使人满意。微软开源的LightGBM（基于GBDT的）则很好的解决这些问题，它主要包含两个算法：

GOSS（从减少样本角度）：排除大部分小梯度的样本，仅用剩下的样本计算信息增益。

EFB（从减少特征角度）：捆绑互斥特征，也就是他们很少同时取非零值（也就是用一个合成特征代替）。

输入

- 输入一个数据集，特征列需要是数值或数值数组类型，标签列需要是整数或字符串类型，标签唯一值为2。

输出

- 输出LightGBM二分类(单机)模型。

算子参数

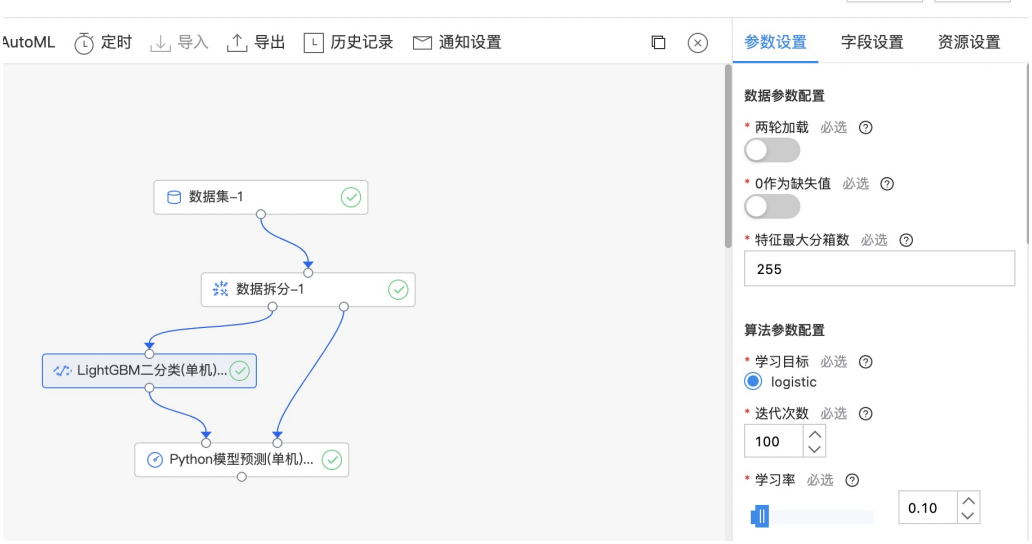
参数名称	是否必选	参数描述	默认值
两轮加载	是	如果数据文件太大而无法放入内存，可以设置两轮加载。	关闭
0作为缺失值	是	将加载数据中的所有0值作为缺失值，否则np.nan表示缺失值。	关闭
特征最大分箱数	是	特征值分箱的最大箱数。较少的分箱数可能会降低训练的准确性，但可能会提高总体训练效果（处理过拟合） 范围：[2, inf)。	255
学习目标	是	二分类任务的学习目标。	logistic
迭代次数	是	提升迭代次数 范围：[1, 200]。	100
学习率	是	步长收缩率 范围：[0.01, 1.0]。	0.10
树最大叶子数	是	每棵树的最大叶子数 范围：[2, 131072]。	31
划分最小信息增益	是	划分树（叶子）节点所需的最小信息增益 范围：[0.0, inf)	0.0
叶子节点最小样本数	是	每个叶子节点的最小样本数。可用于处理过拟合 范围：[0, inf)。	20
叶子节点最小样本权重和	是	每个叶子节点的最小样本权重和。可用于处理过拟合 范围：[0.0, inf)。	0.001
样本采样频率	是	0表示禁用样本采样；k表示每k次迭代执行一次样本采样。要启用样本采样，[样本采样比例]的值也应设置为小于1.0 范围：[0, inf)。	0
样本采样比例	是	随机选择部分样本。可用于加快训练。可用于处理过拟合 范围：[0.01, 1.0]。	1.00
正样本采样比例	是	用于不平衡的二分类问题，将随机采样[正样本数*正样本采样比例]个正样本。应该与[负样本采样比例]一起使用。将此设置为1.0以禁用。如果启用了平衡采样（值设置为小于1.0），则将忽略[样本采样比例] 范围：[0.01, 1.0]	1.00
负样本采样比例	是	用于不平衡的二分类问题，将随机采样[负样本数*负样本采样比例]个负样本。应该与[正样本采样比例]一起使用。将此设置为1.0以禁用。如果启用了平衡采样（值设置为小于1.0），则将忽略[样本采样比例] 范围：[0.01, 1.0]	1.00
样本采样随机种子	是	样本采样随机种子	3
树特征采样比例	是	如果值小于1.0，将在每次迭代（树）上随机选择部分特征。可用于加快训练。可用于处理过拟合 范围：[0.01, 1.0]。	1.00
树节点特征采样比例	是	如果值小于1.0，将在每个树节点上随机选择部分特征。可用于处理过拟合。与树特征采样比例不同，这无法加快训练速度。每个树节点最终的特征采样比例为[树特征采样比例*树节点特征采样比例] 范围：[0.01, 1.0]	1.00
特征采样随机种子	是	特征采样随机种子	2
L1正则化系数	是	权重的L1正则化项 范围：[0.0, inf)。	0.0
L2正则化系数	是	权重的L2正则化项 范围：[0.0, inf)。	0.0
提升算法	是	目前支持两种提升算法：gbdt（梯度提升决策树）、rf（随机森林）；如果选择rf，需要设置「样本采样频率」大于0、「样本采样比例」大于0且小于1，这是rf算法本身隐含约束。	gbdt
线程数	是	0表示OpenMP中的默认线程数。要获得最佳速度，请将其设置为实际CPU内核数，而不是线程数（大多数CPU使用超线程为每个CPU内核生成2个线程）。如果数据集很小，请不要将其设置得太大 范围：[0, 16]	0

字段参数

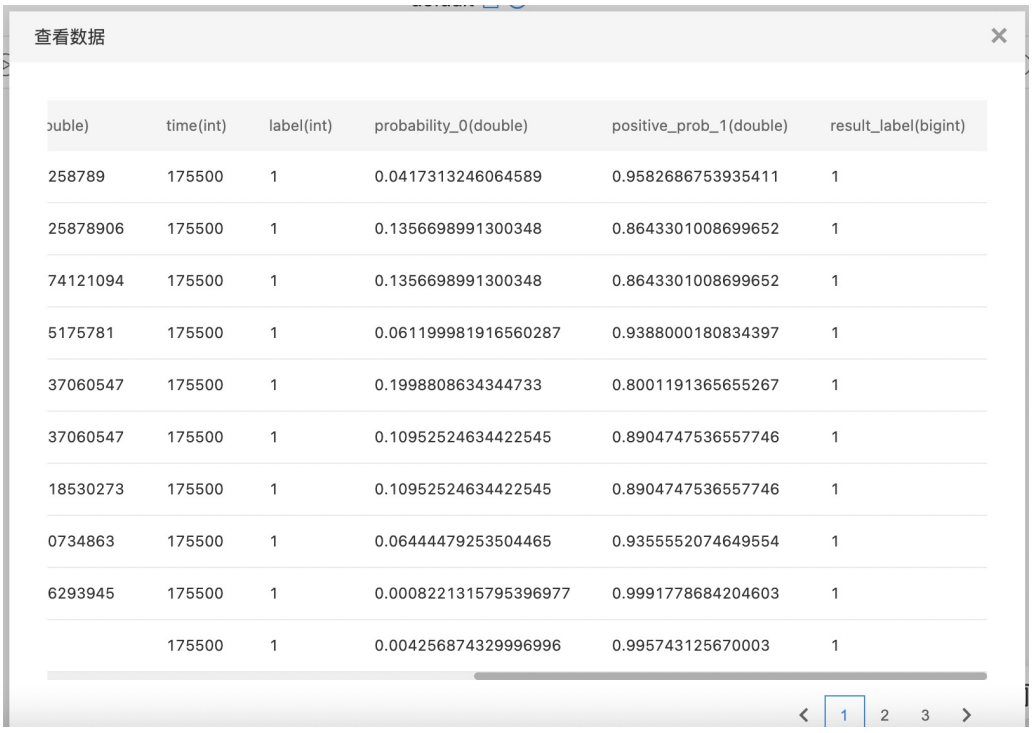
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求是整数或字符串类型，唯一值是2。	无
正样本标签值	否	二分类的正样本的标签值，应在标签列中存在。	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看训练结果。



LightGBM多分类(单机)

LightGBM多分类与二分类算法原理一致，运用sotfmax和ova(一对多)的方式实现多分类。

输入

- 输入一个数据集，特征列需要是数值或数值数组类型，标签列需要是整数或字符串类型，标签唯一值大于2。

输出

- 输出LightGBM多分类(单机)模型，支持查看Top10分类的概率。

算子参数

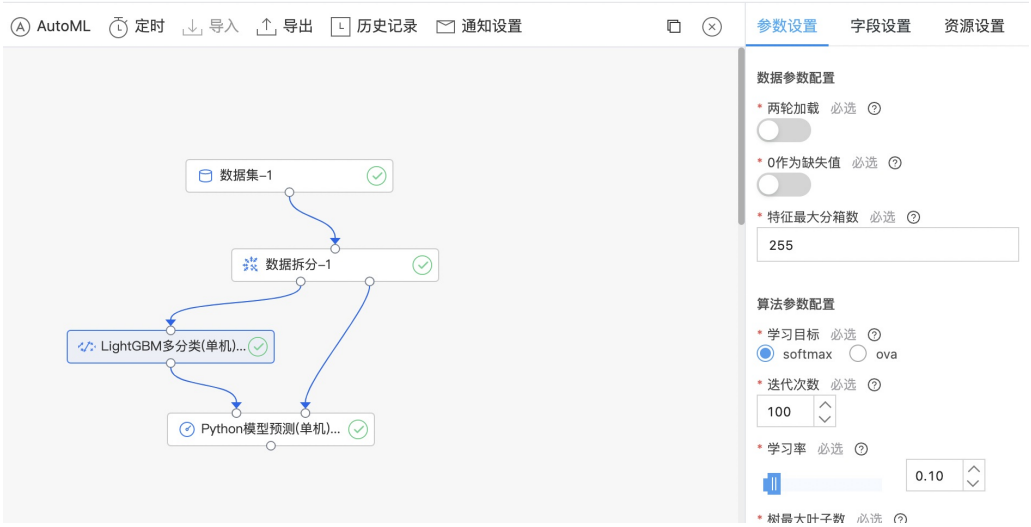
参数名称	是否必选	参数描述	默认值
两轮加载	是	如果数据文件太大而无法放入内存，可以设置两轮加载。	关闭
0作为缺失值	是	将加载数据中的所有0值作为缺失值，否则np.nan表示缺失值。	关闭
特征最大分箱数	是	特征值分箱的最大箱数。较少的分箱数可能会降低训练的准确性，但可能会提高总体训练效果（处理过拟合） 范围：[2, inf)。	255
学习目标	是	多分类任务的学习目标，目前支持：softmax和ova。	softmax
迭代次数	是	提升迭代次数 范围：[1, 200]。	100
学习率	是	步长收缩率 范围：[0.01, 1.0]。	0.10
树最大叶子数	是	每棵树的最大叶子数 范围：[2, 131072]。	31
划分最小信息增益	是	划分树（叶子）节点所需的最小信息增益 范围：[0.0, inf)	0.0
叶子节点最小样本数	是	每个叶子节点的最小样本数。可用于处理过拟合 范围：[0, inf)。	20
叶子节点最小样本权重和	是	每个叶子节点的最小样本权重和。可用于处理过拟合 范围：[0.0, inf)。	0.001
样本采样频率	是	0表示禁用样本采样；k表示每k次迭代执行一次样本采样。要启用样本采样，[样本采样比例]的值也应设置为小于1.0 范围：[0, inf)。	0
样本采样比例	是	随机选择部分样本。可用于加快训练。可用于处理过拟合 范围：[0.01, 1.0]。	1.00
样本采样随机种子	是	样本采样随机种子	3
树特征采样比例	是	如果值小于1.0，将在每次迭代（树）上随机选择部分特征。可用于加快训练。可用于处理过拟合 范围：[0.01, 1.0]。	1.00
树节点特征采样比例	是	如果值小于1.0，将在每个树节点上随机选择部分特征。可用于处理过拟合。与树特征采样比例不同，这无法加快训练速度。每个树节点最终的特征采样比例为[树特征采样比例*树节点特征采样比例] 范围：[0.01, 1.0]	1.00
特征采样随机种子	是	特征采样随机种子	2
L1正则化系数	是	权重的L1正则化项 范围：[0.0, inf)。	0.0
L2正则化系数	是	权重的L2正则化项 范围：[0.0, inf)。	0.0
提升算法	是	目前支持两种提升算法：gbdt（梯度提升决策树）、rf（随机森林）；如果选择rf，需要设置「样本采样频率」大于0、「样本采样比例」大于0且小于1，这是rf算法本身隐含约束。	gbdt
线程数	是	0表示OpenMP中的默认线程数。要获得最佳速度，请将其设置为实际CPU内核数，而不是线程数（大多数CPU使用超线程为每个CPU内核生成2个线程）。如果数据集很小，请不要将其设置得太大 范围：[0, 16]	0

字段参数

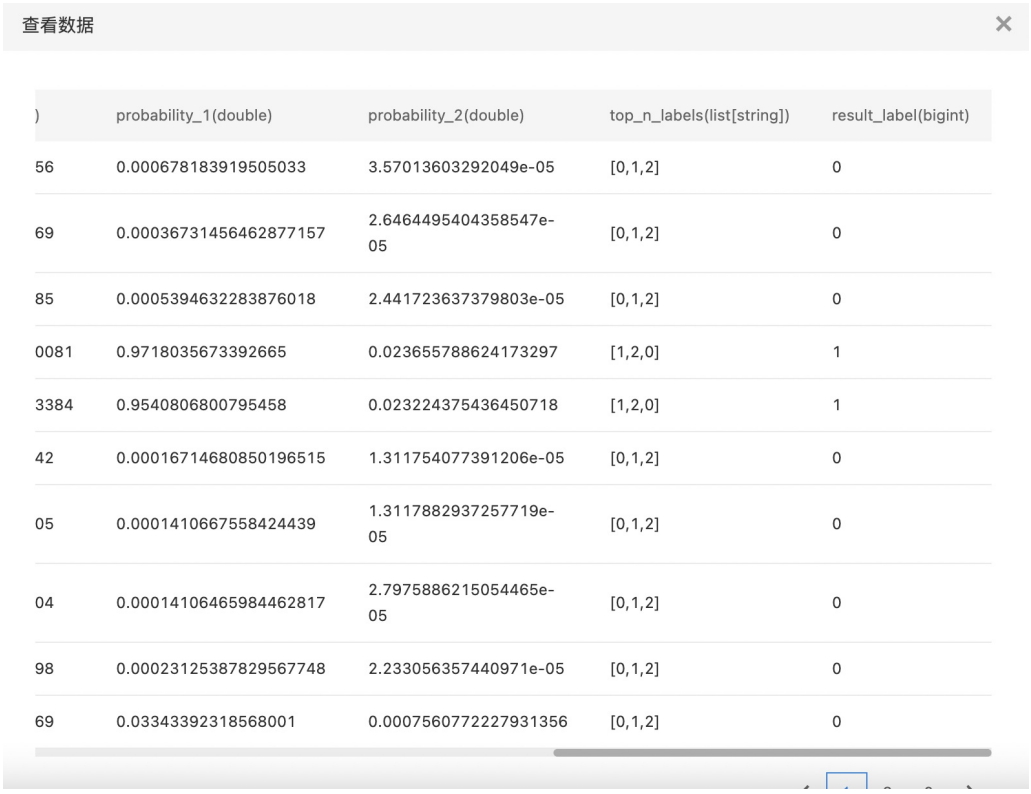
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求是整数或字符串类型。	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看训练结果。



)	probability_1(double)	probability_2(double)	top_n_labels(list[string])	result_label(bigint)
56	0.000678183919505033	3.57013603292049e-05	[0,1,2]	0
69	0.00036731456462877157	2.6464495404358547e-05	[0,1,2]	0
85	0.0005394632283876018	2.441723637379803e-05	[0,1,2]	0
0081	0.9718035673392665	0.023655788624173297	[1,2,0]	1
3384	0.9540806800795458	0.023224375436450718	[1,2,0]	1
42	0.00016714680850196515	1.311754077391206e-05	[0,1,2]	0
05	0.0001410667558424439	1.3117882937257719e-05	[0,1,2]	0
04	0.00014106465984462817	2.7975886215054465e-05	[0,1,2]	0
98	0.00023125387829567748	2.233056357440971e-05	[0,1,2]	0
69	0.03343392318568001	0.0007560772227931356	[0,1,2]	0

LightGBM回归(单机)

LightGBM支持回归任务，不同于分类任务，回归任务的学习目标使用gamma、tweedie、l1、l2等。

输入

- 输入一个数据集，特征列需要是数值或数值数组类型，回归任务标签列需要是数值类型，对于回归任务目标函数gamma，要求标签列是正的，对于回归任务目标函数tweedie，要求标签列是非负的。

输出

- 输出LightGBM回归(单机)模型。

算子参数

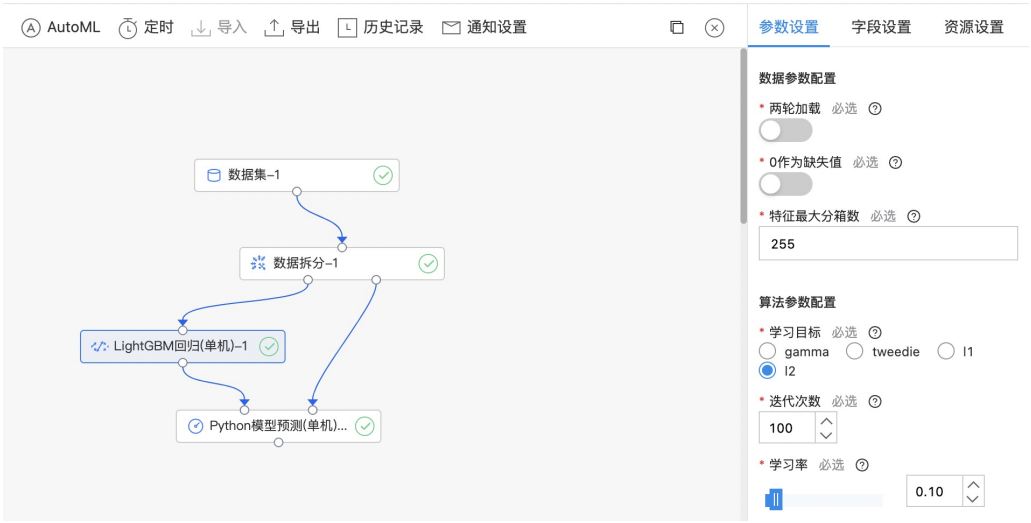
参数名称	是否必选	参数描述	默认值
两轮加载	是	如果数据文件太大而无法放入内存，可以设置两轮加载。	关闭
0作为缺失值	是	将加载数据中的所有0值作为缺失值，否则np.nan表示缺失值。	关闭
特征最大分箱数	是	特征值分箱的最大箱数。较少的分箱数可能会降低训练的准确性，但可能会提高总体训练效果（处理过拟合） 范围：[2, inf)。	255
学习目标	是	回归任务的学习目标，目前支持： gamma tweedie l1 l2	L2
迭代次数	是	提升迭代次数 范围：[1, 200]。	100
学习率	是	步长收缩率 范围：[0.01, 1.0]。	0.10
树最大叶子数	是	每棵树的最大叶子数 范围：[2, 131072]。	31
划分最小信息增益	是	划分树（叶子）节点所需的最小信息增益 范围：[0.0, inf)	0.0
叶子节点最小样本数	是	每个叶子节点的最小样本数。可用于处理过拟合 范围：[0, inf)。	20
叶子节点最小样本权重和	是	每个叶子节点的最小样本权重和。可用于处理过拟合 范围：[0.0, inf)。	0.001
样本采样频率	是	0表示禁用样本采样；k表示每k次迭代执行一次样本采样。要启用样本采样，[样本采样比例]的值也应设置为小于1.0 范围：[0, inf)。	0
样本采样比例	是	随机选择部分样本。可用于加快训练。可用于处理过拟合 范围：[0.01, 1.0]。	1.00
样本采样随机种子	是	样本采样随机种子	3
树特征采样比例	是	如果值小于1.0，将在每次迭代（树）上随机选择部分特征。可用于加快训练。可用于处理过拟合 范围：[0.01, 1.0]。	1.00
树节点特征采样比例	是	如果值小于1.0，将在每个树节点上随机选择部分特征。可用于处理过拟合。与树特征采样比例不同，这无法加快训练速度。每个树节点最终的特征采样比例为[树特征采样比例*树节点特征采样比例] 范围：[0.01, 1.0]	1.00
特征采样随机种子	是	特征采样随机种子	2
L1正则化系数	是	权重的L1正则化项 范围：[0.0, inf)。	0.0
L2正则化系数	是	权重的L2正则化项 范围：[0.0, inf)。	0.0
提升算法	是	目前支持两种提升算法：gbdt（梯度提升决策树）、rf（随机森林）；如果选择rf，需要设置「样本采样频率」大于0、「样本采样比例」大于0且小于1，这是rf算法本身隐含约束。	gbdt
线程数	是	0表示OpenMP中的默认线程数。要获得最佳速度，请将其设置为实际CPU内核数，而不是线程数（大多数CPU使用超线程为每个CPU内核生成2个线程）。如果数据集很小，请不要将其设置得太大 范围：[0, 16]	0

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求是数值类型。	无

使用示例

构建算子结构，配置参数，完成训练。



MxNet算法

MxNet 是一个深度学习框架，旨在提高效率和灵活性，允许混合符号和命令式编程，以最大限度地提高效率和生产力。 MxNet 算法组件：利用MxNet库编写自定义python代码，完成各类深度学习任务。该组件仅提供单机运行的模式。

输入

- 输入一个数据集作为训练集，编写Python代码。

输出

- 输出MxNet算法模型。

算子参数

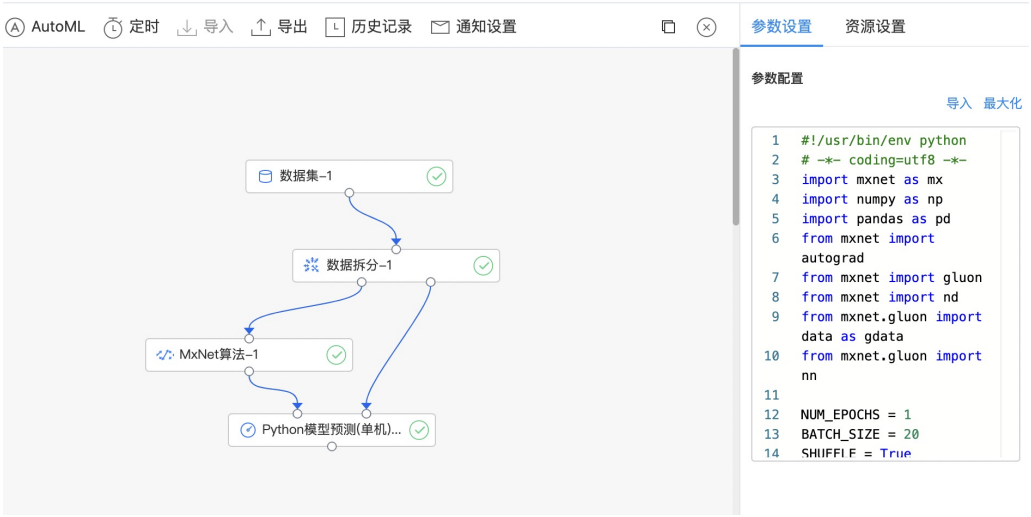
参数名称	是否必选	参数描述	默认值
Python 代码编辑窗口	是	输入Python 代码	

函数说明

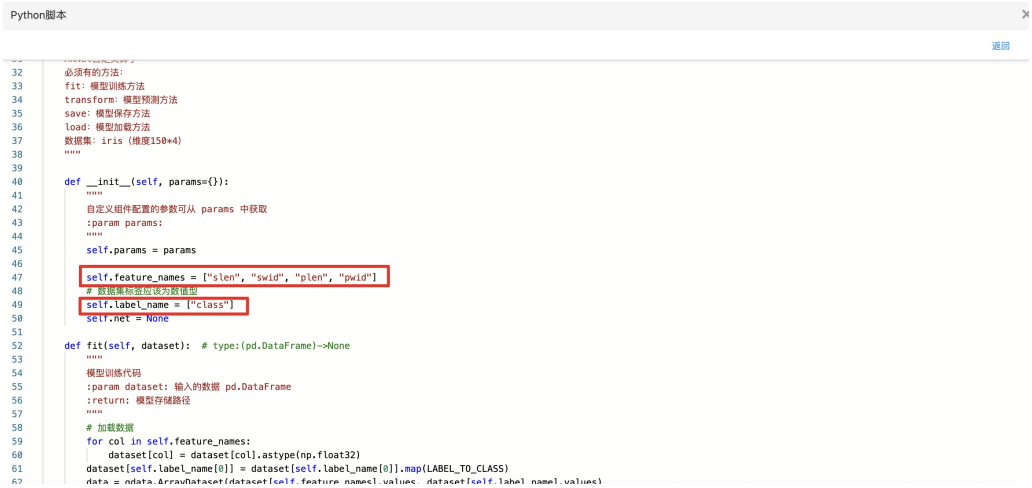
```
 fit：模型训练方法，有且仅有dataset参数
transform：模型预测方法，有且仅有dataset参数，返回的预测结果为pandas.DataFrame
save：模型保存方法,有且仅有model_path参数
load：模型加载方法,有且仅有model_path参数
```

使用示例

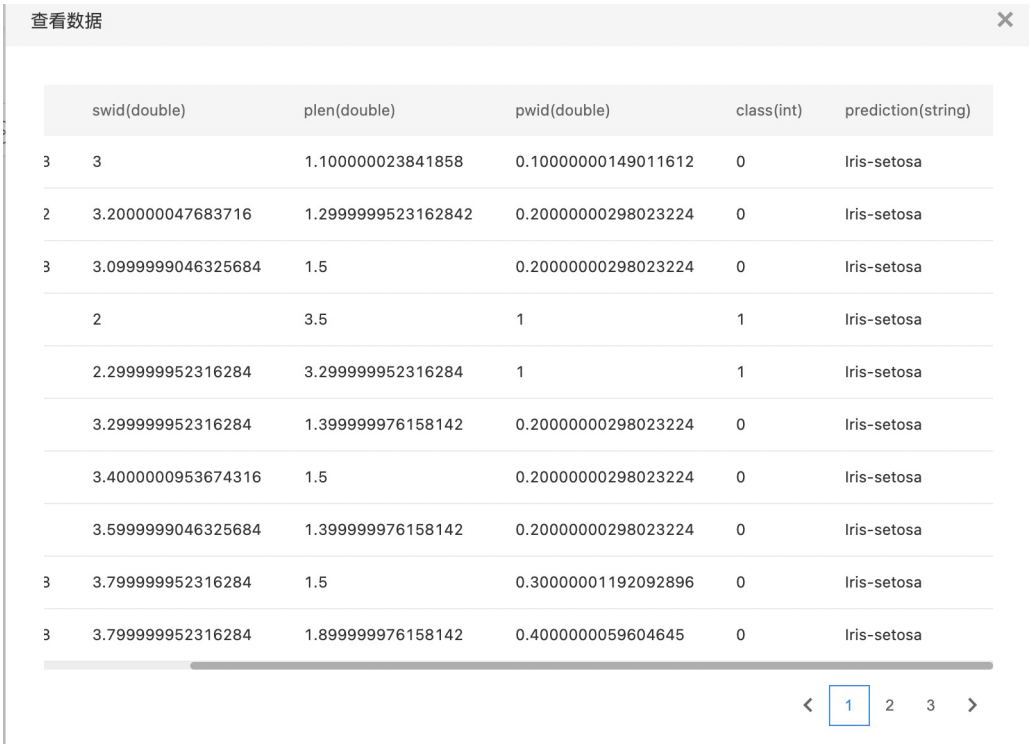
1. 构建算子结构，编写python代码，完成训练。



2. 查看代码，本例中特征列为slen、swid、plen、pwid，标签列为class。



3. 查看训练结果。



Paddle算法

PaddlePaddle（PArallel Distributed Deep LEarning）是一个易用、高效、灵活、可扩展的深度学习框架。 Paddle 算法组件：利用 PaddlePaddle 库写自定义 python 代码，完成各类深度学习任务。

输入

- 输入训练数据集：paddle.reader，每行数据为pandas.core.series.Series格式数据；预测dataset：pandas.DataFrame,每行数据为pandas.core.series.Series格式数据。

输出

- 输出Paddle算法模型。

算子参数

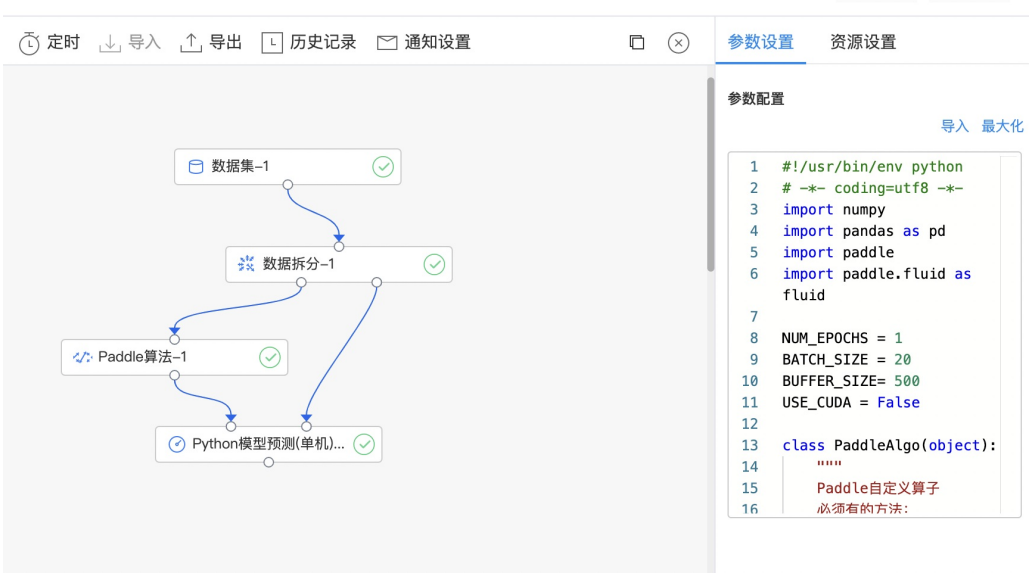
参数名称	是否必选	参数描述	默认值
Python 代码编辑窗口	是	输入Python 代码	

函数说明

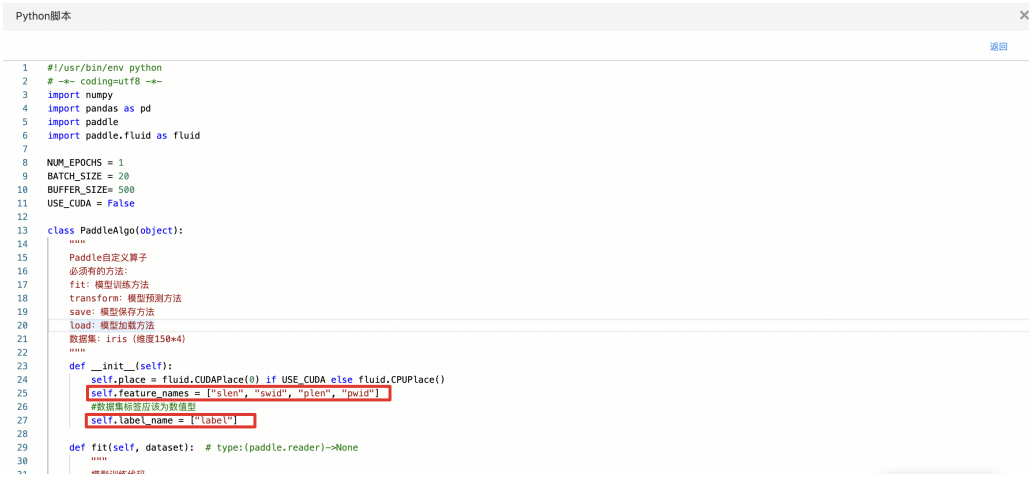
fit：模型训练方法，有且仅有dataset参数
transform：模型预测方法，有且仅有dataset参数，返回的预测结果为pandas.DataFrame
save：模型保存方法,有且仅有model_path参数
load：模型加载方法,有且仅有model_path参数

使用示例

1. 构建算子结构，编写python代码，完成训练。



2. 查看代码，本例中特征列为slen、swid、plen、pwid，标签列为label。



多项式回归

多项式回归，回归函数是回归变量多项式的回归。多项式回归模型是线性回归模型的一种，此时回归函数关于回归系数是线性的。由于任一函数都可以用多项式逼近，因此多项式回归有着广泛应用。 多项式回归模型通常使用最小二乘法来拟合。 实现原理：多项式扩展 + 线性回归。

输入

- 输入一个数据集，数据集的特征列double/int，标记列是Double或Int类型。

输出

- 输出多项式回归模型。

算子参数

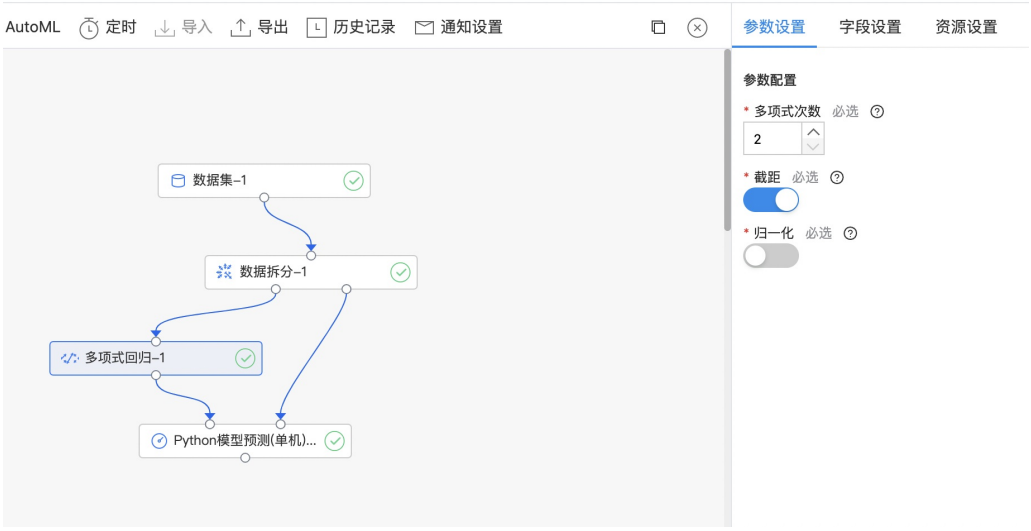
参数名称	是否必选	参数描述	默认值
多项式次数	是	多项式回归中选择的多项式次数 范围：[2, 8]。	2
截距	是	多项式回归中是否使用截距。	开启
归一化	是	多项式回归中是否使用归一化。	关闭

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求是数值类型。	无

使用示例

构建算子结构，配置参数，完成训练。



PyModel模型脚本

PyModel支持将上传的代码文件存放到用户填写的目录里面，同时还支持在算子里面用存储里的代码和数据集训练出模型，放到用户填写的目录里。

输入

- 输入选择存储中的文件，需要填写PyModel模型文件的存放路径。

输出

- 输出PyModel格式的模型。

算子参数

参数名称	是否必选	参数描述	默认值
存储名称	是	选择存储名称	无
资源文件	是	选择上传的资源文件	无
shell命令	是	算子执行的时候，会运行用户输入的shell命令，预期这个shell命令会在用户填写的模型目录产出一个PyModel格式的模型。	sh man.sh
模型目录(相对路径)，模型格式为PyModel	是	模型目录(相对路径)，模型格式为 PyModel	./model_dir

使用示例

1. 上传的示例文件如下，命名为model.py

```
 ###### !/usr/bin/env python
###### -*- coding=utf8 -*-
"""
python script
"""
import pandas as pd


class PyModel(object):
    """
    运行环境为 Python 3
    """

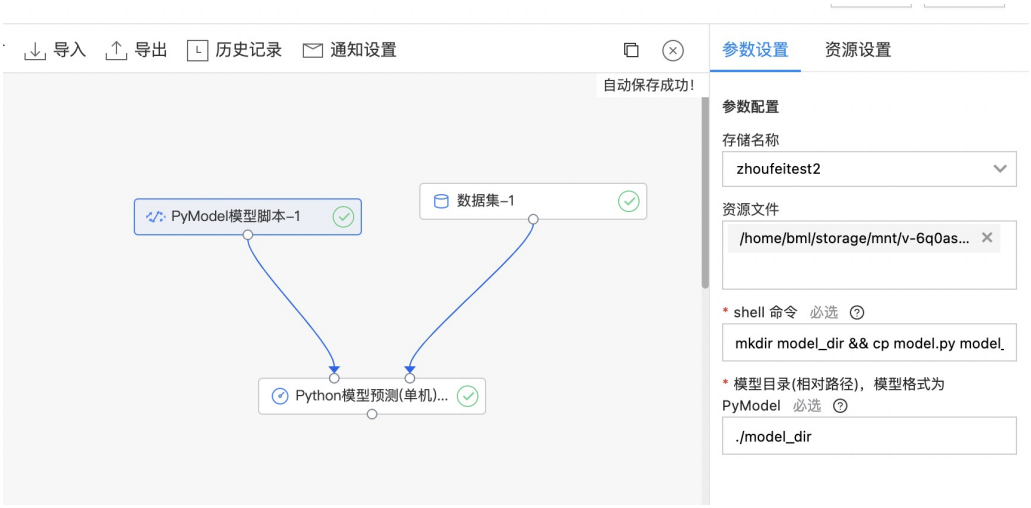
    def transform(self, dataset):  # type:(pd.DataFrame)->pd.DataFrame
        """

        :param dataset:
        :return:
        """
        return pd.DataFrame([{'prediction': 0}])

    def load(self):
        """

        :return:
        """
        pass
```

2. 选择存储名称、资源文件(model.py)，shell脚本输入框填写"mkdir model_dir && cp model.py model_dir"，完成训练。



PyTorch算法

PyTorch 是使用 GPU 和 CPU 优化的深度学习张量库。 自定义 PyTorch 代码组件：利用 PyTorch 库写自定义 python 代码，完成各类深度学习任务。

输入

- 输入一个数据集作为训练集，编写Python代码。

输出

- 输出PyTorch算法模型。

算子参数

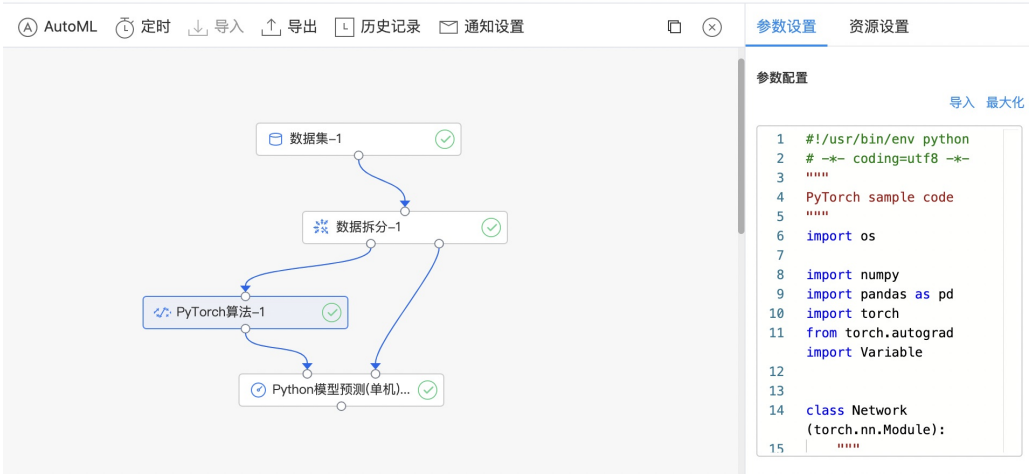
参数名称	是否必选	参数描述	默认值
Python 代码编辑窗口	是	输入Python 代码	

函数说明

fit：模型训练方法，有且仅有dataset参数
transform：模型预测方法，有且仅有dataset参数，返回的预测结果为pandas.DataFrame
save：模型保存方法,有且仅有model_path参数
load：模型加载方法,有且仅有model_path参数

使用示例

1. 构建算子结构，编写python代码，完成训练。



2. 查看代码，本例中特征列为slen、swid、plen、pwid，标签列为class。



Sklearn算法

scikit-learn 是基于 Python 语言的机器学习工具。它是建立在 NumPy ，SciPy 和 matplotlib 上的简单高效的数据挖掘和数据分析工具。 自定义 Sklearn 代码组件：利用 scikit-learn 库写自定义 python 代码，完成各类机器学习任务。

输入

- 输入一个数据集作为训练集，编写Python代码。

输出

- 输出Sklearn算法模型。

算子参数

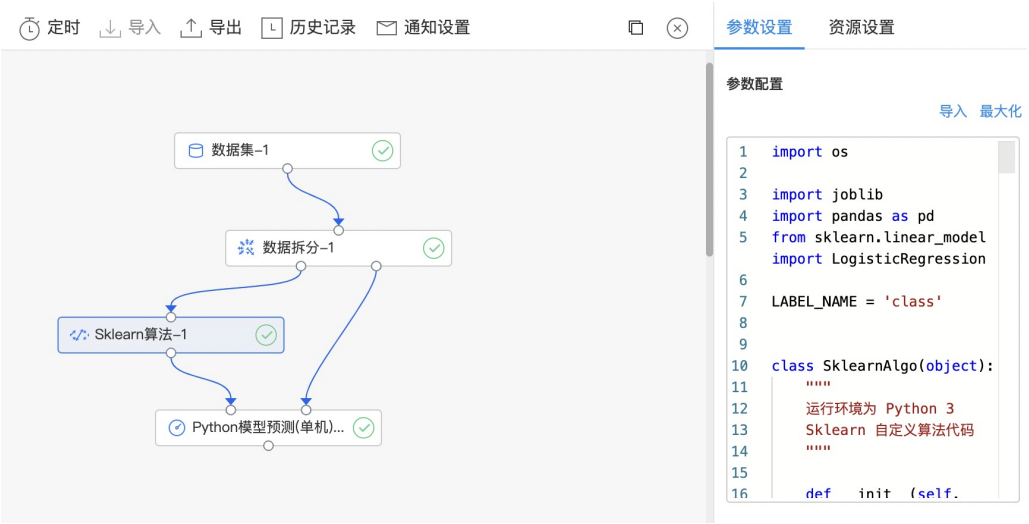
参数名称	是否必选	参数描述	默认值
Python 代码编辑窗口	是	输入Python 代码	

函数说明

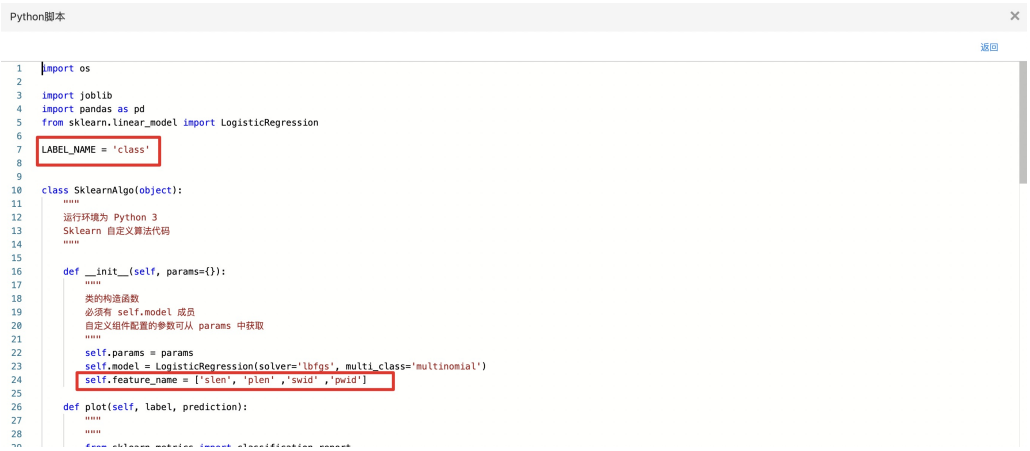
fit：模型训练方法，有且仅有dataset参数
transform：模型预测方法，有且仅有dataset参数，返回的预测结果为pandas.DataFrame
save：模型保存方法,有且仅有model_path参数
load：模型加载方法,有且仅有model_path参数

使用示例

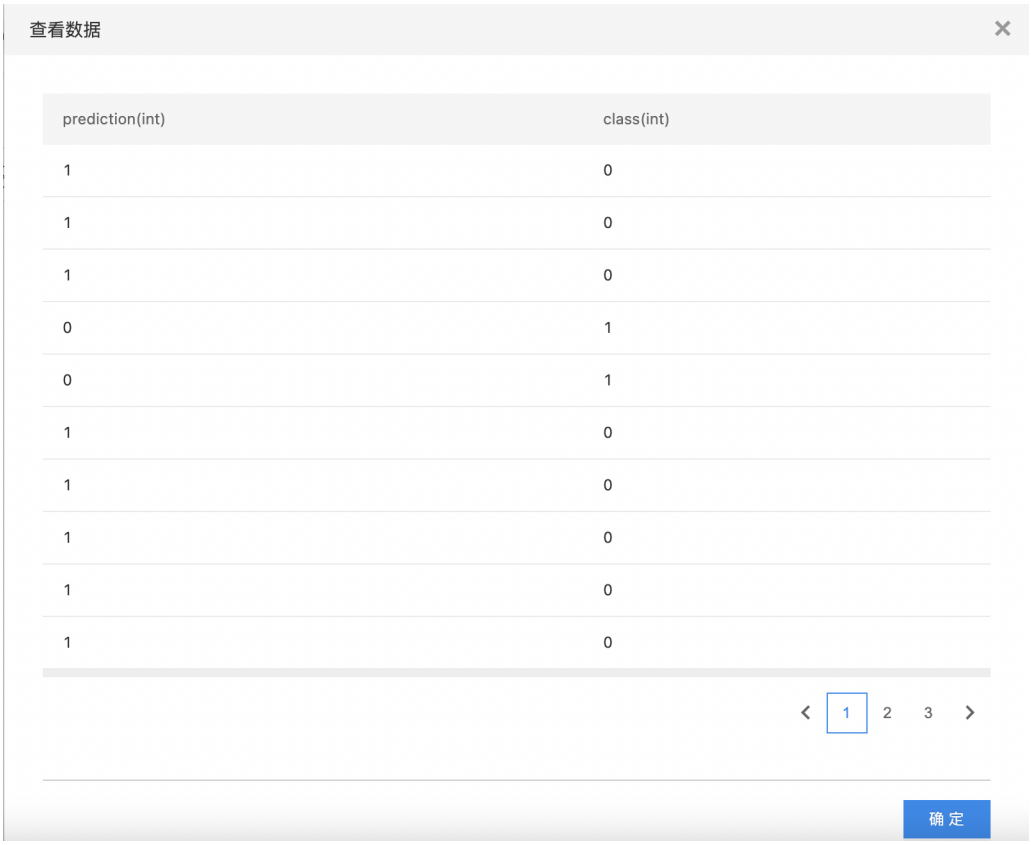
1. 构建算子结构，编写python代码，完成训练。



2. 查看代码，本例中特征列为slen、swid、plen、pwid，标签列为class。



3. 查看训练结果。



Sklearn算法(Pmml模型)

Sklearn算法(Pmml模型)组件与Sklearn算法组件的使用方法一致，不同的是Sklearn算法(Pmml模型)组件中用户可以自定义load和save 防范，并且训练完成后支持导出Pmml格式的模型。

输入

输入数据：一个数据集作为训练集，编写Python代码。

输出

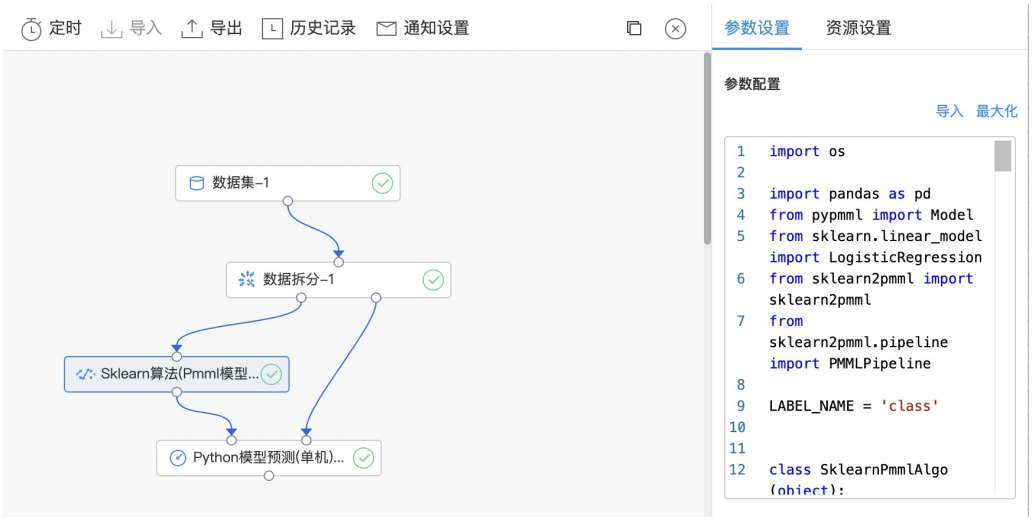
输出数据：Sklearn算法(Pmml模型)。

算子参数

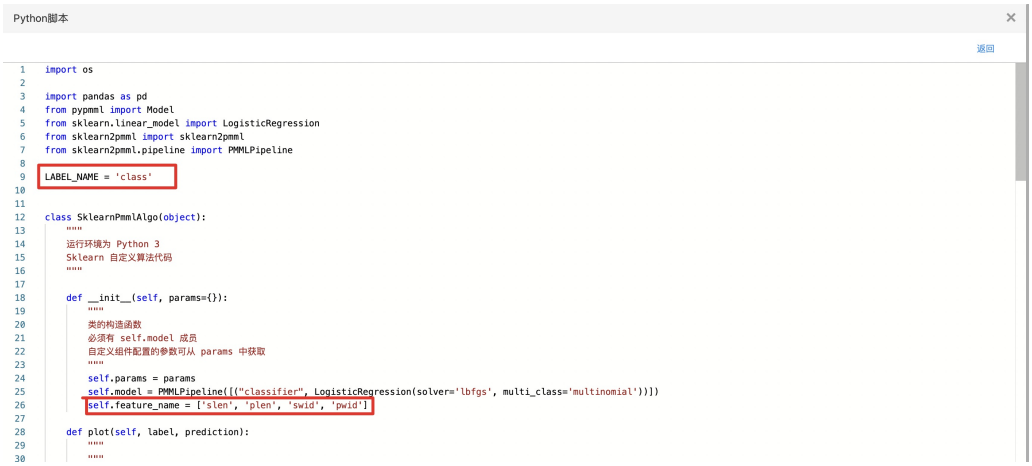
参数名称	是否必选	参数描述	默认值
Python 代码编辑窗口	是	输入Python 代码	

使用示例

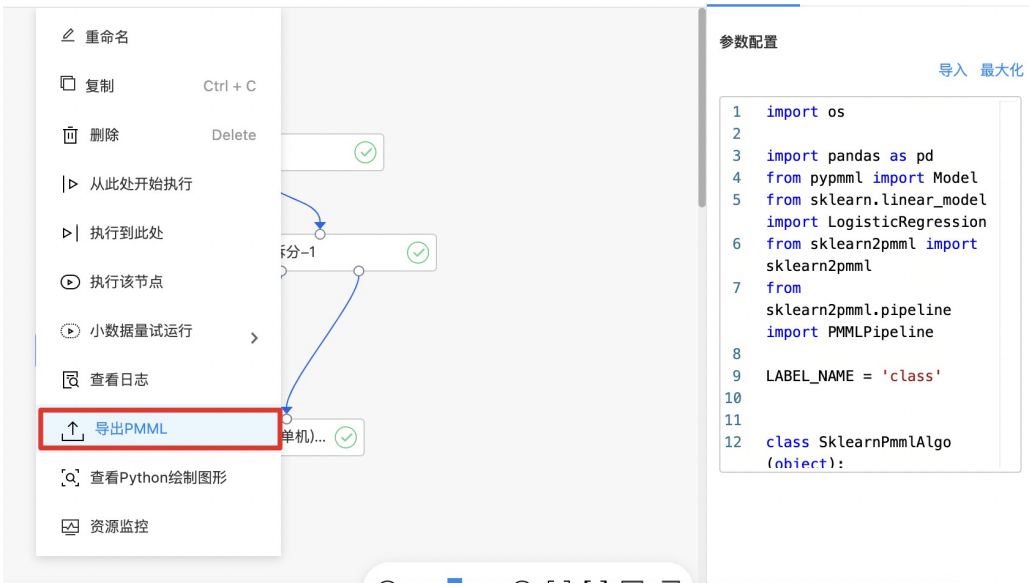
1. 构建算子结构，编写python代码，完成训练。



2. 查看代码，本例中特征列为slen、swid、plen、pwid，标签列为class。



3. 训练完成后的模型可以导出为PMML格式文件。



Tensorflow算法

TensorFlow 是一个端到端开源机器学习平台。它拥有一个包含各种工具、库和社区资源的全面灵活生态系统，可以让研究人员推动机器学习领域的先进技术的发展，并让开发者轻松地构建和部署由机器学习提供支持的应用。 自定义 Tensorflow 代码组件：利用 Tensorflow 库写自定义 python 代码，完成各类深度学习任务。

输入

- 输入一个数据集作为训练集，编写Python代码。

输出

- 输出Tensorflow算法模型。

算子参数

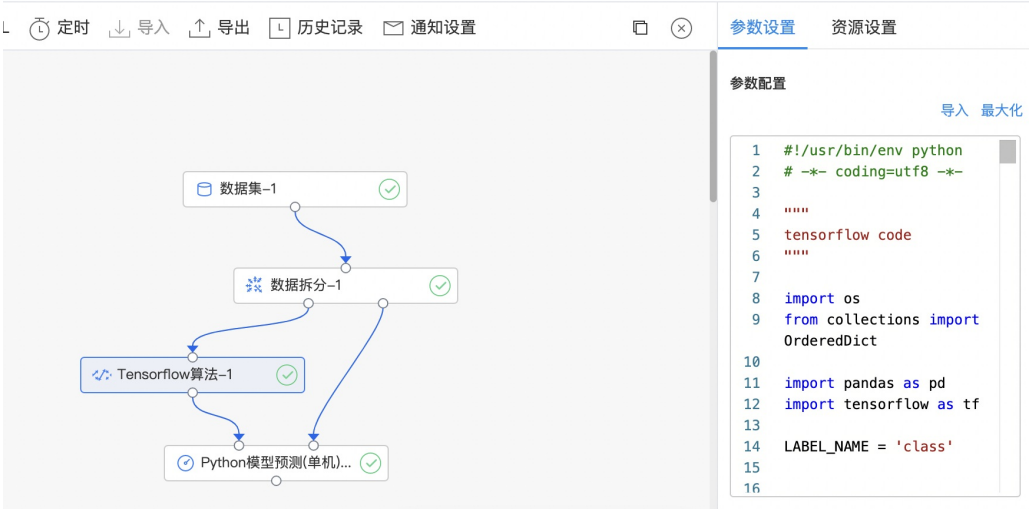
参数名称	是否必选	参数描述	默认值
Python 代码编辑窗口	是	输入Python 代码	

函数说明

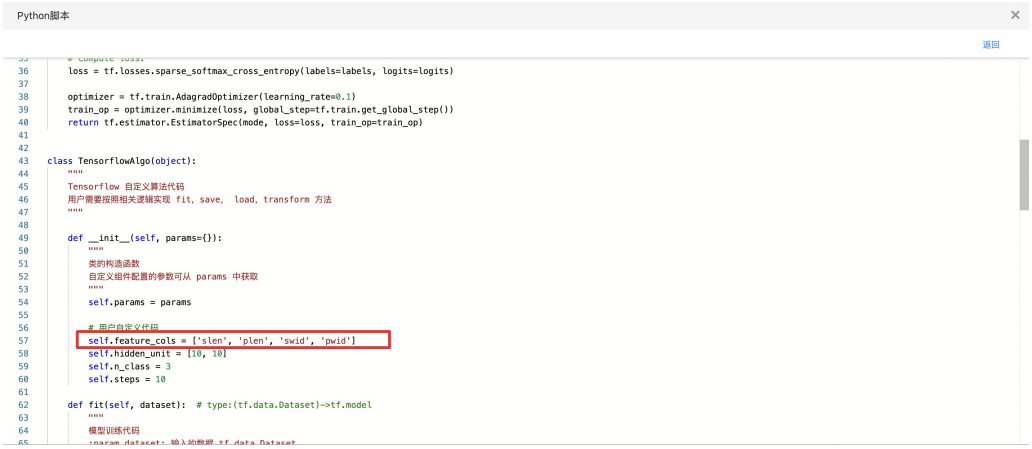
fit：模型训练方法，有且仅有dataset参数
transform：模型预测方法，有且仅有dataset参数，返回的预测结果为pandas.DataFrame
save：模型保存方法,有且仅有model_path参数
load：模型加载方法,有且仅有model_path参数

使用示例

1. 构建算子结构，编写python代码，完成训练。



2. 查看代码，本例中特征列为slen、swid、plen、pwid，标签列为class。



WideAndDeep

WideAndDeep 模型是 Google 发布的一类用于分类和回归的模型，并应用到了 Google Play 的应用推荐中。WideAndDeep 模型的核心思想是结合线性模型的记忆能力（memorization）和 DNN 模型的泛化能力(generalization)，在训练过程中同时优化 2 个模型的参数，从而达到整体模型的预测能力最优。 记忆(memorization)即从历史数据中发现 item 或者特征之间的相关性。 泛化(generalization)即相关性的传递，发现在历史数据中很少或者没有出现的新的特征组合。 WideAndDeep组件：利用 WideAndDeep 模型完成二分类任务。

输入

- 输入一个数据集，需要指定标签列，deep模式配置列，wide模式配置列。

输出

- 输出WideAndDeep模型。

算子参数

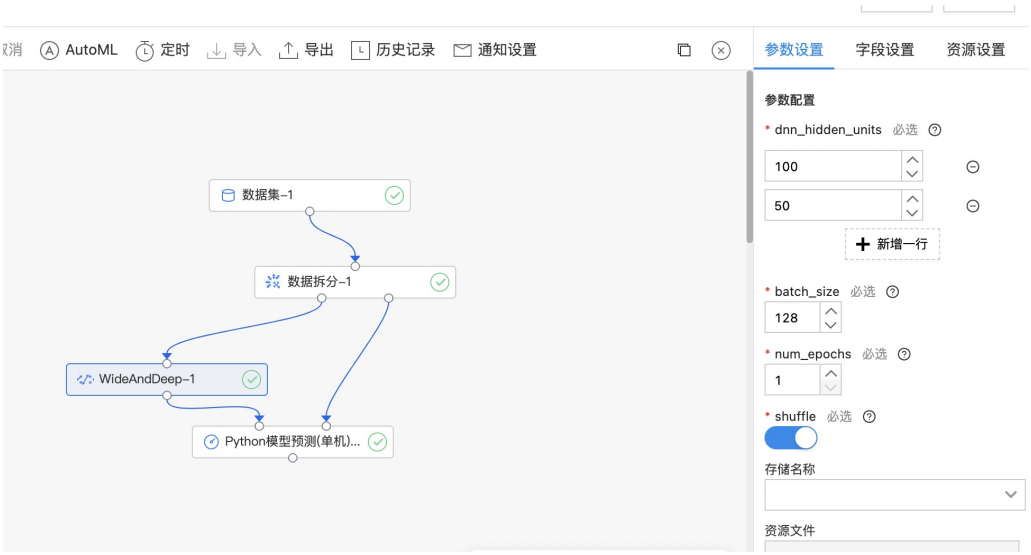
参数名称	是否必选	参数描述	默认值
dnn_hidden_units	是	设置DEEP中神经网络的隐藏层节点数，数组中必须为正整数 范围：[2, inf)。	初始有两行数据，默认值分别是100和50
batch_size	是	训练过程中的batch_size 范围：[1, inf)。	128
num_epochs	是	训练过程中的训练轮数 范围：[1, inf)。	1
shuffle	是	在训练过程中数据是否进行打乱。	开启
存储名称	是	选择项目内存储名称。	无
资源文件	是	选择对应存储内的代码文件。	无

字段参数

参数名称	是否必选	参数描述	默认值
标签列	是	标签列，要求是整数类型且标签列值为0或1，唯一值是2。	无
numeric_column	是	设置需要numeric_column的列，需要为数值类型数据列。	无
embedding_column	是	设置需要embedding_column的列，需要为string类别型数据列。	无
embedding_column中的dimension参数	是	设置embedding_column中的dimension参数 范围：[1, inf)。	8
categorical_column_with_vocabulary_list	否	设置WIDE模型中categorical_column_with_vocabulary_list列，需要为string类别型数据列	无
categorical_column_with_hash_bucket	是	设置WIDE模型中categorical_column_with_hash_bucket列，需要为string类别型数据列。	无
categorical_column_with_hash_bucket中hash_bucket_size参数	是	categorical_column_with_hash_bucket中hash_bucket_size参数范围：[1, inf)。	1000
crossed_column	否	设置WIDE模型中交叉特征列，最少选择两列数据，需要为string类别型数据。	无

使用示例

构建算子结构，配置参数，完成训练。



XGBoost二分类(单机)

XGBoost是一种提升树模型，它是将多个树模型（CART）集成在一起，形成一个很强的分类器。 对于一个样本，每棵树都会预测出一个结果（分数），对于二分类问题来说，学习目标为logistic。

输入

- 输入数据集。特征列需要是数值或数值数组类型，标签列需要是整数或字符串类型，标签唯一值为2。

输出

- 输出XGBoost二分类(单机)模型。

算子参数

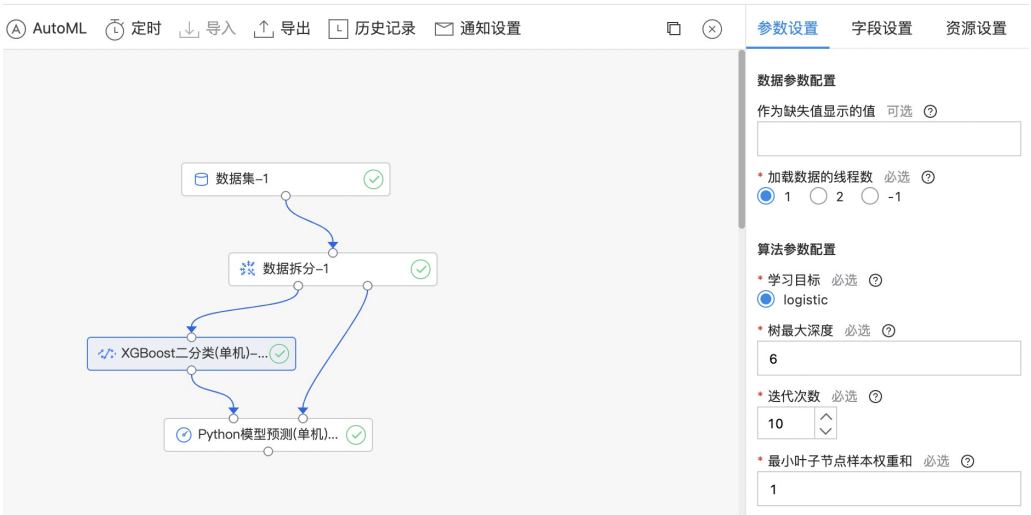
参数名称	是否必选	参数描述	默认值
作为缺失值显示的值	否	输入特征数据中需要作为缺失值显示的值（浮点型）。若没有设置，则默认为np.nan。	无
加载数据的线程数	是	适用并行化时用于加载数据的线程数。默认为1；如果为-1，则使用系统上可用的最大线程数。	1
学习目标	是	二分类任务的学习目标。	logistic
树最大深度	是	每棵树的最大深度。树越深，模型越复杂，越容易过拟合 范围：[1, inf)。	6
迭代次数	是	提升迭代次数，范围：[1, 200]。	10
最小叶子样本权重和	是	叶子节点需要的最小样本权重和，范围：[0.0, inf)。	1
正样本缩放比例	是	控制正负权重的平衡，对于不平衡的类别很有用。 典型值：sum（负实例）/ sum（正实例） 范围：[1e-08, inf)。	1
学习率	是	更新中使用的步长缩小，以防止过拟合 范围：[0.0, 1.0]。	0.30
gamma	是	节点分裂所需的最小损失函数下降值。gamma越大，算法越保守 范围：[0.0, inf)。	0
随机采样比例	是	构造每棵树的所用样本比例（样本采样比例） 范围：[0.01, 1.0]。	1.00
随机采样的特征比例	是	构造每棵树的所用特征比例（特征采样比例） 范围：[0.01, 1.0]。	1.00
L1正则化系数	是	权重的L1正则化项。值越大，模型越保守 范围：[0.0, inf)。	0
L2正则化系数	是	权重的L2正则化项。值越大，模型越保守 范围：[0.0, inf)。	1
随机种子	是	随机数种子	0
并行运行线程数	否	用于训练模型的并行线程数。若没有设置，则默认为最大可用线程数 范围：[1, 16]。	无

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求是整数或字符串类型，唯一值是2。	无
正样本标签值	否	二分类的正样本的标签值，应在标签列中存在。	无

使用示例

1. 构建算子结构，配置参数，完成训练，算子支持查看全量特征重要性。



2. 查看训练结果。

查看数据

(double)	time(int)	label(int)	probability_0(double)	positive_prob_1(double)	result_label(bigint)
15258789	175500	1	0.07169979810714722	0.9283002018928528	1
1525878906	175500	1	0.09967327117919922	0.9003267288208008	1
8474121094	175500	1	0.06928443908691406	0.9307155609130859	1
305175781	175500	1	0.1466817855834961	0.8533182144165039	1
9237060547	175500	1	0.25188708305358887	0.7481129169464111	1
9237060547	175500	1	0.1466817855834961	0.8533182144165039	1
9618530273	175500	1	0.0792771577835083	0.9207228422164917	1
190734863	175500	1	0.12046611309051514	0.8795338869094849	1
076293945	175500	1	0.05245119333267212	0.9475488066673279	1
	175500	1	0.09967327117919922	0.9003267288208008	1

1 2 3

XGBoost多分类(单机)

XGBoost 是一种提升树模型，它是将多个树模型（CART）集成在一起，形成一个很强的分类器。 对于一个样本，每棵树都会预测出一个结果（分数），对于多分类问题来说，学习目标为softprob。

输入

- 输入一个数据集，特征列需要是数值或数值数组类型，多分类任务标签列需要是整数或字符串类型，标签唯一值大于2。

输出

- 输出XGBoost多分类(单机)模型，支持查看Top10分类的概率。

算子参数

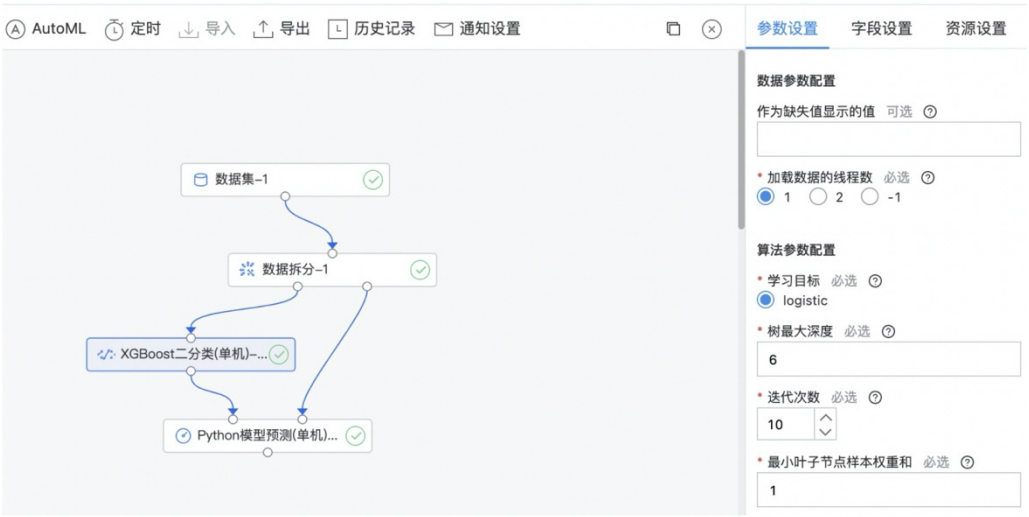
参数名称	是否必选	参数描述	默认值
作为缺失值显示的值	否	输入特征数据中需要作为缺失值显示的值（浮点型）。若没有设置，则默认为np.nan。	无
加载数据的线程数	是	适用并行化时用于加载数据的线程数。默认为1；如果为-1，则使用系统上可用的最大线程数。	1
学习目标	是	多分类任务的学习目标。	softprob
树最大深度	是	每棵树的最大深度。树越深，模型越复杂，越容易过拟合 范围：[1, inf)。	6
迭代次数	是	提升迭代次数，范围：[1, 200]。	10
最小叶子样本权重和	是	叶子节点需要的最小样本权重和，范围：[0.0, inf)。	1
学习率	是	更新中使用的步长缩小，以防止过拟合 范围：[0.0, 1.0]。	0.30
gamma	是	节点分裂所需的最小损失函数下降值。gamma越大，算法越保守 范围：[0.0, inf)。	0
随机采样比例	是	构造每棵树的所用样本比例（样本采样比例） 范围：[0.01, 1.0]。	1.00
随机采样的特征比例	是	构造每棵树的所用特征比例（特征采样比例） 范围：[0.01, 1.0]。	1.00
L1正则化系数	是	权重的L1正则化项。值越大，模型越保守 范围：[0.0, inf)。	0
L2正则化系数	是	权重的L2正则化项。值越大，模型越保守 范围：[0.0, inf)。	1
随机种子	是	随机数种子	0
并行运行线程数	否	用于训练模型的并行线程数。若没有设置，则默认为最大可用线程数 范围：[1, 16]。	无

字段参数

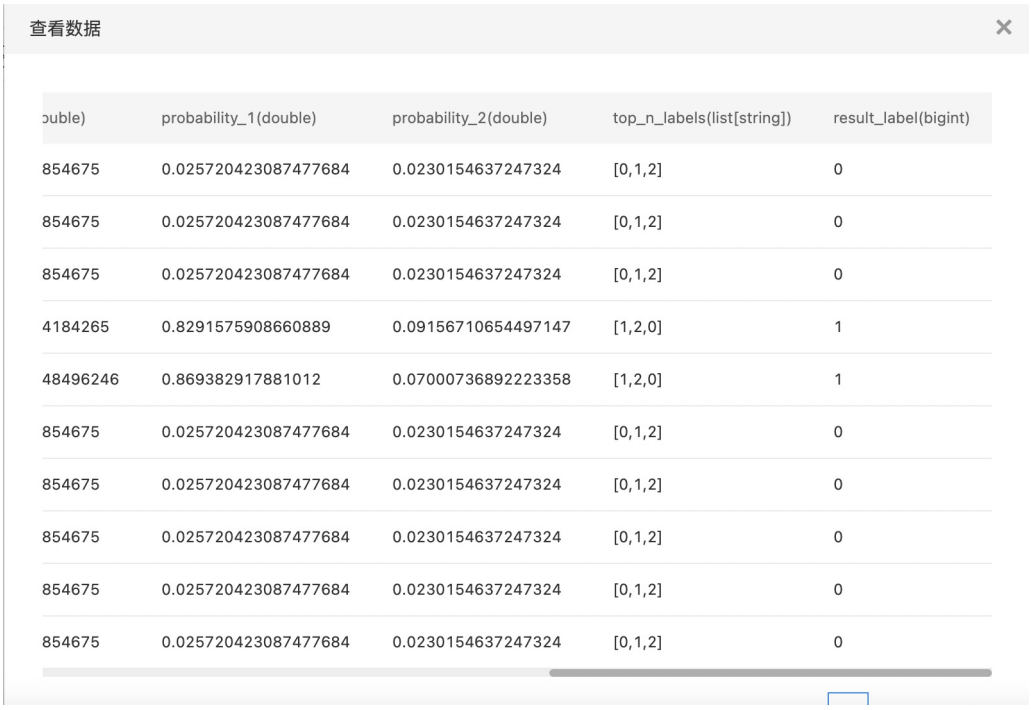
参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求是整数或字符串类型。	无

使用示例

1. 构建算子结构，配置参数，完成训练，算子支持查看全量特征重要性。



2. 查看训练结果。



XGBoost回归 (单机)

XGBoost 是一种迭代的决策树算法，它是由多棵决策树组成，所有树的结论累加起来做最终答案。XGBoost 几乎可用于所有回归问题(线性/非线性)。

输入

- 输入一个数据集，特征列需要是数值或数值数组类型，回归任务标签列需要是数值类型，对于回归任务目标函数gamma及tweedie，要求标签列是非负的。

输出

- 输出XGBoost回归(单机)模型

算子参数

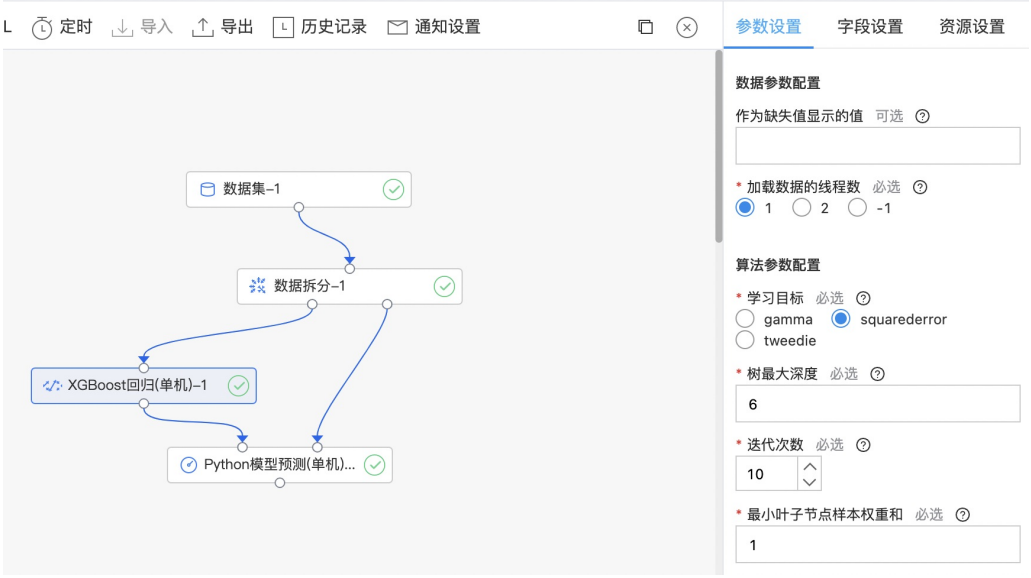
参数名称	是否必选	参数描述	默认值
作为缺失值显示的值	否	输入特征数据中需要作为缺失值显示的值（浮点型）。若没有设置，则默认为np.nan。	无
加载数据的线程数	是	适用并行化时用于加载数据的线程数。默认为1；如果为-1，则使用系统上可用的最大线程数。	1
学习目标	是	回归任务的学习目标，目前支持： gamma squarederror tweedie	squarederror
树最大深度	是	每棵树的最大深度。树越深，模型越复杂，越容易过拟合 范围：[1, inf)。	6
迭代次数	是	提升迭代次数，范围：[1, 200]。	10
最小叶子样本权重和	是	叶子节点需要的最小样本权重和，范围：[0.0, inf)。	1
学习率	是	更新中使用的步长缩小，以防止过拟合 范围：[0.0, 1.0]。	0.30
gamma	是	节点分裂所需的最小损失函数下降值。gamma越大，算法越保守 范围：[0.0, inf)。	0
随机采样比例	是	构造每棵树的所用样本比例（样本采样比例） 范围：[0.01, 1.0]。	1.00
随机采样的特征比例	是	构造每棵树的所用特征比例（特征采样比例） 范围：[0.01, 1.0]。	1.00
L1正则化系数	是	权重的L1正则化项。值越大，模型越保守 范围：[0.0, inf)。	0
L2正则化系数	是	权重的L2正则化项。值越大，模型越保守 范围：[0.0, inf)。	1
随机种子	是	随机数种子	0
并行运行线程数	否	用于训练模型的并行线程数。若没有设置，则默认为最大可用线程数 范围：[1, 16]。	无

字段参数

参数名称	是否必选	参数描述	默认值
特征列	是	预测使用的特征列，要求必须是数值或数值数组类型。	无
标签列	是	预测目标列，要求是数值类型。	无

使用示例

构建算子结构，配置参数，完成训练，算子支持查看全量特征重要性。



NLP算法

NLP算法

Bert命名实体识别

BERT 采用了 Transformer Encoder 的模型来作为语言模型，Transformer模型完全抛弃了 RNN/CNN 等结构，而完全采用 Attention 机制来进行 input-output 之间关系的计算。 Fine-tuning 方式是指在已经训练好的语言模型的基础上，加入少量的 task-specific parameters, 例如对于分类问题在语言模型基础上加一层 softmax 网络，然后在新的语料上重新训练来进行 fine-tune。 Bert命名实体识别模型将句子输入到bert中，在bert输出后连接一个crf层得到最后结果，输入的数据格式应满足BIO的标注格式。

输入

- 输入文本数据集，标注数据常采用BIO的标注方式。

输出

- 输出Bert命名实体识别模型。

算子参数

参数名称	是否必选	参数描述	默认值
batch_size	是	训练过程中的batch_size 范围：[1, inf)。	4
epoch	是	训练过程中的训练轮数 范围：[1, inf)。	1
do_lower_case	是	输入文本是否小写, 对于中文文本应当设置为True。	开启
学习率	是	训练开始时的学习率 范围：[0.0, 1.0]。	3e-05
序列最大长度，超过(序列最大长度-1)的输入字将被截取掉	是	序列最大长度 范围：[2, inf)。	128
warmup_proportion	是	进行线性学习率预热的训练比例 范围：[0.0, 1.0]。	0.1
save_checkpoints_steps	是	保存checkpoint的频率 范围：[1, inf)。	1000
训练集/验证集划分比例	是	训练集合比例设置成 1 则不使用验证集。 范围：[0.01, 1.0]。	0.8
选择预训练模型	是	选择预训练模型	中文训练模型

字段参数

参数名称	是否必选	参数描述	默认值
文本列	是	请选择文本列，句子分字，字之间必须以空格间隔，类型为字符串。	无
标签列	是	请选择标签列，采用BIO格式标注方式，标签之间必须以空格间隔且个数和文本列相同，类型为字符串。	无

使用示例

1. BIO方式标注的数据。

查看数据

index(int)	sentence(string)	label(string)
0	海钓比赛地点在厦门与金门之间的海域。	O O O O O O O B-LOC I-LOC O B-LOC I-LOC O O O O O O
1	这座依山傍水的博物馆由国内一流的设计师主持设计，整个建筑群精美而恢宏。	O O O O O O O

算子参数

参数名称	是否必选	参数描述	默认值
batch_size	是	训练过程中的batch_size 范围：[1, inf)。	4
epoch	是	训练过程中的训练轮数 范围：[1, inf)。	1
do_lower_case	是	输入文本是否小写, 对于中文文本应当设置为True。	开启
学习率	是	训练开始时的学习率 范围：[0.0, 1.0]。	3e-05
序列最大长度，超过(序列最大长度-1)的输入字将被截取掉	是	序列最大长度 范围：[2, inf)。	128
warmup_proportion	是	进行线性学习率预热的训练比例 范围：[0.0, 1.0]。	0.1
save_checkpoints_steps	是	保存checkpoint的频率 范围：[1, inf)。	1000
训练集/验证集划分比例	是	训练集合比例设置成 1 则不使用验证集。 范围：[0.01, 1.0]。	0.8
选择预训练模型	是	选择预训练模型	中文训练模型

字段参数

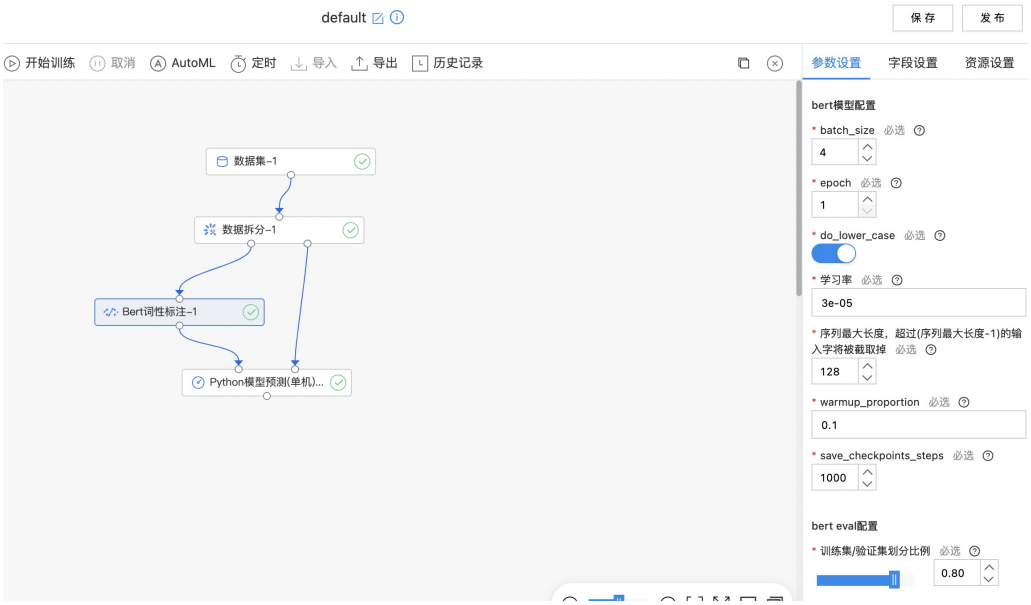
参数名称	是否必选	参数描述	默认值
文本列	是	请选择文本列，句子分字，字之间必须以空格间隔，类型为字符串。	无
标签列	是	请选择标签列，采用BIO格式标注方式，标签之间必须以空格间隔且个数和文本列相同，类型为字符串。	无

使用示例

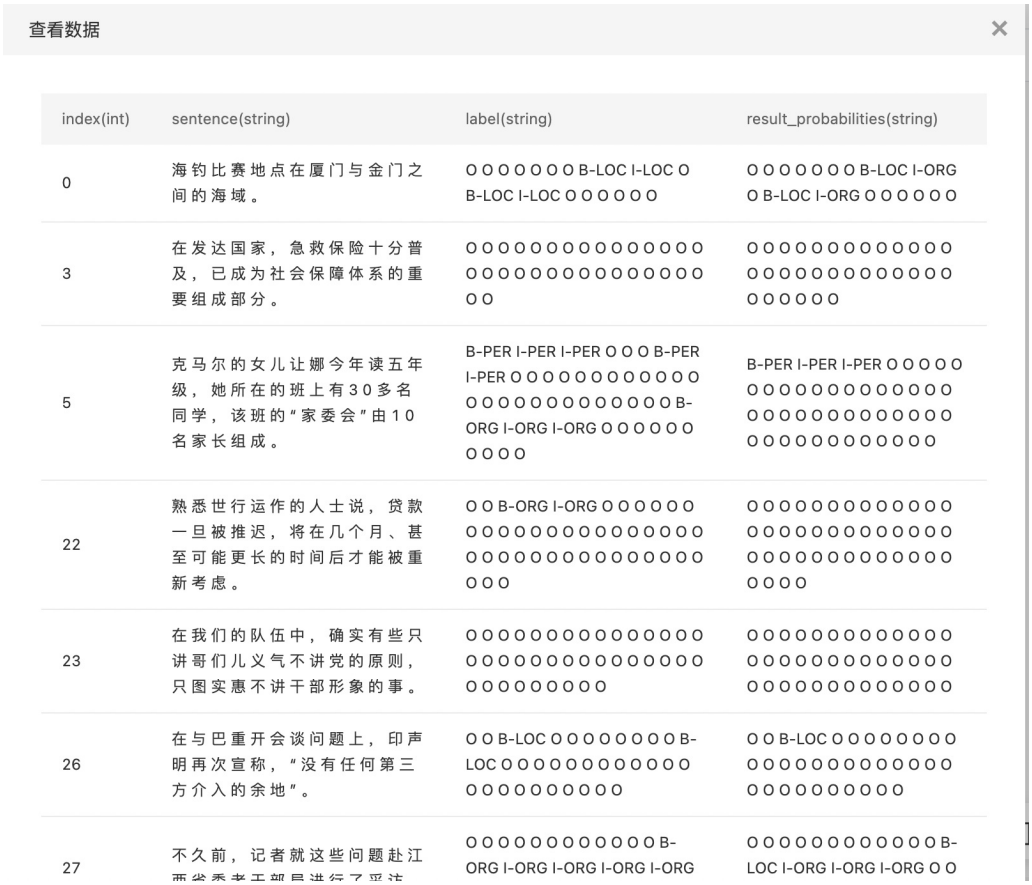
1. BIO方式标注的数据。

查看数据

index(int)	sentence(string)	label(string)
0	海 钓 比 赛 地 点 在 厦 门 与 金 门 之 间 的 海 域 。	O O O O O O O B-LOC I-LOC O B-LOC I-LOC O O O O O O
1	这 座 依 山 傍 水 的 博 物 馆 由 国 内 一 流 的 设 计 师 主 持 设 计 ， 整 个 建 筑 群 精 美 而 恢 宏 。	O O O O O O O



3. 查看预测结果。



index(int)	sentence(string)	label(string)	result_probabilities(string)
0	海钓比赛地点在厦门与金门之间的海域。	O O O O O O O B-LOC I-LOC O B-LOC I-LOC O O O O O O	O O O O O O O B-LOC I-ORG O B-LOC I-ORG O O O O O O
3	在发达国家，急救保险十分普及，已成为社会保障体系的重要组成部分。	O O O O O O O

参数名称	是否必选	参数描述	默认值
batch_size	是	训练过程中的batch_size 范围：[1, inf)。	4
epoch	是	训练过程中的训练轮数 范围：[1, inf)。	1
do_lower_case	是	输入文本是否小写, 对于中文文本应当设置为True。	开启
学习率	是	训练开始时的学习率 范围：[0.0, 1.0]。	3e-05
序列最大长度	是	序列最大长度 范围：[2, inf)。	128
warmup_proportion	是	进行线性学习率预热的训练比例 范围：[0.0, 1.0]。	0.1
save_checkpoints_steps	是	保存checkpoint的频率 范围：[1, inf)。	1000
训练集/验证集划分比例	是	训练集合比例设置成 1 则不使用验证集。 范围：[0.01, 1.0]。	0.8
选择预训练模型	是	选择预训练模型	中文训练模型

字段参数

参数名称	是否必选	参数描述	默认值
文本列	是	按顺序选择两列实体与一列文本（类型为字符串类型）。	无
标签列	是	选择一列标签列。	无

使用示例

构建算子结构，配置参数，完成训练。



Bert文本相似度

BERT 采用了 Transformer Encoder 的模型来作为语言模型，Transformer模型完全抛弃了 RNN/CNN 等结构，而完全采用 Attention 机制来进行 input-output 之间关系的计算。 Fine-tuning 方式是指在已经训练好的语言模型的基础上，加入少量的 task-specific parameters, 例如对于分类问题在语言模型基础上加一层 softmax 网络，然后在新的语料上重新训练来进行 fine-tune。 Bert 文本相似度模型：BERT + Fine-tuning。

输入

- 输入数据集，特征列为两列文本列（类型为字符串类型），标签列为一列文本列（类型为整型）。

输出

- 输出Bert文本相似度模型。

算子参数

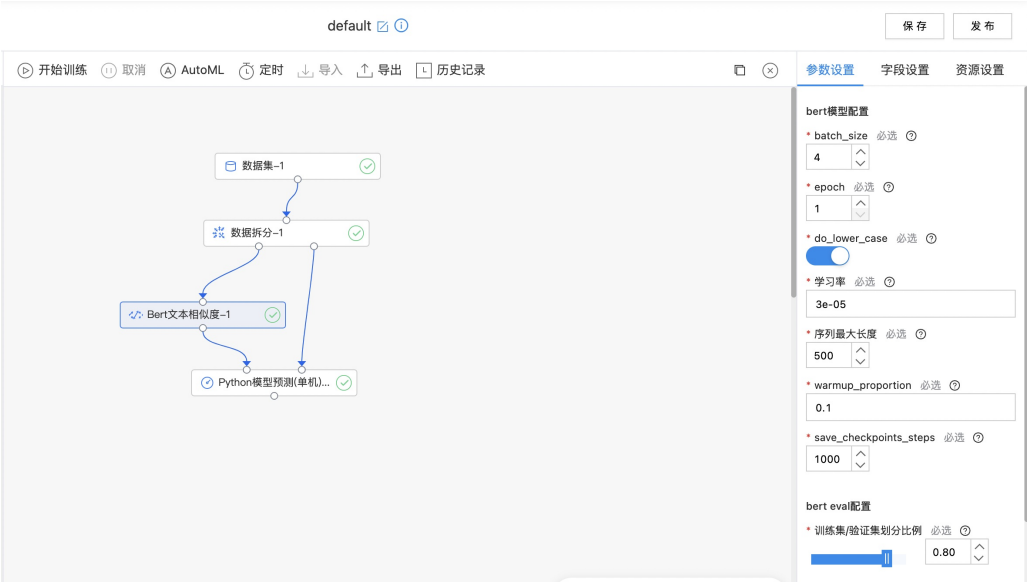
参数名称	是否必选	参数描述	默认值
batch_size	是	训练过程中的batch_size 范围：[1, inf)。	4
epoch	是	训练过程中的训练轮数 范围：[1, inf)。	1
do_lower_case	是	输入文本是否小写, 对于中文文本应当设置为True。	开启
学习率	是	训练开始时的学习率 范围：[0.0, 1.0]。	3e-05
序列最大长度	是	序列最大长度 范围：[2, inf)。	128
warmup_proportion	是	进行线性学习率预热的训练比例 范围：[0.0, 1.0]。	0.1
save_checkpoints_steps	是	保存checkpoint的频率 范围：[1, inf)。	1000
训练集/验证集划分比例	是	训练集合比例设置成 1 则不使用验证集。 范围：[0.01, 1.0]。	0.8
选择预训练模型	是	选择预训练模型。	中文训练模型

字段参数

参数名称	是否必选	参数描述	默认值
文本列	是	选择两列文本列（类型为字符串类型）	无
标签列	是	选择一列文本列（类型为整型）	无

使用示例

构建算子结构，配置参数，完成训练。



Fasttext文本分类

Fasttext 是一种简单有效的句子分类算法， 通过词向量以及 n-gram 向量的平均值计算出句子的向量表示，再通过全连接层网络对句子进行分类。

输入

- 输入一个数据集，特征列需要是字符串列表或字符串类型（如果是字符串类型，假设字符串已经经过了分词，以空格或tab分割），标签列需要是枚举类型。

输出

- 输出Fasttext模型，可以使用python预测组件进行预测。

算子参数

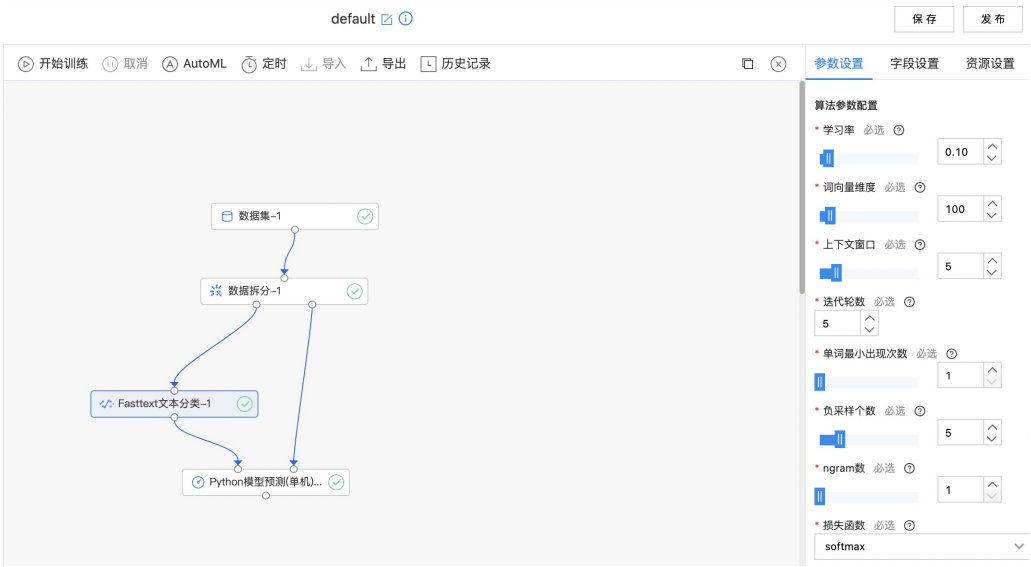
参数名称	是否必选	参数描述	默认值
学习率	是	学习率 范围：[0.01, 1.0]。	0.10
词向量维度	是	词向量维度 范围：[50, 500]。	100
上下文窗口	是	训练词向量考虑的上下文窗口大小 范围：[3, 15]。	5
迭代轮数	是	算法运行的 epoch 数，迭代几轮训练集 范围：[1, 100]。	5
单词最小出现次数	是	小于该数值的单词记为OOV 范围：[1, 10]。	1
负采样个数	是	负采样个数 范围：[1, 20]。	5
ngarm数	是	ngram数 范围：[1, 5]。	1
损失函数	是	损失函数，目前支持： softmax hs ns ova	softmax

字段参数

参数名称	是否必选	参数描述	默认值
输入列	是	必须是字符串列表或字符串类型。如果是字符串类型，需要提前以空格或tab分词。	无
标签列	是	需要是枚举类型。	无

使用示例

构建算子结构，配置参数，完成训练。



LDA主题模型

LDA 在主题模型中占有非常重要的地位，常用来文本分类，它可以将文档集中每篇文档的主题以概率分布的形式给出，从而通过分析一些文档抽取出它们的主题分布后，便可以根据主题分布进行主题聚类或文本分类。

输入

- 输入一个数据集，输入列为字符串数组类型（既经过分词后的数据）。

输出

- 输出LDA主题模型。

算子参数

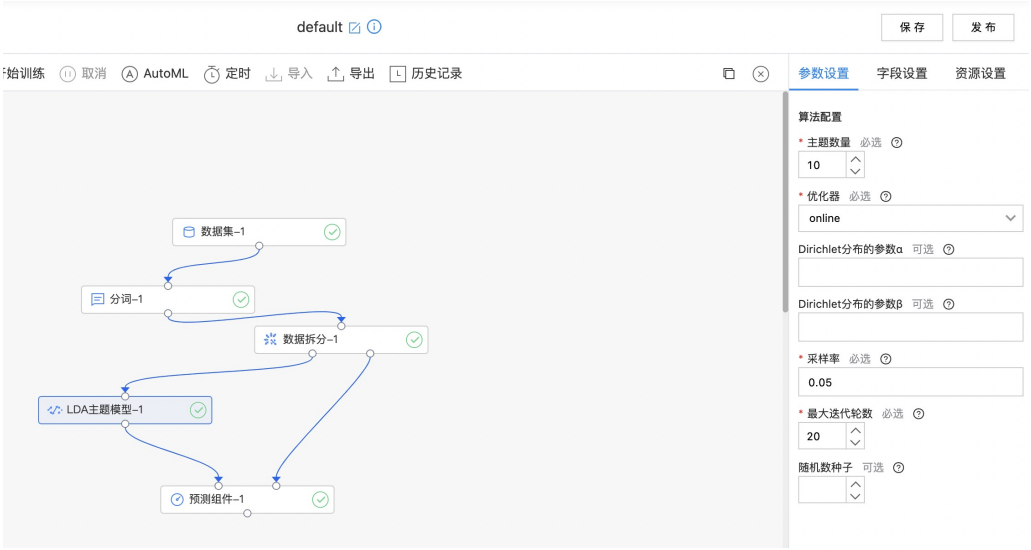
参数名称	是否必选	参数描述	默认值
主题数量	是	主题数量（聚簇中心数量） 范围：[2, inf)。	10
优化器	是	优化器用来学习LDA模型，当前支持： online:Online Variational Bayes和em:Expectation-Maximization	online
Dirichlet分布的参数α	否	文档在主题上分布的先验参数（超参数α),值越大推断出得分布越平滑 范围：[1.000000000000001, inf)。	无
Dirichlet分布的参数β	否	主题在单词上的先验分布参数,值越大推断出得分布越平滑 范围：[1.000000000000001, inf)。	无
采样率	是	小批量梯度下降的每次迭代中要采样和使用的语料的比例 范围：[1.0E-15, 1.0]。	0.05
最大迭代轮数	是	当迭代次数大于该数值时，停止迭代 范围：[1, inf)。	20
随机种子	否	随机数种子。	无

字段参数

参数名称	是否必选	参数描述	默认值
输入列	是	需要做LDA的列，类型需要是字符串数组。	无

使用示例

构建算子结构，配置参数，完成训练。



自然语言处理组件

自然语言处理组件

Word2Vec

Word2Vec 是一种经典的词向量算法，能够从大量文本中学习出各个词语的向量表示，其利用神经网络，可以通过训练，将词映射到 K 维度空间向量，甚至对于表示词的向量进行操作还能和语义相对应，由于其简单和高效引起了很多人的关注。

输入

- 输入一个数据集，输入列是分词的结果，用这个数据集训练词向量。

输出

- 输出python模型，可用于将输入的list string列转化成vector列。

算子参数

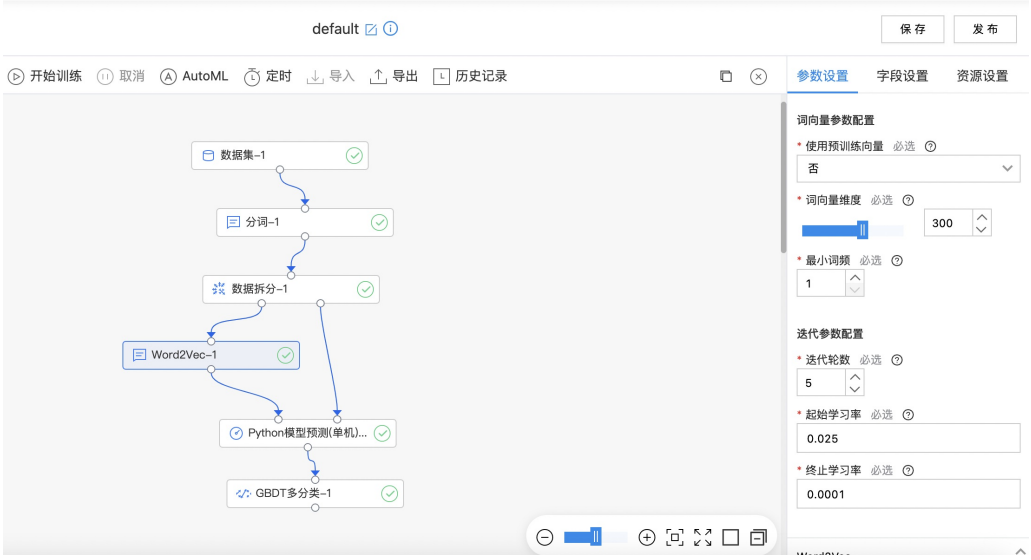
参数名称	是否必选	参数描述	默认值
词向量参数配置	是	是否使用预置的词向量。若开启，词向量维度会设为300，词向量会优先使用预训练词库中的值，预训练词库中没有的词才会训练词向量。	否
词向量维度	是	训练出的词向量的维度 范围：[5, 500]。	300
最小词频	是	训练样本中词频小于此值的词不计算词向量 范围：[1, inf)。	1
迭代轮数	是	算法运行的 epoch 数，迭代几轮训练集 范围：[1, inf)。	5
起始学习率	是	训练开始时的学习率 范围：[0.00025, 0.1]。	0.025
终止学习率	是	训练过程中，学习率会线性地下降到终止学习率 范围：[1e-05, 0.001]。	0.0001
训练算法	是	选择训练算法： CBOW skip-gram	CBOW
softmax优化	是	计算 softmax 时的优化算法： hierarchical softmax	negative sampling
负采样词数	是	负采样选择的词数设置成 0 则不使用负采样。对于小数据集，一般选择 5-20；对于大数据集，一般选择 2-5 个即可 范围：[0, 30]。	5

字段参数

参数名称	是否必选	参数描述	默认值
输入列	是	必须是字符串列表类型，分词后的结果。	无

使用示例

构建算子结构，配置参数，完成训练。



自动摘要

自动摘要是自动将文本转换生成简短摘要的信息压缩技术，要求有足够的信息量、较低的冗余度、较高的可读性。

输入

- 输入文本数据集，用户需要选择要做**自动摘要**的一列。
- 输入列的类型需要是（文本）字符串。

输出

- 输出自动摘要结果数据集，关键句模式与摘要模式的输出列名不同。
  - 关键句模式：在输入数据集的基础上添加了"原始列名*top_sentence_list*"*列，列的类型是字符串数组。*
  - 摘要模式：在输入数据集的基础上添加了"原始列名_summary"列，列的类型是字符串。

算子参数

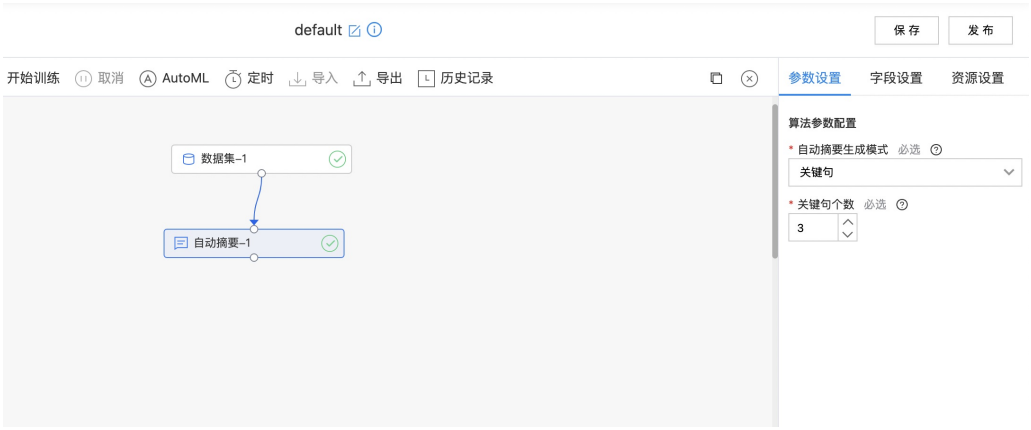
参数名称	是否必选	参数描述	默认值
自动摘要生成模式	是	提供两种自动摘要生成模式：关键句、摘要	关键句
关键句个数	是	需要的关键句的个数。若设置的个数大于生成的关键句列表的总长度，则返回整个关键句列表。 范围：[1, 30]	3

字段参数

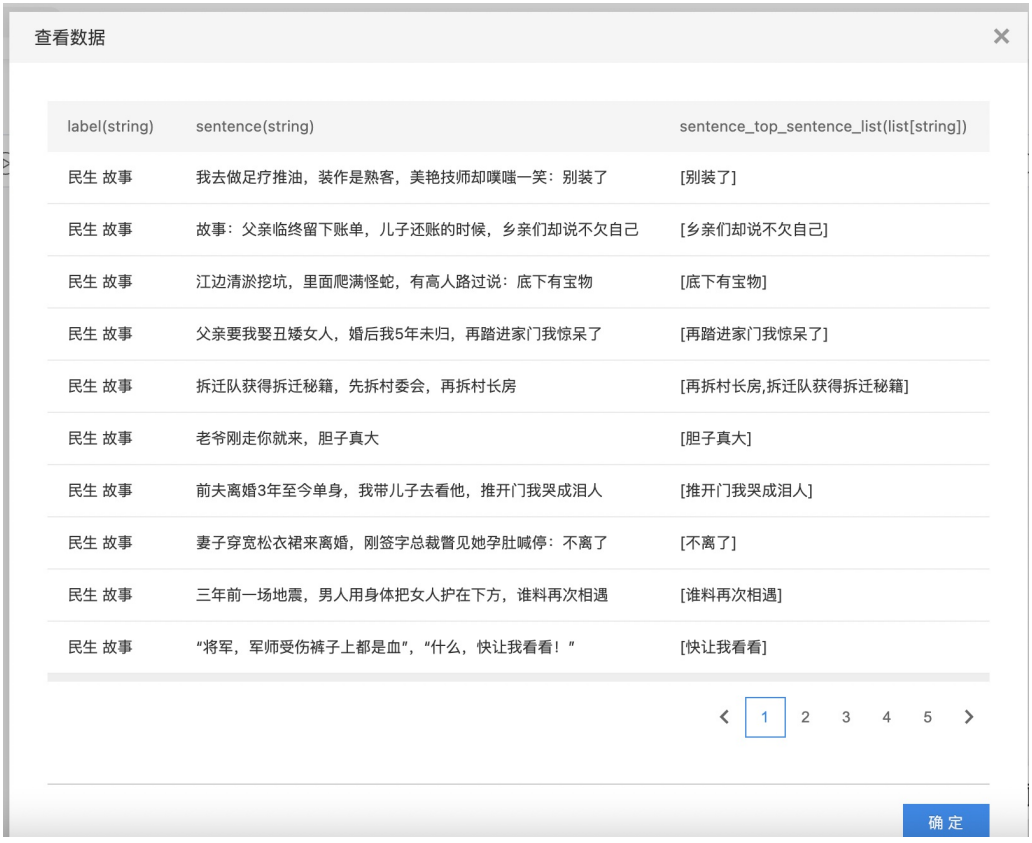
参数名称	是否必选	参数描述	默认值
文本列	是	待处理的文本列，类型为str。	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看自动摘要结果。



依存句法分析

句法分析是自然语言处理中的关键技术之一，其基本任务是确定句子的句法结构或者句子中词汇之间的依存关系。主要包括两方面的内容，一是确定语言的语法体系，即对语言中合法的句子的语法结构给与形式化的定义；另一方面是句法分析技术，即根据给定的语法体系，自动推导出句子的句法结构，分析句子所包含的句法单位和这些句法单位之间的关系。 **输入**

- 输入文本数据集，用户需要选择要做**依存句法分析**的一列。
- 输入列的类型需要是（文本）字符串类型。

输出

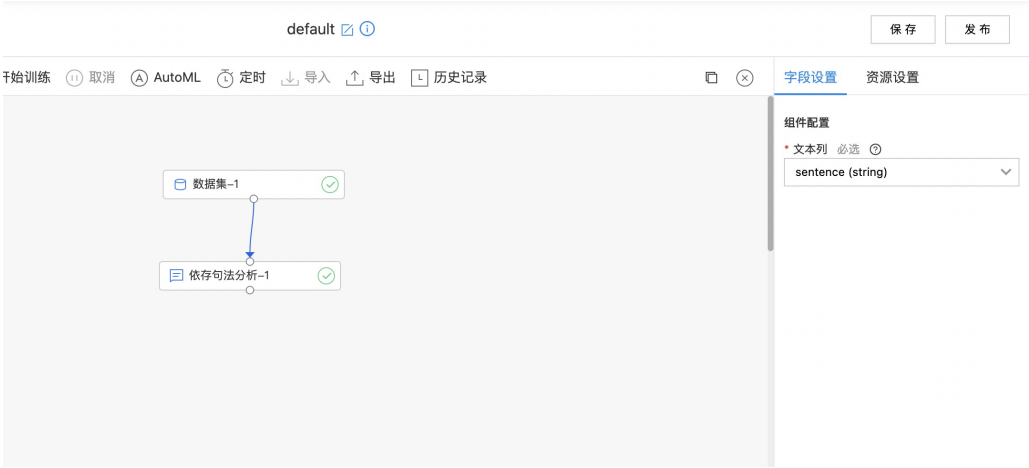
- 输出依存句法分析结果数据集，在输入数据集的基础上添加了"原始列名_parse_dependency"列，列的类型是字符串数组。

字段参数

参数名称	是否必选	参数描述	默认值
文本列	是	待处理的文本列，类型为str。	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看依存句法分析结果。



分词

分词就是将连续的字序列按照一定的规范重新组合成词序列的过程。组件目前支持两种分词方式：精准分词、新词发现。

输入

- 输入数据集，选择需要进行分词的列。

输出

- 输出数据集新增分词结果列。

算子参数

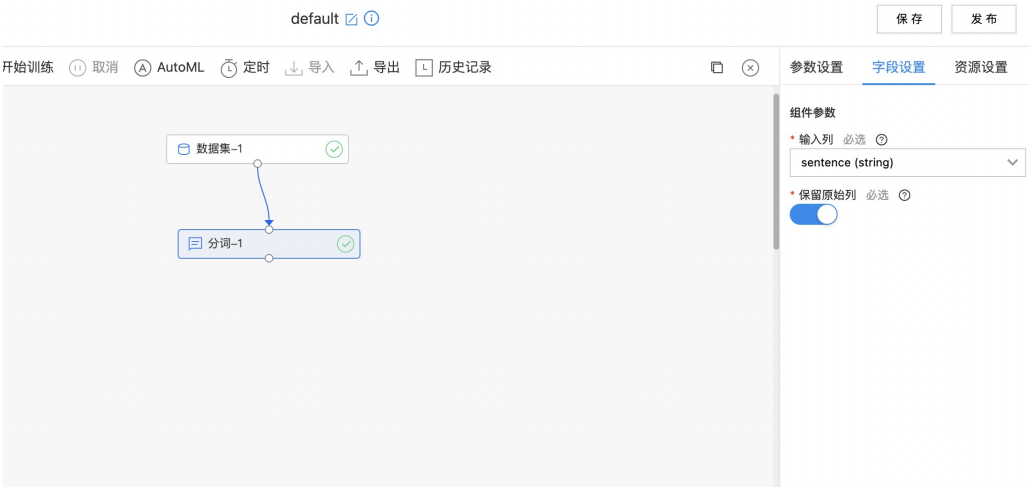
参数名称	是否必选	参数描述	默认值
分词器	是	选择分词器，目前支持精准分词和新词发现。精准分词是比较稳定的分词器，新词发现带有新词发现的功能。	精准分词
人名识别	是	是否开启人名识别。	开启
数字识别	是	是否开启数字识别，开启后[五十八]会被分成一个数字。	开启
量词合并	是	是否把数字和量词合并，开启数字识别后有效，开启后[8号]会被识别成一个词。	开启
词性标注	是	是否标注词性。	关闭

字段参数

参数名称	是否必选	参数描述	默认值
输入列	是	需要分词的列，需要是字符串类型。	无
保留原始列	是	是否保留原始列。如果保留，则输出结果在原始列名前添加[splitted_]前缀如果不保留，则替换原始列。	开启

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看分词结果。

查看数据

label(string)	sentence(string)	splitted_sentence(list[string])
民生 故事	我去做足疗推油，装作是熟客，美艳技师却噗嗤一笑：别装了	[我,去,做,足,疗,推,油,，,装作,是,熟客,，,美艳,技师,却,噗嗤,一,笑,：,别,装,了]
民生 故事	故事：父亲临终留下账单，儿子还账的时候，乡亲们却说不欠自己	[故事,：,父亲,临终,留下,账单,，,儿子,还账,的,时候,，,乡亲,们,却说,不,欠,自己]
民生 故事	江边清淤挖坑，里面爬满怪蛇，有高人路过说：底下有宝物	[江边,清淤,挖坑,，,里面,爬,满,怪,蛇,，,有,高人,路过,说,：,底下,有,宝物]
民生 故事	父亲要我娶丑矮女人，婚后我5年未归，再踏进家门我惊呆了	[父亲,要,我,娶,丑,矮,女人,，,婚后,我,5年,未归,，,再,踏进,家门,我,惊呆,了]
民生 故事	拆迁队获得拆迁秘籍，先拆村委会，再拆村长房	[拆迁,队,获得,拆迁,秘籍,，,先,拆,村委会,，,再,拆,村长,房]
民生 故事	老爷刚走你就来，胆子真大	[老爷,刚,走,你,就,来,，,胆子,真,大]
民生 故事	前夫离婚3年至今单身，我带儿子去看他，推开门我哭成泪人	[前夫,离婚,3年,至今,单身,，,我,带,儿子,去,看,他,，,推开,门,我,哭成,泪人]
民生 故事	妻子穿宽松衣裙来离婚，刚签字总裁瞥见她孕肚喊停：不离了	[妻子,穿,宽松,衣裙,来,离婚,，,刚,签字,总裁,瞥见,她,孕,肚,喊停,：,不离,了]

TF-IDF

TF-IDF（term frequency–inverse document frequency）是一种用于资讯检索和文本挖掘的常用加权技术。 TF-IDF 是一种统计方法，用以评估一个字词对于一个文件集或一份文件对于一个语料库的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。 TF-IDF 加权的各种形式常被搜索引擎应用，作为文件与用户查询之间相关程度的度量或评级。

输入

- 输入数据集中选一列，list[string]格式的feature（也就是分词或词典过滤的结果）。

输出

- 输出数据集中新增TF-IDF结果列，类型是稀疏向量。

算子参数

参数名称	是否必选	参数描述	默认值
哈希特征数	是	对词进行哈希计算时的分桶数。建议要远大于词典长度，以防哈希碰撞 范围：[128, inf)	1048576
词语最少出现次数	是	在不同文档中出现次数少于这个数字的词语在计算IDF时会被忽略 范围：[0, inf)	0

字段参数

参数名称	是否必选	参数描述	默认值
输入列	是	需要做TF-IDF的列，类型需要是字符串数组	无

计算逻辑

- TF（Term Frequency）：

$$\text{词频}(TF) = \frac{\text{某个词在文章中的出现次数}}{\text{文章的总词数}}$$

- IDF（Inverse Document Frequency）

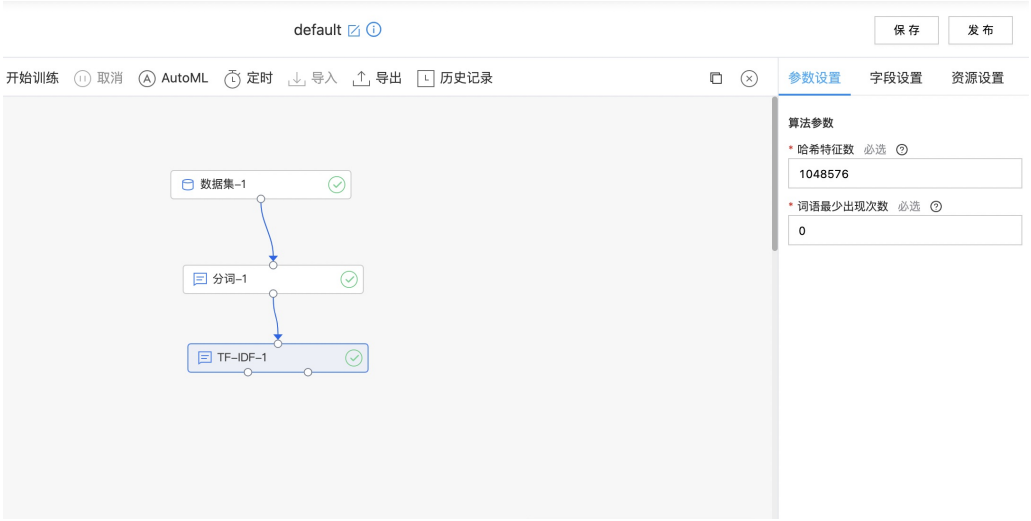
$$\text{逆文档频率}(IDF) = log(\frac{\text{语料库的文档总数}}{\text{包含该词的文档数}+1})$$

- TF-IDF

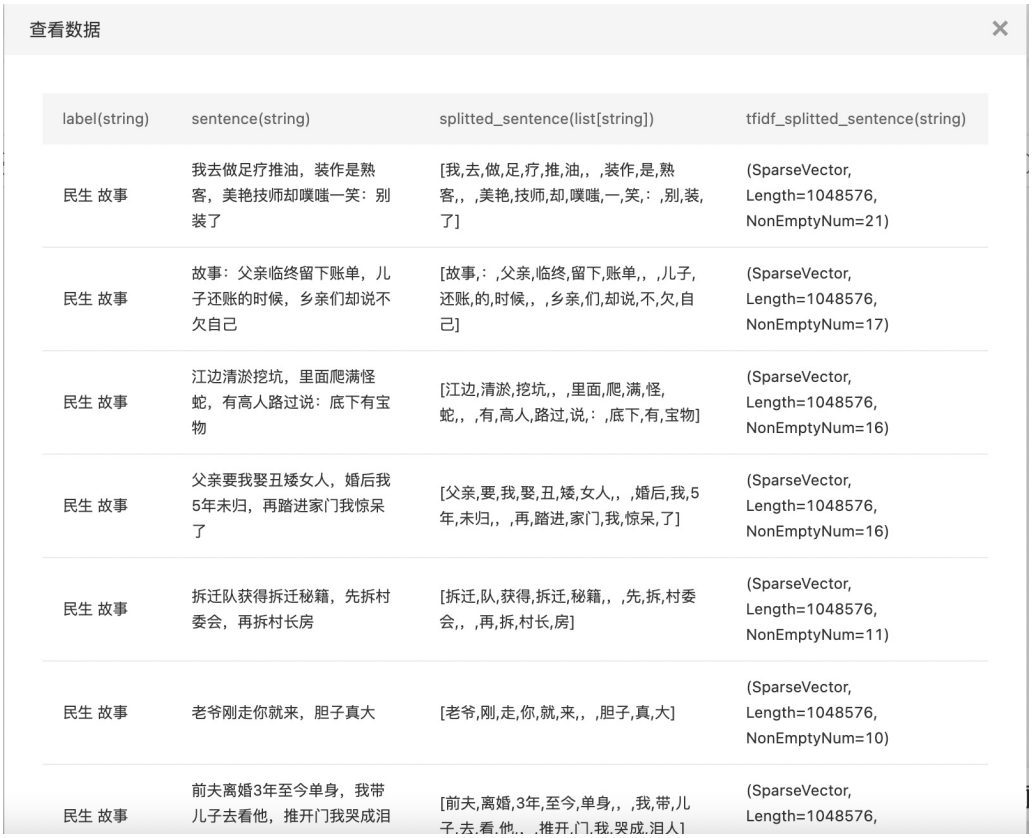
$$TF-IDF = \text{词频}(TF) \times \text{逆文档频率}(IDF)$$

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看TF-IDF处理结果。



label(string)	sentence(string)	splitted_sentence(list[string])	tfidf_splitted_sentence(string)
民生 故事	我去做足疗推油，装作是熟客，美艳技师却噗嗤一笑：别装了	[我,去,做,足,疗,推,油,，,装作,是,熟客,，,美艳,技师,却,噗嗤,一,笑,：,别,装,了]	(SparseVector, Length=1048576, NonEmptyNum=21)
民生 故事	故事：父亲临终留下账单，儿子还账的时候，乡亲们却说不欠自己	[故事,：,父亲,临终,留下,账单,，,儿子,还账,的,时候,，,乡亲,们,却说,不,欠,自己]	(SparseVector, Length=1048576, NonEmptyNum=17)
民生 故事	江边清淤挖坑，里面爬满怪蛇，有高人路过说：底下有宝物	[江边,清淤,挖坑,，,里面,爬,满,怪,蛇,，,有,高人,路过,说,：,底下,有,宝物]	(SparseVector, Length=1048576, NonEmptyNum=16)
民生 故事	父亲要我娶丑矮女人，婚后我5年未归，再踏进家门我惊呆了	[父亲,要,我,娶,丑,矮,女人,，,婚后,我,5年,未归,，,再,踏进,家门,我,惊呆,了]	(SparseVector, Length=1048576, NonEmptyNum=16)
民生 故事	拆迁队获得拆迁秘籍，先拆村委会，再拆村长房	[拆迁,队,获得,拆迁,秘籍,，,先,拆,村委会,，,再,拆,村长,房]	(SparseVector, Length=1048576, NonEmptyNum=11)
民生 故事	老爷刚走你就来，胆子真大	[老爷,刚,走,你,就,来,，,胆子,真,大]	(SparseVector, Length=1048576, NonEmptyNum=10)
民生 故事	前夫离婚3年至今单身，我带儿子去看他，推开门我哭成泪	[前夫,离婚,3年,至今,单身,，,我,带,儿子,去,看,他,，,推开,门,我,哭成,泪人]	(SparseVector, Length=1048576,

词典过滤

输入

- 选择两个输入数据集，第一个是需要过滤的数据集，第二个是词典数据集，根据选择的词典对待过滤数据集中的数据进行过滤
- 当前词典支持的过滤方式有：
    1. 白名单词典 结果集中只保留包含词典中的词
    2. 黑名单词典 结果集中去除包含词典中的词即保留词典中未出现的词
- 可选择是否保留原始列，默认为不保存
    - 如果保留，则输出结果在原始列名前添加filtered_前缀如果不保留，则替换原始列
- 词条是否为正则表达式进行过滤
    - 开启后，词典按正则表达式识别，例如“.*/v”可匹配所有结尾是/v的词。开启后过滤速度会减慢，请谨慎开启

输出

- 词典过滤处理后的数据集。

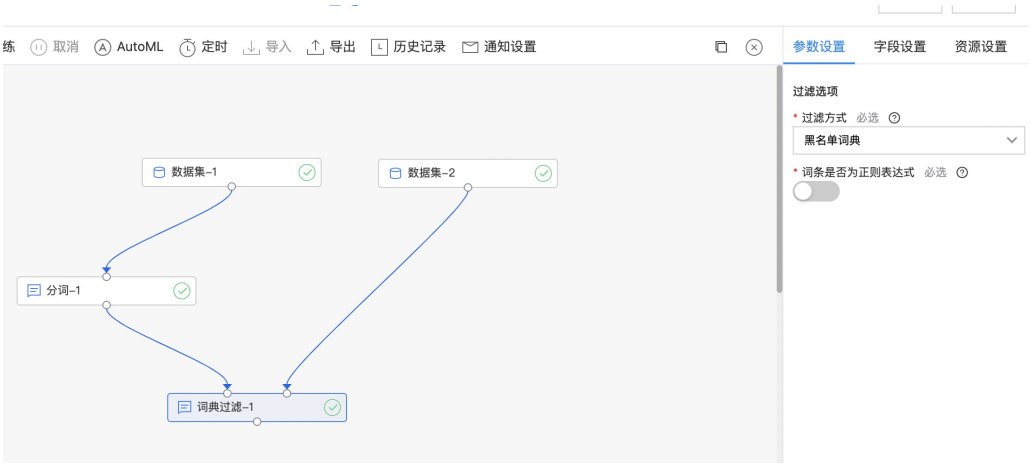
算子参数

参数名称	是否必选	参数描述	默认值
过滤方式	是	选择白名单词典，结果中只保留词典中的词；选择黑名单词典，结果中去除词典中的词	黑名单词典
词条是否为正则表达式	是	开启后，词典按正则表达式识别，例如“.*/v”可匹配所有结尾是/v的词。开启后过滤速度会减慢，请谨慎开启。	关闭

字段参数

参数名称	是否必选	参数描述	默认值
输入列	是	输入数据集中需要过滤的列，列的类型需要是字符串数组。	无
保留原始列	是	是否保留原始列。如果保留，则输出结果在原始列名前添加[filtered_]前缀如果不保留，则替换原始列	关闭
词典列	是	词典数据集中的词典列，每行一个词，列的类型需要是字符串。	无

使用示例



命名实体识别预处理

输入

- 第一个输入为文本数据集，需要选择处理特征列。
- 第二个输入是用户字典，可根据需要选择用户字典的词语列和词性列(非必选)。

输出

- 输出命名实体识别预处理。

算子参数

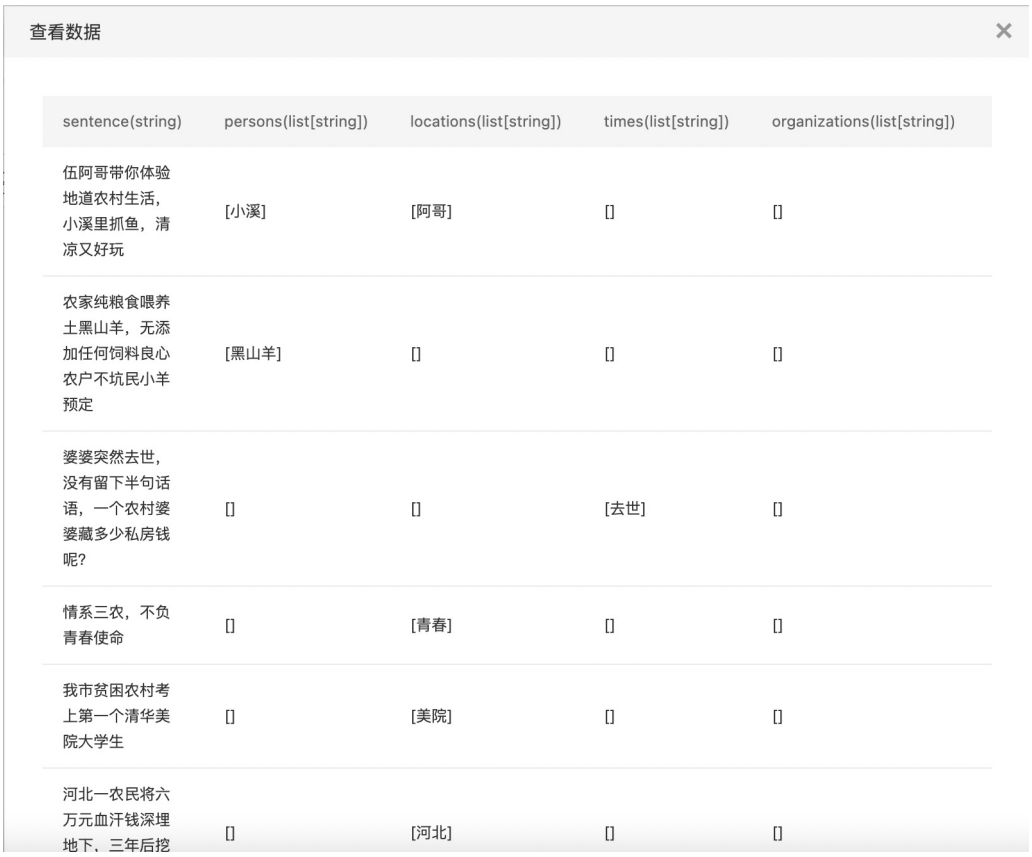
参数名称	是否必选	参数描述	默认值
是否删除重复行	是	是否删除文本数据集中的重复行。	关闭
是否使用lac模型	是	否使用lac模型，lac模型链接可参考https://github.com/baidu/lac	关闭

字段参数

参数名称	是否必选	参数描述	默认值
文本列	是	待处理的文本列，类型为str。	无
词语列	否	用户词典的词语列，类型为str。	无
词性列	否	词性标签集合可参考https://github.com/baidu/lac，类型为str。	无

使用示例

查看命名实体识别预处理结果。



图算法

图算法

FastUnfolding

FastUnfolding 算法是基于模块度对社区划分的算法。FastUnfolding 算法是一种迭代的算法，主要目标是不断划分社区使得划分后的整个网络的模块度不断增大。

输入

- 输入数据集，包括源顶点列，目标顶点列，边权值列（可选）。

输出

- 输出结果数据集，包括两列：node（源/目标顶点名称），community（社区id），community一致表示属于同一个社区。

算子参数

参数名称	是否必选	参数描述	默认值
分辨率	否	非负数值，可改变社区的大小，默认值：1.0；随着分辨率的增大，所划分的社区个数会减少 范围：[0.0, inf)。	1.0

字段参数

参数名称	是否必选	参数描述	默认值
源顶点列	是	选择源顶点列，数值或字符串类型。	无
目标顶点列	是	选择目标顶点列，数值或字符串类型。	无
边权值列	否	选择边权值列，数值类型，要求非负；若没有选择边权值列，则表示边权值均为1。	无

计算逻辑

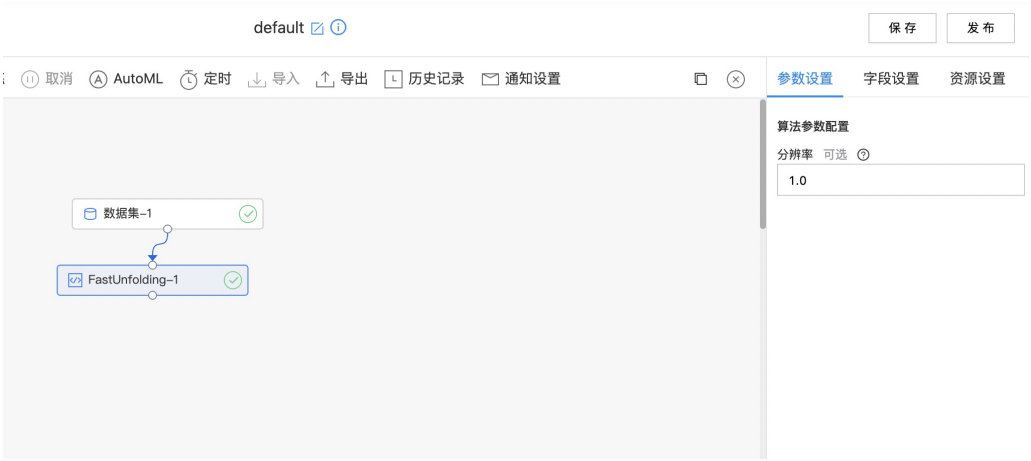
模块度成为度量社区划分优劣的重要标准，划分后的网络模块度值越大，说明社区划分的效果越好。 模块度公式简化形式:

$$Q = \sum_{c}(\frac{\sum_{in}}{2m} - (\frac{\sum_{tot}}{2m})^2)$$

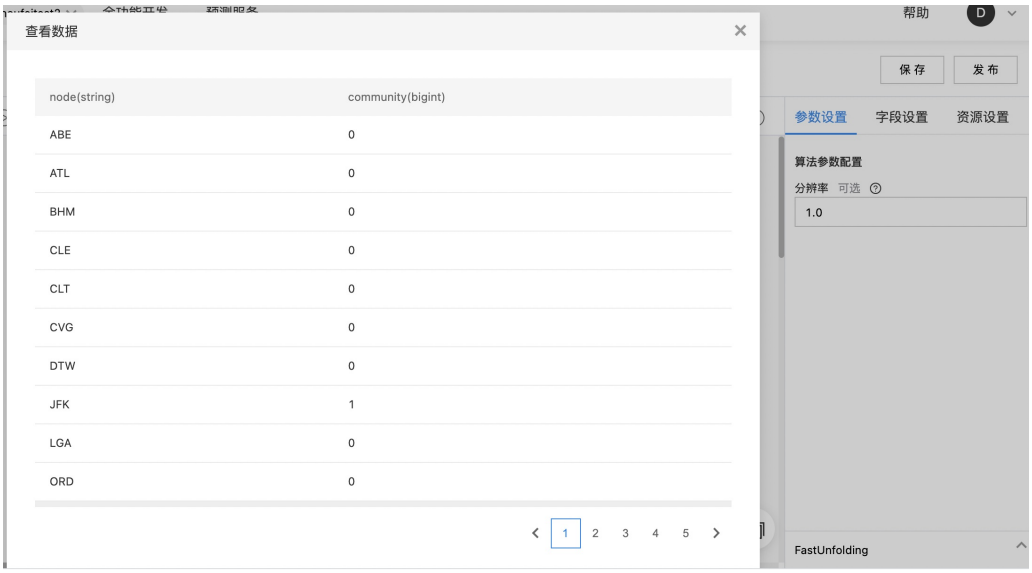
其中 $\sum_{in}$ 表示的是社区 C 内部的权重， $\sum_{tot}$ 表示的是与社区 C 内部的点连接的边的权重，包括社区内部的边以及社区外部的边

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看输出结果。



LINE

LINE也是一种基于邻域相似假设的方法，引入了两个核心概念: first-order proximity (一阶邻近) 和 second-order proximity (二阶邻近).，其中一阶邻近描述的是直接相连的节点之间的关系，而二阶邻近则指的是节点之间虽然不直接相连，但是它们会有一些共同的邻居。

输入

- 输入数据集，包括源顶点列，目标顶点列，边权值列（可选）。

输出

- 输出结果数据集，包括两列：node（源/目标顶点名称），lineVector（相似度向量）。

算子参数

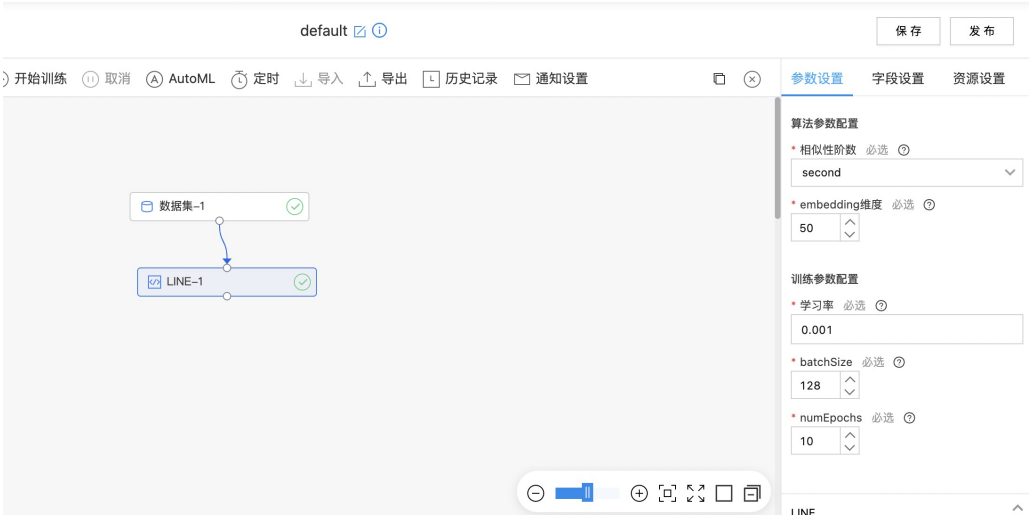
参数名称	是否必选	参数描述	默认值
相似性阶数	是	顶点之间的相似度，当设置相似度为all时，为一阶相似度和二阶相似度的拼接，输出的embedding维度为设置维度的二倍。	second
embedding维度	是	embedding维度，当设置相似度为all时，为一阶相似度和二阶相似度的拼接，输出的embedding维度为设置维度的二倍 范围：[2, inf)。	50
学习率	是	学习率 范围：[1e-05, 1.0]。	0.001
batchSize	是	训练过程中的batch_size 范围：[1, inf)。	128
numEpochs	是	训练过程中的训练轮数 范围：[1, inf)。	10

字段参数

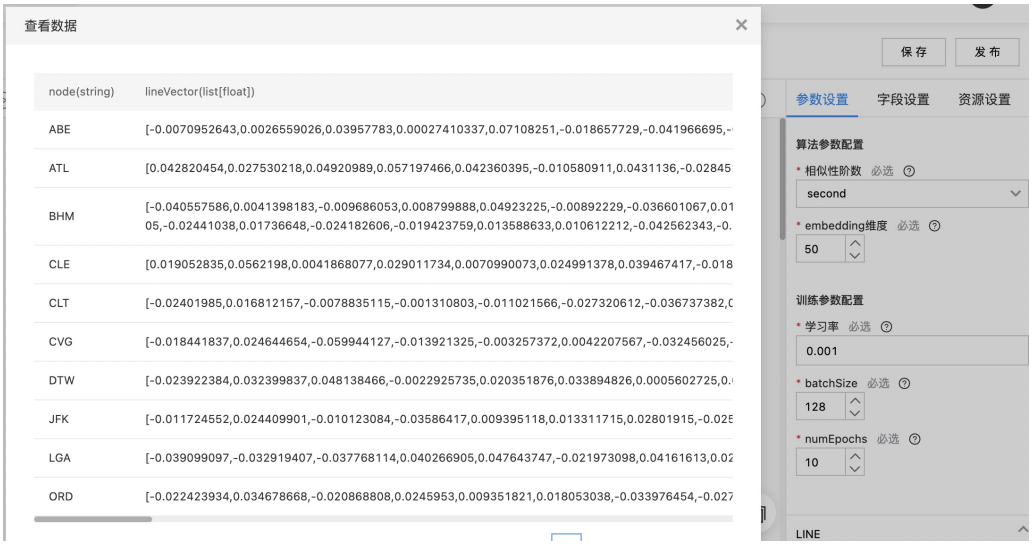
参数名称	是否必选	参数描述	默认值
源顶点列	是	选择源顶点列，数值或字符串类型。	无
目标顶点列	是	选择目标顶点列，数值或字符串类型。	无
边权值列	否	数值类型，要求非负。	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看输出结果。



LPA

LPA算法是一种基于图的半监督学习方法，其基本思路是用已标记节点的标签信息去预测未标记节点的标签信息。利用样本间的关系建立关系完全图模型，在完全图中，节点包括已标注和未标注数据，其边表示两个节点的相似度，节点的标签按相似度传递给其他节点。

输入

- 输入数据集，包括源顶点列，目标顶点列。

输出

- 输出结果数据集，包括两列：node（源/目标顶点名称），community（社区id），community一致表示属于同一个社区。

算子参数

参数名称	是否必选	参数描述	默认值
最大迭代次数	是	LPA的迭代步数 范围：[1, inf)。	5

字段参数

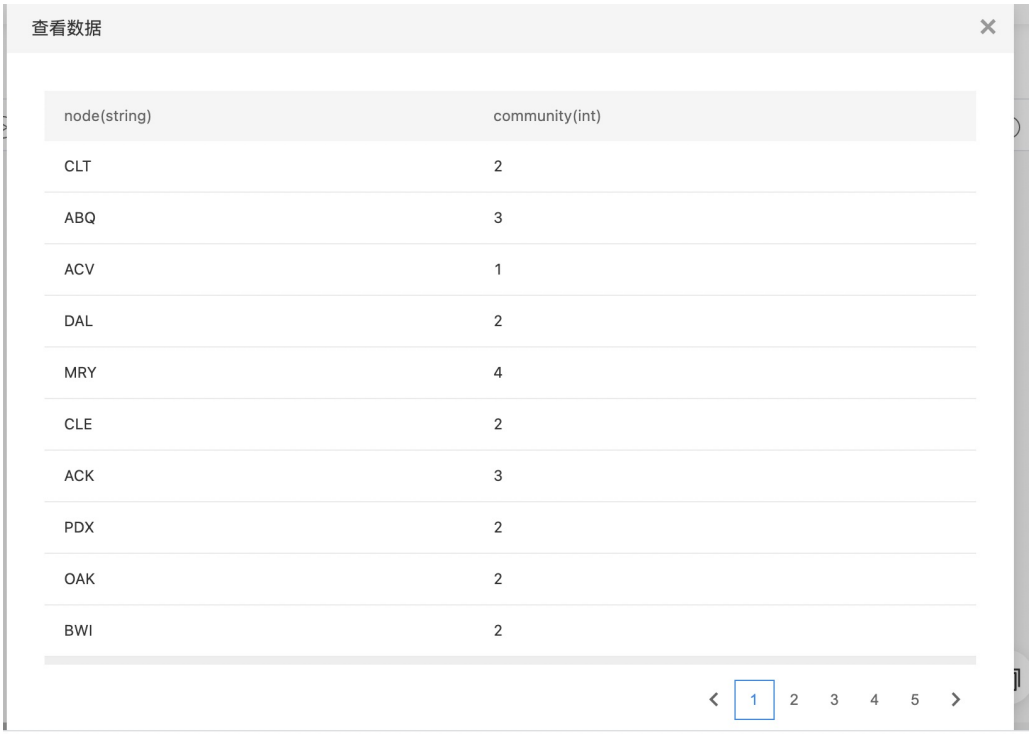
参数名称	是否必选	参数描述	默认值
源顶点列	是	边的起始节点。	无
目标顶点列	是	边的终止节点。	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看输出结果。



PageRank

以网页排序举例，PageRank算法简单来说分为两步：

1. 给每个网页一个PR值（下面用PR值指代PageRank值）

2. 通过（投票）算法不断迭代，直至达到平稳分布为止。

- 如果一个网页被很多其他网页链接到的话说明这个网页比较重要，也就是PageRank值会相对较高
- 如果一个PageRank值很高的网页链接到一个其他的网页，那么被链接到的网页的PageRank值会相应地因此而提高

输入

- 输入数据集，包括源节点列，目的节点列。

输出

- 输出PageRank处理后的结果。

算子参数

参数名称	是否必选	参数描述	默认值
最小容差	否	当本轮pr与上轮的差值的最大值小于最小容差时，迭代结束 范围：[0.0, 1.0]。	无
最大迭代次数	否	当迭代次数大于该数值时，停止迭代。未填写最小容差时，此参数有效。 范围：[1, inf)。	无
阻尼系数	是	该节点随机链接到下一个节点的概率，默认设为0.85 范围：[0.0, 1.0]。	0.85

字段参数

参数名称	是否必选	参数描述	默认值
源节点	是	边的起始节点	无
目的节点	是	边的终止节点	无

计算逻辑

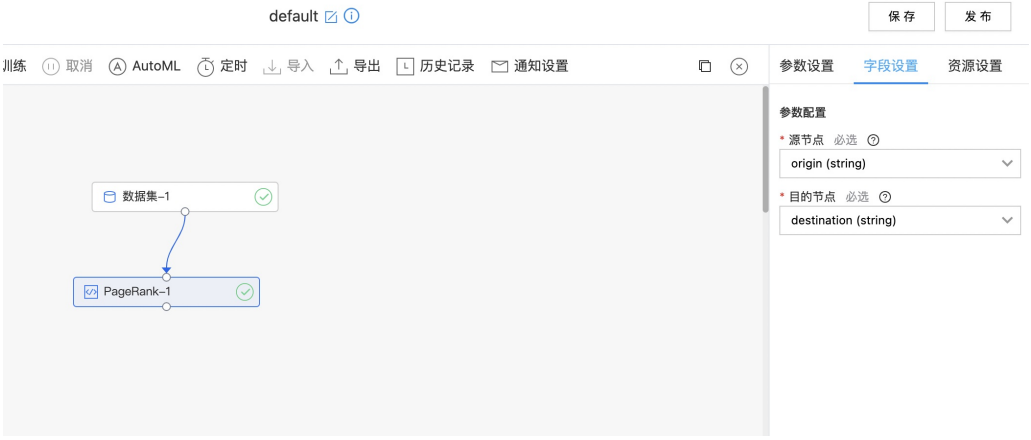
PR值计算公式：

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

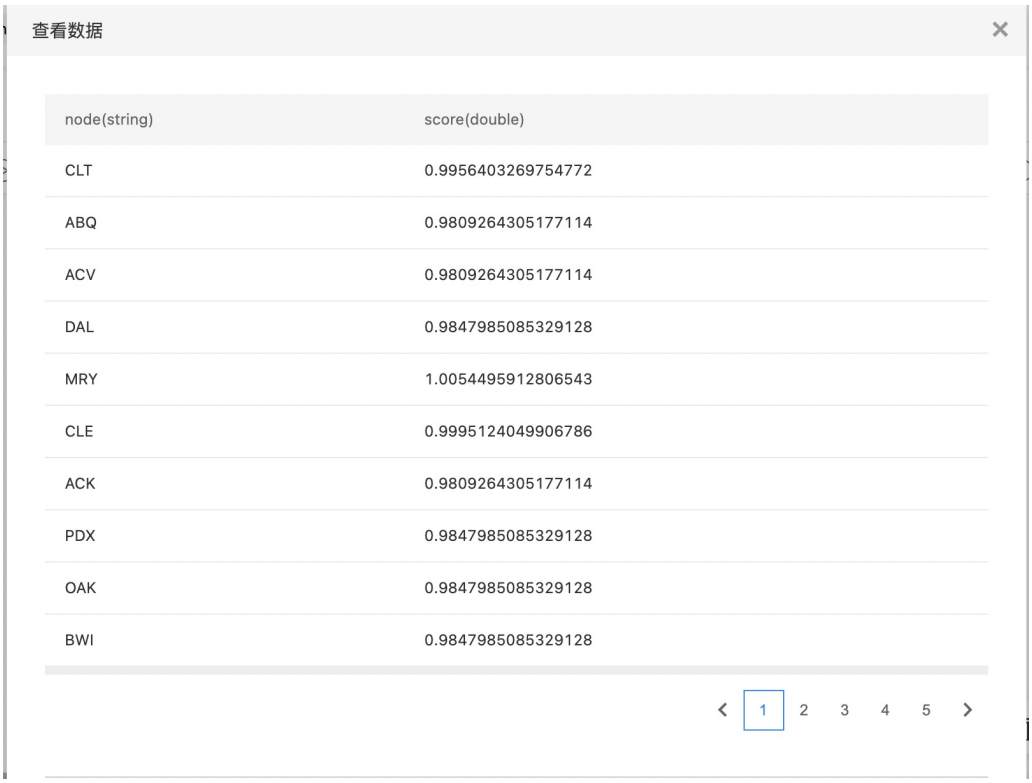
其中B_u是所有链接到网页u的网页集合，网页v是属于集合B_u的一个网页，$L(v)$则是网页v的对外链接数(即出度)

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看输出结果。



PowerIterationClustering

幂迭代聚类(Power iteration clustering，PIC) 是一个可尺度化的有效聚类算法。幂迭代算法是将数据点嵌入到由相似矩阵推导出来的低维子空间中，然后通过k-means算法得出聚类结果。幂迭代算法利用数据归一化的逐对相似度矩阵，采用截断的迭代法，寻找数据集的一个超低维嵌入，低维空间的嵌入是由拉普拉斯矩阵迭代生成的伪特征向量，这种嵌入恰好是有效的聚类指标，使他在真实的数据集上好于谱聚类算法而不需要求解矩阵的特征值。

输入

- 输入一个数据集，需要选择源顶点列，目标顶点列，边权值列 。

输出

- 输出结果数据集，包括两列：node（源/目标顶点名称），cluster(聚类结果)。

算子参数

参数名称	是否必选	参数描述	默认值
聚类中心个数	是	PIC的聚类中心个数 范围：[2, inf)。	2
向量的初始化模式	是	PIC对向量的初始化模式，分为随机初始化和度初始化。	random
最大迭代次数	是	PIC的最大迭代步数 范围：[1, inf)。	100

字段参数

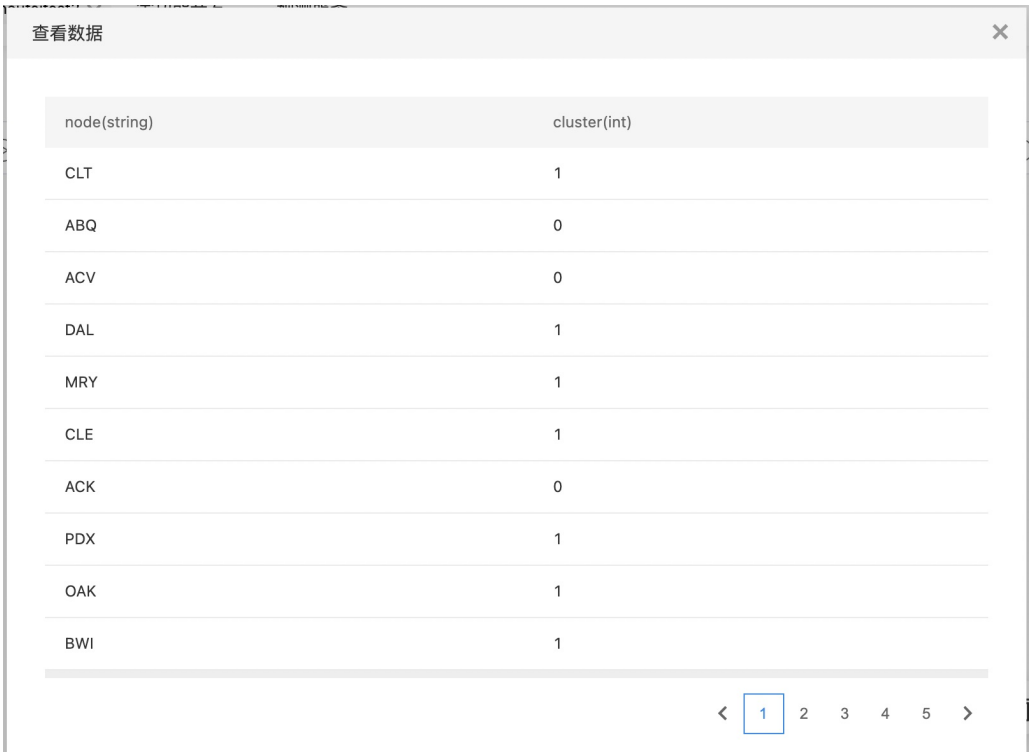
参数名称	是否必选	参数描述	默认值
源顶点列	是	边的起始节点。	无
目标顶点列	是	边的终止节点。	无
边权值列	是	边的权重列，即为点之间的相似度列，需要为数值类型的列且值应该为非负数。	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看输出结果。



统计分析组件

统计分析组件

AutoRegression

自相关函数，求解时间序列的自相关性，可以表现出模型的平稳性。

输入

- 输入是一个数据集，需要选择要做**AR**的两列。选择的检验列需要是数值类型，排序列用于排序。

输出

- 输出是一个结果数据集。包含两列：lag，correlation，列的类型分别是bigint和double。

算子参数

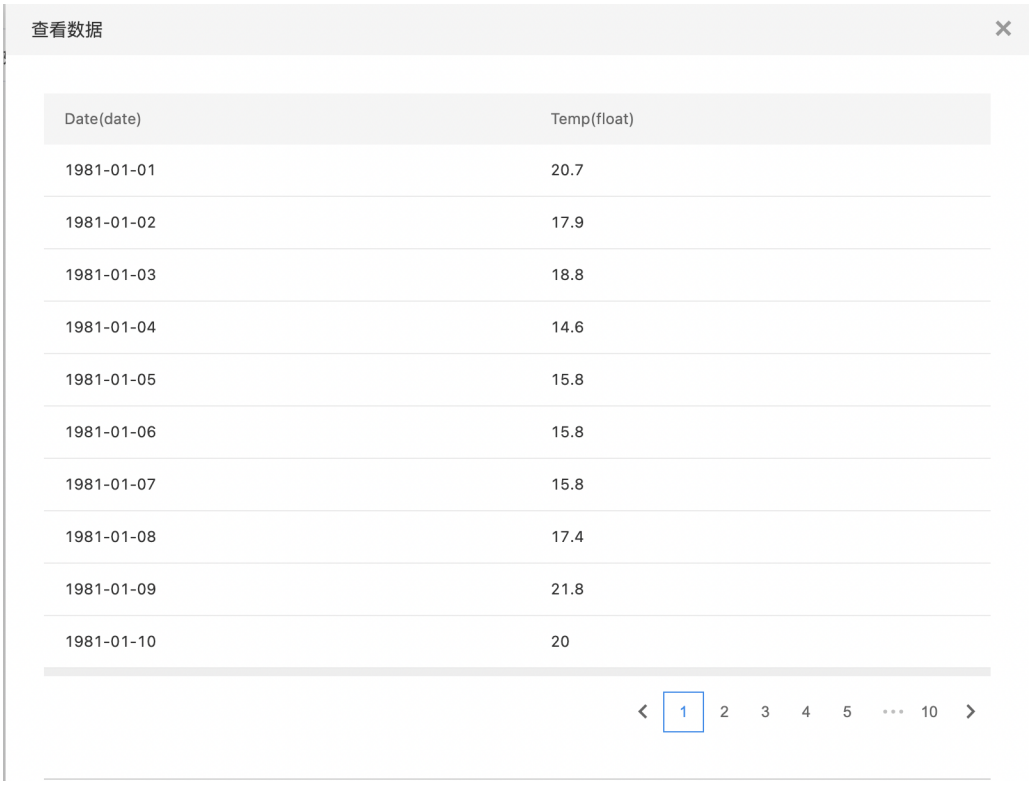
参数名称	是否必选	参数描述	默认值
最大lag	是	统计时考虑的最大lag 范围：[1, 3653]。	100

字段参数

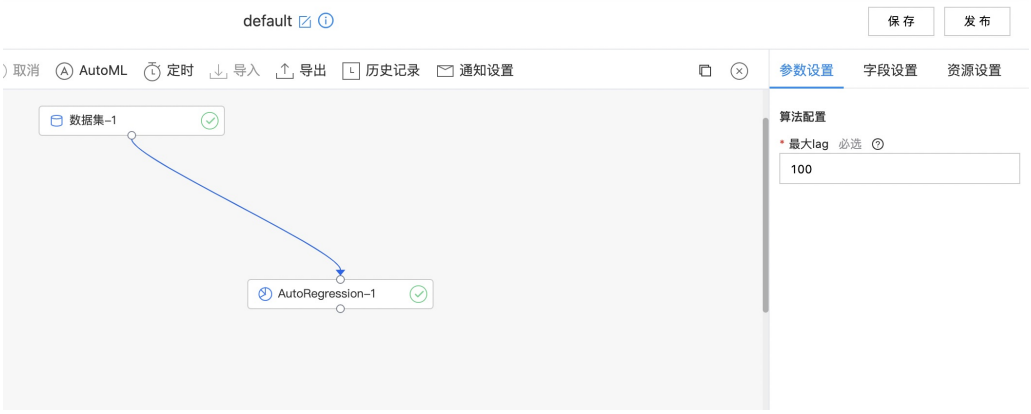
参数名称	是否必选	参数描述	默认值
检验列	是	选择一列检验列，数值类型。	无
排序列	是	AR检验列依据此列由小到大排序，一般选择日期列。	无

使用示例

1. 输入数据集为天气数据，Date代表日期，Temp代表温度。



2. 构建算子结构，配置参数，完成训练。



3. 查看输出数据。

查看数据

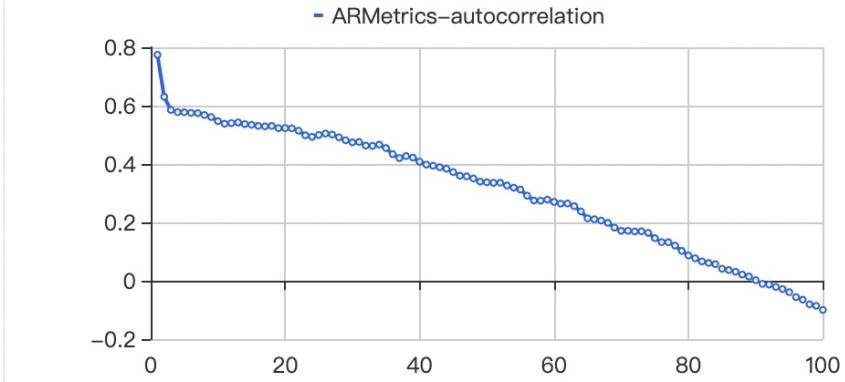
lag(bigint)	correlation(double)
1	0.7742680578193195
2	0.6302866508281394
3	0.5852931552303943
4	0.5777456854906504
5	0.5772801629548401
6	0.5751041517895307
7	0.5743704211242118
8	0.5678263106364088
9	0.5612012986593726
10	0.546686938741883

< 1 2 3 4 5 >

4. 查看分析报告结果。

分析报告

AutoRegression



F检验

F检验（F-test），最常用的别名叫做联合假设检验（英语：joint hypotheses test），此外也称方差比率检验、方差齐性检验。它是一种在零假设（null hypothesis, H0）之下，统计值服从 F-分布 的检验。其通常是用来分析用了超过一个参数的统计模型，以判断该模型中的全部或一部分参数是否适合用来估计母体。

输入

- 输入是一个数据集，需要选择要做**F检验**的一列。选择的列需要是数值类型。进一步地，还需要选择**F检验**对应的标签列。标签列要求是整数或字符串类型。

输出

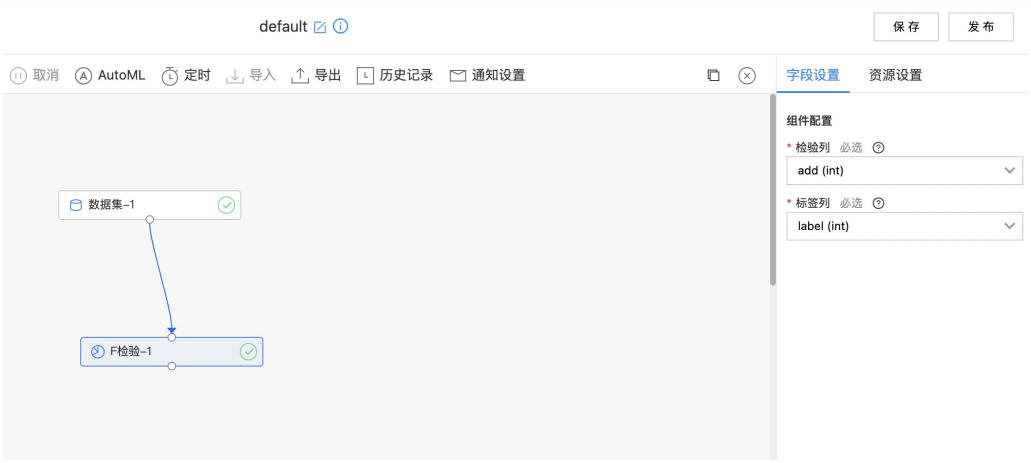
- 输出是一个结果数据集。包含两列：f值（f_value），置信度（p_value）。列的类型是double。

字段参数

参数名称	是否必选	参数描述	默认值
检验列	是	选择一列检验列，数值类型。	无
排序列	是	分组依据，要求是整数或字符串类型；不能存在nan(缺失值)。	无

使用示例

1.构建算子结构，配置参数，完成训练。



2.查看输出数据集结果。



KS检验

KS 检验（Kolmogorov-Smirnov 检验）是基于累计分布函数的，用于检验一个分布是否符合某种理论分布或比较两个经验分布是否有显著差异。系统中实现的为比较两个经验分布是否有显著差异。原假设为两个样本的分布相同，如果 ks 统计量小或者 p 值高，则我们不能拒绝原假设。

输入

- 输入是一个数据集。用户需要选择要做**KS检验**的两列（必须为两列），选择的列需要是数值类型。

输出

- 输出是一个结果数据集。包含两列：KS值（statistic），置信度（pValue），列的类型是double。

算子参数

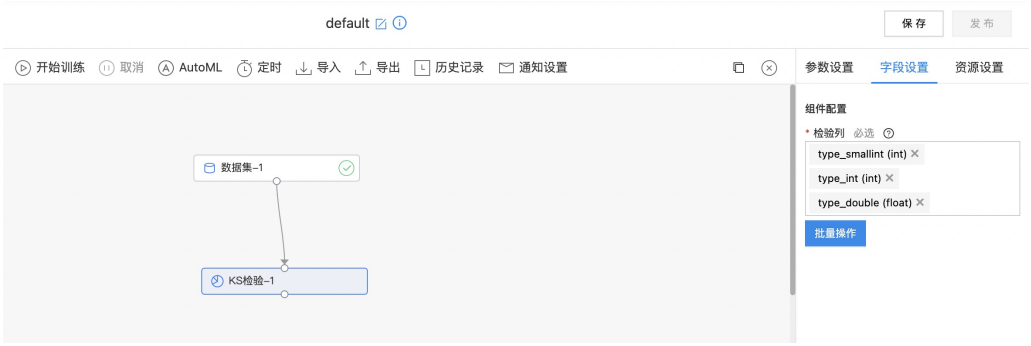
参数名称	是否必选	参数描述	默认值
替代方式	否	替代假设方式： two-sided greater less	two-sided
计算模式	否	计算p值的方式： auto asym exact	auto

字段参数

参数名称	是否必选	参数描述	默认值
检验列	是	必须是两列，并且是数值类型。	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看输出结果。



单样本T检验

T 检验，亦称 student T 检验（Student's T test），主要用于样本含量较小（例如 n < 30），总体标准差 σ 未知的正态分布。T 检验是用 T 分布理论来推论差异发生的概率，从而比较两个平均数的差异是否显著。 单样本 T 检验是检验一个样本平均数与一个已知的总体平均数的差异是否显著。

输入

- 输入是一个数据集。用户需要选择要做**单样本T检验**的一列。选择的列需要是数值类型。进一步地，还需要输入总体均值（零假设中的期望值）。

输出

- 输出是一个结果数据集。包含两列：t值（t_statistic），置信度（p_value），列的类型是double。

算子参数

参数名称	是否必选	参数描述	默认值
nan政策	否	当输入包含nan时如何处理： “propagate”（默认）返回nan “raise”引发错误 “omit”忽略nan值计算	propagate

字段参数

参数名称	是否必选	参数描述	默认值
检验列	是	选择一列检验列，数值类型。	无
总体均值	是	零假设中的期望值。	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看输出结果。



PSI检验

群体稳定性指标 PSI（Population Stability Index）是衡量模型的预测值与实际值偏差大小的指标。

输入

- 输入是一个数据集。用户需要选择要做PSI检验列。选择的列需要是数值类型，其中包括实际值列和预测值列。

输出

- 输出是一个结果数据集。包含一列：稳定度指标（psi），列的类型是double。

算子参数

参数名称	是否必选	参数描述	默认值
分箱方式	是	计算PSI分箱方式支持： bins 均匀分箱 quantiles 分位数分箱	bins
分箱个数	是	计算PSI时分箱个数	10

字段参数

参数名称	是否必选	参数描述	默认值
实际值列	是	实际值，要求是数值类型	无
预测值列	是	预测值，要求是数值类型	无

计算逻辑

PSI计算公式：

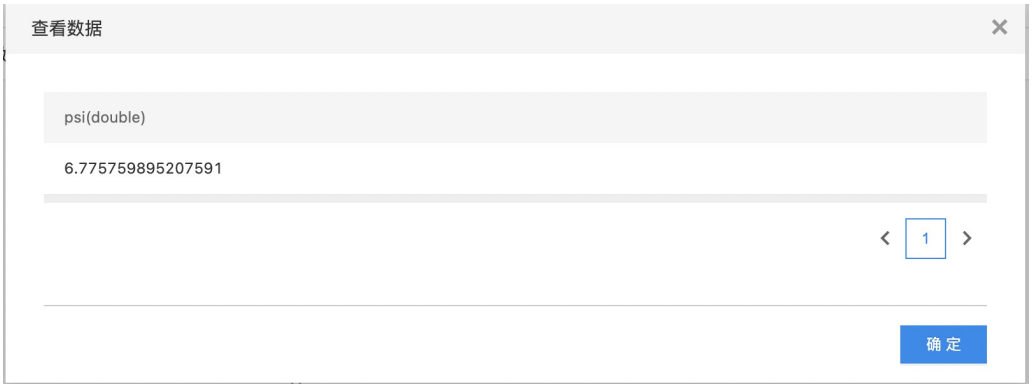
$$psi = sum((_{\text{实际占比}} - _{\text{预期占比}}) * ln(_{\text{实际占比}}/_{\text{预期占比}})).$$

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看输出结果。



双样本T检验

双样本 T 检验利用 T 分布理论来检验两个总体均值是否显著差异。

输入

- 输入是一个数据集。用户需要选择要做**双样本T检验**的两列。选择的列需要是数值类型。

输出

- 输出是一个结果数据集。包含两列：t值（t_statistic），置信度（p_value），列的类型是double。

算子参数

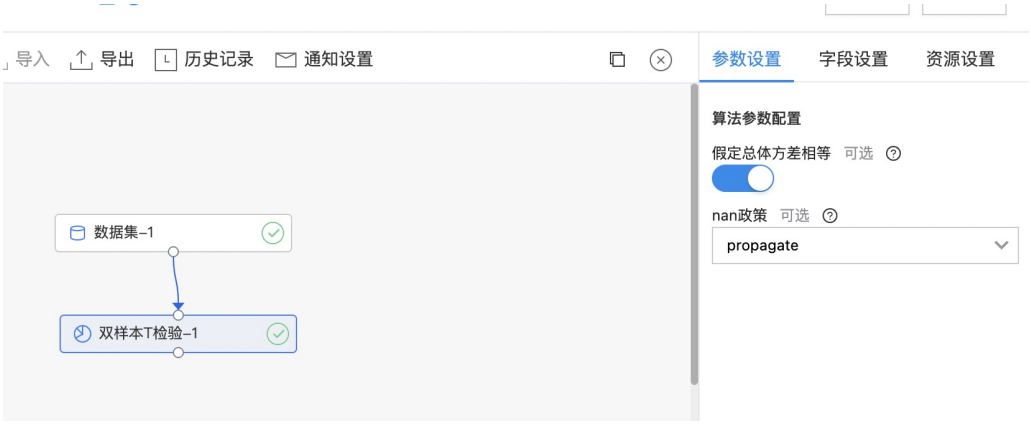
参数名称	是否必选	参数描述	默认值
假定总体方差相等	否	如果为True（默认），则执行一个标准的独立2样本t检验，该检验假定总体方差相等。如果为False，则执行Welch的t检验，该检验不假定总体方差相等。	开启
nan政策	否	当输入包含nan时如何处理： “propagate”（默认）返回nan “raise”引发错误 “omit”忽略nan值计算	propagate

字段参数

参数名称	是否必选	参数描述	默认值
检验列	是	必须是两列，并且是数值类型。	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看输出结果。



U检验

U 检验是一种用来评估两个独立的顺序数据样本是否来自同一个总体的非参数检验。系统中实现的为 Mann-Whitney U 检验（曼-惠特尼 U 检验），它假设两个样本分别来自除了总体均值以外完全相同的两个总体，目的是检验这两个总体的均值是否有显著的差别。

输入

- 输入是一个数据集。用户需要选择要做**U检验**的两列（必须为两列），选择的列需要是数值类型。

输出

- 输出是一个结果数据集。包含两列：U值（u_statistic），置信度（p_value），列的类型是double。

算子参数

参数名称	是否必选	参数描述	默认值
替代方式	是	替代假设方式： two-sided greater less	two-sided
连续性校正	否	是否考虑连续性校正	开启

字段参数

参数名称	是否必选	参数描述	默认值
检验列	是	必须是两列，并且是数值类型。	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看输出结果。



卡方拟合性检验

卡方拟合性检验用于检验单个多项分类名义型变量各分类间的实际观测次数与理论次数之间是否一致的问题，其中实际观测次数是根据样本数据得到的实计数，理论次数则是根据理论或经验得到的期望次数。 卡方拟合性检验的零假设是观测次数与理论次数之间无差异。

输入

- 输入是一个数据集。用户需要选择要做**卡方拟合性检验**的一列，选择的列需要是数值类型。

输出

- 输出是一个结果数据集。包含三列：卡方值，自由度，置信度，列的类型是double。

字段参数

参数名称	是否必选	参数描述	默认值
检验列	是	选择一列检验列，必须是数值类行。	无

计算逻辑

卡方拟合性检验统计量：

卡方拟合性检验统计量:

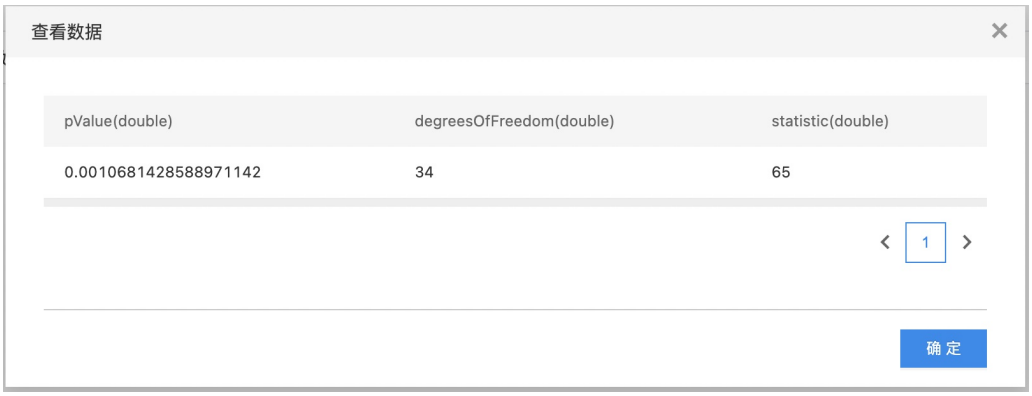
$$\chi^2 = \sum \frac{(A-T)^2}{T}, \text{其中} A \text{为实际观测次数}, T \text{为理论次数}$$

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看输出结果。



卡方独立性检验

卡方独立性检验用于检验两个或两个以上因素（各有两项或以上的分类）之间是否相互影响的问题。所谓独立，即无关联，互不影响，就意味着一个因素各个分类之间的比例关系，在另一个因素的各项分类下都是相同的。 卡方独立性检验的零假设是各因素之间相互独立。 卡方独立性检验统计量与卡方拟合性检验统计量计算公式一致，统称为卡方检验统计量。

输入

- 输入是一个数据集。用户需要选择要做**卡方独立性检验**的两列（必须是两列），选择的列需要是数值类型。

输出

- 输出是一个结果数据集。包含三列：卡方值，自由度，置信度，列的类型是double。

字段参数

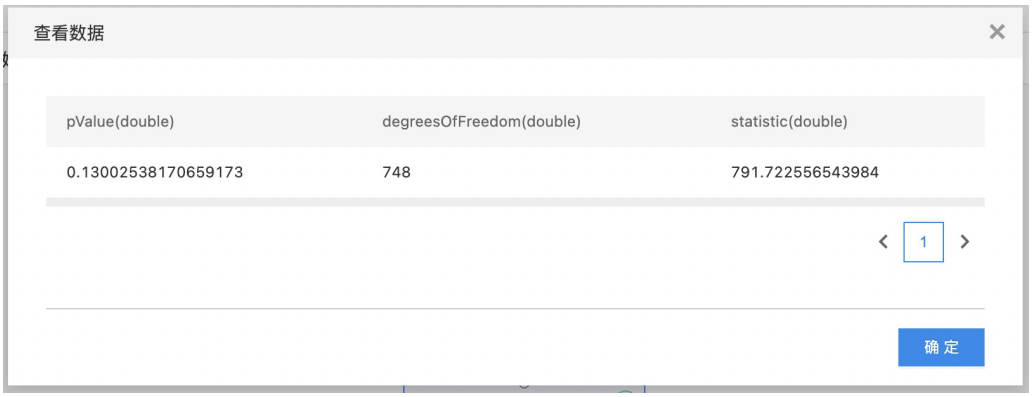
参数名称	是否必选	参数描述	默认值
检验列	是	必须是两列，并且是数值类型。	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看训练结果。



相关系数矩阵

相关系数矩阵是由矩阵各列对应的变量间的相关系数构成的。也就是说，相关系数矩阵的第 i 行第 j 列元素是矩阵第 i 列和第 j 列对应变量的相关系数。组件支持皮尔森（pearson）相关系数和斯皮尔曼（spearman）相关系数。

输入

- 输入是一个数据集，选择任意几列数据，所有选择的列都必须是数字类型。

输出

- 输出数据之间的相关系数矩阵。

字段参数

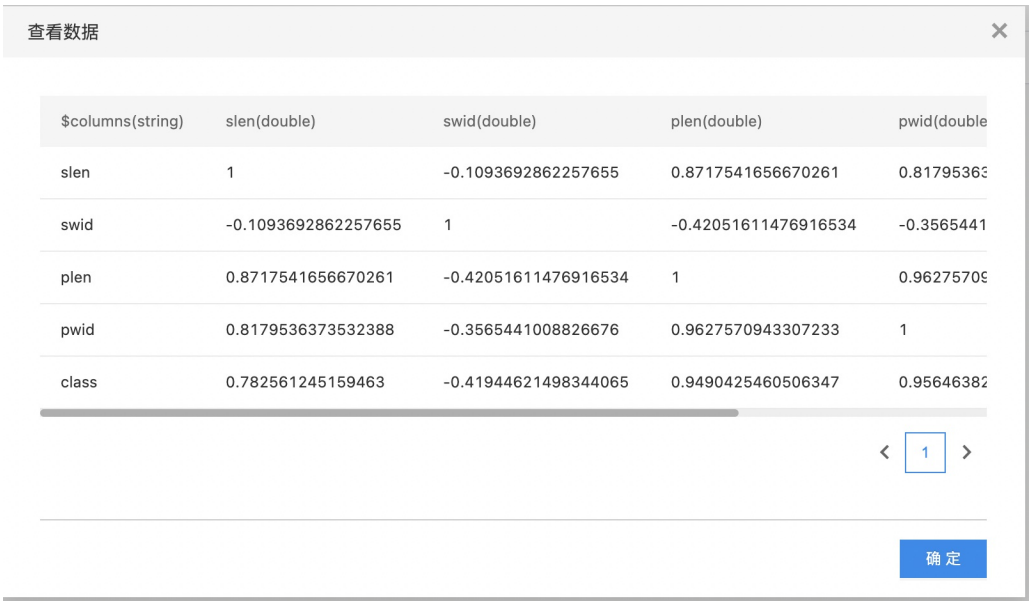
参数名称	是否必选	参数描述	默认值
计算方法	是	相关系数的计算方法： pearson spearman	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看训练结果。



协方差

协方差在概率论和统计学中用于衡量两个变量的总体误差。而方差是协方差的一种特殊情况，即当两个变量是相同的情况。 如果协方差为正，说明两个变量同向变化，协方差越大说明同向程度越高；如果协方差为负，说明两个变量反向变化，协方差越小说明反向程度越高。

输入

- 输入是一个数据集。用户需要选择要计算**协方差**的两列（必须是两列），选择的列需要是数值类型。

输出

- 输出协方差计算结果。

字段参数

参数名称	是否必选	参数描述	默认值
选择两列数据	是	必须是数值类型。	无

计算逻辑

协方差计算函数：

$$cov(x,y) = E(XY) - E(X)E(Y)$$

因为我们是统计样本数据,所以应该使用样本系的方差,计算公式:

$$cov(x,y) = E(XY) - E(X)E(Y) * n/(n-1)$$

其中$E[\cdot]$表示变量的期望

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看输出数据。



直方图

在统计学中，直方图是一种对数据分布情况的图形表示，是一种二维统计图表，它的两个坐标分别是统计样本和该样本对应的某个属性的度量。

输入

- 输入是一个数据集，选择任意几列需要绘制直方图的数据，数据必须是数字类型。

输出

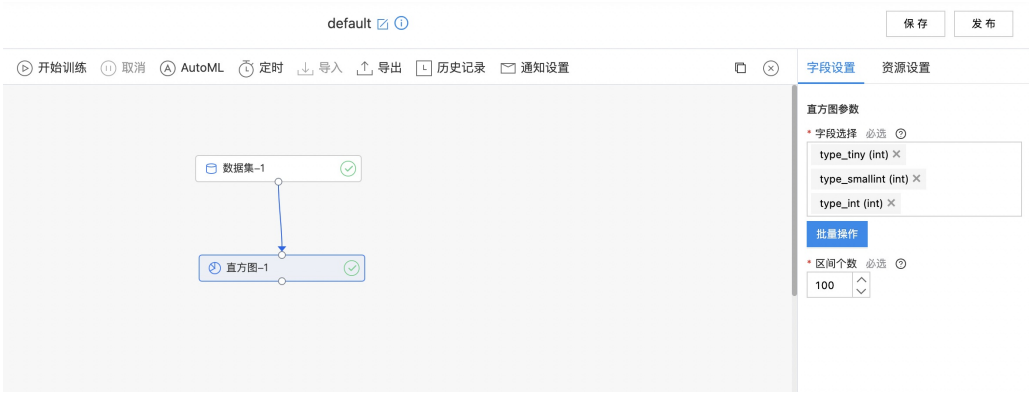
- 所有选择列的直方图数据
  - 格式: start0,end0,count0;start1,end1,count1;.....;start(n-1),end(n-1),count(n-1)
  - start,end: 区间的最小值和最大值
  - count:区间的数据个数
- 右键查看分析报告展示所选数据的直方图。

字段参数

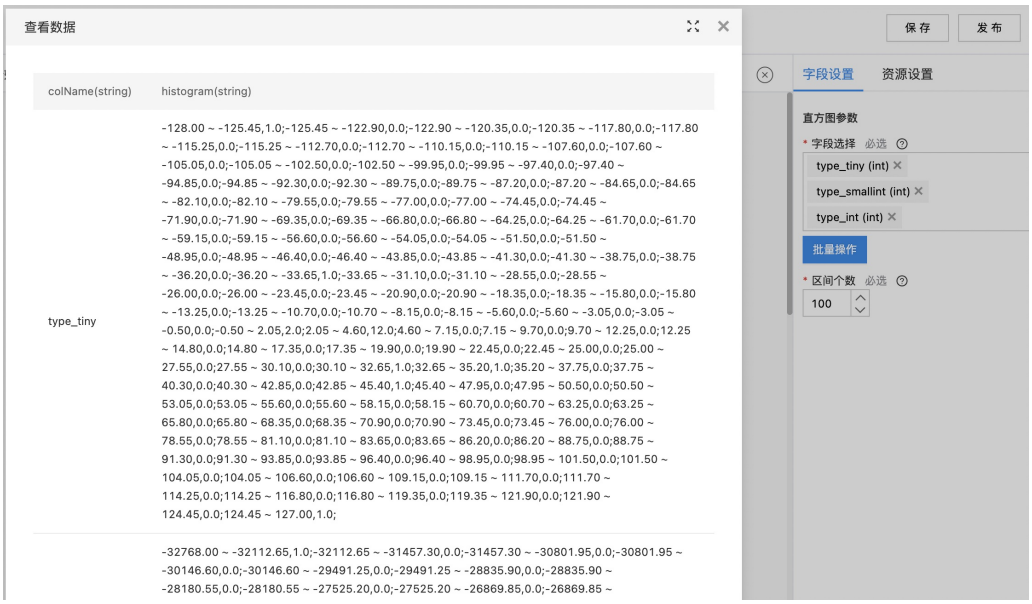
参数名称	是否必选	参数描述	默认值
字段选择	是	需要是数值类型。	无
区间个数	是	配置区间个数，范围：[2, inf)。	100

使用示例

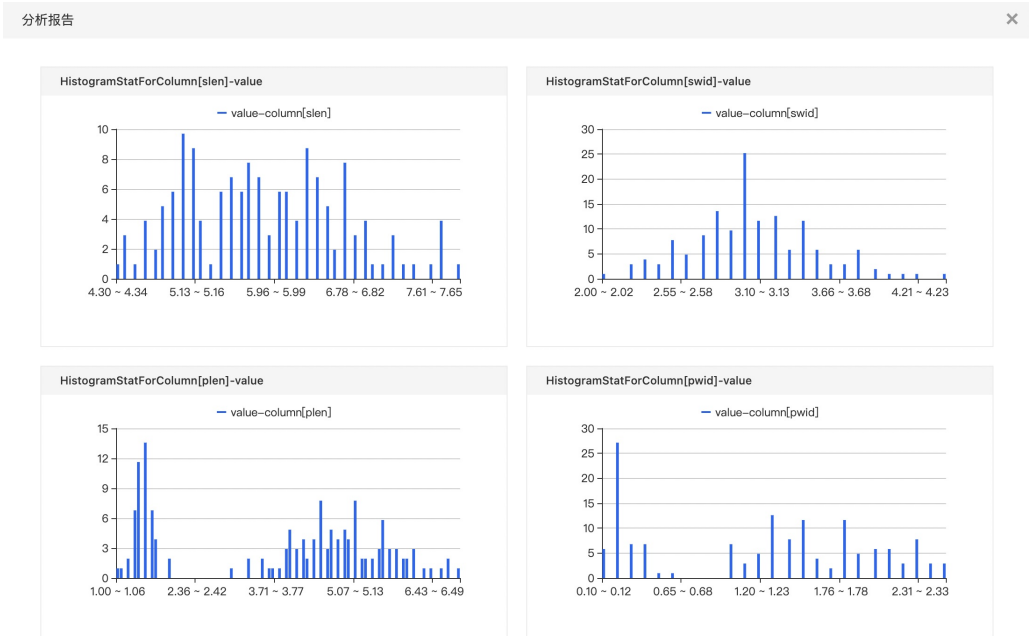
1. 构建算子结构，配置参数，完成训练。



2. 查看输出数据。



3. 查看分析报告结果。



洛伦兹曲线

洛伦兹曲线研究的是国民收入在国民之间的分配问题，它用以比较和分析一个国家在不同时代或者不同国家在同一时代的财富不平等状况。 洛伦兹曲线的横轴表示人口（按收入由低到高分组）的累积百分比，纵轴表示收入的累积百分比。 洛伦兹曲线的弯曲程度有重要意义。一般来讲，它反映了收入分配的不平等程度。弯曲程度越大，收入分配越不平等，反之亦然。

输入

- 输入是一个数据集，选择任意几列需要绘制洛伦兹曲线的数据，配置区间个数。

输出

- 输出洛伦兹曲线对应值与绘制图标。

字段参数

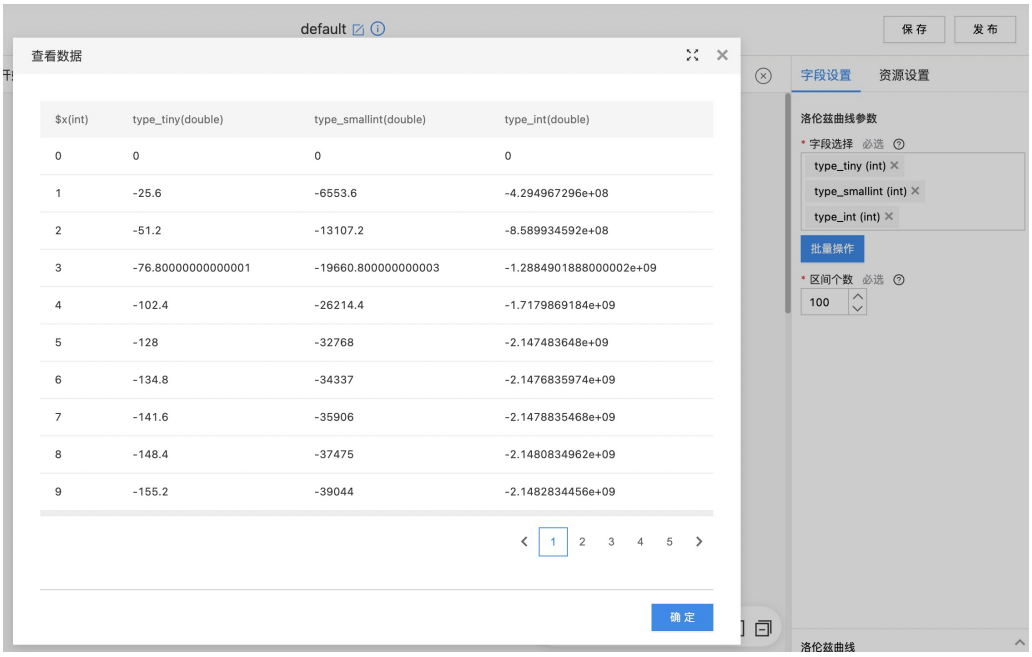
参数名称	是否必选	参数描述	默认值
字段选择	是	需要是数值类型。	无
区间个数	是	配置区间个数，范围：[2, inf)。	100

使用示例

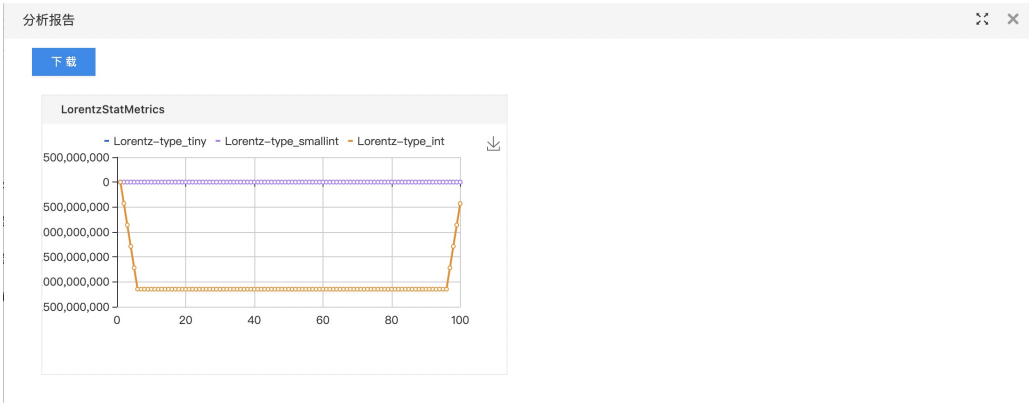
1. 构建算子结构，配置参数，完成训练。



2. 查看输出数据。



3. 查看分析报告。



正态检验

利用观测数据判断总体是否服从正态分布的检验称为正态检验，它是统计判决中重要的一种特殊的拟合优度假设检验。常用的正态检验方法有正态概率纸、柯尔莫可洛夫-斯米洛夫检验等。

输入

- 输入是一个数据集，需要制定进行**正态检验**的数据列，列类型是数值类型。

输出

- 输出正态检验的置信度和kstest值。

字段参数

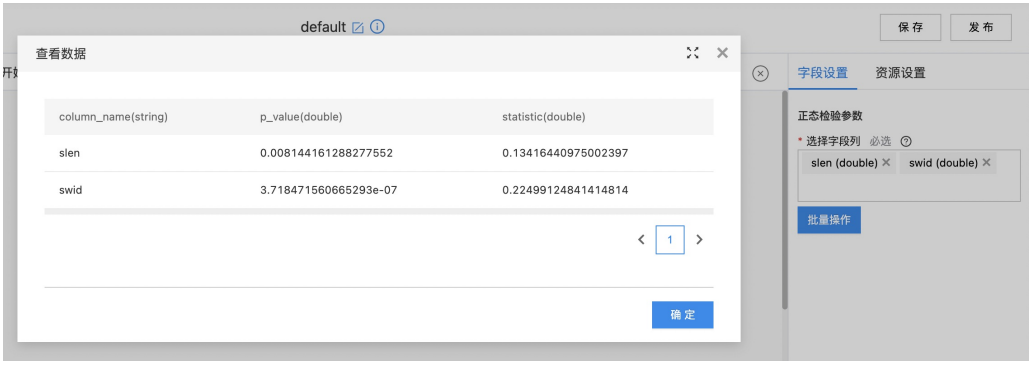
参数名称	是否必选	参数描述	默认值
选择字段列	是	需要是数值类型。	无

使用示例

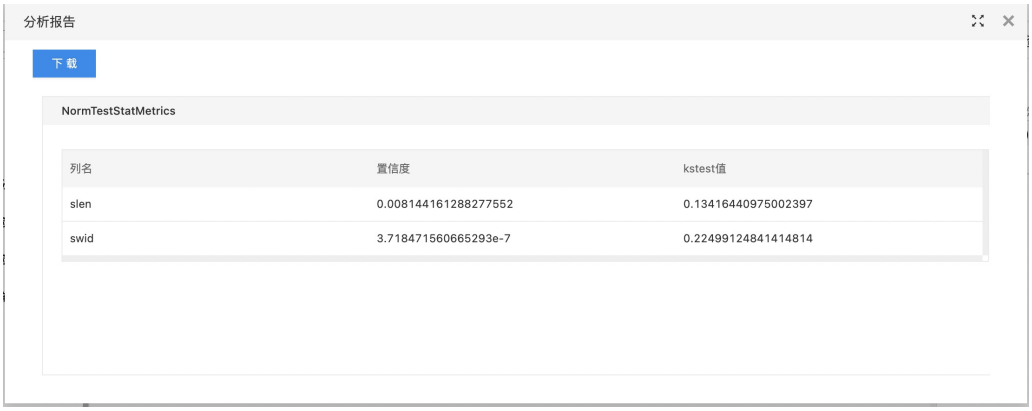
1. 构建算子结构，配置参数，完成训练。



2. 查看输出结果。



3. 查看分析报告。



皮尔森系数

柯尔莫可洛夫-斯米洛夫检验基于累计分布函数，用以检验两个经验分布是否不同或一个经验分布与另一个理想分布是否不同。 K-S检验的基本思路是：先将理论累积频率分布与观测的经验累积频率分布加以比较，求出它们最大的偏离值，然后在给定的显著性水平上检验这种偏离值是否是偶然出现的。

输入

- 输入是一个数据集，需要选择计算皮尔森系数的两列数据，必须是数值类型。

输出

- 输出是计算后的皮尔森系数结果。

字段参数

参数名称	是否必选	参数描述	默认值
选择两列数据	是	必须是数值类型。	无

计算逻辑

皮尔森系数计算公式:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

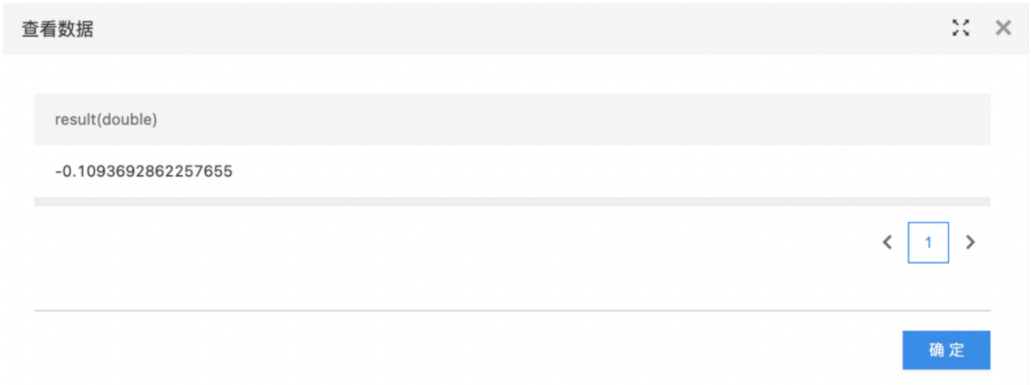
其中$cov(X,Y)$表示X与Y的协方差，σ_X表示X的标准差，σ_Y表示Y的标准差

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 输出皮尔森系数计算结果。



全表统计

统计各种指标信息，包括总数、NAN、正无穷数量、负无穷数量、标准误差、偏度、峰度、二阶矩、三阶矩、四阶矩、二阶中心距、三阶中心距、四阶中心距、总和等。

输入

- 输入是一个数据集，选择需要进行统计的数据列，仅会统计数值类型的列。

输出

- 输出已选择数据列的统计结果。

字段参数

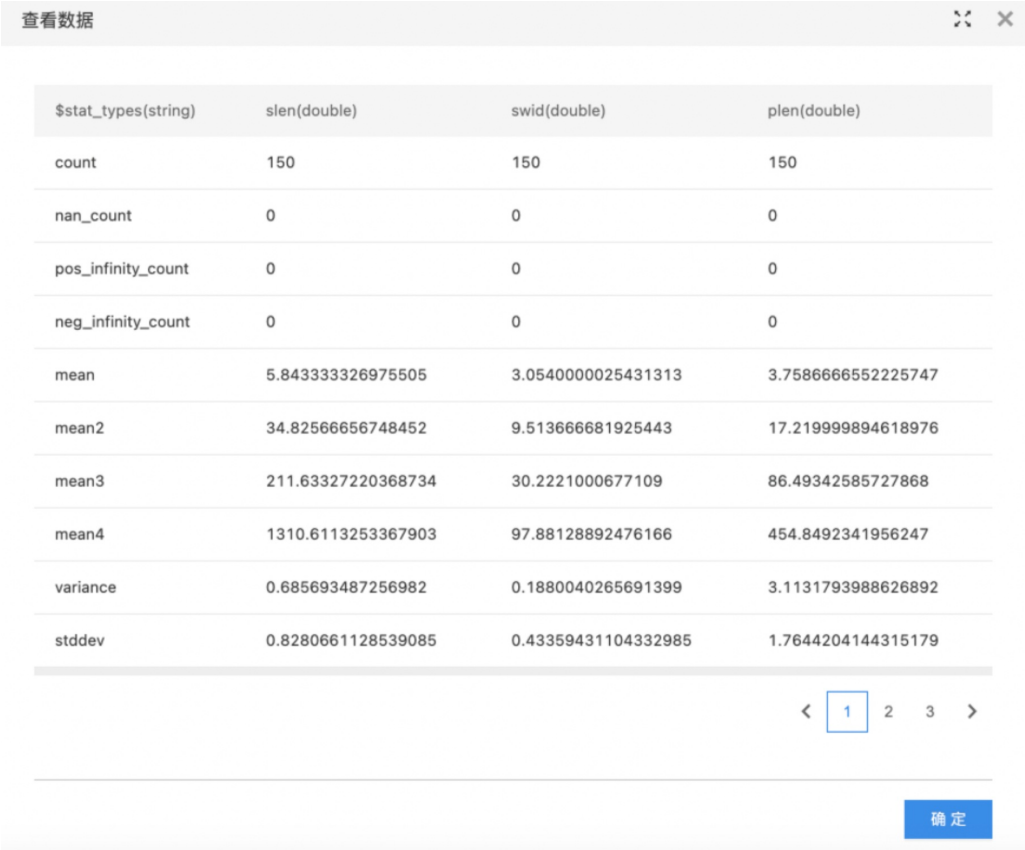
参数名称	是否必选	参数描述	默认值
选择需要统计的列	是	仅会统计数值类型的列。	无

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 右键“全表统计"组件，选择“查看数据” > "输出数据集"，查看输出结果。



预测组件

预测组件

特征工程预测

特征工程预测是专门用于特征工程组件模型预测的组件，特征工程组件中生成模型的算法都可以采用该组件进行预测操作。

输入

- 输入Python 模型和预测数据集。

输出

- 输出Python模型预测结果数据集。

预测组件

预测组件是专门用于算法组件模型预测的组件，算法组件产出的模型可以采用该组件进行预测操作。

输入

- 输入Python 模型和预测数据集。

输出

- 输出Python模型预测结果数据集。

Python特征工程预测(单机)

Python特征工程预测(单机)组件用于Python 特征工程组件的预测，只支持单机运行。

输入

- 输入Python 模型和预测数据集。

输出

- 输出Python模型预测结果数据集。

Python特征工程预测(分布式)

Python特征工程预测(分布式)组件用于Python 特征工程组件的预测，支持多机运行。

输入

- 输入Python 模型和预测数据集。

输出

- 输出Python模型预测结果数据集。

Python模型预测(单机)

Python 模型预测(单机)组件用于python 模型组件的预测，只支持单机运行。

输入

- 输入Python 模型和预测数据集。

输出

- 输出Python模型预测结果数据集。

Python模型预测(分布式)

Python 模型预测(分布式)组件用于python 模型组件的预测，支持多小·小·机运行。

输入

- 输入Python 模型和预测数据集。

输出

- 输出Python模型预测结果数据集。

模型评估组件

模型评估组件

二分类评估

评估模块支持计算 AUC、KS 及 F1 score，同时输出数据用于画 PR 曲线、ROC 曲线、KS 曲线、LIFT chart、Gain chart，同时也支持分组评估。

输入

- 最多可输入4个数据集，用户需要选择原始标签列、预测标签列和正样本标签值，还可以提供scoreColumn获得更多指标。

输出

- 第一个输出是summary数据集，第二个输出是metricsByThreshold数据集，运行后在组件的右键菜单可查看图形化的衡量指标。
- summary数据集：只有一行，每一列是一个评价指标，包括以下指标：accuracy、precision、recall、FPR、f1score、auc（roc曲线下的面积。如果参数未提供scoreColumn则没有此项指标）、area under pr（pr曲线下的面积。如果参数未提供scoreColumn则没有此项指标）。
- metricsByThreshold数据集：描述了阈值变化时fp率、precision、recall、f1score的变化。如果参数未提供scoreColumn则此数据集为空。
- 右键菜单查看模型评估报告：包含多张可视化的衡量指标图。

字段参数

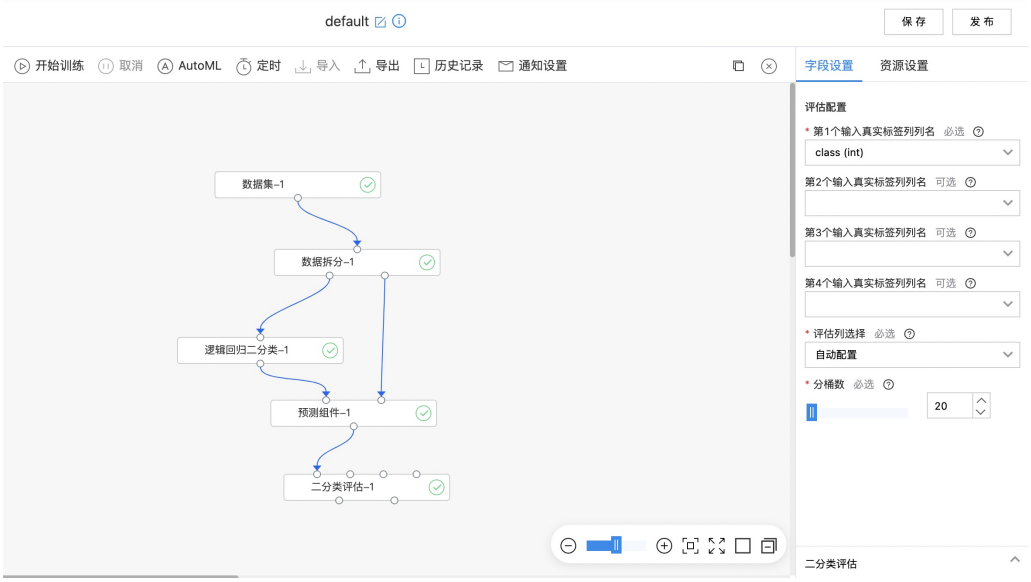
参数名称	是否必选	参数描述	默认值
真实标签列列名	是	该列代表真实的标签，需要是非数组类型，唯一值最多是2，最多可选择4个真实标签列(仅第1个必选)。	无
评估列选择	是	自动配置会根据系统默认字段配置字段。手动配置可以手动选择要预测标签列。	自动配置
评估内容配置	是	基础评估只进行基本指标的评估。详细评估在有预测概率列时使用，提供详细的评估。	基础评估
预测概率列	否	输入数据集中的该列代表从0到1的预测概率，概率越大表示越接近正样本。必须是Double类型	
正样本标签	否	二分类的正样本的标签值	无
预测标签列列名	否	该列代表预测得到的标签，是评估的对象，最多可选择4个预测标签列。	无
分桶数	是	计算ROC曲线、F-score by threshold等指标时把数据分成多少份 范围：[5, 1000]	20

评估指标

名称	介绍
accuracy	准确率
precision	精准率
recall	召回率
FPR	伪阳性率
f1score	F1分数
auc	roc曲线下面积
area under pr	pr曲线下面积

使用示例

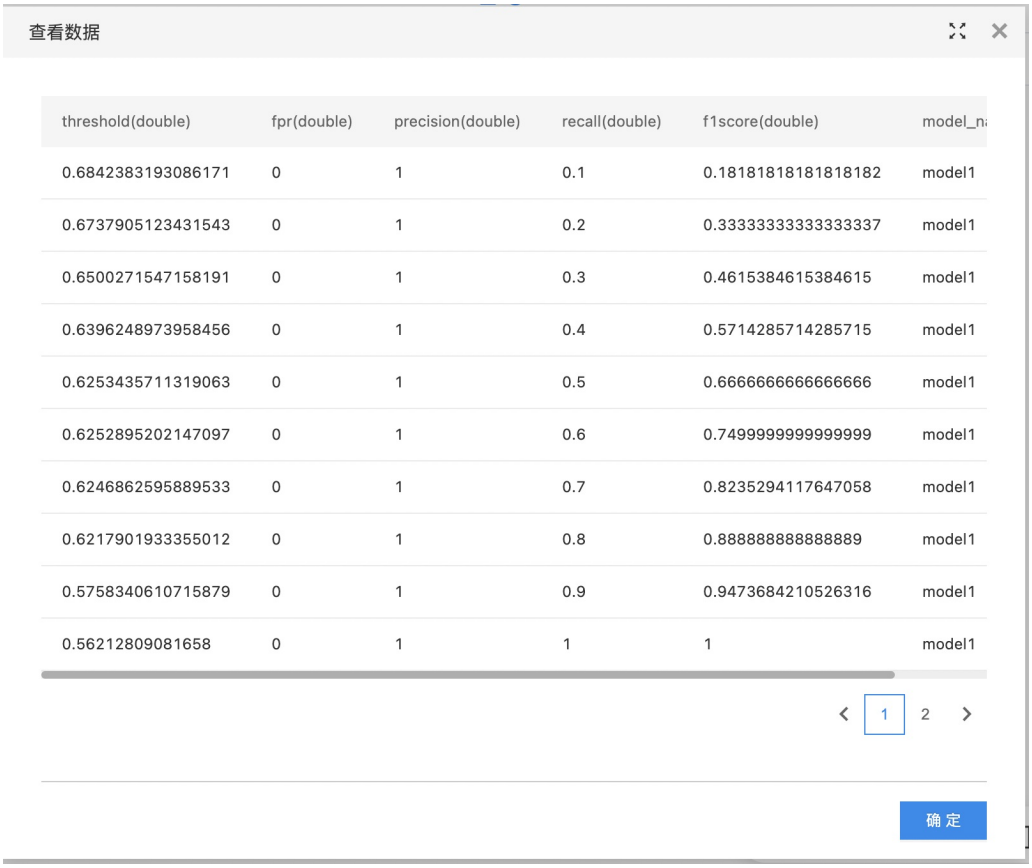
1. 搭建算子结构如下图所示，配置二分类评估组件参数，运行算子。



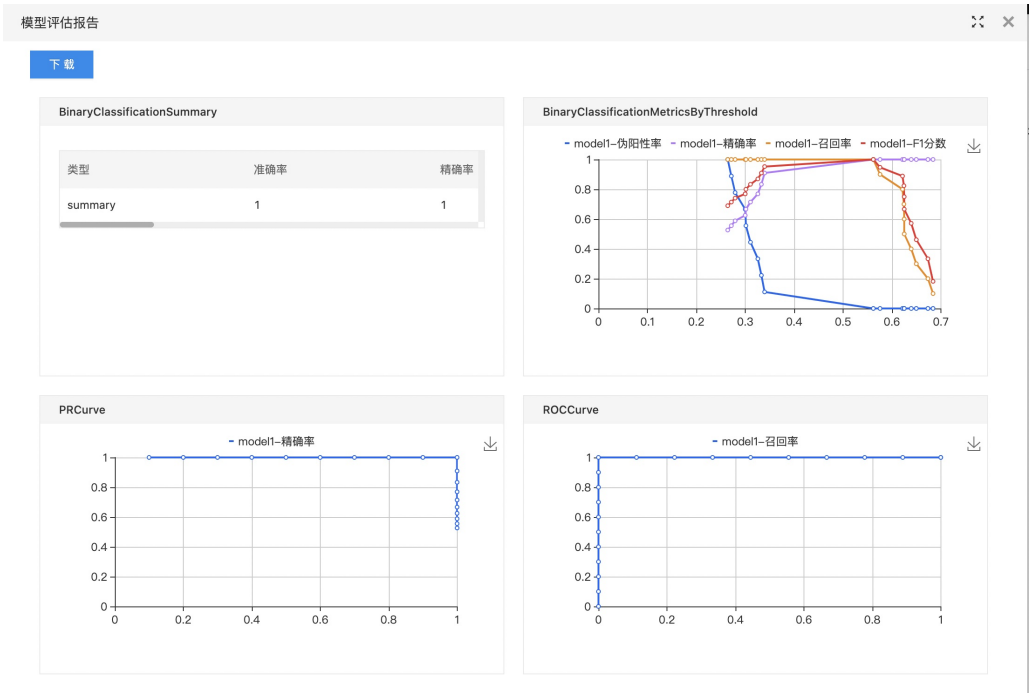
2. 查看summary数据集。

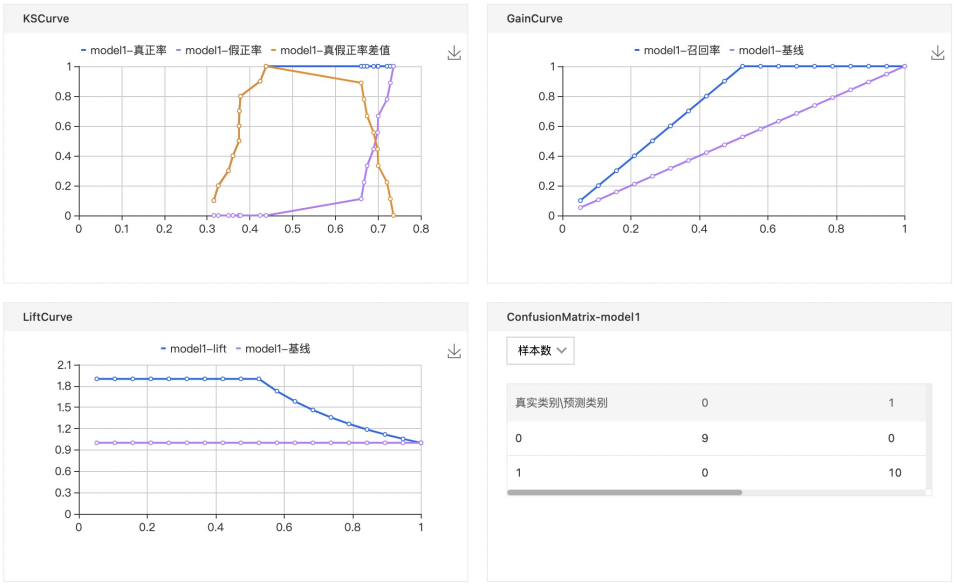


3. 查看metricsByThreshold数据集。



4. 右键查看模型评估报告。





多分类评估

基于分类模型的预测结果和原始结果，评价多分类算法模型的优劣，指标包括 Accuracy、F1-Score 等。

输入

- 最多可输入4个数据集，用户需要选择原始标签列、预测标签列。

输出

- 第一个输出是summary数据集，第二个输出是分类指标表，右键可以查看模型评估报告。
- summary数据集只有一行，每一列是一个评价指标，包括以下指标：accuracy、weighted recall、weighted precision、weighted FPR、weighted f1score。
- metricsByLabel数据集:描述了不同标签的fp率、precision、recall、f1score。
- 模型评估报告：展示了上面两个表格与混淆矩阵。

字段参数

参数名称	是否必选	参数描述	默认值
真实标签列列名	是	该列代表真实的标签，需要是非数组类型，最多选择4个真实标签列列名(仅第1个必选)。	无
评估列选择	是	自动配置会根据系统默认字段配置字段。手动配置可以手动选择要预测标签列。	自动配置
标签列列名	否	该列代表预测得到的标签，是评估的对象，最多选择4个标签列列名。	无

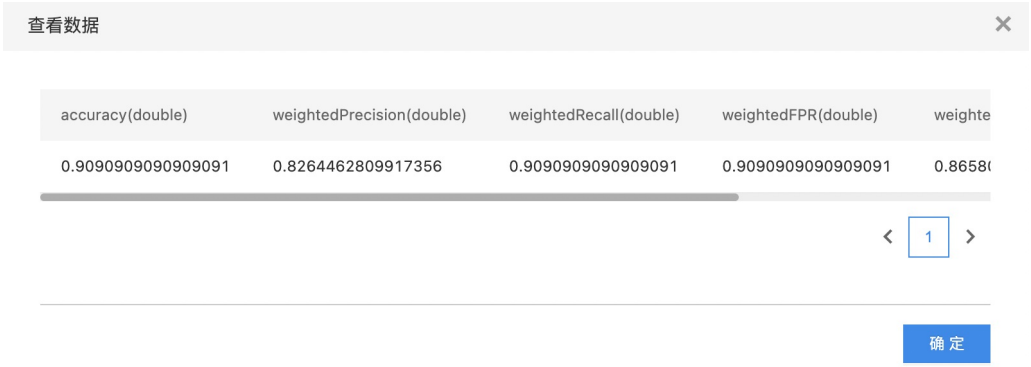
评估指标

名称	介绍
weighted recall	加权精准率
weighted precision	加权召回率
weighted FPR	加权伪阳性
weighted f1score	加权f1分数
recall	召回率
precision	精准率
fpr	fp率
f1score	f1分数
accuracy	准确率

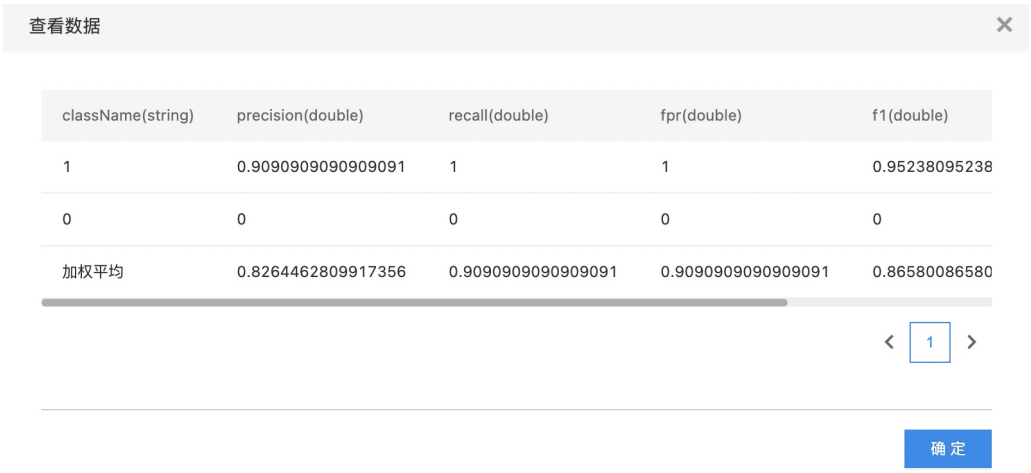
使用示例

1. 多分类与二分类的使用方法一样，将模型更换为多分类即可，该示例只展示评估指标。

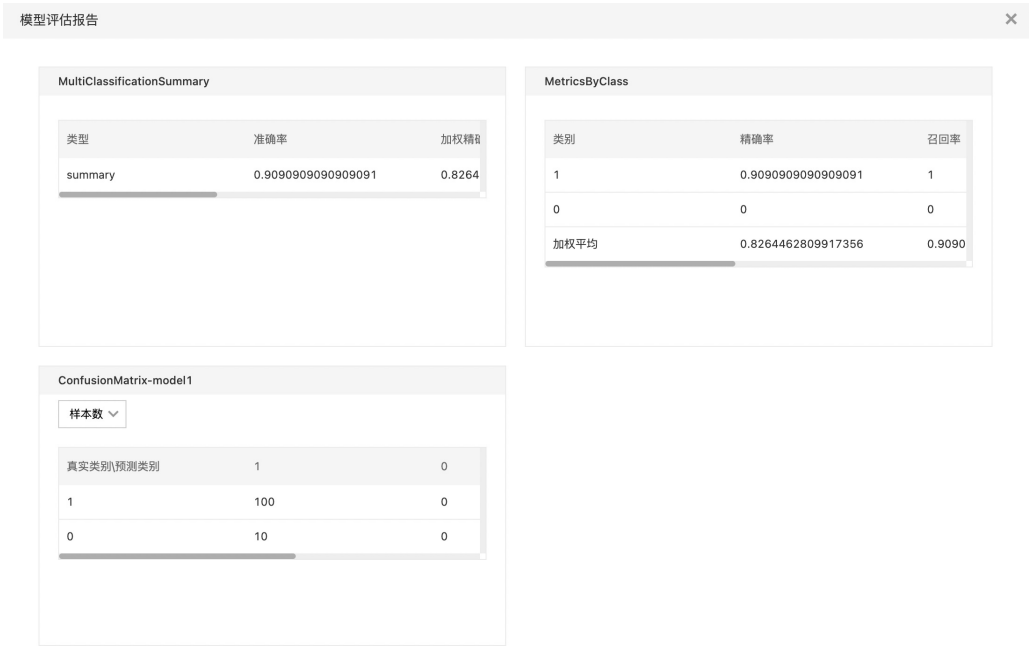
2. 查看summary数据集。



3. 查看分类指标表。



4. 查看模型评估报告。



回归评估

基于预测结果和原始结果，评价回归算法模型的优劣，包含指标和残差直返图。其中指标包括 SST、SSE、SSR、R2、R、 MSE、RMSE、MAE、MAD、MAPE、count、yMean 和 predictMean。

输入

- 最多可输入4个数据集，用户需要选择原始标签列、预测标签列。

输出

- 第一个输出是summary数据表，第二个输出是误差统计，对误差划分为10等份，统计每个区间内样本数量，右键可以查看模型评估报告。

字段参数

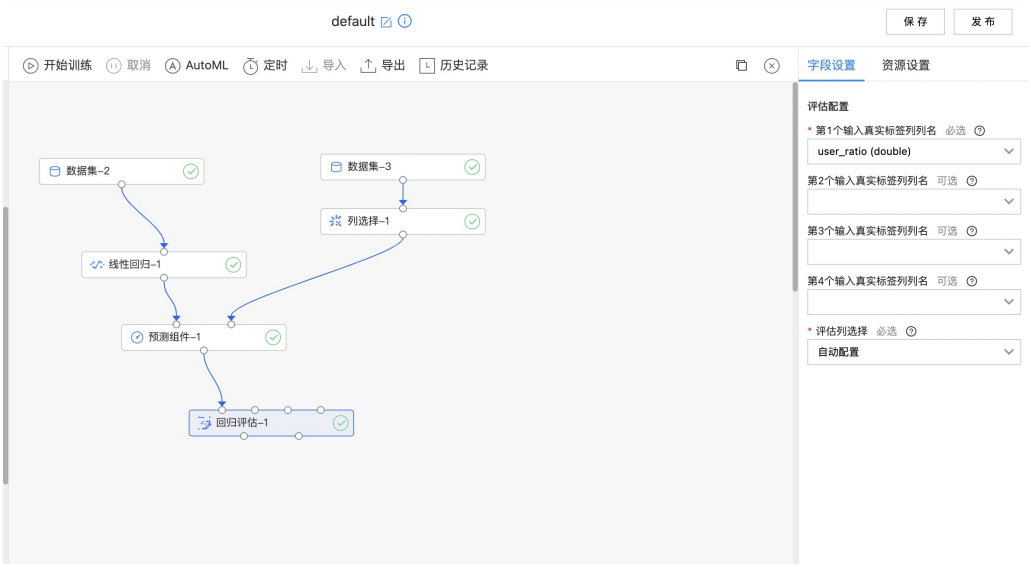
参数名称	是否必选	参数描述	默认值
真实标签列列名	是	该列代表真实的标签，需要是数值类型，最多可选择4个真实标签列(仅第1个必选)。	无
评估列选择	是	自动配置会根据系统默认字段配置字段。手动配置可以手动选择要预测标签列。	自动配置
预测标签列列名	否	该列代表预测得到的标签，是评估的对象，最多可选择4个预测标签列。	无

评估指标

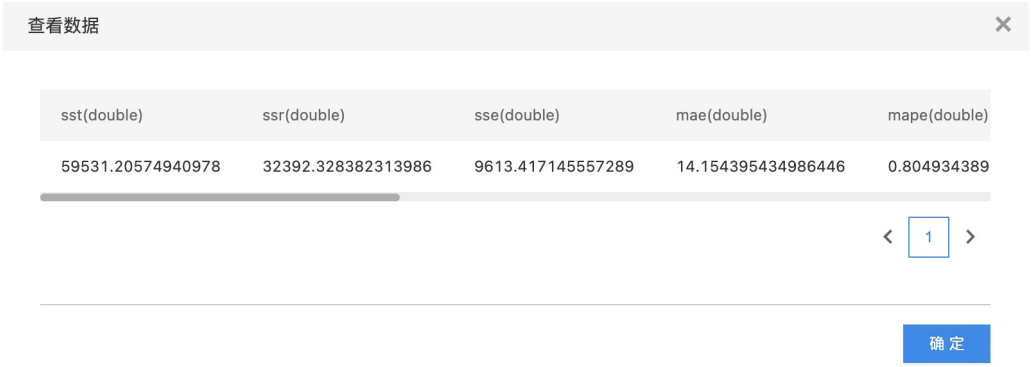
名称	介绍
SST	总平方和
SSE	误差平方和
SSR	回归平方和
R2	判定系数
MSE	均方误差
RMSE	均方根误差
MAE	平均绝对误差
MAPE	平均绝对百分误差
explained variance	可解释方差
yMean	原始列均值
predictionMean	预测列均值

使用示例

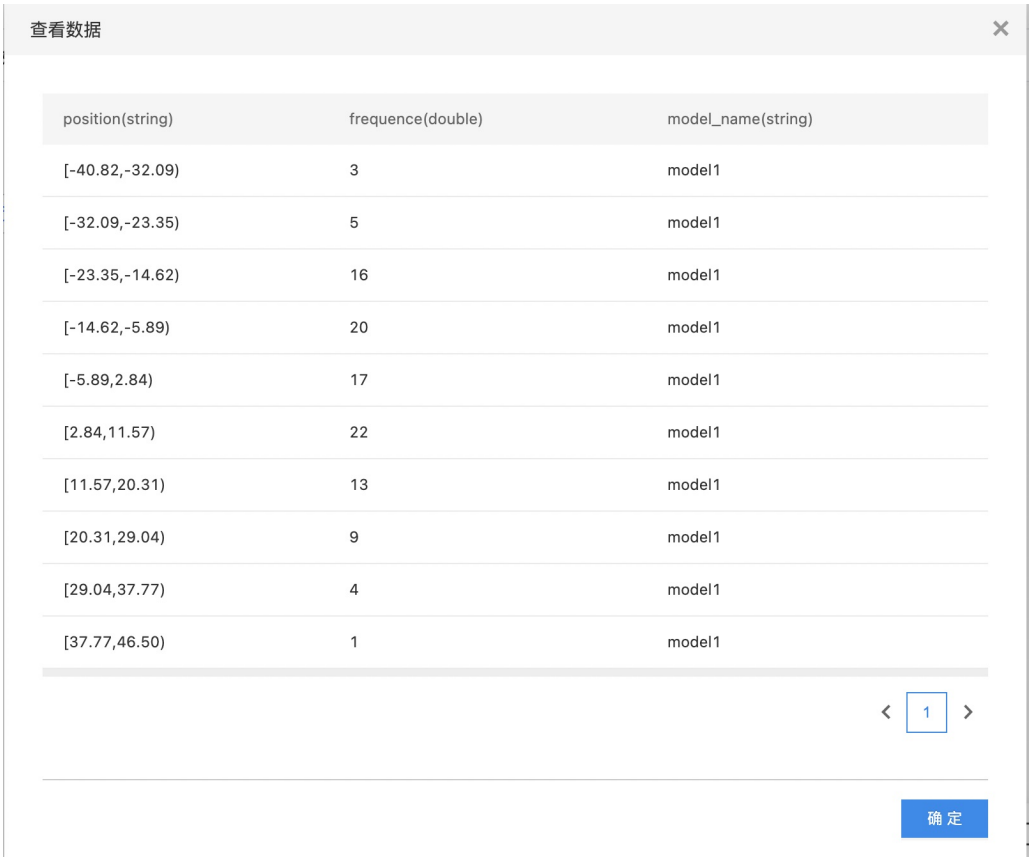
1. 搭建算子结构如下图所示，配置回归评估组件参数，运行算子。



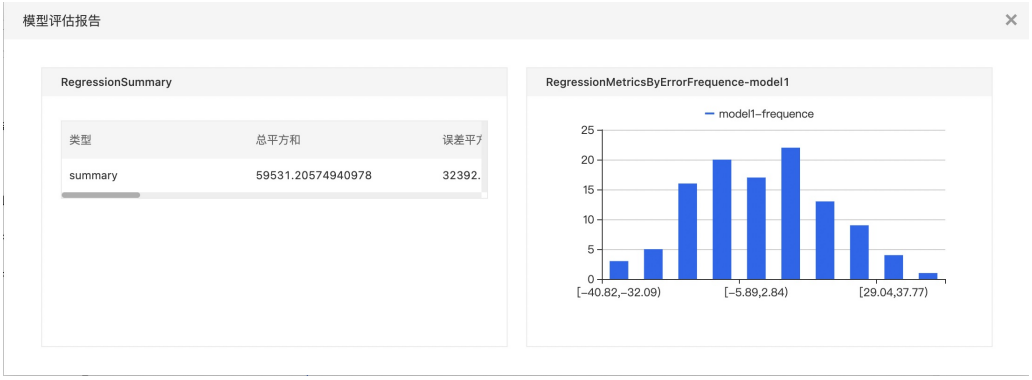
2. 查看summary数据集。



3. 查看误差统计。



4. 查看模型评估报告。



聚类评估

基于原始数据和聚类模型，评价聚类模型的优劣，包含指标和图标。

输入

- 输入聚类评估后的数据集，需要选择模型评估时使用的特征列，要求必须是模型存储特征值一致，预测标签列，离质心距离。

输出

- 第一个输出是summary数据集，第二个输出是每个cluster统计信息，右键可以查看模型评估报告。

字段参数

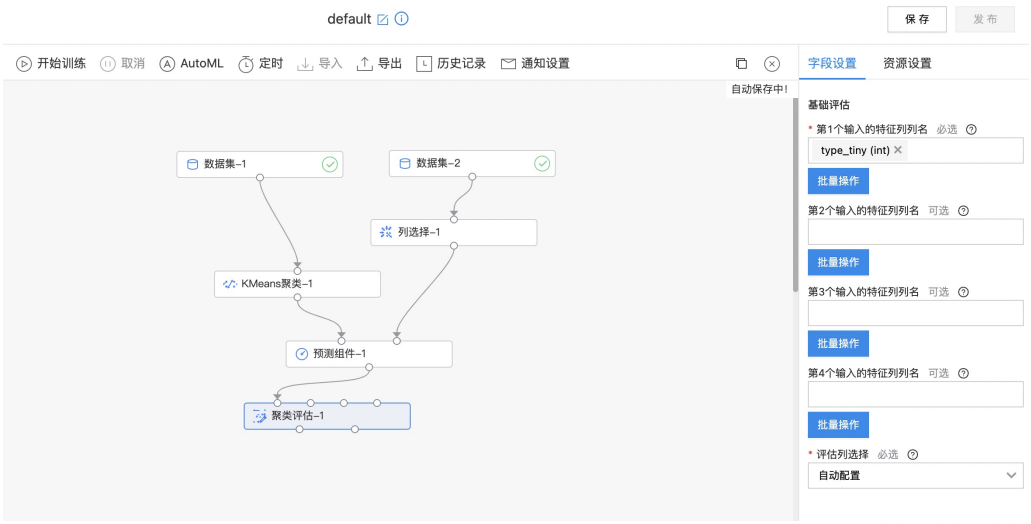
参数名称	是否必选	参数描述	默认值
特征列列名	是	模型评估时使用的特征列，要求必须是模型存储特征值一致，必须是数值或数值数组类型，最多配置4个特征列多选框(仅第1个必选)。	无
评估列选择	是	自动配置会根据系统默认字段配置字段。手动配置可以手动选择要预测标签列。	自动配置
预测标签列名	否	该列代表预测的聚类簇索引，必须是整数类型。	无
距离质心距离列	否	该列代表每个例子距离所有质心的距离，必须是数值数组类型。	无
真实标签列列名	否	该列代表每个例子距离所有质心的距离，必须是数值数组类型。	无

评估指标

名称	介绍
Silhouette coefficient	轮廓系数
tot_withinss	凝聚度
totss	总离差平方和
betweenss	分离度
Average Distance to Cluster Center	距质心平均距离
Max Distance to Cluster Center	距质心最大距离
Number of Points	簇内样本数

使用示例

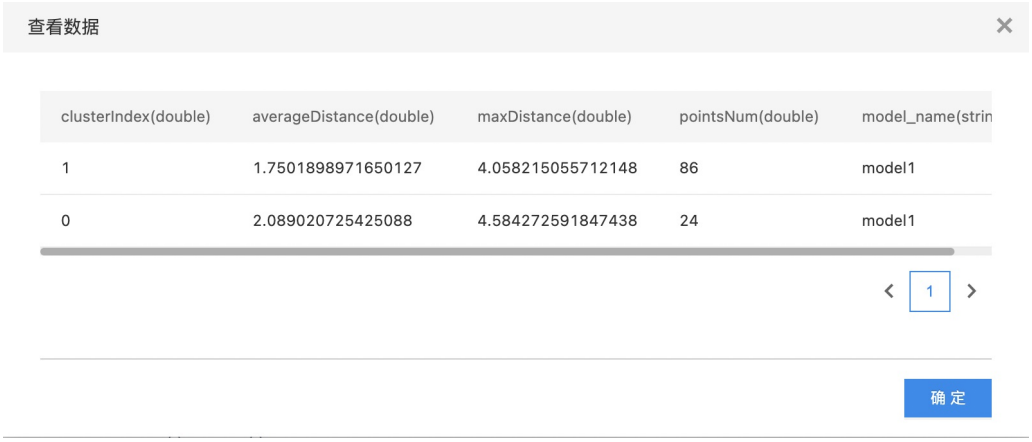
1. 搭建算子结构如下图所示，配置聚类评估组件参数，运行算子。



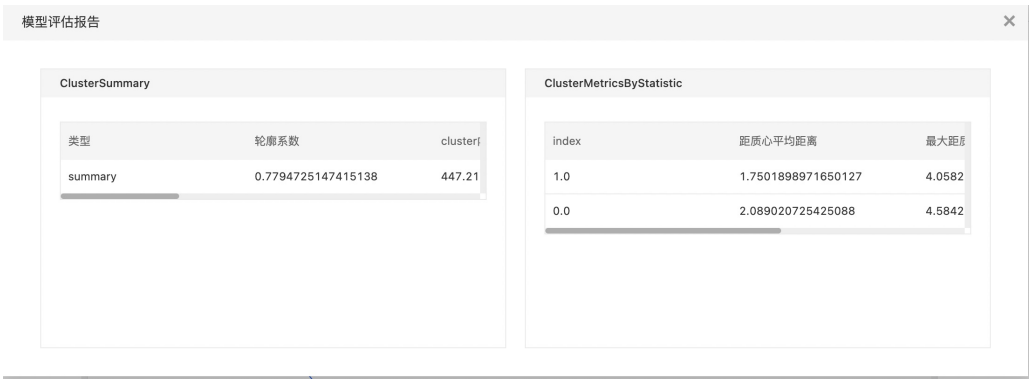
2. 查看summary数据集。



3. 查看cluster统计信息。



4. 查看模型评估报告。



异常检测评估

评估模块支持计算 AUC、KS，同时输出数据用于画 PR 曲线、ROC 曲线 KS 曲线，同时也支持分组评估。

输入

- 输入异常检测处理后的数据集，评估需要标签列和分数列。标签列的唯一值必须是2，需要指明异常标签是哪个，默认是1；分数列的值越大越可能是异常；如果使用自动配置，使用组件库中异常检测算子的预测结果诸位输入，则无需配置分数列，只需配置标签列。

输出

- 第一个输出是summary数据集，第二个输出是metricsByThreshold数据集，右键可以查询模型评估报告。

字段参数

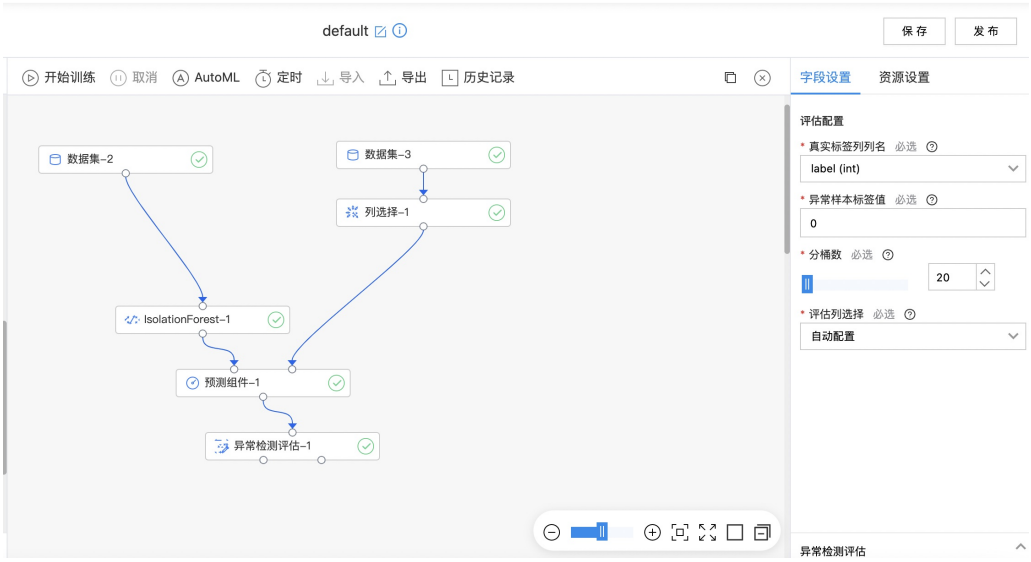
参数名称	是否必选	参数描述	默认值
真实标签列列名	是	该列代表真实的标签，需要是非数组类型，唯一值最多是2。	无
异常样本标签值	是	真实标签列中异常样本的标签值。	无
分桶数	是	计算时把数据分成多少份 范围：[5, 1000]。	20
评估列选择	是	自动配置会根据系统默认字段配置字段。手动配置可以手动选择要预测标签列。	自动配置
预测分数列	是	输入数据集中的该列代表异常分数，分数越大表示越可能是异常值。必须是Double类型	无

评估指标

名称	介绍
auc_roc	auc_roc
auc_pr	auc_pr值
ks	ks值

使用示例

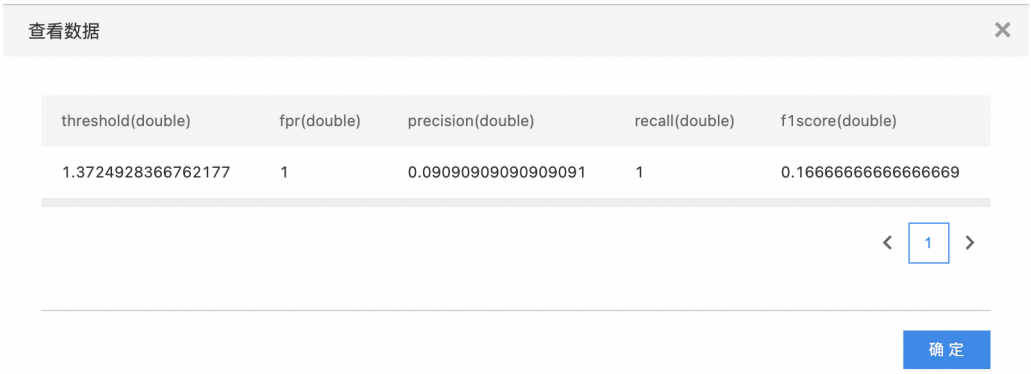
1. 搭建算子结构如下图所示，配置异常检测评估组件参数，运行算子。



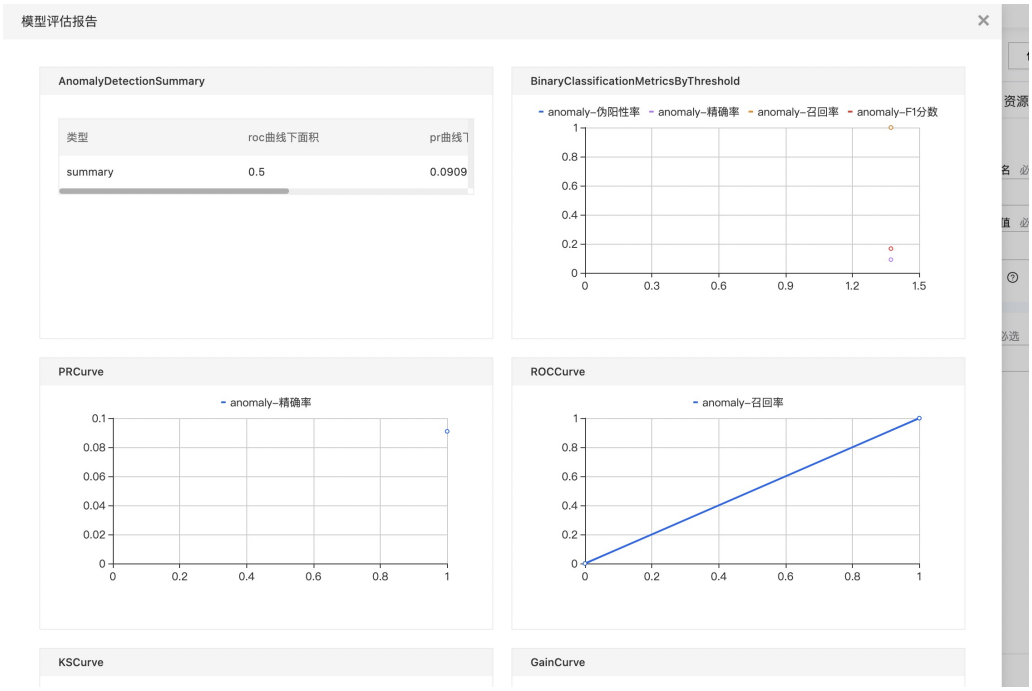
2. 查看summary数据集。



3. 查看metricsByThreshold数据集。



4. 查看模型评估报告。



自定义Python评估

自定义 Python 评估组件：写自定义 python 代码，完成模型评估任务。

输入

- 输入数据集，编辑python代码自定义评估组件。

输出

- 输出自定义评估组件结果，右键查看python绘制图表。

算子参数

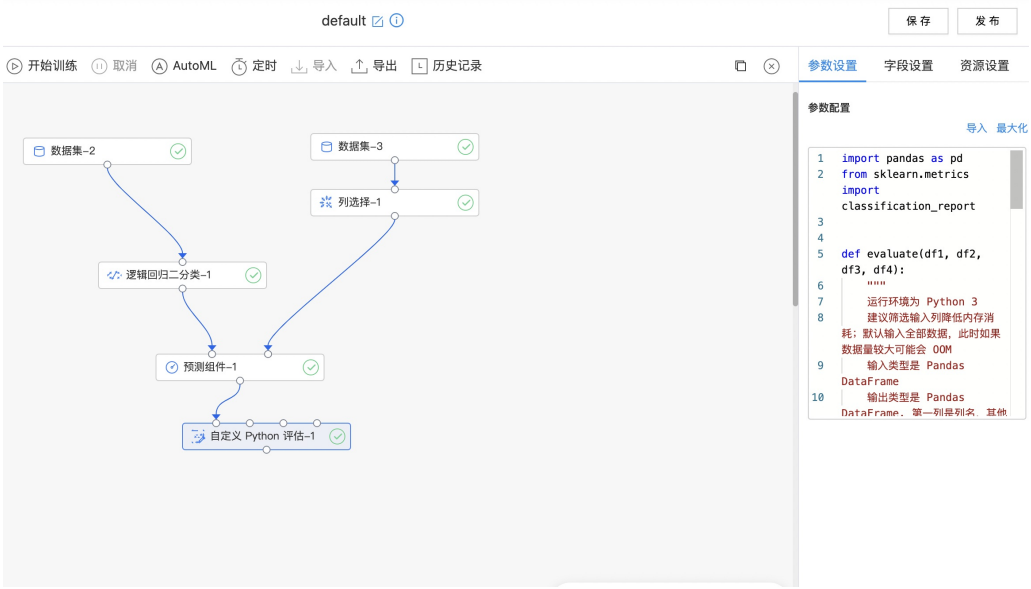
参数名称	是否必选	参数描述	默认值
python代码编辑窗口	是	编写python代码，自定义评估组件	

字段参数

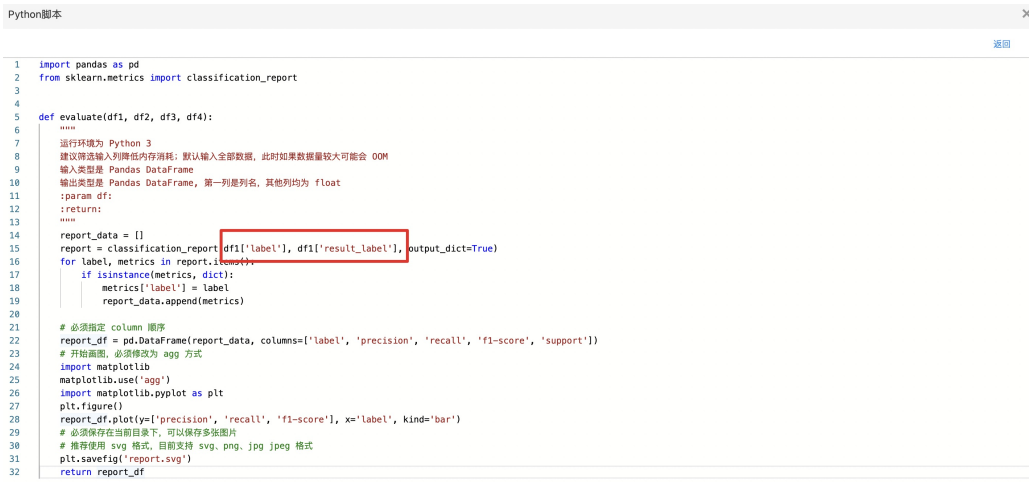
参数名称	是否必选	参数描述	默认值
第1个输入列列名，默认加载全部列	否	只加载选择的列，减少内存占用	无
第2个输入列列名，默认加载全部列	否	只加载选择的列，减少内存占用	无
第3个输入列列名，默认加载全部列	否	只加载选择的列，减少内存占用	无
第4个输入列列名，默认加载全部列	否	只加载选择的列，减少内存占用	无

使用示例

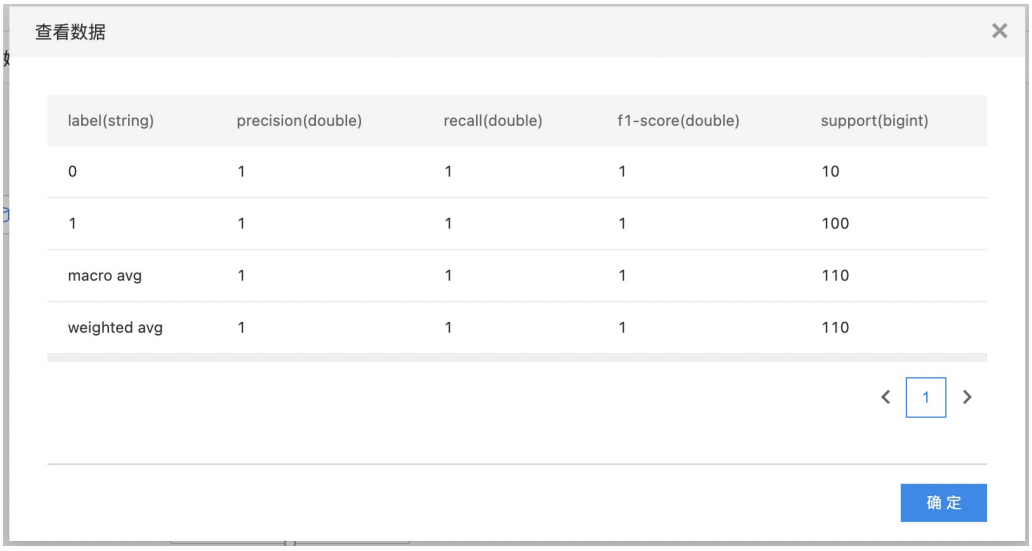
1. 搭建算子结构如下图所示，编写python代码，运行算子。



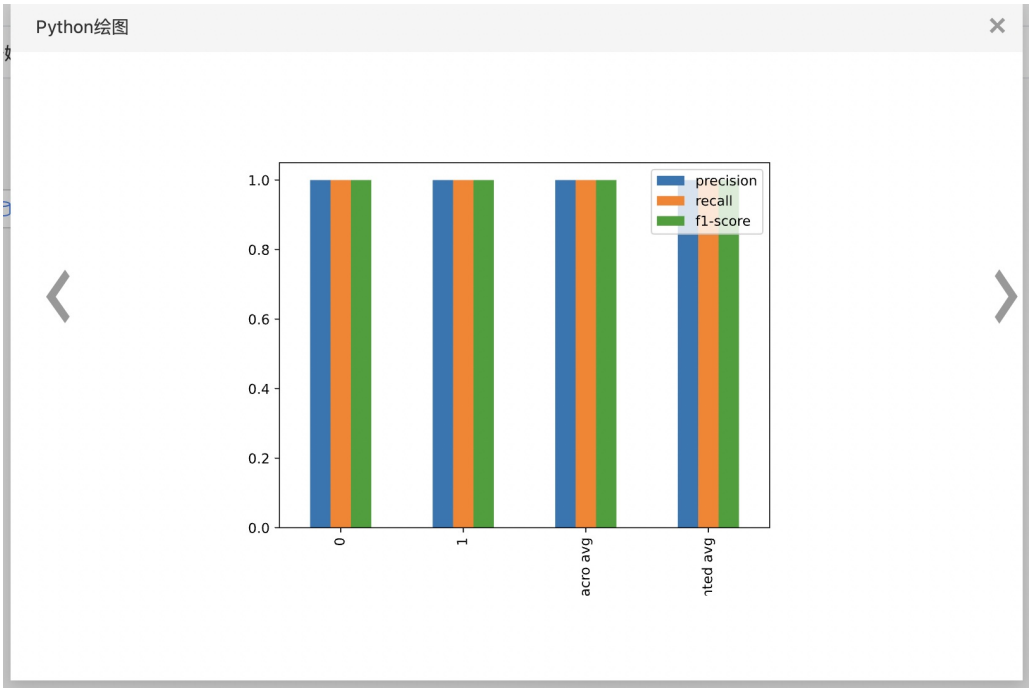
2. 代码如下，**classification_report**中参数**df1['label']**和**df1['result_label']**需要与输入数据集的真实标签列名与预测标签列名对应。



3. 查看输出结果。



4. 查看python绘制图表。



时间序列组件

时间序列组件

ARIMA

ARIMA(Autoregressive Integrated Moving Average)差分自回归移动平均模型，时间序列预测分析方法之一。 ARIMA(p,d,q)中，AR是自回归，p为自回归项数；MA为移动平均，q为移动平均项数，d为使之成为平稳序列所做的差分次数(阶数)。

输入

- 输入一个数据集，需要指定排序列、数值列。
- 排序列为对时间序列进行排序的列（基本为数值列）要求数值非空且有序，数值列为时间序列，根据此列进行时间序列运算，要求非空的数值列。

输出

- 输出一个数据集，该数据集只含有一列(时间序列的预测结果，行数为预测步长。

算子参数

参数名称	是否必选	参数描述	默认值
ARIMA中p值	是	ARIMA中p值，代表AR中的自回归项数，该值必须为正 范围：[0, inf)	1
ARIMA中d值	是	ARIMA中d值，代表差分次数，该值必须为正 范围：[0, 2]	0
ARIMA中q值	是	ARIMA中q值，代表MA中的移动平均项数，该值必须为正 范围：[0, inf)	0
预测的步长	是	基于时间序列预测的未来时间步长 范围：[1, inf)	1

字段参数

参数名称	是否必选	参数描述	默认值
排序列	是	时间序列依据此列由小到大排序，一般选择日期列且不能有重复值	无
数值列	是	时间序列的数值列，步长与结果数据集的行数一致，该列必须为数值	无

计算逻辑

1.AR自回归模型公式：

$$y_t = \mu + \sum_{i=1}^{p} \gamma_i y_{t-i} + \varepsilon_t$$

其中y_t是当前值，μ是常数项，p是自回归项数，γ_i是自相关系数，ε_t是误差

2.MA移动平均模型公式：

$$y_t = \mu + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t$$

移动平均模型关注的是自回归模型中的误差项的累加，其中q是移动平均项数

3.ARMA自回归移动平均模型公式：

$$y_t = \mu + \sum_{i=1}^{p} \gamma_i y_{t-i} + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

p代表自回归项数，q代表移动平均项数，d代表差分阶数。

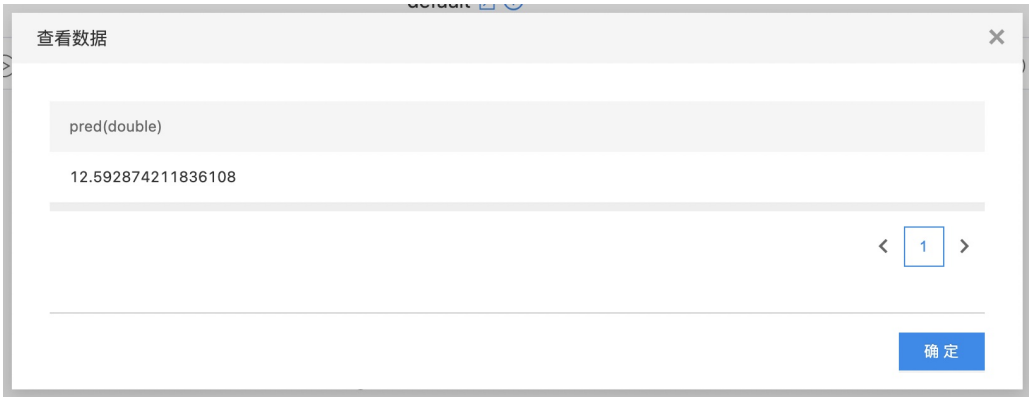
可见，ARMA的公式由AR和MA结合而成，在此基础上结合差分法，就是ARMIA算法。

使用示例

1. 构建算子结构，配置参数，完成训练。



2. 查看训练结果。



AutoARIMA

AutoARIMA差分整合移动平均自回归模型，时间序列预测分析方法之一。AutoARIMA通过超参搜索查询最优的ARIMA模型。 Auto-ARIMA通过进行差异测试来确定差分d，然后在定义的**p、q起始值**和**p、q最大值**，的范围内拟合模型。如果启用了季节性可选参数，在确定季节性差分d的最佳顺序之后，还将寻求找出最佳的p、q超参数。

注意：由于平稳性问题，auto-ARIMA可能找不到会收敛的合适模型。如果是这种情况，将抛出ValueError，建议在重新拟合之前采取诱导平稳性的措施，或者选择新的搜索范围。

输入

- 输入一个数据集，包括排序列、数值列。
- 排序列为对时间序列进行排序的列（基本为数值列）要求数值非空且有序；数值列为时间序列列根据此列进行时间序列运算，要求非空的数值列。

输出

- 第一个输出是模型报告，包含搜索出的模型，模型的AIC、BIC、HQIC、Log Likehood。
- 第二个输出是预测结果，行数与预测的步长一致。

算子参数

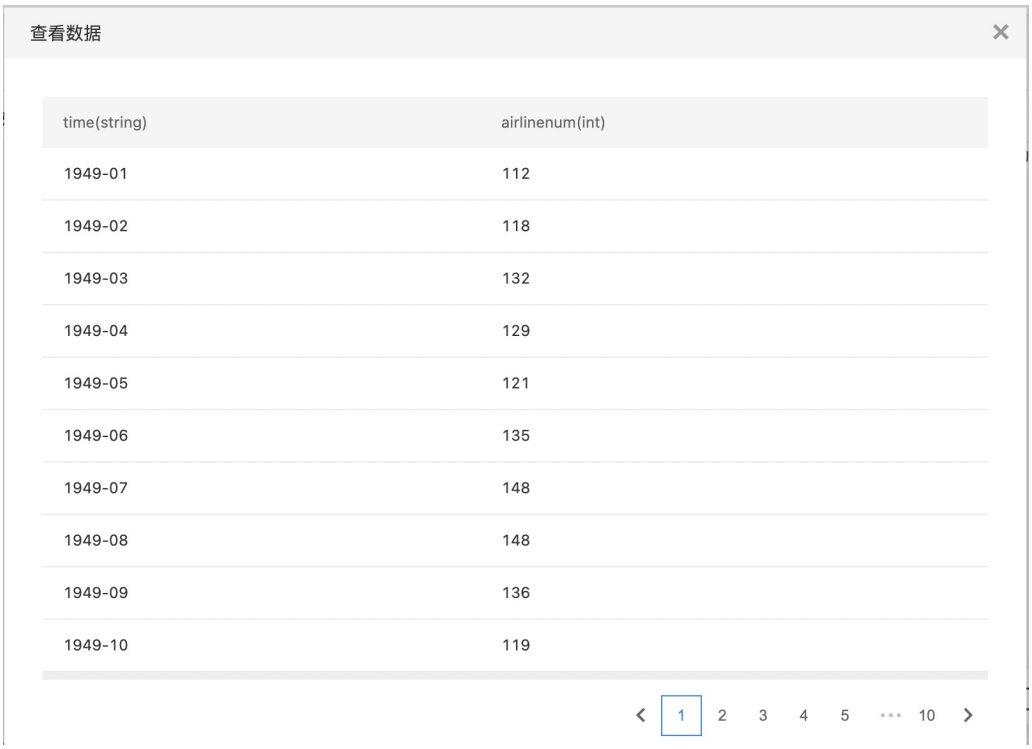
参数名称	是否必选	参数描述	默认值
是否训练季节性ARIMA	是	是否适合季节性的ARIMA，默认是真。注意，如果季节性为真且频率为1时，季节性将为假。	开启
季节性频率	是	每个季节周期观测的频率。例如，每年观测一次（非季节性数据）则频率为1，每个季度观测一次则频率为4，每个月观测一次则频率为12，每周观测一次则频率为52，每天观测一次则频率为7。 范围：[1, inf)。	1
p、q起始值	是	p为自回归(AR)模型的阶数(或时滞数)。q是MA模型的阶数，起始值必须是一个正整数。 范围：[1, inf)。	2
p、q最大值	是	p、q最大值, 必须是个大于等于p、q起始值的正整数 范围：[1, inf)。	5
差分d	是	一阶微分的阶数，必须为-1或正整数。设置为-1时将根据测试结果自动选择该值。如果为-1，运行时间会相对较长 范围：[-1, inf)。	-1
差分d最大值	是	差分d最大值, 必须是个大于等d的正整数 范围：[1, inf)。	2
季节性p、q起始值	是	季节性模型设置的p、q最大值，必须是个大于等于p、q起始值的正整数。 范围：[1, inf)。	1
季节性p、q最大值	是	季节性模型设置的p、q最大值，必须是一个正整数。 范围：[1, inf)。	2
季节性差分d	是	季节性差分d，必须为-1或正整数。设置为-1时将根据测试结果自动选择该值。如果为-1，运行时间会相对较长 范围：[-1, inf)。	-1
季节性差分d最大值	是	季节性模型设置的差分d最大值，必须是个大于等于季节性差分d的正整数。 范围：[0, inf)。	1
最优化方法	是	搜索模型过程中采用的最优化方法： nm：Nelder-Mead bfgs：Broyden-Fletcher-Goldfarb-Shann lbfgs：有限内存BFGS powell：Powell方法 cg：共轭梯度法	bfgs
显著性	是	测试显著性 范围：[0.0, 1.0]。	0.05
最大迭代次数	是	搜索模型的最大迭代次数 范围：[1, inf)。	50
预测的步长	是	基于时间序列预测的未来时间步长 范围：[1, inf)。	10

字段参数

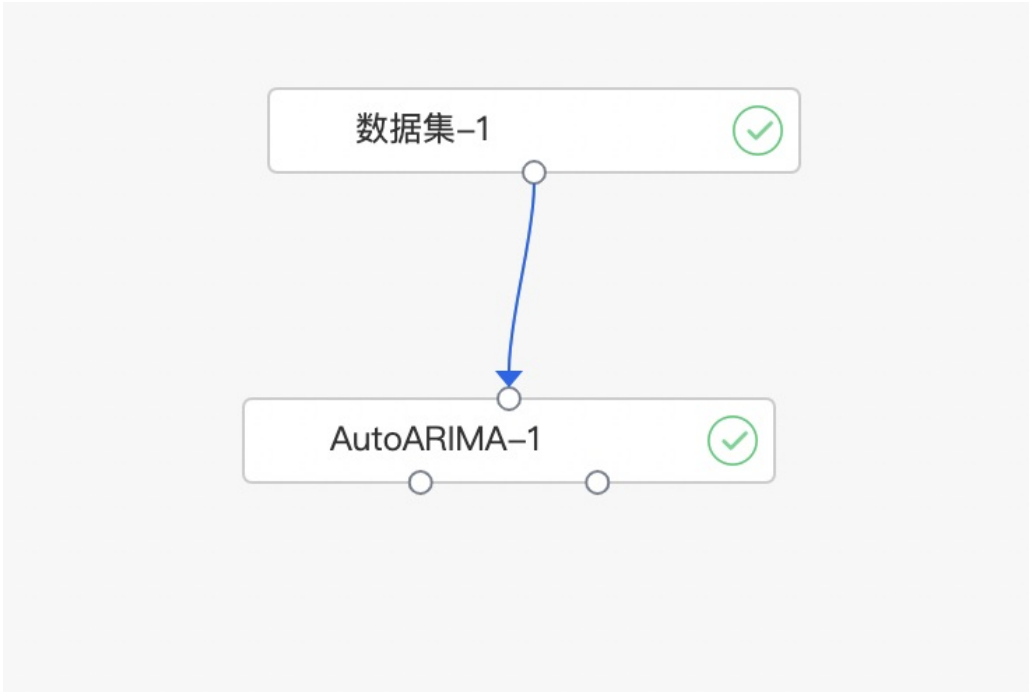
参数名称	是否必选	参数描述	默认值
排序列	是	时间序列依据此列由小到大排序，一般选择日期列且不能有重复值。	无
数值列	是	时间序列的数值列，该列必须为数值。	无

使用示例

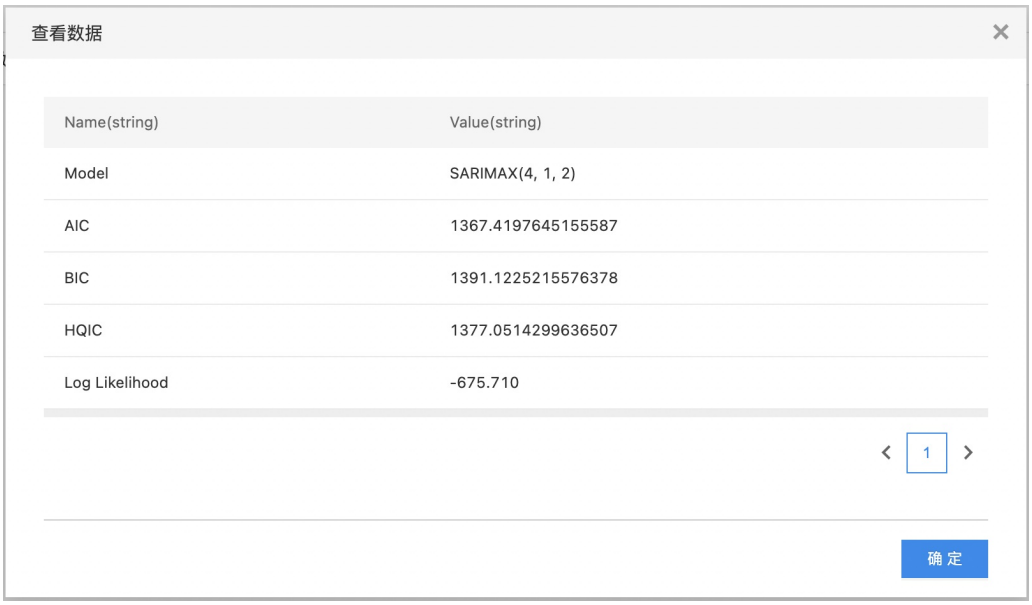
1. 查看原始数据，数据集包含两列：time-时间列、airlinenum-数值列，可以得知航线数量是每天观测一次。



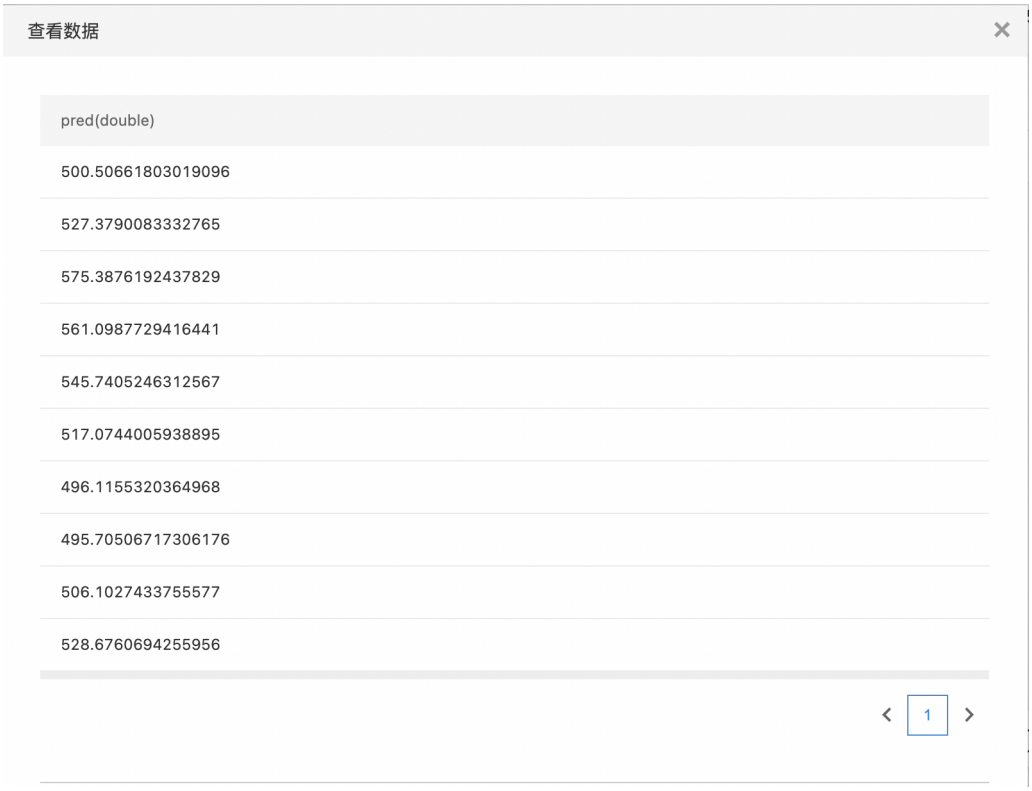
2. 算子连接方式如下，选择默认参数，完成训练。



3. 查看模型报告。



4. 查看预测结果。



AR

AR（auto-regressive）模型是自回归模型，时间序列预测分析方法之一。其原理是利用观测点前若干时刻的变量的线性组合来描述观测点后若干时刻变量的值，属于线性回归模型。

输入

- 输入一个数据集，需要指定排序列、数值列。
- 排序列为对时间序列进行排序的列（基本为数值列）要求数值非空且有序，数值列为时间序列列根据此列进行时间序列运算，要求非空的数值列。

输出

- 输出一个数据集，该数据集只含有一列(时间序列的预测结果)，行数为预测步长。

算子参数

参数名称	是否必选	参数描述	默认值
AR中的p值	是	AR中p值 ，代表自回归项数，范围：[1, inf)。	1
预测的步长	是	基于时间序列预测的未来时间步长，步长与结果数据集的行数一致，范围：[1, inf)。	1

字段参数

参数名称	是否必选	参数描述	默认值
排序列	是	时间序列依据此列由小到大排序，一般选择日期列且不能有重复值	无
数值列	是	时间序列的数值列，该列必须为数值	无

使用示例

构建算子结构，配置参数，完成训练。



DeepAR

DeepAR 是Amazon提出的基于深度学习的时间序列预测方法。DeepAR 是一种适用于时间序列预测的监督学习算法，该算法使用递归神经网络(RNN) 生成点预测和概率预测。传统的时间序列预测方法（ARIMA、Holt-Winters' 等）往往针对一维时间序列本身建模，难以利用额外特征。此外，传统方法的预测目标通常是序列在每个时间步上的取值。与之相比，基于神经网络的 DeepAR 方法可以很方便地将额外的特征纳入考虑，且其预测目标是序列在每个时间步上取值的概率分布。

输入

- 输入一个数据集，需要指定排序列、数值列。排序列为对时间序列进行排序的列要求为符合此列应为日期列且需要满足Timestamp中日期类型。数值列为时间序列列根据此列进行时间序列运算，要求非空的数值列。

输出

- 输出一个数据集，该数据集只含有一列(时间序列的预测结果)，行数为预测步长。

算子参数

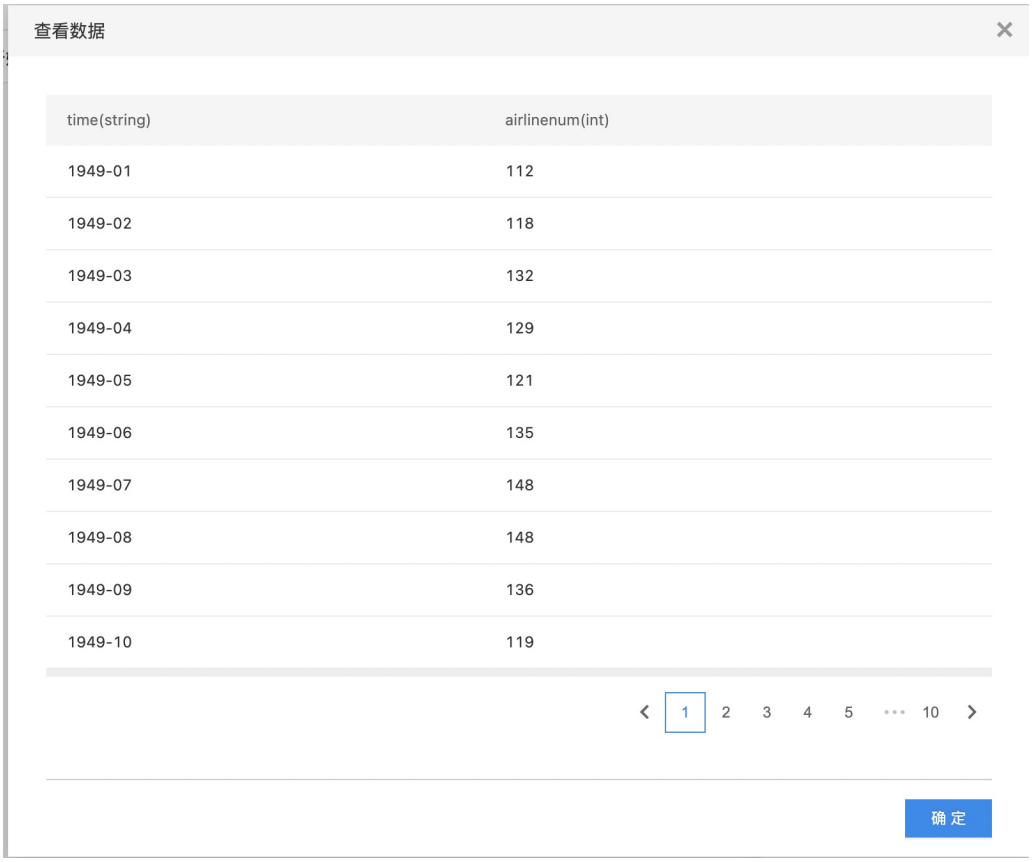
参数名称	是否必选	参数描述	默认值
DeepAR中RNN的类型	是	DeepAR中RNN的类型： lstm gru	lstm
DeepAR中的RNN层数	是	DeepAR中RNN层数 范围：[1, inf)	2
DeepAR中RNN的神经元个数	是	DeepAR中RNN的神经单元个数 范围：[10, inf)	40
时间序列中的观测频率	是	时间序列中的观测频率，需要和排序列时间频率一致且为Pandas中频率类型，通常用M、D、H等，例如1M代表1个月，1D代表1天，1H代表1小时。	无
预测的步长	是	基于时间序列预测的未来时间步长 范围：[1, inf)	1
训练学习率	是	训练中的学习率 范围：[1e-06, 1.0]	0.001
训练轮数	是	训练轮数 范围：[1, inf)	100
训练batch的大小	是	训练batch大小 范围：[1, inf)	32
DeepAR中的DropOut	是	DeepAR中的DropOut 范围：[0.0, 1.0]	0.1

字段参数

参数名称	是否必选	参数描述	默认值
排序列	是	时间序列依据此列由小到大排序，此列应为日期列且需要满足Timestamp中日期类型且不能有重复值	无
数值列	是	时间序列的数值列，该列必须为数值	无

使用示例

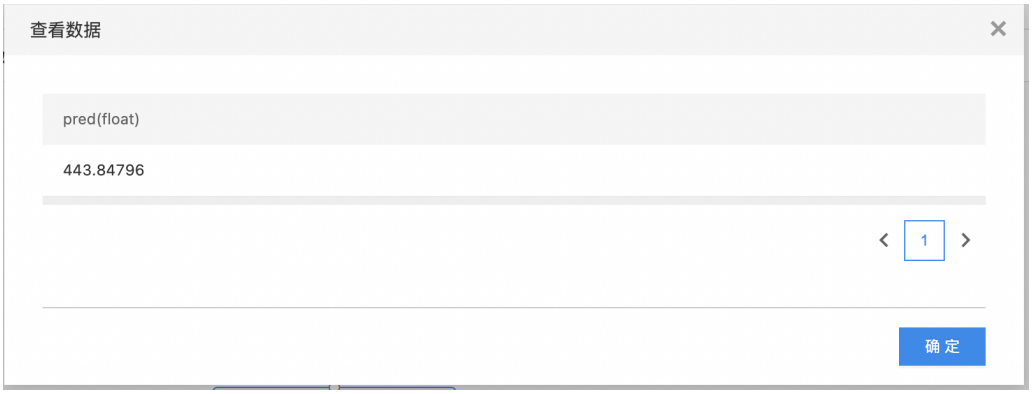
1. 查看原始数据。



2. 构建算子结构，配置参数，在本示例中，时间序列中的观测频率为"1M"，完成训练。



3. 查看输出结果。



Prophet

Prophet是基于非线性趋势与年度、每周和每日季节性以及假日效应的附加模型预测时间序列数据的过程，支持趋势、季节性周期变化及节假日效应。

输入

输入两个数据集，其中训练数据集必选，节假日数据集可选。

输出

输出一个数据集

算子参数

参数名称	是否必选	参数描述	默认值
growth	是	增长趋势项。支持两种增长趋势项模型：linear，logistic。	linear
automatic_changepoint_selection	是	自动变点选择。若选"no"，则表示手动指定变点。	Yes
change point_range	是	变点选择范围。基于时间序列的前指定比例范围的历史数据选择变点， 范围：[0.0, 1.0]。	0.80
n_changepoints	是	变点数。从历史数据前changepoint_range比例中均匀选择的变点个数， 范围：[0, inf)	25
change point_prior_scale	是	变点灵活度。调整自动变点选择的灵活性。较大的值将允许更多的变点，较小的值将允许较少的变点。值越大，模型对历史数据的拟合程度越强。建议取值在0.001和0.5之间， 范围：[1e-15, inf)	0.05
yearly_seasonality	是	适应年季节性。支持三种模式：auto，true，false。	Auto
weekly_seasonality	是	适应周季节性。支持三种模式：auto，true，false。	Auto
daily_seasonality	是	适应日季节性。支持三种模式：auto，true，false。	Auto
seasonality_mode	是	季节性方式。支持两种方式：additive(加法模型），multiplicative（乘法模型）。	additive
seasonality_prior_scale	是	季节性强度。调整季节性模型的强度，较大的值可使模型适应较大的季节性波动，较小的值会抑制季节性。建议取值在0.01和10之间， 范围：[1e-15, inf)。	10.0
holiday_prior_scale	是	节假日强度。调整节假日成分模型的强度，仅输入holidays时有效。建议取值在0.01和10之间， 范围：[1e-15, inf)。	10.0
mcmc_samples	是	MCMC样本。非负整数，如果大于0，将对指定数量的MCMC样本进行完整的贝叶斯推断，如果为0，将进行最大后验概率估计 范围：[0, inf)。	0
Interval_samples	是	区间宽度。为预测提供的不确定性区间的宽度，如果mcmc_samples为0，这将只是使用外推生成模型的最大后验概率估计的趋势中的不确定性；如果mcmc_samples大于0，这将被整合到所有模型参数中，包括季节性的不确定性。 范围：[-1.0, 1.0]	0.80
uncertainty_samples	是	模拟样本。用于估计不确定性区间的模拟抽样次数 范围：[1, inf)。	1000

预测参数配置

参数名称	是否必选	参数描述	默认值
periods	是	向前预测的整数周期数。 范围：[0, inf)。	7
freq	是	日期频率。频率可以有倍数，例如，5H表示5个小时。请输入有效频率。常用频率单位：S(秒)，T(分)，H(时)，D(日历日)，B(工作日)，W(周)，MS(月初)，M(月底)，Q(季度末)，Y(年末)。	D
include_history	是	包括历史日期。将历史日期包括在预测数据中。如果选择不包括历史日期，periods值需要设置为大于0。	no

字段参数

参数名称	是否必选	参数描述	默认值
时间列	是	时间序列。要求是日期类型或日期格式的字符串类型。时间列不可以包含空值。日期格式样例：2021-01-01 00:00:00。	无
预测列	是	时间序列预测列。要求是数值类型。预测列可以包含空值，要求非空值样本不少于2。	无
节假日名称列	否	节假日名称列。要求是字符串类型。输入holidays时有效且必选。节假日名称列不可以包含空值。	无
节假日时间列	否	节假日时间列。要求是日期类型或日期格式的字符串类型。输入holidays时有效且必选。节假日时间列不可以包含空值。日期格式样例：2021-01-01 00:00:00。	无

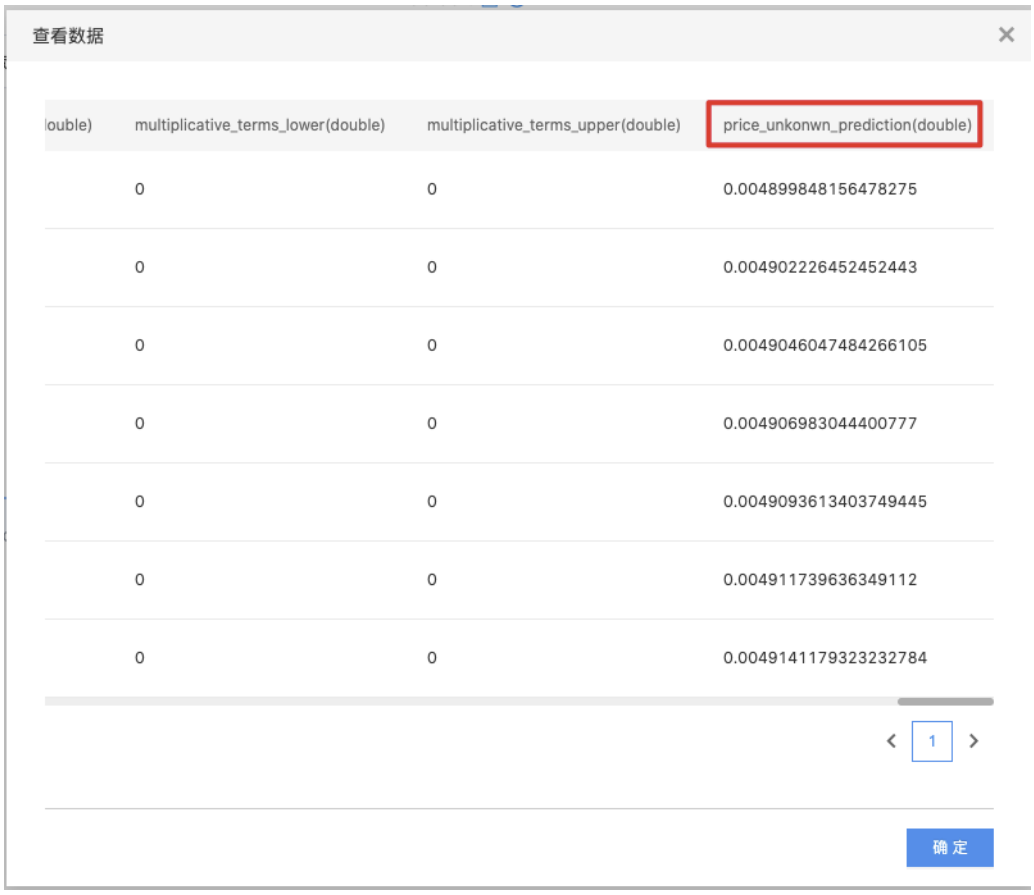
使用示例

1. 构建算子结构，配置参数，完成训练。



2. 右键“Prophet"组件，选择“查看数据” > "输出数据集"，查看输出结果。

   以”_prediction“结尾的列为预测结果列。



模型仓库

模型仓库简介

模型仓库是整个BML中模型的中央存储仓库，模型仓库可导入所有训练任务生成的模型并进行统一管理。

模型仓库支持的模型类型

技术方向	类型	部署方式
视觉	图像分类	公有云部署、EasyEdge本地部署
	物体检测	公有云部署、EasyEdge本地部署
	实例分割	公有云部署、EasyEdge本地部署
自然语言处理	文本分类	公有云部署
	短文本相似度	公有云部署
	序列标注	公有云部署
	文本实体抽取	公有云部署
结构化数据	表格数据预测	公有云部署
通用	N/A	公有云部署

从训练任务导入模型

BML支持多种训练方式，在训练任务执行成功后，可以将其生成的模型导入到模型仓库中进行模型的统一管理。

- 从预置模型调参导入模型
  - 在BML中通过预置模型调参完成模型训练后，可以将满足要求的模型发布到模型仓库中。
  - 一个预置模型调参项目的不同任务生成的模型只能发布到模型仓库中一个模型的不同版本下。
- 从Notebook导入模型

- 在Notebook中保存模型并通过校验后，可以将其发布到模型仓库中。
- 一个Notebook中保存的不同模型只能发布到模型仓库中一个模型的不同版本下。

- 从自定义作业导入模型
  - 自定义作业运行成功后，可以将其生成的模型发布到模型仓库中。
  - 一个自定义作业产生的模型只能发布到模型仓库中一个新的模型下。

从本地导入模型

BML也支持用户将本地已训练好的模型导入到模型仓库中，从而将模型部署为在线服务或转换为可行性离线部署的SDK等。

从本地导入时，BML当前支持的模型类型及相关框架如下所示：

模型类型	支持的AI框架	支持的部署方式
图像分类	PaddlePaddle	公有云部署、EasyEdge本地部署
	TensorFlow	公有云部署、EasyEdge本地部署
	Pytorch	公有云部署、EasyEdge本地部署
	Caffe	EasyEdge本地部署
	MXNet	EasyEdge本地部署
	Darknet	EasyEdge本地部署
	ONNX	EasyEdge本地部署
物体检测	PaddlePaddle	公有云部署、EasyEdge本地部署
	Caffe	EasyEdge本地部署
	Darknet	EasyEdge本地部署
表格数据预测	Sklearn	公有云部署
	XGBoost	公有云部署

创建模型

在模型仓库中，一个模型可以包含多个版本。从训练任务中导入模型或是直接导入本地模型，都对应于一个模型版本，在导入模型前应先创建模型。

前提条件

无。

操作步骤

1. 在左侧菜单栏中单击“模型仓库”>“模型管理”，进入模型管理页面。
2. 单击“新建”，系统显示“新建模型”弹出框。
3. 在“新建模型”弹出框中填写模型信息。

   在新建模型过程中可以直接从训练任务导入模型，也可以在创建后再从本地或训练任务导入模型。

   创建模型的操作如下所示：



4. 单击“确认”，完成模型创建。

   新创建模型中包含的模型版本数量为0，可以通过新建版本向其中导入模型。

   一个模型下各个版本应该都来自于同一个训练任务或者都是从本地导入的。

名称	ID	类型	来源	版本数量	创建时间	操作
demo	11589	图像分类	--	0	2021/07/15 15:04:05	新建版本 删除

从训练任务导入模型

在新建版本时可以从预置模型调参、Notebook、自定义作业中导入模型。

前提条件

存在成功的训练任务，已创建模型且并未发布。

操作步骤

1. 在左侧导航栏中选择“模型仓库”>“模型管理”。
2. 在模型列表页中，单击已创建的模型“Demo”所在行的“新建版本”，进入“新建版本”页面。
3. 填写如图所示的信息并从下拉菜单中选择要导入的模型的名称和版本：



4. 单击“提交”，完成模型导入。

从本地导入模型

在新建版本时可以导入本地模型。

前提条件

已创建模型，且该模型的模型来源为“本地上传”。

操作步骤

1. 在左侧导航栏中选择“模型仓库”>“模型管理”。
2. 在模型列表页中，单击“来源”为“本地上传”的模型所在行的“新建版本”，进入“新建版本”页面。
3. 根据页面提示填写相关信息以及上传文件：



- 对于图像分类、物体检测在上传文件时以及提交时进行文件校验，如校验出错，请根据提示进行修改。
- 对于表格数据预测在上传时不作校验，请自行保证文件的正确性。
- 不同类型的模型在部署时接口格式是不同的，请确保一个模型下的各个版本的模型类型一致。

4. 单击“提交”，完成模型导入。

查看模型

对于来自预置模型调参任务的模型，在导入模型仓库后可以查看其评估报告。

前提条件

已成功导入来自预置模型调参任务的模型。

操作步骤

1. 在左侧导航栏中选择“模型仓库”>“模型管理”。
2. 在模型列表页中，单击已创建的模型“Demo”所在行的“版本列表”，进入“Demo”模型的版本列表页面，如下所示。



3. 在版本列表中，单击“评估报告”即可对应版本的详情页面。
   - 只有来自预置模型调参的模型才具有评估报告。
   - 来自“Notebook”和“自定义作业”的模型无评估报告。
   - 从本地导入的模型无评估报告。

校验模型

通过校验功能，可以快速查看模型的评估效果。

从训练任务导入的模型支持模型校验；对于导入的模型，只有PaddlePaddle、TensorFlow、PyTorch以及Sklearn、XGBoost框架/库的模型支持校验，其它框架的不支持校验。

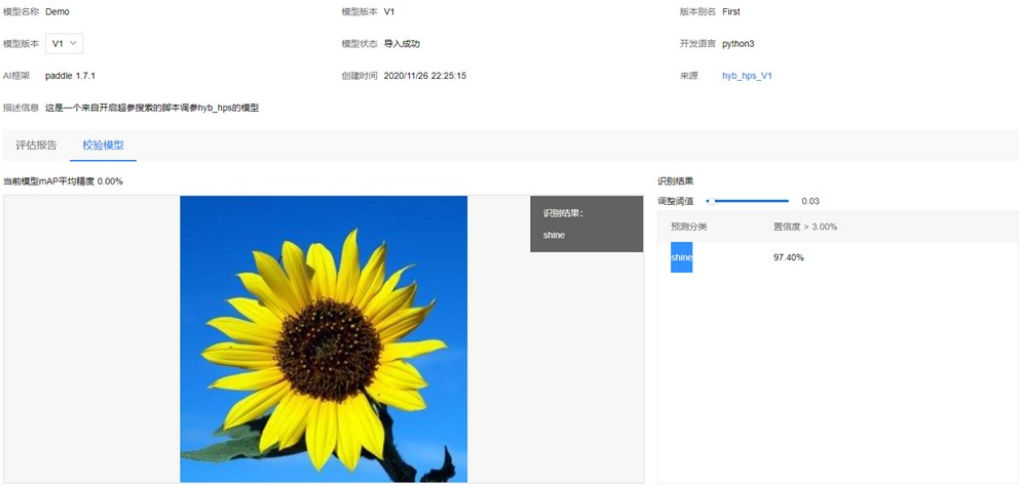
前提条件

已成功导入模型。

操作步骤

1. 在左侧导航栏中选择“模型仓库”>“模型管理”。
2. 在模型列表页中，单击已创建的模型“Demo”所在行的“新建版本”，进入“新建版本”页面。
3. 在版本列表中，单击“校验模型”，进入对应版本的校验页面。
4. 单击“启动模型校验”。

   校验模型需要在后端完成模型的部署，需要一定时间，请耐心等待。
5. 在校验服务启动后，上传图片完成校验，如下图所示：



服务代码文件示例

Sklearn服务代码文件示例

Sklearn服务代码文件示例

在模型仓库中导入基于Sklearn库的机器学习模型时，除需导入模型文件外，也需要导入服务代码文件，其中服务代码文件用于在线部署模型时进行模型文件的加载以及进行必要的预处理和后处理逻辑。

Sklearn模型服务代码示例如下所示：

```
 ##### !/usr/bin/env python
##### -*- coding: utf-8 -*-
##### ********************************************************************************
#
##### Copyright (c) 2020 Baidu.com, Inc. All Rights Reserved
#
##### ********************************************************************************
import numpy as np

ERROR_CODE_FAILED_DECODING_INPUT = 336005
MESSAGE_FAILED_DECODING_INPUT = "Failed decoding input"
ERROR_CODE_MISSING_REQUIRED_PARAMETER = 336006
MESSAGE_MISSING_REQUIRED_PARAMETER = "Missing required parameter"


class CustomException(RuntimeError):
    """
    进行模型验证和部署服务必需的异常类，缺少该类在代码验证时将会失败
    在处理异常数据或者请求时，推荐在`PredictWrapper`中的自定义预处理preprocess和后处理postprocess函数中抛出`CustomException`类，
    并为`message`指定准确可读的错误信息，以便在服务响应包中的`error_msg`参数中返回。
    """
    def __init__(self, error_code, message, orig_error=None):
        """ init with error_code, message and origin exception """
        super(CustomException, self).__init__(message)
        self.error_code = error_code
        self.orig_error = orig_error


class PredictWrapper(object):
    """ 模型服务预测封装类，支持用户自定义对服务请求数据的预处理和模型预测结果的后处理函数 """

    def __init__(self, model_path, use_gpu, logger):
        """
        根据`model_path`初始化`PredictWrapper`类，如解析label_list.txt，加载模型输出标签id和标签名称的映射关系
        :param model_path: 该目录下存放了用户选择的模型版本中包含的所有文件
        """
        # 加载模型
        model_filename = model_path + '/model.pkl'
        with open(model_filename, 'rb') as pk_fin:
            import pickle
            self._model = pickle.load(pk_fin)

    def preprocess(self, request_body):
        """
        自定义对请求体的预处理，针对图像类模型服务，包括对图片对图像的解析、转化等
        :param request_body: 请求体的json字典
        :return:
            data: 用于模型预测的输入。
            infer_args: 用于模型预测的其他参数
            request_context: 透传给自定义后处理函数`postprocess`的参数，例如指定返回预测结果的top N，过滤低score的阈值threshold.
        """
        try:
            features = request_body['features']
            features_np = np.array(features)
        except KeyError:
            raise CustomException(error_code=ERROR_CODE_MISSING_REQUIRED_PARAMETER,
                                  message=MESSAGE_FAILED_DECODING_INPUT)
        except Exception as e:
            raise CustomException(error_code=ERROR_CODE_FAILED_DECODING_INPUT,
                                  message=MESSAGE_FAILED_DECODING_INPUT, orig_error=e)
        return features_np, {}, {}

    def predict(self, data, infer_args):
        """
        模型预测
        :param data: 预处理后的数据
        :param infer_args: 预处理返回的`infer_args`
```

```
        :param infer_args: 预处理返回的 infer_args
        :return: infer_result 预测结果
        """
        return self._model.predict(data, **infer_args)

    def postprocess(self, infer_result, request_context):
        """
        对ml模型预测结果进行后处理
        :param infer_result: ML模型的预测结果
        :param request_context: 自定义预处理函数中返回的`request context`
        :return: request results 请求的处理结果
        """
        if not isinstance(infer_result, np.ndarray):
            infer_result = np.array(infer_result)
        return {'categories': infer_result.tolist()}
```

XGBoost服务代码文件示例

XGBoost服务代码文件示例

在模型仓库中导入基于XGBoost库的机器学习模型时，除需导入模型文件外，也需要导入服务代码文件，其中服务代码文件用于在线部署模型时进行模型文件的加载以及进行必要的预处理和后处理逻辑。

XGBoost模型服务代码示例如下所示：

```
 ##### !/usr/bin/env python
##### -*- coding: utf-8 -*-
##### ********************************************************************************
#
##### Copyright (c) 2020 Baidu.com, Inc. All Rights Reserved
#
##### ********************************************************************************
import numpy as np

ERROR_CODE_FAILED_DECODING_INPUT = 336005
MESSAGE_FAILED_DECODING_INPUT = "Failed decoding input"
ERROR_CODE_MISSING_REQUIRED_PARAMETER = 336006
MESSAGE_MISSING_REQUIRED_PARAMETER = "Missing required parameter"

from xgboost import DMatrix


class CustomException(RuntimeError):
    """
    进行模型验证和部署服务必需的异常类，缺少该类在代码验证时将会失败
    在处理异常数据或者请求时，推荐在`PredictWrapper`中的自定义预处理preprocess和后处理postprocess函数中抛出`CustomException`
类，
    并为`message`指定准确可读的错误信息，以便在服务响应包中的`error_msg`参数中返回。
    """
    def __init__(self, error_code, message, orig_error=None):
        """ init with error_code, message and origin exception """
        super(CustomException, self).__init__(message)
        self.error_code = error_code
        self.orig_error = orig_error


class PredictWrapper(object):
    """ 模型服务预测封装类，支持用户自定义对服务请求数据的预处理和模型预测结果的后处理函数 """

    def __init__(self, model_path, use_gpu, logger):
        """
        根据`model_path`初始化`PredictWrapper`类，如解析label_list.txt，加载模型输出标签id和标签名称的映射关系
        :param model_path: 该目录下存放了用户选择的模型版本中包含的所有文件
        """
        # 加载模型
        model_filename = model_path + '/model.pkl'
        try:
            with open(model_filename, 'rb') as pk_fin:
                import pickle
                self._model = pickle.load(pk_fin)
```

```
            self._model = pickle.load(pkl_fin)
        except:
            import joblib
            self._model = joblib.load(model_filename)

        algo = self._model.__class__.__name__
        self._is_booster = (algo == 'Booster')

    def preprocess(self, request_body):
        """
        自定义对请求体的预处理，针对图像类模型服务，包括对图片对图像的解析、转化等
        :param request_body: 请求体的json字典
        :return:
            data: 用于模型预测的输入。
            infer_args: 用于模型预测的其他参数
            request_context: 透传给自定义后处理函数`postprocess`的参数，例如指定返回预测结果的top N，过滤低score的阈值threshold.
        """
        try:
            features = request_body['features']
            features = np.array(features)
            if self._is_booster:
                features = DMatrix(features)
        except KeyError:
            raise CustomException(error_code=ERROR_CODE_MISSING_REQUIRED_PARAMETER,
                                  message=MESSAGE_MISSING_REQUIRED_PARAMETER)
        except Exception as e:
            raise CustomException(error_code=ERROR_CODE_FAILED_DECODING_INPUT,
                                  message=MESSAGE_FAILED_DECODING_INPUT, orig_error=e)
        return features, {}, {}

    def predict(self, data, infer_args):
        """
        模型预测
        :param data: 预处理后的数据
        :param infer_args: 预处理返回的`infer_args`
        :return: infer_result 预测结果
        """
        return self._model.predict(data, **infer_args)

    def postprocess(self, infer_result, request_context):
        """
        对ml模型预测结果进行后处理
        :param infer_result: ML模型的预测结果
        :param request_context: 自定义预处理函数中返回的`request context`
        :return: request results 请求的处理结果
        """
        if not isinstance(infer_result, np.ndarray):
            infer_result = np.array(infer_result)
        return {'categories': infer_result.tolist()}
```

预测部署

公有云部署管理

公有云部署简介

公有云部署即将模型中的模型部署为在线服务，从而以REST API的方式提供推理预测能力。

公有云部署概要

模型仓库中的模型与公有云部署即在线服务是一一对应的关系，即模型仓库中一个模型包含多个版本时，这些不同的版本只能部署到同一个在线服务中。

在线服务当前仅允许一个模型版本处于上线状态，若上线时有其它模型版本在线，则会将当前版本下线并上线新的版本。

对于在线的模型版本，可以执行扩缩容操作。

服务状态说明

服务状态及其含义如下所示：

模型状态	说明
运行中	服务运行中，会产生费用，当不适用时请下线模型
休眠中	不占用计算资源，不会产生费用，当有新的请求流量时，系统会自动将模型置为运行状态
停止	无运行的模型，不占用资源，不产生费用，无法处理新的预测请求
更新中	当执行模型版本切换时，服务处于更新中的状态
部署失败	由于暂无资源、模型文件不正确等导致模型部署失败，从而无法提供预测服务
模型待编辑	尚未配置资源的模型处于该状态

公有云部署支持的模型类型

模型仓库中的如下类型的模型支持共有云部署：

- 来自BML训练任务的模型，包括脚本调参、Notbeook以及自定义作业
- 对于用户从本地导入的模型，则如下类型的模型支持公有云部署：
    - 框架为PaddlePaddle、TensorFlow、PyTorch的深度学习模型
    - 基于Sklearn、XGBoost的机器学习模型

服务休眠策略

服务休眠是指若服务在指定的时长内无请求，则释放资源并进入休眠状态，该状态下不会产生费用，当接受到新的请求后，服务会自动恢复到正常状态（约1分钟），但休眠以及恢复过程中，接收的请求无法正常处理。

休眠策略状态说明：

休眠策略状态	说明
否	未配置休眠策略，新建的服务默认无休眠策略，处于该状态
生效中	表示当前休眠策略处于有效期内
未生效	表示当前尚未进入休眠策略的有效期
已失效	表示当前时间已超出了休眠策略的有效期

公有云部署

公有云部署是最快捷的模型部署方式，不同类型的模型在执行公有云部署时流程基本一致，当部署后在线API的接口与模型有关。

根据不同的模型类型，可以参考如下章节完成模型部署：

- 视觉模型公有云部署
- NLP模型公有云部署
- 表格预测模型公有云部署
- 通用模型公有云部署

配置休眠策略

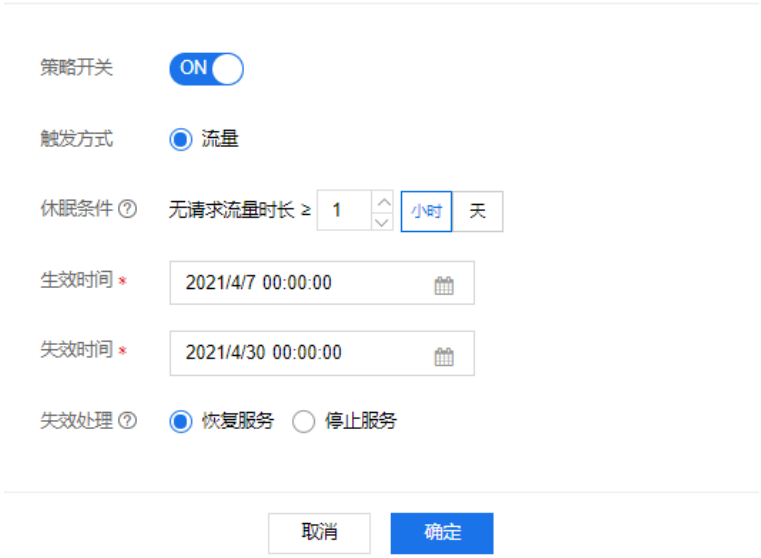
公有云部署支持休眠策略，从而实现服务的自动休眠，以帮助用户减少支出。

前提条件

已创建的在线服务支持配置休眠策略。

操作步骤

1. 在左侧导航栏中选择“模型部署”>“公有云部署”。
2. 在服务列表页中，单击已创建服务所在行的“设置休眠策略”。
3. 配置休眠策略，如下图所示：



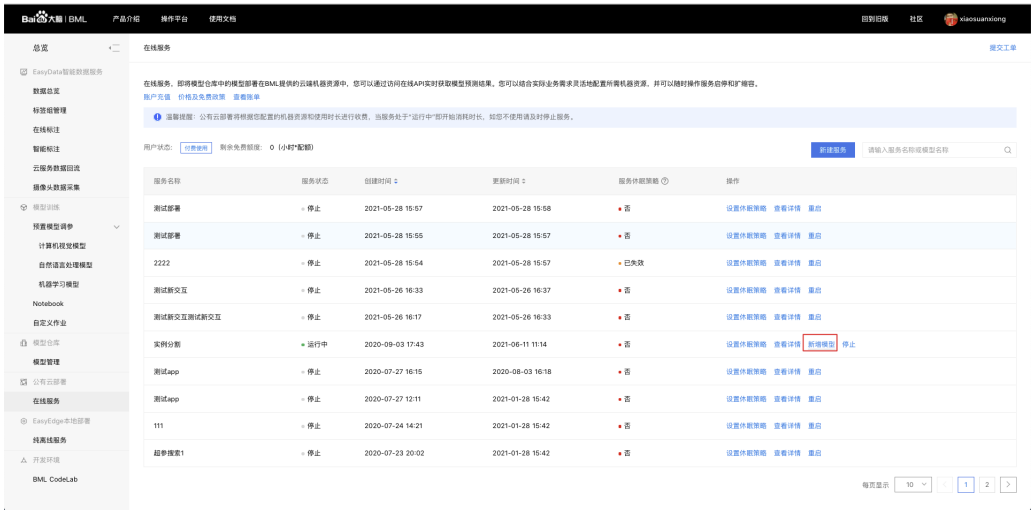
4. 单击“确定”完成配置。

配置AB测试版本

BML在线服务中，同一服务支持同时部署两个模型上线，并且可以自由的调节流量分配占比 **前提条件** 已创建的在线服务，处于**运行中**状态时，允许添加一个AB测试版本模型上线 **操作步骤**

1. 在左侧导航栏中选择“模型部署”>“在线服务”

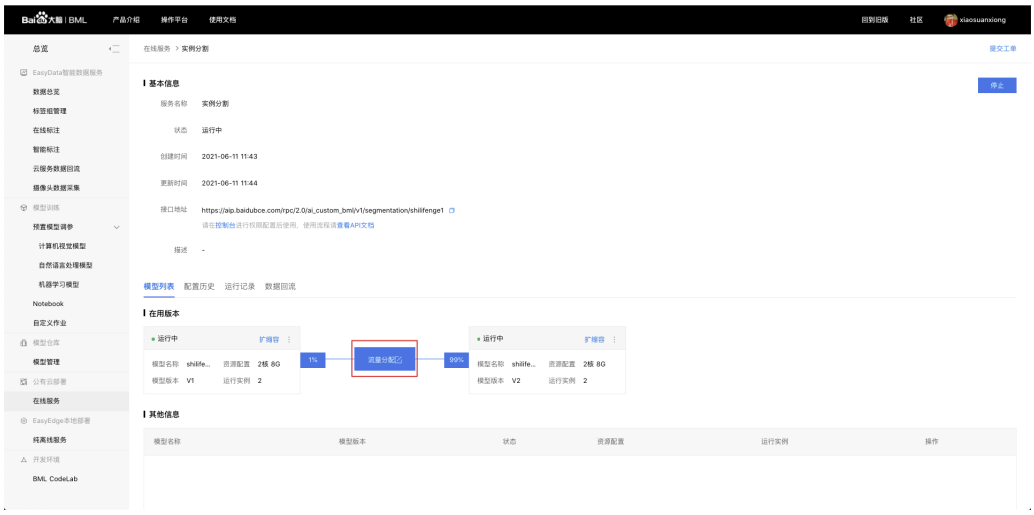
2. 服务列表中，对于处于**运行中**状态的服务，点击"新增版本"，添加AB测试版本



3. 配置AB测试版本模型，包括流量占比和资源配置

4. 点击部署，完成部署后，该服务下将有两个模型同时运行，访问流量按比例随机分配至其中一个。两个模型分别根据各自资源配置进行计费

5. 点击详情界面中的"流量占比"按钮，可以调节两个在线模型的流量占比



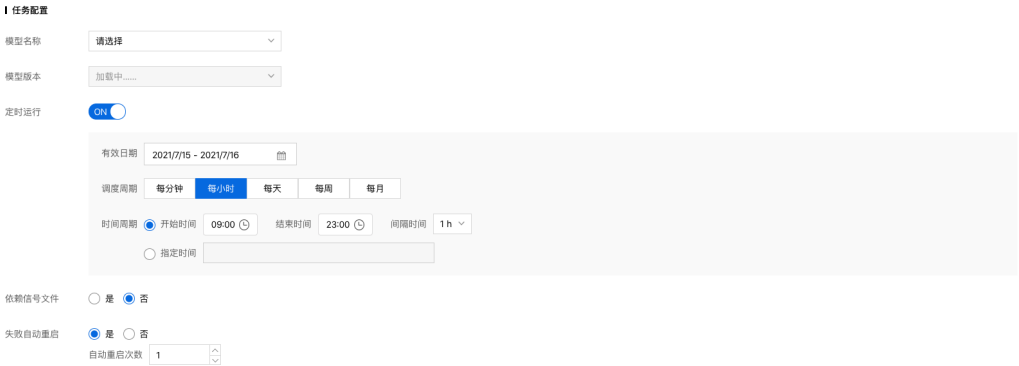
批量预测服务

批量预测是一种进行批量数据推理的方式。用户可以上传批量数据进行推理，在数据处理完成后自动停止。支持设置定时循环任务。

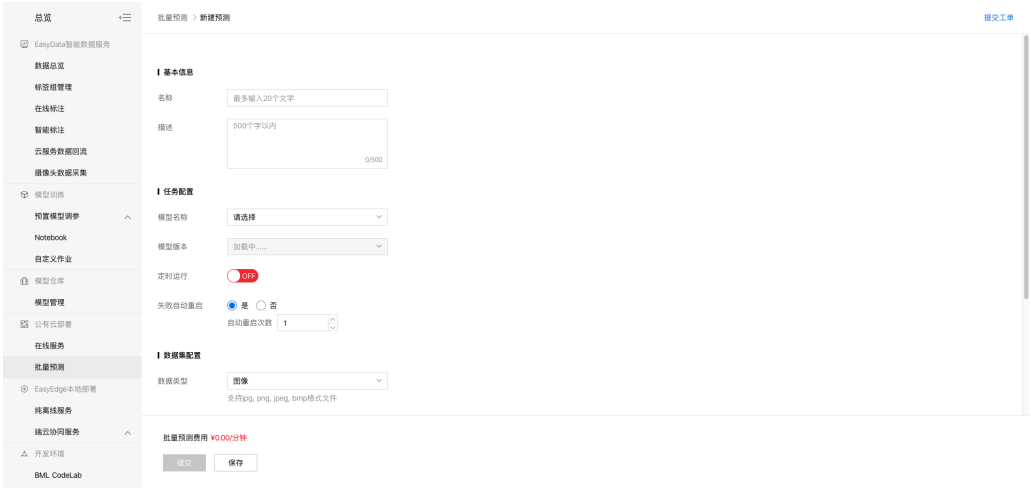
> 批量预测服务的入口在公有云部署的tab下，点击新建预测即可进入填写批量预测服务的详细表单。



> 批量预测中提供了定时运行的服务，在填写表单中您可以按照任务需求，按照分钟、小时、天、周、月的维度设置任务间隔周期，发起定时运行任务。



> 完成表单填写点击提交即可发起一个新的批量预测任务。



视觉模型部署

视觉任务模型部署整体说明

模型仓库中的视觉模型可部署在公有云服务器、私有服务器，封装成可离线运行的设备端SDK，或直接购买软硬一体方案，灵活适配各种使用场景及运行环境

您可以点击进入预训练网络与部署方式适配性页面，查询各预训练网络对下表所示的部署方式、硬件、系统的支持情况

部署方式	支持的硬件	支持的系统
公有云部署	可集成公有云API即可	不限制
私有服务器部署[私有API]	通用X86 CPU 英伟达 GPU	Linux
私有服务器部署[服务器端SDK]	通用X86 CPU	Linux
	英伟达 GPU	Linux/Windows
通用设备端SDK	通用ARM	Linux/Android/iOS
	海思NNIE	Linux
	华为达芬奇NPU	Android
	华为NPU	Android
	高通骁龙	Android
	苹果A仿生芯片	iOS
	通用X86 CPU	Windows
	英特尔®神经计算棒	Linux/Windows
专项硬件适配SDK[软硬一体方案]	Baidu-EdgeBoard(FZ)	Linux
	Baidu-EdgeBoard(VMX)	Linux/Windows
	Nvidia-Jetson(Nano/TX2/Xavier)	Linux

公有云API

训练完成的模型存储在云端，可通过独立Rest API调用模型，实现AI能力与业务系统或硬件设备整合

具有完善的鉴权、流控等安全机制，GPU集群稳定承载高并发请求

支持查找云端模型识别错误的数据，纠正结果并将其加入模型迭代的训练集，不断优化模型效果

私有服务器部署

可将训练完成的模型部署在私有CPU/GPU服务器上，支持私有API和服务器端SDK两种集成方式,可在内网/无网环境下使用模型，确保数据隐私

- 私有API：将模型以Docker形式在本地服务器（仅支持Linux）上部署为http服务，可调用与公有云API功能相同的接口。可纯离线完成部署，服务调用便捷
- 服务器端SDK：将模型封装成适配本地服务器（支持Linux和Windows）的SDK，可集成在其他程序中运行。首次联网激活后即可纯离线运行，占用服务器资源更少，使用方法更灵活 提供基础版、加速版（已支持通用x86、通用ARM芯片）两种版本，可根据业务场景需求选择。

训练完成的模型被打包成适配智能硬件的SDK，可进行设备端离线计算。满足推理阶段数据敏感性要求、更快的响应速度要求

支持iOS、Android、Linux、Windows四种操作系统，基础接口封装完善，满足灵活的应用侧二次开发

软硬一体方案

模型训练完成后，可前往AI市场购买EasyDL&EdgeBoard软硬一体方案

可应用于工业分拣、视频监控等多种设备端离线计算场景，让离线AI落地更轻松。了解更多

公有云部署

视觉任务公有云部署

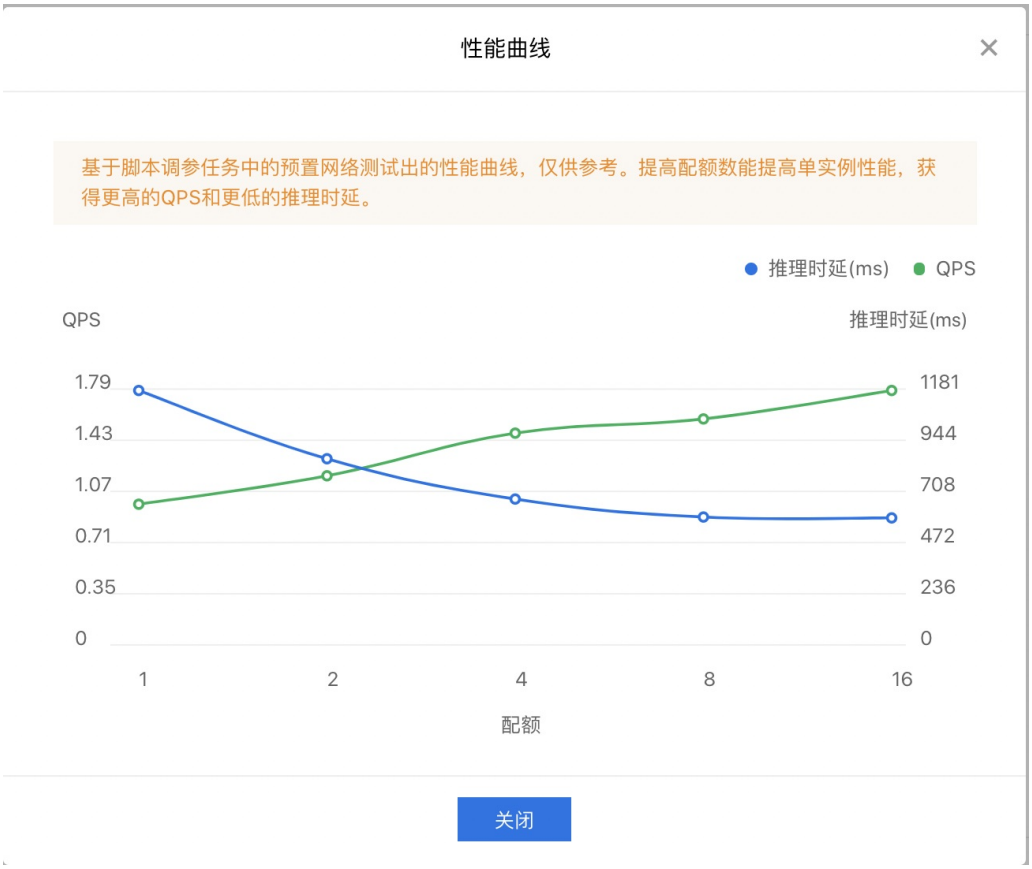
模型仓库中的视觉模型，发布为公有云部署时储在云端，可通过独立Rest API调用模型，实现AI能力与业务系统或硬件设备整合。
BML具有完善的鉴权、流控等安全机制，并配置丰富的资源集群稳定承载高并发请求。 并且支持查找云端模型识别错误的数据，纠正结果并将其加入模型迭代的训练集，不断优化模型效果

公有云部署 模型部署流程

1. 将训练完成的模型发布到模型仓库，在模型仓库中的『版本列表』中点击『公有云部署』或者点击左侧导航栏的公有云部署，进入公有云部署界面后点击『部署模型』。
2. 在部署界面，首先进行服务配置：自定义服务名称、接口地址后缀，选择需要部署的模型，
3. 对每个部署模型还需要进行资源配置，配置模型预测服务所需机器资源。目前资源开放CPU类型，支持按配额与实例进行灵活配置，计费说明请点击查看



4. 当部署模型为脚本调参任务产出的模型时，系统会自动显示该模型在当前配置下的预估时延和QPS。点击"性能曲线"，即可查看**单实例**下的推理时延，QPS与配额的曲线关系，您可以根据性能曲线配置合理的配额。



5. 配置完成后，点击开始部署即可一键部署模型，并获得专用的API地址。

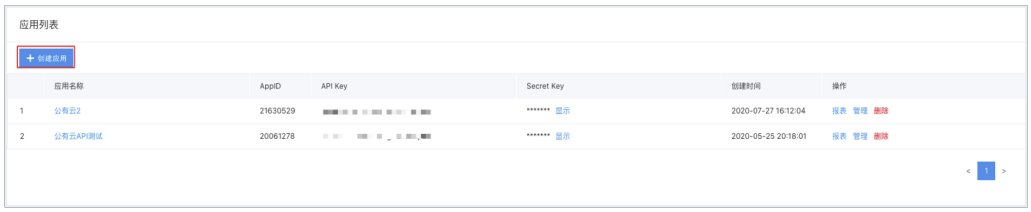
> 说明：
>
> 1、系统将运行公有云部署的机器资源1核cpu和4G内存折算为1配额，提高配额数量能提升QPS和降低推理时延
>
> 2、实例为cpu资源运行的实例数，为保证服务的稳定性，实例数至少需要配置为2。实例数与QPS成正比关系，提高实例数时，QPS能正比增加

接口赋权

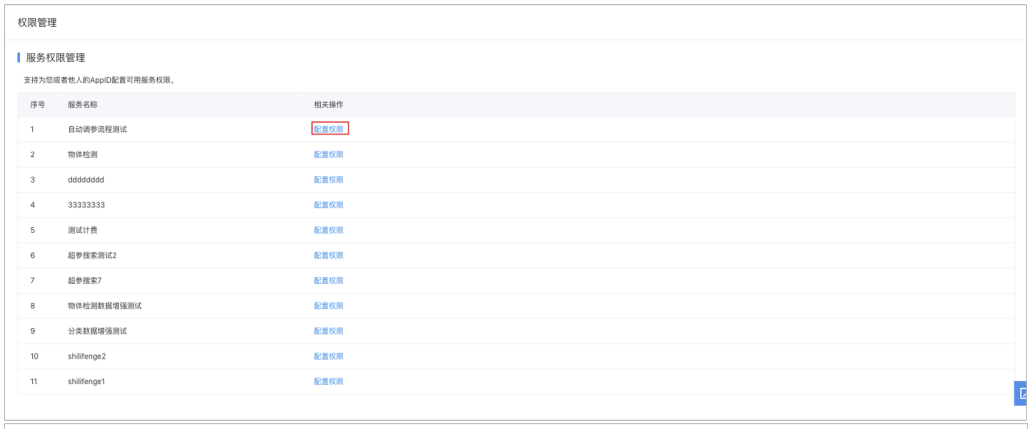
1. 正式使用还需要对接口进行赋权，在已发布的服务的详情界面，点击『控制台』进入BML控制台。



2. 在控制台中创建一个应用，获得由一串数字组成的appid，然后就可以参考接口文档正式使用了



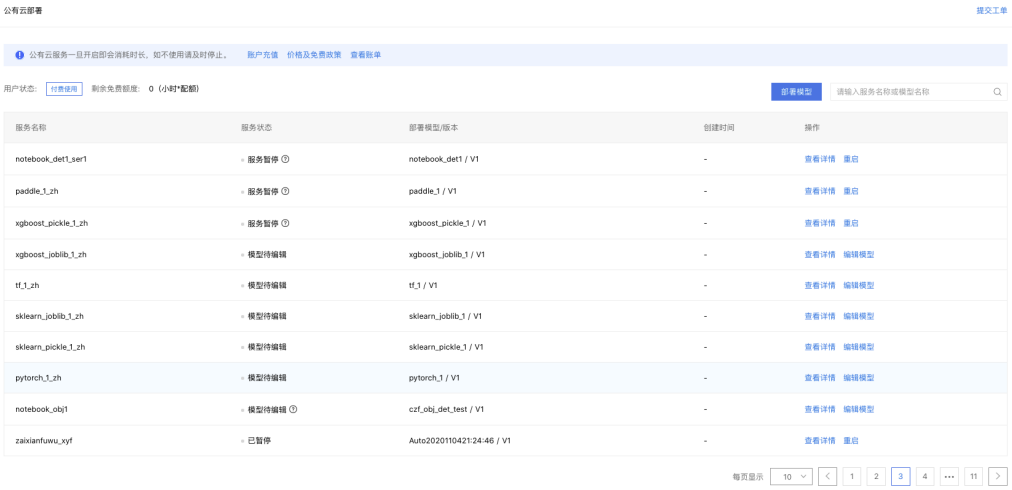
3. 如需为第三方用户配置调用接口的权限，可以前往BML控制台-公有云服务管理-权限管理配置





公有云部署列表

公有云部署列表信息由历史提交公有云服务组成，显示服务的基本信息。



如图，在预测服务列表页支持以下操作：

查看详情：点击进入该公有云服务的详情界面

重启：对已停止或部署失败的服务进行重新创建，恢复对应的服务为可用状态

停止：停止该公有云服务，用户不再使用相关服务后可进行这一操作，停止后将不进行计费

编辑模型：对于已经部署为公有云服务，但尚未配置机器资源的模型，点击可以配置机器资源

图像分类-单图单标签API参考文档

本文档主要说明定制化模型发布后获得的API如何使用，如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动
- 加入BML官方QQ群（群号:868826008）联系群管

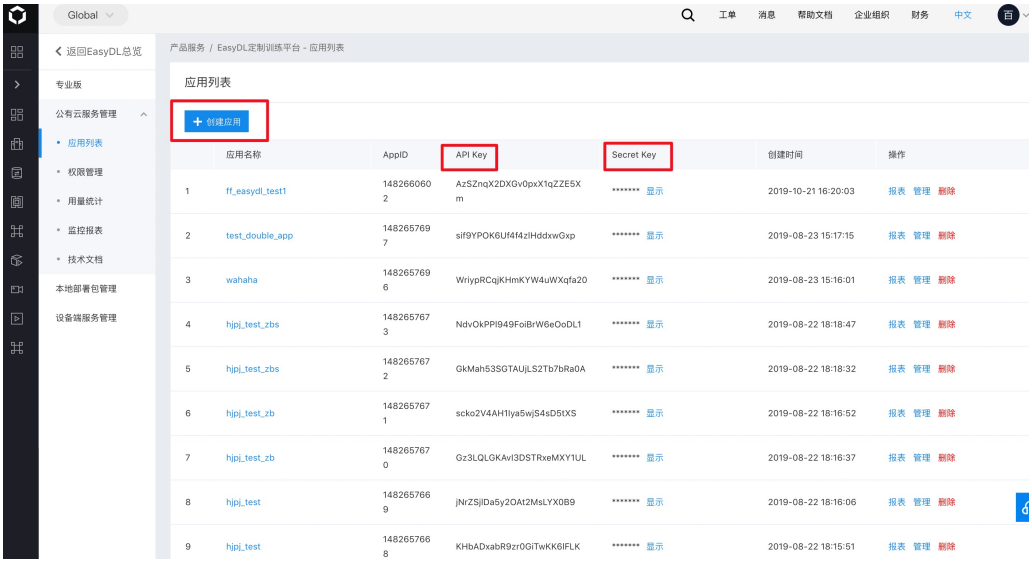
接口描述

基于自定义训练出的图像分类模型，实现个性化图像识别。模型训练完毕后发布可获得定制化图像分类API

接口鉴权

1、在BML控制台创建应用

2、应用列表页获取AK SK



请求说明

请求示例

HTTP 方法：`POST`

请求URL： 请首先进行自定义模型训练，完成训练后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

注意：如果出现336001的错误码很可能是因为请求方式错误，与其他图像识别服务不同的是定制化图像识别服务以json方式请求。

Body请求示例：

```
{
  "image": "<base64数据>",
  "top_num": 5
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
image	是	string	-	图像数据，base64编码，要求base64编码后大小不超过4M，最短边至少15px，最长边最大4096px,支持jpg/png/bmp格式 **注意请去掉头部**
top_num	否	number	-	返回分类数量，默认为6个

请求代码示例

提示一：使用示例代码前，请记得替换其中的示例Token、图片地址或Base64信息。

提示二：部分语言依赖的类或库，请在代码注释中查看下载地址。

PHP

Java

Python2

Python3

C++

```php
<?php
/**
 * 发起http post请求(REST API), 并获取REST请求的结果
 * @param string $url
 * @param string $param
 * @return - http response body if succeeds, else false.
 */
function request_post($url = '', $param = '')
{
    if (empty($url) || empty($param)) {
        return false;
    }

    $postUrl = $url;
    $curlPost = $param;
    // 初始化curl
    $curl = curl_init();
    curl_setopt($curl, CURLOPT_URL, $postUrl);
```

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
results	否	array(object)	分类结果数组
+name	否	string	分类名称
+score	否	number	置信度

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如Access Token失效返回：

```
{
 "error_code": 110,
 "error_msg": "Access token invalid or no longer valid"
}
```

需要重新获取新的Access Token再次请求即可。

错误码	错误信息	描述
1	Unknown error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
2	Service temporarily unavailable	服务暂不可用，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
3	Unsupported openapi method	调用的API不存在，请检查后重新尝试
4	Open api request limit reached	集群超限额
6	No permission to access data	无权限访问该用户数据
13	Get service token failed	获取token失败
14	IAM Certification failed	IAM鉴权失败
15	app not exists or create failed	应用不存在或者创建失败
17	Open api daily request limit reached	每天请求量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
18	Open api qps request limit reached	QPS超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
19	Open api total request limit reached	请求总量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
100	Invalid parameter	无效的access_token参数，请检查后重新尝试
110	Access token invalid or no longer valid	access_token无效
111	Access token expired	access token过期
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（868826008）或工单联系技术支持团队
336001	Invalid Argument	入参格式有误，比如缺少必要参数、图片base64编码错误等等，可检查下图片编码、代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数或者代码格式有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336004	输入文件大小不合法	图片超出大小限制，图片限4M以内，请根据接口文档检查入参格式，有疑问请通过QQ群（868826008）或工单联系技术支持团队
336005	图片解码失败	图片编码错误（非jpg,bmp,png等常见图片格式），请检查并修改图片格式
336006	缺失必要参数	image字段缺失（未上传图片）
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请通过QQ群（868826008）或工单联系技术支持团队

图像分类-单图多标签API参考文档

本文档主要说明定制化模型发布后获得的API如何使用，如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动
- 加入BML官方QQ群（群号:868826008）联系群管

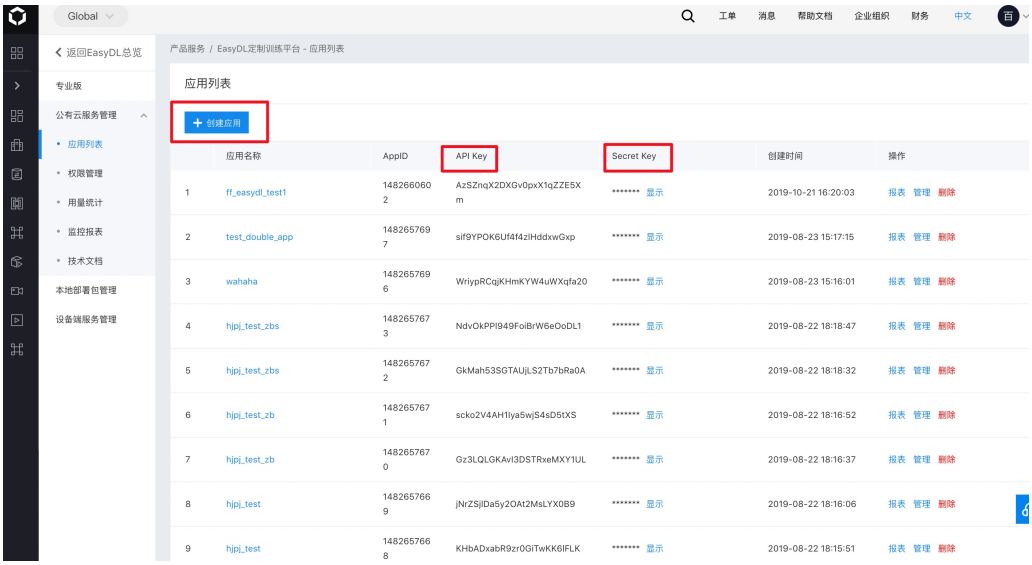
接口描述

基于自定义训练出的图像分类模型，实现个性化图像识别。模型训练完毕后发布可获得定制化图像分类API

接口鉴权

1、在BML——控制台创建应用

2、应用列表页获取AK SK



请求说明

请求示例

HTTP 方法：`POST`

请求URL： 请首先进行自定义模型训练，完成训练后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

注意：如果出现336001的错误码很可能是因为请求方式错误，与其他图像识别服务不同的是定制化图像识别服务以json方式请求。

Body请求示例：

```
{
  "image": "<base64数据>",
  "threshold": 0.5
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
image	是	string	-	图像数据，base64编码，要求base64编码后大小不超过4M，最短边至少15px，最长边最大4096px,支持jpg/png/bmp格式 **注意请去掉头部**
threshold	否	number	-	阈值，默认值为推荐阈值（0-1之间），具体值可在我的训练任务列表-模型效果查看

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
results	否	array(object)	分类结果数组
+name	否	string	分类名称
+score	否	number	置信度

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如Access Token失效返回：

```
 {
  "error_code": 110,
  "error_msg": "Access token invalid or no longer valid"
}
```

需要重新获取新的Access Token再次请求即可。

错误码	错误信息	描述
1	Unknown error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
2	Service temporarily unavailable	服务暂不可用，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
3	Unsupported openapi method	调用的API不存在，请检查后重新尝试
4	Open api request limit reached	集群超限额
6	No permission to access data	无权限访问该用户数据
13	Get service token failed	获取token失败
14	IAM Certification failed	IAM鉴权失败
15	app not exists or create failed	应用不存在或者创建失败
17	Open api daily request limit reached	每天请求量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
18	Open api qps request limit reached	QPS超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
19	Open api total request limit reached	请求总量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
100	Invalid parameter	无效的access_token参数，请检查后重新尝试
110	Access token invalid or no longer valid	access_token无效
111	Access token expired	access token过期
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（868826008）或工单联系技术支持团队
336001	Invalid Argument	入参格式有误，比如缺少必要参数、图片base64编码错误等等，可检查下图片编码、代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数或者代码格式有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336004	输入文件大小不合法	图片超出大小限制，图片限4M以内，请根据接口文档检查入参格式，有疑问请通过QQ群（868826008）或工单联系技术支持团队
336005	图片解码失败	图片编码错误（非jpg,bmp,png等常见图片格式），请检查并修改图片格式
336006	缺失必要参数	image字段缺失（未上传图片）
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请通过QQ群（868826008）或工单联系技术支持团队

物体检测API参考文档

本文档主要说明定制化模型发布后获得的API如何使用，如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动
- 加入BML官方QQ群（群号:868826008）联系群管

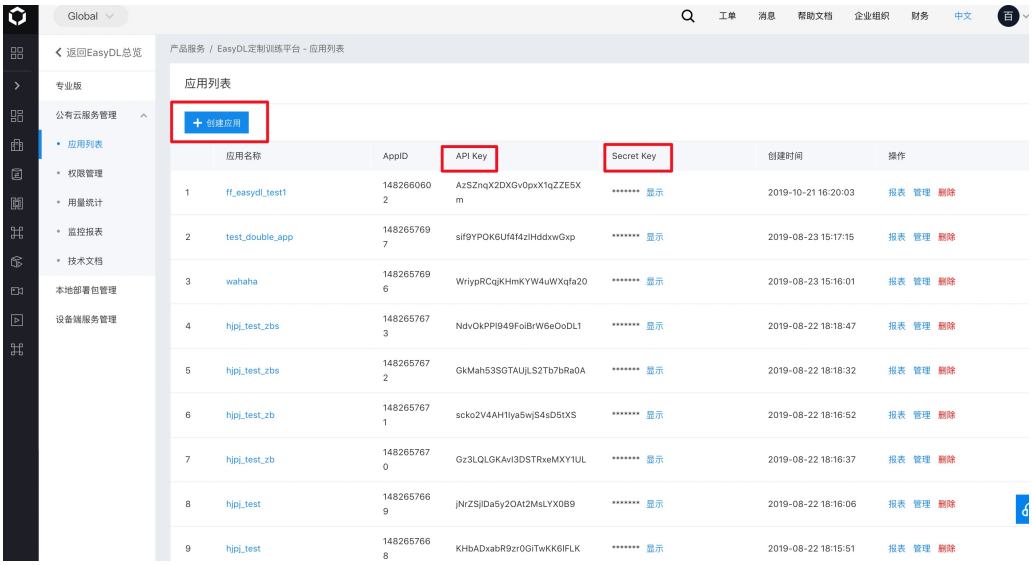
接口描述

基于自定义训练出的物体检测模型，实现个性化图像识别。模型训练完毕后发布可获得定制物体检测API

接口鉴权

1、在BML——控制台创建应用

2、应用列表页获取AK SK



请求说明

请求示例

HTTP 方法：POST

请求URL： 请首先进行自定义模型训练，完成训练后申请上线，上线成功后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

注意：如果出现336001的错误码很可能是因为请求方式错误，与其他图像识别服务不同的是定制化图像识别服务以json方式请求。

Body请求示例：

```
{
  "image": "<base64数据>"
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
image	是	string	-	图像数据，base64编码，要求base64编码后大小不超过4M，最短边至少15px，最长边最大4096px,支持jpg/png/bmp格式 **注意请去掉头部**
threshold	否	number	-	阈值，默认值为推荐阈值（0-1之间），具体值可在我的训练任务列表-模型效果查看

请求代码示例

提示一：使用示例代码前，请记得替换其中的示例Token、图片地址或Base64信息。

提示二：部分语言依赖的类或库，请在代码注释中查看下载地址。

PHP

Java

Python2

Python3

C++

```php
<?php
/**
 * 发起http post请求(REST API), 并获取REST请求的结果
 * @param string $url
 * @param string $param
 * @return - http response body if succeeds, else false.
 */
function request_post($url = '', $param = '')
{
    if (empty($url) || empty($param)) {
        return false;
    }

    $postUrl = $url;
    $curlPost = $param;
    // 初始化curl
    $curl = curl_init();
    curl_setopt($curl, CURLOPT_URL, $postUrl);
```

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
results	否	array(object)	识别结果数组
+name	否	string	分类名称
+score	否	number	置信度
+location	否		
++left	否	number	检测到的目标主体区域到图片左边界的距离
++top	否	number	检测到的目标主体区域到图片上边界的距离
++width	否	number	检测到的目标主体区域的宽度
++height	否	number	检测到的目标主体区域的高度

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如Access Token失效返回：

```
{
 "error_code": 110,
 "error_msg": "Access token invalid or no longer valid"
}
```

需要重新获取新的Access Token再次请求即可。

错误码	错误信息	描述
1	Unknown error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
2	Service temporarily unavailable	服务暂不可用，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
3	Unsupported openapi method	调用的API不存在，请检查后重新尝试
4	Open api request limit reached	集群超限额
6	No permission to access data	无权限访问该用户数据
13	Get service token failed	获取token失败
14	IAM Certification failed	IAM鉴权失败
15	app not exsits or create failed	应用不存在或者创建失败
17	Open api daily request limit reached	每天请求量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
18	Open api qps request limit reached	QPS超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
19	Open api total request limit reached	请求总量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
100	Invalid parameter	无效的access_token参数，请检查后重新尝试
110	Access token invalid or no longer valid	access_token无效
111	Access token expired	access token过期
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（868826008）或工单联系技术支持团队
336001	Invalid Argument	入参格式有误，比如缺少必要参数、图片base64编码错误等等，可检查下图片编码、代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数或代码格式有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336004	输入文件大小不合法	图片超出大小限制，图片限4M以内，请根据接口文档检查入参格式，有疑问请通过QQ群（868826008）或工单联系技术支持团队
336005	图片解码失败	图片编码错误（非jpg,bmp,png等常见图片格式），请检查并修改图片格式
336006	缺失必要参数	image字段缺失（未上传图片）
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请通过QQ群（868826008）或工单联系技术支持团队

实例分割API参考文档

本文档主要说明定制化模型发布后获得的API如何使用，如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动
- 加入BML官方QQ群（群号:868826008）联系群管

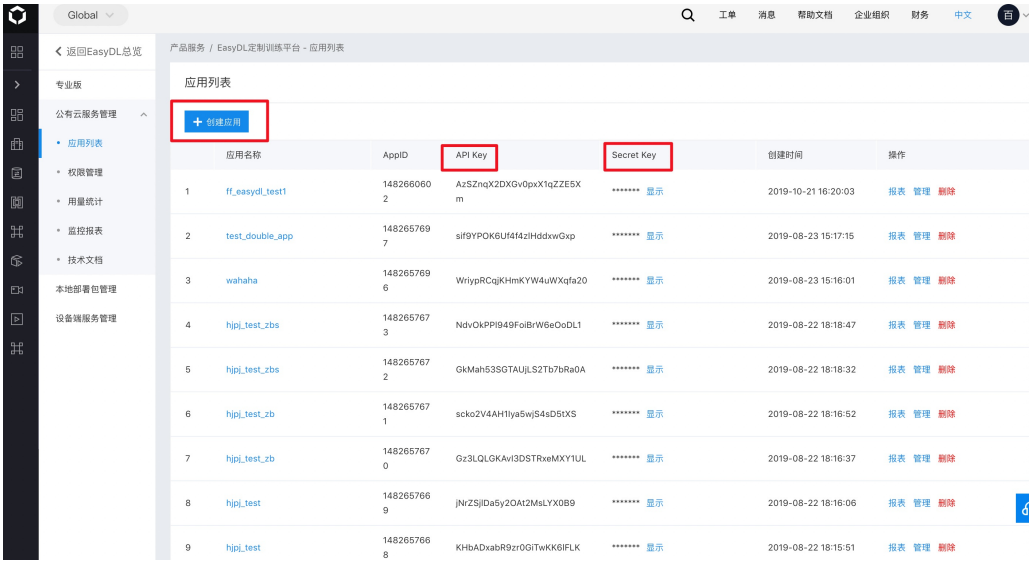
接口描述

基于自定义训练出的实例分割模型，实现个性化图像识别。模型训练完毕后发布可获得定制实例分割API

接口鉴权

1、在BML——控制台创建应用

2、应用列表页获取AK SK



请求说明

请求示例

HTTP 方法：`POST`

请求URL： 请首先进行自定义模型训练，完成训练后申请上线，上线成功后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

注意：如果出现336001的错误码很可能是因为请求方式错误，与其他图像识别服务不同的是定制化图像识别服务以json方式请求。

Body请求示例：

```
{
  "image": "<base64数据>"
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
image	是	string	-	图像数据，base64编码，要求base64编码后大小不超过4M，最短边至少15px，最长边最大4096px，通用算法训练的模型发布API支持jpg/png/bmp格式。肺炎CT影像识别专用算法训练发布的API在此基础上支持dicom格式，限制4M以内。 **注意请去掉头部**
threshold	否	number	-	默认值为推荐阈值，请在我的模型列表-模型效果查看推荐阈值

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
results	否	array(object)	识别结果数组
+name	否	string	分类名称
+score	否	number	置信度
+location	否		
++left	否	number	检测到的目标主体区域到图片左边界的距离
++top	否	number	检测到的目标主体区域到图片上边界的距离
++width	否	number	检测到的目标主体区域的宽度
++height	否	number	检测到的目标主体区域的高度
+mask	否	array	基于游程编码的字符串，编码内容为和原图宽高相同的布尔数组：若数组值为0，代表原图此位置像素点不属于检测目标，若为1，代表原图此位置像素点属于检测目标。下载解码SDK

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如Access Token失效返回：

```
{
 "error_code": 110,
 "error_msg": "Access token invalid or no longer valid"
}
```

需要重新获取新的Access Token再次请求即可。

错误码	错误信息	描述
1	Unknown error	服务器内部错误，请再次请求， 如果持续出现此类错误，有疑问请通过QQ群（868826008）或工单联系技术支持团队。
2	Service temporarily unavailable	服务暂不可用，请再次请求， 如果持续出现此类错误，有疑问请通过QQ群（868826008）或工单联系技术支持团队。
3	Unsupported openapi method	调用的API不存在，请检查后重新尝试
4	Open api request limit reached	集群超限额
6	No permission to access data	无权限访问该用户数据
13	Get service token failed	获取token失败
14	IAM Certification failed	IAM鉴权失败
15	app not exists or create failed	应用不存在或者创建失败
17	Open api daily request limit reached	每天请求量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度智能云控制台内提交工单反馈
18	Open api qps request limit reached	QPS超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度智能云控制台内提交工单反馈
19	Open api total request limit reached	请求总量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度智能云控制台内提交工单反馈
100	Invalid parameter	无效的access_token参数，请检查后重新尝试
110	Access token invalid or no longer valid	access_token无效
111	Access token expired	access token过期
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请在百度智能云控制台内提交工单反馈
336001	Invalid Argument	入参格式有误，比如缺少必要参数、图片base64编码错误等等，可检查下图片编码、代码格式是否有误。有疑问请在百度云控制台内提交工单反馈
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数或代码格式有误。有疑问请在百度智能云控制台内提交工单反馈
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请在百度智能云控制台内提交工单反馈
336004	输入文件大小不合法	图片超出大小限制，图片限4M以内，请根据接口文档检查入参格式，有疑问请在百度智能云控制台内提交工单反馈
336005	图片解码失败	图片编码错误（非jpg,bmp,png等常见图片格式），请检查并修改图片格式
336006	缺失必要参数	image字段缺失（未上传图片）
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请在百度智能云控制台内提交工单反馈

文字识别API参考文档

本文档主要说明定制化模型发布后获得的API如何使用，如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动
- 加入BML官方QQ群（群号:868826008）联系群管

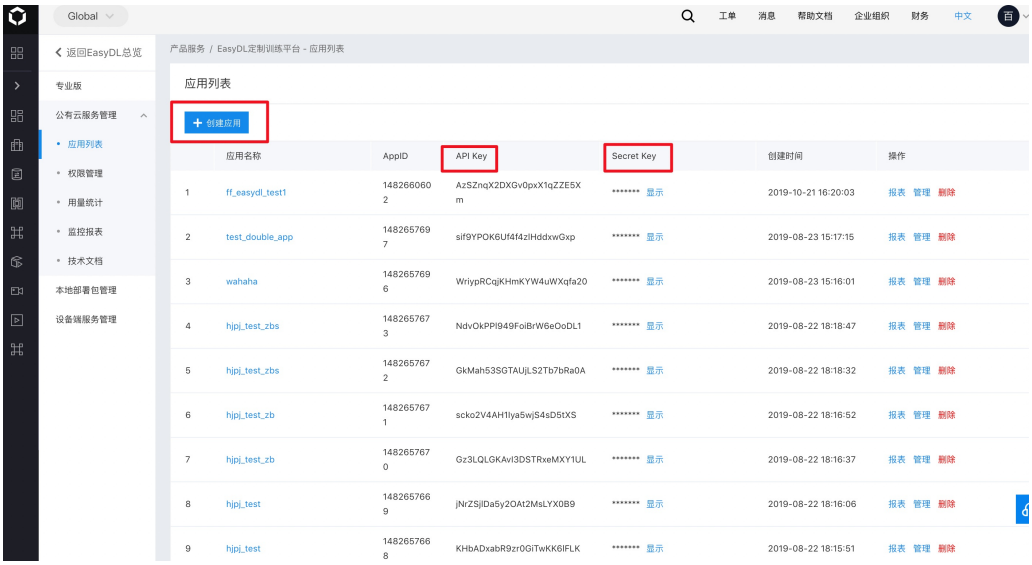
接口描述

基于自定义训练出的文字识别模型，实现个性化文字识别。模型训练完毕后发布可获得定制化文字识别API。

接口鉴权

1、在BML控制台创建应用

2、应用列表页获取AK SK



请求说明

请求示例

HTTP 方法：`POST`

请求URL： 请首先进行自定义模型训练，完成训练后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

注意：如果出现336001的错误码很可能是因为请求方式错误，与其他图像识别服务不同的是定制化图像识别服务以json方式请求。

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
image	是	string	-	图像数据，base64编码，要求base64编码后大小不超过4M，最短边至少15px，最长边最大4096px,支持jpg/png/bmp格式 **注意请去掉头部**
language_type	否	string	auto_detect、CHN_ENG、ENG、JAP、KOR、FRE、SPA、POR、GER、ITA、RUS、DAN、DUT、MAL、SWE、IND、POL、ROM、TUR、GRE、HUN	识别语言类型，默认为CHN_ENG。可选值包括：- auto_detect：自动检测语言，并识别 - CHN_ENG：中英文混合 - ENG：英文 - JAP：日语 - KOR：韩语 - FRE：法语 - SPA：西班牙语 - POR：葡萄牙语 - GER：德语 - ITA：意大利语 - RUS：俄语 - DAN：丹麦语 - DUT：荷兰语 - MAL：马来语 - SWE：瑞典语 - IND：印尼语 - POL：波兰语 - ROM：罗马尼亚语 - TUR：土耳其语 - GRE：希腊语 - HUN：匈牙利语 - THA：泰语 - VIE：越南语 - ARA：阿拉伯语 - HIN：印地语

请求代码示例

提示一：使用示例代码前，请记得替换其中的示例Token、图片地址或Base64信息。

提示二：部分语言依赖的类或库，请在代码注释中查看下载地址。

```
PHP
```

Java

Python3

C++

```php
<?php
/**
* 发起http post请求(REST API), 并获取REST请求的结果
* @param string $url
* @param string $param
* @return - http response body if succeeds, else false.
*/
function request_post($url = '', $param = '')
{
    if (empty($url) || empty($param)) {
        return false;
    }

    $postUrl = $url;
    $curlPost = $param;
    // 初始化curl
    $curl = curl_init();
    curl_setopt($curl, CURLOPT_URL, $postUrl);
```

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
words_results	是	array[]	识别结果数组
words_result_num	是	uint32	识别结果数，表示words_result的元素个数
+score	否	number	置信度

返回示例

```
{
  "log_id": 1390582998516105216,
  "words_result_num": 2
  "words_result": [
     {
        "words": " OCR"
     },
     {
        "words": "百度通用文字识别高精度版"
     }
  ]
}
```

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如Access Token失效返回：

```
{
 "error_code": 110,
 "error_msg": "Access token invalid or no longer valid"
}
```

需要重新获取新的Access Token再次请求即可。

错误码	错误信息	描述
1	Unknown error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
2	Service temporarily unavailable	服务暂不可用，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
3	Unsupported openapi method	调用的API不存在，请检查后重新尝试
4	Open api request limit reached	集群超限额
6	No permission to access data	无权限访问该用户数据
13	Get service token failed	获取token失败
14	IAM Certification failed	IAM鉴权失败
15	app not exists or create failed	应用不存在或者创建失败
17	Open api daily request limit reached	每天请求量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
18	Open api qps request limit reached	QPS超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
19	Open api total request limit reached	请求总量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
100	Invalid parameter	无效的access_token参数，请检查后重新尝试
110	Access token invalid or no longer valid	access_token无效
111	Access token expired	access token过期
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（868826008）或工单联系技术支持团队
336001	Invalid Argument	入参格式有误，比如缺少必要参数、图片base64编码错误等等，可检查下图片编码、代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数或者代码格式有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336004	输入文件大小不合法	图片超出大小限制，图片限4M以内，请根据接口文档检查入参格式，有疑问请通过QQ群（868826008）或工单联系技术支持团队
336005	图片解码失败	图片编码错误（非jpg,bmp,png等常见图片格式），请检查并修改图片格式
336006	缺失必要参数	image字段缺失（未上传图片）
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请通过QQ群（868826008）或工单联系技术支持团队

私有服务器部署

视觉模型如何部署在私有服务器

在发布模型页面中，可以申请私有服务器部署，将模型部署在私有服务器中。

支持「私有API」和「服务器端SDK」两种集成方式：

- 私有API：将模型以Docker形式在私有服务器（仅支持Linux）上部署为http服务，可调用与公有云API功能相同的接口。可纯离线完成部署，服务调用便捷

- 服务器端SDK：将模型封装成适配私有服务器（支持Linux和Windows）的SDK，可集成在其他程序中运行。首次联网激活后即可纯离线运行，占用服务器资源更少，使用方法更灵活

私有API

图像分类-单图单标签私有API集成文档

本文档主要说明定制化模型本地部署后，如何使用本地API。如还未训练模型，请先前往BML进行训练。

如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动
- 加入BML官方QQ群（群号:868826008）联系群管

部署包使用说明

BML定制化图像分类模型的本地部署通过EasyPack实现，目前提供单机一键部署的方式。

在BML控制台申请、下载部署包后，可以参考EasyPack-单机一键部署将软件包部署在本地服务器上。部署成功后，启动服务，即可调用与在线API功能类似的接口。

授权说明

本地部署包根据服务器硬件（CPU单机或GPU单卡）进行授权，只能在申请时提交的硬件指纹所属的硬件上使用。

部署包测试期为1个月，如需购买永久授权，可提交工单咨询

API参考

请求说明

请求示例

HTTP 方法：`POST`

请求URL： 请首先在BML进行自定义模型训练，完成训练后申请本地部署，本地部署成功后拼接url。

请求URL： http://{IP}:{PORT}/{DEPLOY_NAME}/ImageClassification IP：服务本地部署所在机器的ip地址 PORT：服务部署后获取的端口 DEPLOY_NAME：申请时填写的本地服务名称

Header如下：

参数	值
Content-Type	application/json

注意：如果出现336001的错误码很可能是因为请求方式错误，与其他图像识别服务不同的是定制化图像识别服务以json方式请求。

Body请求示例：

```
{
  "image": "<base64数据>",
  "top_num": 5
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
image	是	string	-	图像数据，base64编码，要求base64编码后大小不超过4M，最短边至少15px，最长边最大4096px,支持jpg/png/bmp格式 **注意请去掉头部**
top_num	否	number	-	返回分类数量，默认为6个

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
results	否	array(object)	分类结果数组
+name	否	string	分类名称
+score	否	number	置信度

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如缺少必要出入参时返回：

```
{
 "error_code": 336001,
 "error_msg": "Invalid Argument"
}
```

错误码	错误信息	描述
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（868826008）或工单联系技术支持团队
336001	Invalid Argument	入参格式有误，比如缺少必要参数、图片base64编码错误等等，可检查下图片编码、代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336004	输入文件大小不合法	图片超出大小限制，图片限4M以内，请根据接口文档检查入参格式，有疑问请通过QQ群（868826008）或工单联系技术支持团队
336005	图片解码失败	图片编码错误（非jpg,bmp,png等常见图片格式），请检查并修改图片格式
336006	缺失必要参数	image字段缺失（未上传图片）
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请通过QQ群（868826008）或工单联系技术支持团队
337000	Auth check failed	离线鉴权调用失败

图像分类-单图多标签私有API集成文档

本文档主要说明定制化模型本地部署后，如何使用本地API。如还未训练模型，请先前往BML进行训练。

如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动
- 加入BML官方QQ群（群号:868826008）联系群管

部署包使用说明

BML定制化图像分类模型的本地部署通过EasyPack实现，目前提供单机一键部署的方式。

在BML 控制台申请、下载部署包后，可以参考EasyPack-单机一键部署将软件包部署在本地服务器上。部署成功后，启动服务，即可调用与在线API功能类似的接口。

授权说明

本地部署包根据服务器硬件（CPU单机或GPU单卡）进行授权，只能在申请时提交的硬件指纹所属的硬件上使用。

部署包测试期为1个月，如需购买永久授权，可提交工单咨询

API参考

请求说明

请求示例

HTTP 方法：`POST`

请求URL： 请首先在BML进行自定义模型训练，完成训练后申请本地部署，本地部署成功后拼接url。

请求URL： http://{IP}:{PORT}/{DEPLOY_NAME}/ImageClassification IP：服务本地部署所在机器的ip地址 PORT：服务部署后获取的端口 DEPLOY_NAME：申请时填写的本地服务名称

Header如下：

参数	值
Content-Type	application/json

注意：如果出现336001的错误码很可能是因为请求方式错误，与其他图像识别服务不同的是定制化图像识别服务以json方式请求。

Body请求示例：

```
 {
   "image": "<base64数据>",
   "threshold": 5
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
image	是	string	-	图像数据，base64编码，要求base64编码后大小不超过4M，最短边至少15px，最长边最大4096px,支持jpg/png/bmp格式 **注意请去掉头部**
threshold	否	number	-	阈值，默认值为推荐阈值（0-1之间），具体值可在我的训练任务列表-模型效果查看

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
results	否	array(object)	分类结果数组
+name	否	string	分类名称
+score	否	number	置信度

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如缺少必要出入参时返回：

```
{
  "error_code": 336001,
  "error_msg": "Invalid Argument"
}
```

错误码	错误信息	描述
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（868826008）或工单联系技术支持团队
336001	Invalid Argument	入参格式有误，比如缺少必要参数、图片base64编码错误等等，可检查下图片编码、代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336004	输入文件大小不合法	图片超出大小限制，图片限4M以内，请根据接口文档检查入参格式，有疑问请通过QQ群（868826008）或工单联系技术支持团队
336005	图片解码失败	图片编码错误（非jpg,bmp,png等常见图片格式），请检查并修改图片格式
336006	缺失必要参数	image字段缺失（未上传图片）
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请通过QQ群（868826008）或工单联系技术支持团队
337000	Auth check failed	离线鉴权调用失败

物体检测私有API集成文档

本文档主要说明定制化模型本地部署后，如何使用本地API。如还未训练模型，请先前往BML进行训练。

如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动
- 加入BML官方QQ群（群号:868826008）联系群管

部署包使用说明

BML定制化物体检测模型的本地部署通过EasyPack实现，目前提供单机一键部署的方式。

在BML控制台申请、下载部署包后，可以参考EasyPack-单机一键部署将软件包部署在本地服务器上。部署成功后，启动服务，即可调用与在线API功能类似的接口。

授权说明

本地部署包根据服务器硬件（CPU单机或GPU单卡）进行授权，只能在申请时提交的硬件指纹所属的硬件上使用。

部署包测试期为1个月，如需购买永久授权，可提交工单咨询

API参考

请求说明

请求示例

HTTP 方法：`POST`

请求URL： 请首先在BML进行自定义模型训练，完成训练后申请本地部署，本地部署成功后拼接url。

请求URL： http://{IP}:{PORT}/{DEPLOY_NAME}/ObjectDetection IP：服务本地部署所在机器的ip地址 PORT：服务部署后获取的端口 DEPLOY_NAME：申请时填写的本地服务名称

Header如下：

参数	值
Content-Type	application/json

注意：如果出现336001的错误码很可能是因为请求方式错误，与其他图像识别服务不同的是定制化图像识别服务以json方式请求。

Body请求示例：

```
 {
   "image": "<base64数据>"
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
image	是	string	-	图像数据，base64编码，要求base64编码后大小不超过4M，最短边至少15px，最长边最大4096px,支持jpg/png/bmp格式 **注意请去掉头部**
threshold	否	number	-	默认值为推荐阈值，请在我的模型列表-模型效果查看推荐阈值

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
results	否	array(object)	识别结果数组
+name	否	string	分类名称
+score	否	number	置信度
+location	否		
++left	否	number	检测到的目标主体区域到图片左边界的距离
++top	否	number	检测到的目标主体区域到图片上边界的距离
++width	否	number	检测到的目标主体区域的宽度
++height	否	number	检测到的目标主体区域的高度

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如缺少必要出入参时返回：

```
 {
  "error_code": 336001,
  "error_msg": "Invalid Argument"
}
```

错误码	错误信息	描述
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（868826008）或工单联系技术支持团队
336001	Invalid Argument	入参格式有误，比如缺少必要参数、图片base64编码错误等等，可检查下图片编码、代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336004	输入文件大小不合法	图片超出大小限制，图片限4M以内，请根据接口文档检查入参格式，有疑问请通过QQ群（868826008）或工单联系技术支持团队
336005	图片解码失败	图片编码错误（非jpg,bmp,png等常见图片格式），请检查并修改图片格式
336006	缺失必要参数	image字段缺失（未上传图片）
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请通过QQ群（868826008）或工单联系技术支持团队
337000	Auth check failed	离线鉴权调用失败

如何发布私有API

训练完毕后，可以在左侧导航栏中找到【EasyEdge本地部署】，选择通用【服务器】，点击【发布新服务】，进入发布界面。



step1.部署方式选择服务器，集成方式选择私有API后，选择需要发布的模型及版本，上传指纹文件，勾选业务场景需要的操作系统与硬件芯片。设置完毕点击下一步



step2. 填写完基本信息之后，点击"提交"，即可申请将模型封装成可离线计算的部署包，申请通过后，即可下载部署包。



服务器端SDK

视觉任务服务器端SDK简介

本文档主要说明定制化模型发布后获得的服务器端SDK如何使用，如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动
- 加入BML官方QQ群（群号:813291890）联系群管

SDK说明

服务器端SDK支持Linux、Windows两种操作系统。以下为具体的系统、硬件环境支持：

操作系统	系统支持	硬件环境要求
Linux		CPU: x86_64 NVIDIA GPU: x86_64
Windows	64位 Windows7 及以上	NVIDIA GPU: x86_64 环境依赖： .NET Framework 4.5 Visual C++ Redistributable Packages for Visual Studio 2013 Visual C++ Redistributable Packages for Visual Studio 2015 GPU依赖： CUDA 9.x + cuDNN 7.x

单次预测耗时参考

> 根据具体设备、线程数不同，数据可能有波动，请以实测为准

芯片类型	实测硬件	图像分类模型	物体检测模型	备注
x86-64	Xeon E5-2650 v4	4-2725ms	118-3154ms	不同模型网络差别较大
Nvidia-GPU	Tesla P4	2-2413ms	35-87ms	不同模型网络差别较大

激活&使用步骤

服务器端SDK的激活与使用分以下三步：

① 在BML控制台下载SDK、获取序列号

② 本地运行SDK，并完成首次联网激活

> 通过左侧导航栏查看不同操作系统SDK的开发文档

③ 正式使用

SDK常见问题

> 通过左侧导航栏查看不同操作系统SDK的FAQ

以下是通用FAQ，如您的问题仍未解决，欢迎加入官方QQ群813291890咨询群管

1、激活失败怎么办？

①可能是当前序列号已被其他设备激活使用，请核实序列号后用未被激活的序列号重新激活

②序列号填写错误，请核实序列号后重新激活

③同一台设备绑定同一个序列号激活次数过多（超过50次），请更换序列号后重试

④首次激活需要联网，网络环境不佳或无网络环境，请检查网络环境后重试

⑤模型发布者和序列号所属账号非同一账号，如果存在这种异常建议更换账号获取有效序列号

⑥序列号已过有效期，请更换序列号后重试

视觉任务服务器端LinuxSDK集成文档-Python

简介

本文档介绍 EasyDL 的 Linux Python SDK 的使用方法，适用于 EasyDL 和 BML。

EasyDL 通用版：

- 网络类型支持：图像分类，物体检测，图像分割，声音分类，表格预测
- 硬件支持：
  - Linux x86_64 CPU (基础版，加速版)
  - Linux x86_64 Nvidia GPU (基础版，加速版)
- 语言支持：Python 3.5, 3.6, 3.7

BML：

- 网络类型支持：图像分类，物体检测，声音分类
- 硬件支持：
  - Linux x86_64 CPU (基础版)
  - Linux x86_64 Nvidia GPU (基础版)
- 语言支持：Python 3.5, 3.6, 3.7

Release Notes

时间	版本	说明
2022.09.15	1.3.3	EasyDL CPU普通版新增支持表格预测
2022.05.27	1.3.1	CPU、GPU普通版新增支持BML Cloud小目标检测模型
2021.12.22	1.2.7	声音分类模型升级
2021.10.20	1.2.6	CPU基础版、CPU加速版、GPU基础版推理引擎优化升级
2021.08.19	1.2.5	CPU基础版、CPU无损加速版、GPU基础版新增支持EasyDL小目标检测
2021.06.29	1.2.4	CPU、GPU新增EasyDL目标跟踪支持；新增http server服务启动demo
2021.03.09	1.2.2	EasyDL CPU加速版新增支持分类、高性能检测和均衡检测的量化压缩模型
2021.01.27	1.2.1	EasyDL经典版分类高性能模型升级；支持更多模型
2020.12.18	1.2.0	推理引擎升级；接口升级；性能优化
2020.09.17	1.1.19	支持更多模型
2020.08.11	1.1.18	性能优化
2020.06.23	1.1.17	支持更多EasyDL专业版模型
2020.04.16	1.1.15	技术优化；升级 OpenVINO 版本
2020.03.12	1.1.14	新增声音识别python sdk
2020.02.12	1.1.13	新增口罩模型支持
2020.01.16	1.1.12	预测函数默认使用推荐阈值
2019.12.26	1.1.11	EasyDL 专业版支持 SDK 加速版
2019.12.04	1.1.10	支持图像分割
2019.10.21	1.1.9	支持 EasyDL 专业版
2019.08.29	1.1.8	CPU 加速版支持
2019.07.19	1.1.7	提供模型更新工具
2019.05.16	1.1.3	NVIDIA GPU 支持
2019.03.15	1.1.0	架构与功能完善
2019.02.28	1.0.6	引擎功能完善
2019.02.13	1.0.5	paddlepaddle 支持
2018.11.30	1.0.0	第一版！

> 2020-12-18: 【接口升级】 序列号的配置接口从1.2.0版本开始已升级为新接口，以前的方式被置为deprecated，并将在未来的版本中移除。请尽快考虑升级为新的接口方式，具体使用方式可以参考下文介绍以及demo工程示例，谢谢。

快速开始

1. 安装依赖

- 根据引擎的不同，SDK 依赖了不同的底层引擎。根据所需自行安装。
- 使用声音分类SDK需要安装额外依赖 * pip 安装 `resampy pydub six librosa` 音频默认格式支持wav文件预测，如果需要预测mp3等其他音频格式的数据需要系统额外安装`ffmpeg`（windows系统的ffmpeg已基在sdk中无需额外安装，linux系统需要手动安装）
- 使用表格预测SDK需要安装额外依赖 *pip安装`brotlipy==0.7.0 certifi==2020.6.20 joblib==1.0.1 kaggle==1.5.12 Pillow py4j pycosat python-dateutil python-slugify ruamel_yaml text-unidecode threadpoolctl flask pandas==1.0.5 scikit-learn==0.23.2 lightgbm==2.2.3 catboost==0.24.1 xgboost==1.2.0 numpy==1.19.5 scipy==1.5.2 psutil==5.7.2 pypmml==0.9.7 torch==1.8.0 jieba==0.42.1 pyod==0.8.5 pyarrow==6.0.0 scikit-optimize==0.9.0 pyspark==3.3.0`* 另外ml算法安装（目前只支持python3.7） pip install `BaiduAI_TabularInfer-0.0.0-cp37-cp37m-linux_x86_64.whl` **安装 paddlepaddle**
- 使用`x86_64 CPU 基础版`预测时必须安装（目标跟踪除外）：

```
python -m pip install paddlepaddle==2.2.2 -i https://mirror.baidu.com/pypi/simple
```

> 若 CPU 为特殊型号，如赛扬处理器（一般用于深度定制的硬件中），请关注 CPU 是否支持 avx 指令集。如果不支持，请在paddle官网安装 noavx 版本

- 使用`NVIDIA GPU 基础版`预测时必须安装（目标跟踪除外）：

```
python -m pip install paddlepaddle-gpu==2.2.2.post101 -f https://www.paddlepaddle.org.cn/whl/linux/mkl/avx/stable.html #CUDA10.1的PaddlePaddle
python -m pip install paddlepaddle-gpu==2.2.2 -i https://mirror.baidu.com/pypi/simple #CUDA10.2的PaddlePaddle
python -m pip install paddlepaddle-gpu==2.2.2.post110 -f https://www.paddlepaddle.org.cn/whl/linux/mkl/avx/stable.html #CUDA11.0的PaddlePaddle
python -m pip install paddlepaddle-gpu==2.2.2.post111 -f https://www.paddlepaddle.org.cn/whl/linux/mkl/avx/stable.html #CUDA11.1的PaddlePaddle
python -m pip install paddlepaddle-gpu==2.2.2.post112 -f https://www.paddlepaddle.org.cn/whl/linux/mkl/avx/stable.html #CUDA11.2的PaddlePaddle
```

> 不同cuda版本的环境，请参考paddle文档安装合适的 paddle 版本。 不被 paddle 支持的 cuda 和 cudnn 版本，EasyEdge 暂不支持**安装 OpenVINO** 使用`x86_64 CPU 加速版` SDK 预测时必须安装。

1）请参考 OpenVINO toolkit 文档安装 2021.4版本, 安装时可忽略Configure the Model Optimizer及后续部分

2）运行之前，务必设置环境变量

```
source /opt/intel/openvino_2021/bin/setupvars.sh
```

安装 cuda、cudnn

- 使用`Nvidia GPU 加速版`预测时必须安装。 依赖的版本为 cuda9.0、cudnn7。版本号必须正确。

安装 pytorch（torch >= 1.7.0)

- 目标跟踪模型的预测必须安装pytorch版本1.7.0及以上（包含：`Nvidia GPU 基础版`、`x86_64 CPU 基础版`）。
- 目标跟踪模型`Nvidia GPU 基础版`还需安装依赖cuda、cudnn。

> 关于不同版本的pytorch和CUDA版本的对应关系：pytorch官网 目标跟踪模型还有一些列举在requirements.txt里的依赖（包括torch >= 1.7.0），均可使用pip下载安装。

```
pip3 install -r requirements.txt
```

2. 安装 easyedge python wheel 包

```
pip3 install -U BaiduAI_EasyEdge_SDK-{版本号}-cp36-cp36m-linux_x86_64.whl
```

具体名称以 SDK 包中的 whl 为准。

3. 使用序列号激活

获取序列号



修改`demo.py`填写序列号

```
pred = edge.Program()
pred.set_auth_license_key("这里填写序列号")
```

4. GPU 加速版 使用 GPU 加速版，在安装完 whl 之后，必须：

1. 从这里下载 TensorRT7.0.0.11 for cuda9.0，并把解压后的 lib 放到 C++ SDK 的 lib 目录或系统 lib 目录

2. 运行时，必须在系统库路径中包含 C++ SDK 下的`lib`目录。如设置`LD_LIBRARY_PATH`

```
cd ${SDK_ROOT}

**1. 安装 python wheel 包**
tar -xzvf python/*.tar.gz
pip install -U {对应 Python 版本的 wheel 包}

**2. 设置 LD_LIBRARY_PATH**
tar -xzvf cpp/*.tar.gz
export EDGE_ROOT=$(readlink -f $(ls -h | grep "baidu_easyedge_linux_cpp"))
export LD_LIBRARY_PATH=$EDGE_ROOT/lib

**3. 运行 demo**
python3 demo.py {RES文件夹路径} {测试图片路径}
```

如果是使用 C++ SDK 自带的编译安装的 OpenCV，LD_LIBRARY_PATH 还需要包括 C++ SDK的 build 目录下的`thirdparty/lib`目录

如果没有正确设置`LD_LIBRARY_PATH`，运行时可能报错：

```
ImportError: libeasyedge.so.0.4.3: cannot open shared object file: No such file or directory
ImportError: libopencv_core.so.3.4: cannot open shared object file: No such file or directory
```

5. 测试 Demo

5.1 图片预测

输入对应的模型文件夹（默认为`RES`）和测试图片路径，运行：

```
python3 demo.py {model_dir} {image_name.jpg}
```

测试效果：



5.2 视频预测（适用于目标跟踪）

输入对应的模型文件夹（默认为RES）和测试视频文件路径 / 摄像头id / 网络视频流地址，运行：

```
 **video_type: 输入源类型 type:int**
**1 本地视频文件**
**2 摄像头的index**
**3 网络视频流**
**video_src: 输入源地址，如视频文件路径、摄像头index、网络流地址 type: string**
python3 demo.py {model_dir} {video_type} {video_src}
```

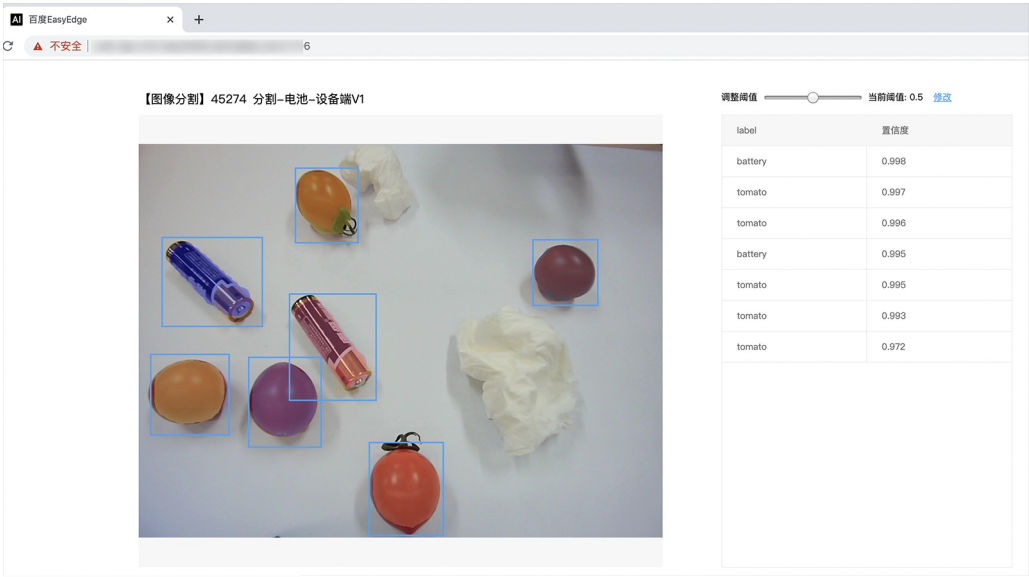
6. 测试Demo HTTP 服务 输入对应的模型文件夹（默认为RES）、序列号、设备ip和指定端口号，运行：

```
python3 demo_serving.py {model_dir} {serial_key} {host, default 0.0.0.0} {port, default 24401}
```

后，会显示：

```
Running on http://0.0.0.0:24401/
```

字样，此时，开发者可以打开浏览器，http://{设备ip}:24401，选择图片或者视频来进行测试。也可以参考`demo_serving.py`里http_client_test()函数请求http服务进行推理。





使用说明

使用流程 demo.py

```
import BaiduAI.EasyEdge as edge

pred = edge.Program()
pred.set_auth_license_key("这里填写序列号")
pred.init(model_dir={RES文件夹路径}, device=edge.Device.CPU, engine=edge.Engine.PADDLE_FLUID)
pred.infer_image({numpy.ndarray的图片})
pred.close()
```

demo_serving.py

```
 import BaiduAI.EasyEdge as edge
from BaiduAI.EasyEdge.serving import Serving

server = Serving(model_dir={RES文件夹路径}, license=serial_key)
**请参考同级目录下demo.py里:**
**pred.init(model_dir=xx, device=xx, engine=xx, device_id=xx)**
**对以下参数device\device_id和engine进行修改**
server.run(host=host, port=port, device=edge.Device.CPU, engine=edge.Engine.PADDLE_FLUID)
```

初始化

- 接口

```
    def init(self,
        model_dir,
        device=Device.CPU,
        engine=Engine.PADDLE_FLUID,
        config_file='conf.json',
        preprocess_file='preprocess_args.json',
        model_file='model',
        params_file='params',
        label_file='label_list.txt',
        infer_cfg_file='infer_cfg.json',
        device_id=0,
        thread_num=1
        ):
    """
    Args:
        model_dir: str
        device: BaiduAI.EasyEdge.Device，比如：Device.CPU
        engine: BaiduAI.EasyEdge.Engine， 比如： Engine.PADDLE_FLUID
        config_file: str
        preprocess_file: str
        model_file: str
        params_file: str
        label_file: str 标签文件
        infer_cfg_file: 包含预处理、后处理信息的文件
device_id: int 设备ID
        thread_num: int CPU的线程数

    Raises:
        RuntimeError, IOError
    Returns:
        bool: True if success
    """
```

使用 NVIDIA GPU 预测时，必须满足：

- 机器已安装 cuda, cudnn
- 已正确安装对应 cuda 版本的 paddle 版本
- 通过设置环境变量FLAGS_fraction_of_gpu_memory_to_use设置合理的初始内存使用比例

使用 CPU 预测时，可以通过在 init 中设置 thread_num 使用多线程预测。如：

```
pred.init(model_dir=_model_dir, device=edge.Device.CPU, engine=edge.Engine.PADDLE_FLUID, thread_num=1)
```

预测图像

- 接口

```
    def infer_image(self, img,
                    threshold=0.3,
                    channel_order='HWC',
                    color_format='BGR',
                    data_type='numpy'):
        """

        Args:
            img: np.ndarray or bytes
            threshold: float
                only return result with confidence larger than threshold
            channel_order: string
                channel order HWC or CHW
            color_format: string
                color format order RGB or BGR
            data_type: string
                image data type

        Returns:
            list

        """
```

- 返回格式: [dict1, dict2, ...]

字段	类型	取值	说明
confidence	float	0~1	分类或检测的置信度
label	string		分类或检测的类别
index	number		分类或检测的类别
x1, y1	float	0~1	物体检测，矩形的左上角坐标 (相对长宽的比例值)
x2, y2	float	0~1	物体检测，矩形的右下角坐标（相对长宽的比例值）
mask	string/numpy.ndarray	图像分割的mask	

关于矩形坐标

x1 * 图片宽度 = 检测框的左上角的横坐标

y1 * 图片高度 = 检测框的左上角的纵坐标

x2 * 图片宽度 = 检测框的右下角的横坐标

y2 * 图片高度 = 检测框的右下角的纵坐标

可以参考 demo 文件中使用 opencv 绘制矩形的逻辑。

结果示例

- i) 图像分类

```
{
  "index": 736,
  "label": "table",
  "confidence": 0.9
}
```

- ii) 物体检测

```
{
  "y2": 0.91211,
  "label": "cat",
  "confidence": 1.0,
  "x2": 0.91504,
  "index": 8,
  "y1": 0.12671,
  "x1": 0.21289
}
```

- iii) 图像分割

```
{
"name": "cat",
  "score": 1.0,
  "location": {
   "left": ...,
   "top": ...,
   "width": ...,
   "height": ...,
  },
  "mask": ...
}
```

mask字段中，data_type为`numpy`时，返回图像掩码的二维数组

```
{
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
{0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
{0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
{0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
}
其中1代表为目标区域，0代表非目标区域
```

data_type为`string`时，mask的游程编码，解析方式可参考 demo

预测视频（目前仅限目标跟踪模型调用）

- 接口

```
def infer_frame(self, frame, threshold=None):
 """
 视频推理(抽帧之后)
 :param frame:
 :param threshold:
 :return:
 """
```

- 返回格式`dict`

字段	类型	说明
pos	dict1	当前帧每一个类别的追踪目标的像素坐标(tlwh)
id	dict2	当前帧每一个类别的追踪目标的id
score	dict3	当前帧每一个类别的追踪目标的识别置信度
label	dict4	class_idx(int)与label(string)的对应关系
class_num	int	追踪类别数

预测声音

- 使用声音分类SDK需要安装额外依赖 *pip 安装* `resampy pydub` 音频默认格式支持wav文件预测，如果需要预测mp3等其他音频格式的数据需要系统额外安装`ffmpeg`（windows系统的ffmpeg已集成在sdk中无需额外安装，linux系统需要手动安装）

- 接口

```
    def infer_sound(self, sound_binary,
                threshold=0.3):
        """

        Args:
            sound_binary: sound_binary
            threshold: confidence

        Returns:
            list

        """
```

- 返回格式: [dict1, dict2, ...]

字段	类型	取值	说明
confidence	float	0~1	分类的置信度
label	string		分类的类别
index	number		分类的类别

升级模型 适用于经典版升级模型，执行bash update_model.sh，根据提示，输入模型路径、激活码、模型ID、模型版本，等待模型更新完毕即可。

FAQ

Q: EasyDL 离线 SDK 与云服务效果不一致，如何处理？ A: 后续我们会消除这部分差异，如果开发者发现差异较大，可联系我们协助处理。

Q: 运行时报错 "非法指令" 或 " illegal instruction" A: 可能是 CPU 缺少 avx 指令集支持，请在paddle官网 下载 noavx 版本覆盖安装

Q: NVIDIA GPU预测时，报错显存不足： A: 如以下错误字样：

```
 paddle.fluid.core.EnforceNotMet: Enforce failed. Expected allocating <= available, but received allocating:20998686233 >
available:19587333888.
Insufficient GPU memory to allocation. at [/paddle/paddle/fluid/platform/gpu_info.cc:170]
```

请在运行 Python 前设置环境变量，通过export FLAGS_fraction_of_gpu_memory_to_use=0.3来限制SDK初始使用的显存量，0.3表示初始使用30%的显存。如果设置的初始显存较小，SDK 会自动尝试 allocate 更多的显存。

Q: 我想使用多线程预测，怎么做？ 如果需要多线程预测，可以每个线程启动一个Progam实例，进行预测。 demo.py文件中有相关示例代码。

> 注意： 对于CPU预测，SDK内部是可以使用多线程，最大化硬件利用率。参考init的thread_num参数。

Q: 运行SDK报错 Authorization failed

情况一：日志显示 Http perform failed: null respond 在新的硬件上首次运行，必须联网激活。

SDK 能够接受HTTP_PROXY 的环境变量通过代理处理自己的网络请求。如

```
 export HTTP_PROXY="http://192.168.1.100:8888"
./easyedge_demo ...
```

情况二：日志显示failed to get/check device id(xxx)或者Device fingerprint mismatch(xxx) 此类情况一般是设备指纹发生了变更，包括（但不局限于）以下可能的情况：

- MAC地址变化
- 磁盘变更
- BIOS重刷

以及系统相关信息。

遇到这类情况，请确保硬件无变更，如果想更换序列号，请先删除 `~/.baidu/easyedge` 目录，再重新激活。

视觉任务服务器端LinuxSDK集成文档-C++

简介

本文档介绍EasyEdge/EasyDL的Linux CPP SDK的使用方法。

- 网络类型支持：图像分类，物体检测，图像分割，目标追踪
- 硬件支持：
  - CPU 基础版: - intel x86_64 * - AMD x86_64 - 龙芯 loongarch64 - 飞腾 aarch64
  - CPU 加速版 - Intel Xeon with Intel®AVX2 and AVX512 - Intel Core Processors with AVX2 - Intel Atom Processors with SSE - AMD Core Processors with AVX2
  - NVIDIA GPU: x86_64 PC
  - 寒武纪 Cambricon MLU270
  - 比特大陆计算卡SC5+
  - 百度昆仑XPU K200
    - x86_64 - 飞腾 aarch64 - 百度昆仑XPU R200
    - x86_64 - 飞腾 aarch64
  - 华为Atlas 300
  - 海光DCU: x86_64 PC
  - 寒武纪 MLU370 on x86_64
- 操作系统支持：Linux

根据开发者的选择，实际下载的版本可能是以下版本之一：

- EasyDL图像
  - x86 CPU 基础版
  - x86 CPU 加速版
  - Nvidia GPU 基础版
  - Nvidia GPU 加速版
  - x86 mlu270基础版
  - x86 SC5+基础版
  - Phytium MLU270基础版
  - Phytium XPU基础版
  - Phytium Atlas300I基础版
  - Hygon DCU基础版

性能数据参考算法性能及适配硬件

> *intel 官方合作，拥有更好的适配与性能表现。

Release Notes

时间	版本	说明
2022.10.27	1.7.1	新增语义分割模型http请求示例；升级海光DCU SDK，需配套rocm4.3版本使用；Linux GPU基础版下线适用于CUDA10.0及以下版本的SDK；Linux GPU加速版升级推理引擎版本

2022.09.15	1.7.0	Linux GPU加速版升级预测引擎；Linux GPU加速版适用于CUDA9.0、CUDA10.0的SDK为deprecated，未来移除；新增实例分割高性能模型离线部署；性能优化
2022.07.28	1.6.0	Linux CPU普通版、Linux GPU普通/加速版、Jetson新增目标追踪模型接入实时流的demo
2022.05.27	1.5.1	CPU、GPU普通版新增支持BML Cloud小目标检测模型
2022.05.18	1.5.0	GPU加速版max_batch_size参数含义变更；修复GPU加速版并发预测时部分图片结果预测错误及耗时增加问题；CPU普通版预测引擎升级；新增版本号头文件；新增飞腾Atlas300I支持，并且在EasdDL新增多种加速版本；示例代码移除frame_buffer，新增更安全高效的safe_queue; 新增Tensor In/Out接口和Demo
2022.04.25	1.4.1	EasyDL, BML升级支持paddle2模型
2022.03.25	1.4.0	新增支持海光服务器搭配海光DCU加速卡;
2021.12.22	1.3.5	GPU加速版支持自定义模型文件缓存路径；新增支持飞腾MLU270服务器、飞腾XPU服务器
2021.10.20	1.3.4	CPU加速版推理引擎优化升级，新增支持飞腾CPU、龙芯CPU服务器、比特大陆计算卡SC5+ BM1684、寒武纪MLU270；大幅提升EasyDL GPU加速版有损压缩加速模型的推理速度
2021.08.19	1.3.2	CPU、GPU普通版及无损加速版新增支持EasyDL小目标检测，CPU普通版、GPU普通版支持检测模型的batch预测
2021.06.29	1.3.1	CPU普通版、GPU普通版支持分类模型的batch预测，CPU加速版支持分类、检测模型的batch预测；GPU加速版支持CUDA11.1；视频流解析支持调整分辨率；预测引擎升级
2021.05.13	1.3.0	新增视频流接入支持；模型发布新增多种加速方案选择；目标追踪支持x86平台的CPU、GPU加速版；展示已发布模型性能评估报告
2021.03.09	1.2.1	GPU新增目标追踪支持, http server服务支持图片通过base64格式调用，EasyDL高性能检测模型和均衡检测模型CPU加速版新增量化压缩模型
2021.01.27	1.1.0	EasyDL经典版分类高性能模型升级；部分SDK不再需要单独安装OpenCV
2020.12.18	1.0.0	1.0版本发布！安全加固升级、性能优化、引擎升级、接口优化等多项更新
2020.11.26	0.5.8	EasyDL经典版分类模型CPU加速版里新增量化压缩模型
2020.10.29	0.5.7	新增CPU加速版支持：EasyDL经典版高精度、超高精度物体检测模型和EasyDL经典版图像分割模型
2020.09.17	0.5.6	性能优化，支持更多模型
2020.08.11	0.5.5	提升预测速度；支持百度昆仑芯片
2020.05.15	0.5.3	优化性能，支持专业版更多模型
2020.04.16	0.5.2	支持CPU加速版；CPU基础版引擎升级；GPU加速版支持多卡多线程
2020.03.12	0.5.0	x86引擎升级；更新本地http服务接口；GPU加速版提速，支持批量图片推理
2020.01.16	0.4.7	ARM引擎升级；增加推荐阈值支持
2019.12.26	0.4.6	支持海思NNIE
2019.11.02	0.4.5	移除curl依赖；支持自动编译OpenCV；支持EasyDL 专业版 Yolov3; 支持EasyDL经典版高精度物体检测模型升级
2019.1	0.4.4	ARM引擎升级，性能提升30%； 支持EasyDL专业版模型

0.25		
2019.09.23	0.4.3	增加海思NNIE加速芯片支持
2019.08.30	0.4.2	ARM引擎升级；支持分类高性能与高精度模型
2019.07.25	0.4.1	引擎升级，性能提升
2019.07.25	0.4.0	支持Xeye, 细节完善
2019.06.11	0.3.3	paddle引擎升级；性能提升
2019.05.16	0.3.2	新增NVIDIA GPU支持；新增armv7l支持
2019.04.25	0.3.1	优化硬件支持
2019.03.29	0.3.0	ARM64 支持；效果提升
2019.02.20	0.2.1	paddle引擎支持；效果提升
2018.11.30	0.1.0	第一版！

> 2022-5-18: 【接口变更】 `PREDICTOR_KEY_GTURBO_MAX_BATCH_SIZE`含义变更。 变更前：预测输入图片数**不大于**该值均可。 变更后：预测输入图片数需**等于**该值。SDK内部对该接口变更做了兼容处理，在输入图片数小于该值时依然可正常运行，但预测性能会和等于该值时一致。推荐根据实际输入图片数量需求修改该值，尽可能保持最小。

> 2020-12-18: 【接口升级】 参数配置接口从1.0.0版本开始已升级为新接口，以前的方式被置为deprecated，并将在未来的版本中移除。请尽快考虑升级为新的接口方式，具体使用方式可以参考下文介绍以及demo工程示例，谢谢。 【关于SDK包与RES模型文件夹配套使用的说明】 我们强烈建议用户使用部署tar包中配套的SDK和RES。 更新模型时，如果SDK版本号有更新，请务必同时更新SDK，旧版本的SDK可能无法正确适配新发布出来部署包中的RES模型。

快速开始

SDK在以下环境中测试通过

- x86_64, Ubuntu 16.04, gcc 5.4
- x86_64, Ubuntu 18.04, gcc 7.4
- Tesla P4, Ubuntu 16.04, cuda 9.0, cudnn 7.5
- x86_64, Ubuntu 16.04, gcc 5.4, XTCL r1.0
- aarch64, Kylin V10, gcc 7.3
- loongarch64, Kylin V10, gcc 8.3
- Bitmain SC5+ BM1684, Ubuntu 18.04, gcc 5.4
- x86_64 MLU270，Ubuntu 18.04, gcc 7.5
- phytium MLU270，Kylin V10，gcc 7.3.0
- phytium XPU，Kylin V10，gcc 7.3.0
- hygon DCU, CentOS 7.8 gcc 7.3.0
- XPU K200, x86_64, Ubuntu 18.04

- XPU K200 aarch64, Ubuntu 18.04
- XPU R200, x86_64, Ubuntu 18.04
- XPU R200 aarch64, Ubuntu 18.04
- MLU370, x86_64, Centos7.6.1810

依赖包括

- cmake 3+
- gcc 5.4 (需包含 GLIBCXX_3.4.22) ，gcc / glibc版本请以实际SDK ldd的结果为准
- opencv3.4.11 (可选)
- cuda9.0_cudnn7 (使用NVIDIA-GPU时必须)
- XTCL 1.0.0.187 (使用昆仑服务器时必须)
- Rocm4.0, Miopen 2.11(使用海光DCU服务器时必须)

1. 安装依赖

以下步骤均可选，请开发者根据实际运行环境选择安装。

(可选) 安装cuda&cudnn

在NVIDIA GPU上运行必须(包括GPU基础版，GPU加速版)

对于GPU基础版，若开发者需求不同的依赖版本，请在PaddlePaddle官网 下载对应版本的`libpaddle_fluid.so`或参考其文档进行编译，覆盖lib文件夹下的相关库文件。

(可选) 安装TensorRT

在NVIDIA GPU上运行GPU加速版必须

下载包中提供了对应 cuda10.0、cuda10.2、cuda11.0+四个版本的 SDK，cuda9.0 和 cuda10.0 的 SDK 默认依赖的 TensorRT 版本为 TensorRT7.0.0.11，cuda10.2 及以上的 SDK 默认依赖的 TensorRT 版本为 TensorRT8.4，请在这里下载对应 cuda 版本的 TensorRT，并把其中的lib文件拷贝到系统lib目录，或其他目录并设置环境变量。

(可选) 安装XTCL ***使用昆仑服务器及对应SDK时必须*** 请安装与1.0.0.187版本兼容的XTCL。必要时，请将运行库路径添加到环境变量。

(可选) 安装Rocm、Miopen ***使用海光DCU服务器对应SDK时必须*** 海光DCU SDK依赖Rocm 4.3和Miopen 2.14版本，推荐使用以下easyedge镜像，SDK镜像内运行，镜像拉取方式(`docker pull registry.baidubce.com/easyedge/hygon_dcu_infer:1.0.2.rocm4.3`)，关于海光DCU使用更多细节可参考paddle文档

2. 使用序列号激活 在控制台获取的序列号请通过参数配置结构体`EdgePredictorConfig`的成员函数`set_config(easyedge::params::PREDICTOR_KEY_SERIAL_NUM, "this-is-serial-num")`设置。

具体请参考SDK自带的`Demo.cpp`文件的使用方法。

3. 测试Demo

模型资源文件默认已经打包在开发者下载的SDK包中。Demo工程直接编译即可运行。

> 请先将tar包整体拷贝到具体运行的设备中，再解压缩编译； 在Intel CPU上运行CPU加速版，如果thirdparty里包含openvino文件夹的，必须在编译或运行demo程序前执行以下命令: `source ${cpp_kit位置路径}/thirdparty/openvino/bin/setupvars.sh` 或者执行 `source ${cpp_kit位置路径}/thirdparty/openvino/setupvars.sh`(openvino-2022.1+)

请在官网获取序列号，填写在`demo.cpp`中



> 部分SDK中已经包含预先编译的二进制，如 bin/easyedge_demo, bin/easyedge_serving，配置LD_LIBRARY_PATH后，可直接运行:
> LD_LIBRARY_PATH=../lib ./bin/easyedge_serving

编译运行：

```
 cd src
mkdir build && cd build
cmake .. && make
./easyedge_image_inference {模型RES文件夹} {测试图片路径}
**如果是NNIE引擎，使用sudo运行**
sudo ./easyedge_image_inference {模型RES文件夹} {测试图片路径}
```

如果希望SDK自动编译安装所需要的OpenCV库，修改cmake的option`EDGE_BUILD_OPENCV`为`ON`即可。 SDK会自动从网络下载opencv源码，并编译需要的module、链接。注意，此功能必须需联网。

```
cmake -DEDGE_BUILD_OPENCV=ON
```

若需自定义library search path或者gcc路径，修改CMakeList.txt即可。

demo运行效果：




```
 > ./easyedge_image_inference ../../../../RES 2.jpeg
2019-02-13 16:46:12,659 INFO [EasyEdge] [easyedge.cpp:34] 140606189016192 Baidu EasyEdge Linux Development Kit
0.2.1(20190213)
2019-02-13 16:46:14,083 INFO [EasyEdge] [paddlev2_edge_predictor.cpp:60] 140606189016192 Allocate graph success.
2019-02-13 16:46:14,326 DEBUG [EasyEdge] [paddlev2_edge_predictor.cpp:143] 140606189016192 Inference costs 168 ms
1, 1:txt_frame, p:0.994905 loc: 0.168161, 0.153654, 0.920856, 0.779621
Done
```

> 对于支持批量预测的模型和SDK，可在使用前修改demo_image_inference或demo_batch_inference里的batch_size再编译、执行。

> 详情请参考下方使用说明-其他配置部分

4. 测试Demo HTTP 服务

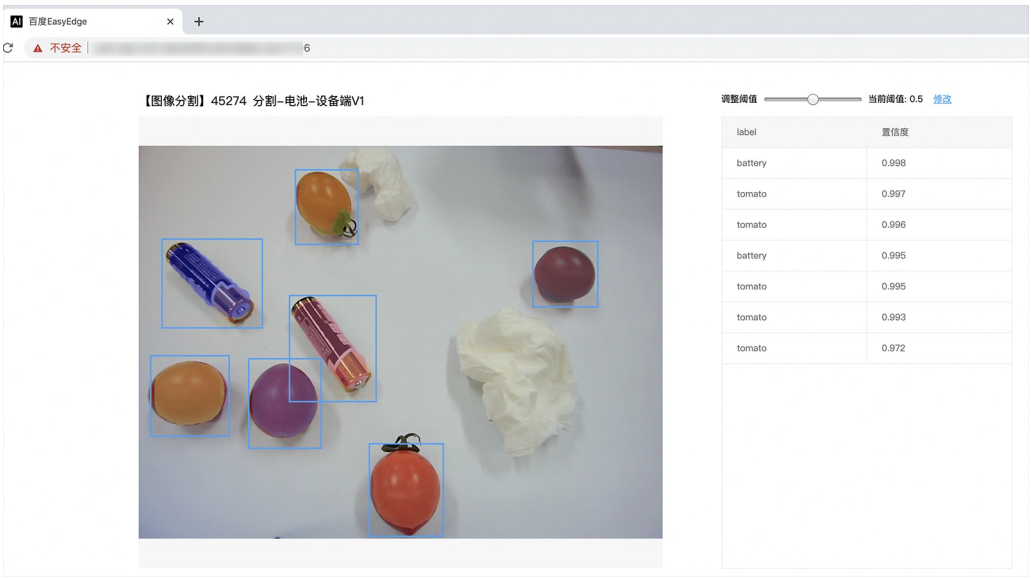
编译demo完成之后，会同时生成一个http服务 运行

```
 **./easyedge_serving {res_dir} {serial_key} {host, default 0.0.0.0} {port, default 24401}**
./easyedge_serving ../../../RES "1111-1111-1111-1111" 0.0.0.0  24401
```

后，日志中会显示

```
HTTP is now serving at 0.0.0.0:24401
```

字样，此时，开发者可以打开浏览器，`http://{设备ip}:24401`，选择图片来进行测试。



> 对于目标追踪的模型，请选择一段视频，并耐心等待结果

图片加载失败

同时，可以调用HTTP接口来访问服务，具体参考下文接口说明。

使用说明

使用该方式，将运行库嵌入到开发者的程序当中。

使用流程

> 请优先参考Demo的使用流程。遇到错误，请优先参考文件中的注释解释，以及日志说明。

```
    // step 1: 配置模型资源目录
    EdgePredictorConfig config;
    config.model_dir = {模型文件目录};

    // step 2: 创建并初始化Predictor；在这里选择合适的引擎
    auto predictor = global_controller()->CreateEdgePredictor(config);


    // step 3-1: 预测图像
    auto img = cv::imread({图片路径});
    std::vector<EdgeResultData> results;
    predictor->infer(img, results);

  // step 3-2: 预测视频
  std::vector<EdgeResultData> results;
  FrameTensor frame_tensor;
    VideoConfig video_config;
    video_config.source_type = static_cast<SourceType>(video_type);  // source_type 定义参考头文件 easyedge_video.h
    video_config.source_value = video_src;
    /*
    ... more video_configs, 根据需要配置video_config的各选项
    */
    auto video_decoding = CreateVideoDecoding(video_config);
    while (video_decoding->next(frame_tensor) == EDGE_OK) {
        results.clear();
        if (frame_tensor.is_needed) {
            predictor->infer(frame_tensor.frame, results);
            render(frame_tensor.frame, results, predictor->model_info().kind);
        }
        //video_decoding->display(frame_tensor); // 显示当前frame，需在video_config中开启配置
        //video_decoding->save(frame_tensor); // 存储当前frame到视频，需在video_config中开启配置
    }
```

> 输入图片不限制大小

SDK参数配置 SDK的参数通过`EdgePredictorConfig::set_config`和`global_controller()->set_config`配置。`set_config`的所有key在`easyedge_xxxx_config.h`中。其中

- `PREDICTOR`前缀的key是不同模型相关的配置，通过`EdgePredictorConfig::set_config`设置
- `CONTROLLER`前缀的key是整个SDK的全局配置，通过`global_controller()->set_config`设置

以序列号为例，KEY的说明如下：

```
  /**
  * @brief 序列号设置；序列号不设置留空时，SDK将会自动尝试使用本地已经激活成功的有效期内的序列号
  * 值类型：string
  * 默认值：空
  */
static constexpr auto PREDICTOR_KEY_SERIAL_NUM = "PREDICTOR_KEY_SERIAL_NUM";
```

使用方法如下：

```
  EdgePredictorConfig config;
config.model_dir = ...;
config.set_config(params::PREDICTOR_KEY_SERIAL_NUM, "1DB7-1111-1111-D27D");
```

具体支持的运行参数配置列表可以参考开发工具包中的头文件的详细说明。

> 相关配置均可以通过环境变量的方法来设置，对应的key名称加上前缀`EDGE_`即为环境变量的key。如序列号配置的环境变量key为`EDGE_PREDICTOR_KEY_SERIAL_NUM`，如指定CPU线程数的环境变量key为`EDGE_PREDICTOR_KEY_CPU_THREADS_NUM`。 注意：通过代码设置的配置会覆盖通过环境变量设置的值。

初始化

- 接口

```
auto predictor = global_controller()->CreateEdgePredictor(config);
predictor->init();
```

若返回非0，请查看输出日志排查错误原因。

预测图像

- 接口

```
/**
 * @brief
 * 通用接口
 * @param image: must be BGR , HWC format (opencv default)
 * @param result
 * @return
 */
virtual int infer(
        cv::Mat& image, std::vector<EdgeResultData>& result
) = 0;

/**
 * @brief
 * 批量图片推理接口
 * @param image: must be BGR , HWC format (opencv default)
 * @param result
 * @return
 */
virtual int infer(
        std::vector<cv::Mat>& image, std::vector<std::vector<EdgeResultData>>& result
) = 0;
```

图片的格式务必为opencv默认的BGR, HWC格式。

- 返回格式

`EdgeResultData`中可以获取对应的分类信息、位置信息。

```
struct EdgeResultData {
  int index;  // 分类结果的index
  std::string label;  // 分类结果的label
  float prob;  // 置信度

  // 物体检测、图像分割时才有意义
  float x1, y1, x2, y2;  // (x1, y1): 左上角，  (x2, y2): 右下角； 均为0~1的长宽比例值。

  // 图像分割的模型，该字段才有意义
  // 请注意：图像分割时，以下两个字段会比较大，使用完成之后请及时释放EdgeResultData
  cv::Mat mask;  // 0, 1 的mask
  std::string mask_rle;  // Run Length Encoding，游程编码的mask

  // 目标追踪模型，该字段才有意义
  int trackid;  // 轨迹id
  int frame;  // 处于视频中的第几帧
  EdgeTrackStat track_stat;  // 跟踪状态
};
```

关于矩形坐标

x1 * 图片宽度 = 检测框的左上角的横坐标

y1 * 图片高度 = 检测框的左上角的纵坐标

x2 * 图片宽度 = 检测框的右下角的横坐标

y2 * 图片高度 = 检测框的右下角的纵坐标

关于图像分割mask

```
 cv::Mat mask为图像掩码的二维数组
{
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
 {0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
 {0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
 {0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
}
其中1代表为目标区域，0代表非目标区域
```

关于图像分割mask_rle

该字段返回了mask的游程编码，解析方式可参考 http demo

以上字段可以参考demo文件中使用opencv绘制的逻辑进行解析

预测视频

SDK 提供了支持摄像头读取、视频文件和网络视频流的解析工具类`VideoDecoding`，此类提供了获取视频帧数据的便利函数。通过`VideoConfig`结构体可以控制视频/摄像头的解析策略、抽帧策略、分辨率调整、结果视频存储等功能。对于抽取到的视频帧可以直接作为SDK infer 接口的参数进行预测。

- 接口

class`VideoDecoding`：

```
 /**
 * @brief 获取输入源的下一帧
 * @param frame_tensor
 * @return
 */
virtual int next(FrameTensor &frame_tensor) = 0;

/**
 * @brief 显示当前frame_tensor中的视频帧
 * @param frame_tensor
 * @return
 */
virtual int display(const FrameTensor &frame_tensor) = 0;

/**
 * @brief 将当前frame_tensor中的视频帧写为本地视频文件
 * @param frame_tensor
 * @return
 */
virtual int save(FrameTensor &frame_tensor) = 0;

/**
 * @brief 获取视频的fps属性
 * @return
 */
virtual int get_fps() = 0;
 /**
  * @brief 获取视频的width属性
  * @return
  */
virtual int get_width() = 0;

/**
 * @brief 获取视频的height属性
 * @return
 */
virtual int get_height() = 0;
```

struct VideoConfig

```
 /**
 * @brief 视频源、抽帧策略、存储策略的设置选项
 */
struct VideoConfig {
    SourceType source_type;          // 输入源类型
    std::string source_value;        // 输入源地址，如视频文件路径、摄像头index、网络流地址
    int skip_frames{0};              // 设置跳帧，每隔skip_frames帧抽取一帧，并把该抽取帧的is_needed置为true
    int retrieve_all{false};         // 是否抽取所有frame以便于作为显示和存储，对于不满足skip_frames策略的frame，把所抽取帧的
is_needed置为false
    int input_fps{0};                // 在采取抽帧之前设置视频的fps
    Resolution resolution{Resolution::kAuto}; // 采样分辨率，只对camera有效

    bool enable_display{false};
    std::string window_name{"EasyEdge"};
    bool display_all{false};         // 是否显示所有frame，若为false，仅显示根据skip_frames抽取的frame

    bool enable_save{false};
    std::string save_path;           // frame存储为视频文件的路径
    bool save_all{false};            // 是否存储所有frame，若为false，仅存储根据skip_frames抽取的frame

    std::map::string, std::string> conf;
};
```

source_type：输入源类型，支持视频文件、摄像头、网络视频流三种，值分别为1、2、3。 source_value： 若source_type为视频文件，该值为指向视频文件的完整路径；若source_type为摄像头，该值为摄像头的index，如对于/dev/video0的摄像头，则index为0；若source_type为网络视频流，则为该视频流的完整地址。 skip_frames：设置跳帧，每隔skip_frames帧抽取一帧，并把该抽取帧的is_needed置为true，标记为is_needed的帧是用来做预测的帧。反之，直接跳过该帧，不经过预测。 retrieve_all：若置该项为true，则无论是否设置跳帧，所有的帧都会被抽取返回，以作为显示或存储用。 input_fps：用于抽帧前设置fps。 resolution：设置摄像头采样的分辨率，其值请参考easyedge_video.h中的定义，注意该分辨率调整仅对输入源为摄像头时有效。 conf：高级选项。部分配置会通过该map来设置。

注意：

1. 如果使用VideoConfig的display功能，需要自行编译带有GTK选项的opencv，默认打包的opencv不包含此项。

2. 使用摄像头抽帧时，如果通过resolution设置了分辨率调整，但是不起作用，请添加如下选项：

```
video_config.conf["backend"] = "2";
```

3.部分设备上的CSI摄像头尚未兼容，如遇到问题，可以通过工单、QQ交流群或微信交流群反馈。

具体接口调用流程，可以参考SDK中的demo_video_inference。

设置序列号

请在网页控制台中申请序列号，并在init初始化前设置。 LinuxSDK 首次使用需联网授权。

```
EdgePredictorConfig config;
config.set_config(easyedge::params::PREDICTOR_KEY_SERIAL_NUM, "this-is-serial-num");
```

日志配置

设置 EdgeLogConfig 的相关参数。具体含义参考文件中的注释说明。

```
EdgeLogConfig log_config;
log_config.enable_debug = true;
global_controller()->set_log_config(log_config);
```

http服务

1. 开启http服务 http服务的启动可以参考demo_serving.cpp文件。

```
/**
  * @brief 开启一个简单的demo http服务。
  * 该方法会block直到收到sigint/sigterm。
  * http服务里，图片的解码运行在cpu之上，可能会降低推理速度。
  * @tparam ConfigT
  * @param config
  * @param host
  * @param port
  * @param service_id service_id  user parameter, uri '/get/service_id' will respond this value with 'text/plain'
  * @param instance_num 实例数量，根据内存/显存/时延要求调整
  * @return
  */
template<typename ConfigT>
int start_http_server(
        const ConfigT &config,
        const std::string &host,
        int port,
        const std::string &service_id,
        int instance_num = 1);
```

2. 请求http服务

> 开发者可以打开浏览器，`http://{设备ip}:24401`，选择图片或视频来进行测试。

http 请求方式一：无额外编码

- 图片测试：不使用图片base64格式

URL中的get参数：

参数	说明	默认值
threshold	阈值过滤， 0~1	如不提供，则会使用模型的推荐阈值

HTTP POST Body即为图片的二进制内容(无需base64, 无需json)

Python请求示例 (针对非语义分割模型)

```
import requests

with open('./1.jpg', 'rb') as f:
    img_data = f.read()
    result = requests.post(
     'http://127.0.0.1:24401/',
     params={'threshold': 0.1},
     data=img_data).json()
```

Python请求示例 (针对语义分割模型)

```
import requests

with open('./1.jpg', 'rb') as f:
 img_data = f.read()
 res = requests.post('http://127.0.0.1:24401/',
  data=img_data)
 with open("gray_result.png", "wb") as fb:
  fb.write(res.content) # 语义分割模型是像素点级别输出，可将api返回结果保存为灰度图，每个像素值代表该像素分类结果
```

Java请求示例

- 视频测试

Python请求示例 (注意：区别于图片预测，需指定`Content-Type`；否则会调用图片推理接口)

```
import requests

with open('./1.mp4', 'rb') as f:
    video_data = f.read()
    result = requests.post(
     'http://127.0.0.1:24401/',
     params={'threshold': 0.1},
     headers={'Content-Type': 'video'},
     data=video_data).json()
```

http 请求方法二：图片使用base64格式 HTTP方法：POST Header如下：

参数	值
Content-Type	application/json

Body请求填写：

- 分类网络： body 中请求示例

```
{
 "image": "<base64数据>"
 "top_num": 5
}
```

body中参数详情

参数	是否必选	类型	可选值范围	说明
image	是	string	-	图像数据，base64编码，要求base64图片编码后大小不超过4M,最短边至少15px，最长边最大4096px，支持jpg/png/bmp格式 **注意去掉头部**
top_num	否	number	-	返回分类数量，不填该参数，则默认返回全部分类结果

- 检测和分割网络： Body请求示例：

```
{
 "image": "<base64数据>"
}
```

body中参数详情：

参数	是否必选	类型	可选值范围	说明
image	是	string	-	图像数据，base64编码，要求base64图片编码后大小不超过4M,最短边至少15px，最长边最大4096px，支持jpg/png/bmp格式 **注意去掉头部**
threshold	否	number	-	默认为推荐阈值，也可自行根据需要进行设置

http 返回数据

字段	类型说明	其他
error_code	Number	0为成功,非0参考message获得具体错误信息
results	Array	内容为具体的识别结果。其中字段的具体含义请参考预测图像-返回格式一节
cost_ms	Number	预测耗时ms，不含网络交互时间

请求示例 (针对非语义分割模型)

```
 import base64
import requests

def main():
    with open("1.jpg 【图片路径】", 'rb') as f:
        result = requests.post("http://{服务ip地址}:24401/", json={
            "image": base64.b64encode(f.read()).decode("utf8")
        })
        # print(result.request.body)
        # print(result.request.headers)
        print(result.content)

if __name__ == '__main__':
    main()
```

请求示例 (针对语义分割模型)

```
 import base64
import requests

def main():
 with open("1.jpg 【图片路径】", 'rb') as f:
  res = requests.post("http://{服务ip地址}:24401/", json={"image": base64.b64encode(f.read()).decode("utf8")})
 with open("gray_result.png", "wb") as fb:
  fb.write(res.content) # 语义分割模型是像素点级别输出，可将api返回结果保存为灰度图，每个像素值代表该像素分类结果

if __name__ == '__main__':
    main()
```

返回示例

```
{
  "cost_ms": 52,
  "error_code": 0,
  "results": [
    {
      "confidence": 0.94482421875,
      "index": 1,
      "label": "IronMan",
      "x1": 0.059185408055782318,
      "x2": 0.18795496225357056,
      "y1": 0.14762254059314728,
      "y2": 0.52510076761245728
    },
    {
      "confidence": 0.94091796875,
      "index": 1,
      "label": "IronMan",
      "x1": 0.79151463508605957,
      "x2": 0.92310667037963867,
      "y1": 0.045728668570518494,
      "y2": 0.42920106649398804
    }
  ]
}
```

其他配置

1. 日志名称、HTTP 网页标题设置

通过global_controller的`set_config`方法设置：

```
global_controller()->set_config(easyedge::params::CONTROLLER_KEY_LOG_BRAND, "MY_BRAND");
```

效果如下：



2. CPU线程数设置

CPU线程数可通过 EdgePredictorConfig::set_config配置

```
 EdgePredictorConfig config;
config.set_config(easyedge::params::PREDICTOR_KEY_CPU_THREADS_NUM, 4);
```

3. 批量预测设置

```
 int batch_size = 2; // 使用前修改batch_size再编译、执行
while (get_next_batch(imgs, img_files, batch_size, start_index)) {
    ...
}
```

GPU 加速版 预测接口 GPU 加速版 SDK 除了支持上面介绍的通用接口外，还支持图片的批量预测，预测接口如下：

```
 /**
 * @brief
 * GPU加速版批量图片推理接口
 * @param image: must be BGR，HWC format (opencv default)
 * @param result
 * @return
 */
virtual int infer(
        std::vector<cv::Mat>& image,
        std::vector<std::vector<EdgeResultData>>& result
) = 0;

/**
 * @brief
 * GPU加速版批量图片推理接口，带阈值
 * @related infer(cv::Mat & image, EdgeColorFormat origin_color_format, std::vector<EdgeResultData> &result, float threshold)
 */
virtual int infer(
            std::vector<cv::Mat> &images,
            EdgeColorFormat origin_color_format,
            std::vector<std::vector<EdgeResultData>> &results,
            float threshold
) = 0;
```

批量图片的预测接口的使用要求在调用 init 接口的时候设置一个有效的 PREDICTOR_KEY_GTURBO_MAX_BATCH_SIZE，其含义见下方参数配置接口的介绍。

运行参数选项 在上面的内容中我们介绍了如何使用EdgePredictorConfig进行运行参数的配置。针对GPU加速版开发工具包，目前EdgePredictorConfig的运行参数所支持的Key包括如下项：

```
 /**
 * @brief 当有同类型的多个设备的时候，使用哪一个设备，如：
 * GPU: 使用哪张GPU卡
 * EdgeBoard(VMX)，Movidius NCS ：使用哪一张加速卡
 * 值类型：int
 * 默认值：0
 */
static constexpr auto PREDICTOR_KEY_DEVICE_ID = "PREDICTOR_KEY_DEVICE_ID";

/**
 * @brief 生成最大 batch_size 为 max_batch_size 的优化模型，单次预测图片数量可以小于或等于此值（推荐等于此值，见release notes）
 * 值类型: int
 * 默认值：4
 */
static constexpr auto PREDICTOR_KEY_GTURBO_MAX_BATCH_SIZE = "PREDICTOR_KEY_GTURBO_MAX_BATCH_SIZE";
```

```
/**
 * @brief 设置device对应的GPU卡可以支持的最大并发量
 * 实际预测的时候对应GPU卡的最大并发量不超过这里设置的范围，否则预测请求会排队等待预测执行
 * 值类型: int
 * 默认值：1
 */
static constexpr auto PREDICTOR_KEY_GTURBO_MAX_CONCURRENCY = "PREDICTOR_KEY_GTURBO_MAX_CONCURRENCY";

/**
 * @brief 是否开启fp16模式预测，开启后预测速度会更快，但精度会略有降低。并且需要硬件支持fp16
 * 值类型: bool
 * 默认值：false
 */
static constexpr auto PREDICTOR_KEY_GTURBO_FP16 = "PREDICTOR_KEY_GTURBO_FP16";

/**
 * @brief 模型编译等级
 * 1：如果当前max_batch_size与历史编译产出的max_batch_size不相等时，则重新编译模型（推荐）
 * 2：无论历史编译产出的max_batch_size为多少，均根据当前max_batch_size重新编译模型
 * 值类型: int
 * 默认值：1
 */
static constexpr auto PREDICTOR_KEY_GTURBO_COMPILE_LEVEL = "PREDICTOR_KEY_GTURBO_COMPILE_LEVEL";

/**
 * @brief GPU工作空间大小设置
 * workspace_size = workspace_prefix * (1 << workspace_offset)
 * workspace_offset: 10 = KB, 20 = MB, 30 = GB
 * 值类型: int
 * 默认值：WORKSPACE_PREFIX: 100, WORKSPACE_OFFSET: 20，即100MB
 */
static constexpr auto PREDICTOR_KEY_GTURBO_WORKSPACE_PREFIX = "PREDICTOR_KEY_GTURBO_WORKSPACE_PREFIX";
static constexpr auto PREDICTOR_KEY_GTURBO_WORKSPACE_OFFSET = "PREDICTOR_KEY_GTURBO_WORKSPACE_OFFSET";

/**
 * @brief 需要使用的dla core
 * 值类型: int
 * 默认值：-1(不使用)
 */
static constexpr auto PREDICTOR_KEY_GTURBO_DLA_CORE = "PREDICTOR_KEY_GTURBO_DLA_CORE";

/**
 * @brief 自定义缓存文件存储路径
 * 值类型: string
 * 默认值: ~/.baidu/easyedge/mcache/{model_id * 1000000 + release_id}
 */
static constexpr auto PREDICTOR_KEY_GTURBO_CACHE_DIR = "PREDICTOR_KEY_GTURBO_CACHE_DIR";

/**
 * @brief 自定义缓存文件命名，默认即可
 * 值类型: string
 * 默认值: 根据配置自动生成
 */
static constexpr auto PREDICTOR_KEY_GTURBO_CACHE_NAME = "PREDICTOR_KEY_GTURBO_CACHE_NAME";

/**
 * @brief 序列号设置；序列号不设置留空时，SDK将会自动尝试使用本地已经激活成功的有效期内的序列号
 * 值类型：string
 * 默认值：空
 */
static constexpr auto PREDICTOR_KEY_SERIAL_NUM = "PREDICTOR_KEY_SERIAL_NUM";
```

`PREDICTOR_KEY_GTURBO_CACHE_NAME`：首次加载模型会先对模型进行编译优化，通过此值可以设置优化后的产出文件名，这在多进程加载同一个模型的时候是有用的。

`PREDICTOR_KEY_GTURBO_CACHE_DIR`：首次加载模型经过编译优化后，产出的优化文件会存储在这个位置，可以按需修改。

`PREDICTOR_KEY_GTURBO_WORKSPACE_PREFIX`、`PREDICTOR_KEY_GTURBO_WORKSPACE_OFFSET`：设置运行时可以被用来使用的最大临时显

存。

PREDICTOR_KEY_GTURBO_MAX_BATCH_SIZE：此值用来控制批量图片预测可以支持的最大图片数，实际预测的时候单次预测图片数需等于此值。

PREDICTOR_KEY_DEVICE_ID：设置需要使用的 GPU 卡号。

PREDICTOR_KEY_GTURBO_COMPILE_LEVEL：模型编译等级。通常模型的编译会比较慢，但编译产出是可以复用的。可以在第一次加载模型的时候设置合理的 max_batch_size 并在之后加载模型的时候直接使用历史编译产出。是否使用历史编译产出可以通过此值 compile_level 来控制。当此值为 0 时，表示忽略当前设置的 max_batch_size 而仅使用历史产出（无历史产出时则编译模型）；当此值为 1 时，会比较历史产出和当前设置的 max_batch_size 是否相等，如不等，则重新编译；当此值为 2 时，无论如何都会重新编译模型。

PREDICTOR_KEY_GTURBO_MAX_CONCURRENCY：通过此值设置单张 GPU 卡上可以支持的最大 infer 并发量，其上限取决于硬件限制。init 接口会根据此值预分配 GPU 资源，建议结合实际使用控制此值，使用多少则设置多少。注意：此值的增加会降低单次 infer 的速度，建议优先考虑 batch inference 和 multi predictor。

PREDICTOR_KEY_GTURBO_FP16：默认是 fp32 模式，置 true 可以开启 fp16 模式预测，预测速度会有所提升，但精度也会略微下降，权衡使用。注意：不是所有模型都支持 fp16 模式。目前已知不支持fp16的模型包括：图像分类高精度模型。

多线程预测 GPU 加速版 SDK 的多线程分为单卡多线程和多卡多线程两种。 单卡多线程：创建一个 predictor，并通过 PREDICTOR_KEY_GTURBO_MAX_CONCURRENCY 控制单卡所支持的最大并发量，只需要 init 一次，多线程调用 infer 接口。 多卡多线程：多卡的支持是通过创建多个 predictor，每个 predictor 对应一张 GPU 卡，predictor 的创建和 init 的调用放在主线程，通过多线程的方式调用 infer 接口。

已知问题 1. 多线程时图片按线程分配不均 或 不同batch size的图片交叉调用infer接口时，部分结果错误 A：EasyDL图像分类高精度模型在有些显卡上可能存在此问题，可以考虑填充假图片数据到图片比较少的线程或batch以使得infer间的图片绝对平均。

2. 显存持续增长或遇到 terminate called after throwing an instance of 'std::runtime_error' what(): Failed to create object A：部分显卡存在此问题，如果遇到此问题，请确认没有频繁调用 init 接口，通常调用 infer 接口即可满足需求。

3. 开启 fp16 后，预测结果错误 A：不是所有模型都支持 fp16 模式。目前已知的不支持fp16的模型包括：图像分类高精度模型。目前不支持的将会在后面的版本陆续支持。

昆仑服务器 昆仑服务器SDK支持将EasyDL的模型部署到昆仑服务器上。SDK提供的接口风格一致，简单易用，轻松实现快速部署。Demo的测试可参考上文中的测试Demo部分。

参数配置接口 在上面的内容中我们介绍了如何使用EdgePredictorConfig进行运行参数的配置。针对昆仑服务器开发工具包，目前EdgePredictorConfig的运行参数所支持的Key包括如下项：

```
 /**
 * @brief 序列号设置；序列号不设置留空时，SDK将会自动尝试使用本地已经激活成功的有效期内的序列号
 * 值类型：string
 * 默认值：空
 */
static constexpr auto PREDICTOR_KEY_SERIAL_NUM = "PREDICTOR_KEY_SERIAL_NUM";

/**
 * @brief 当有同类型的多个设备的时候，使用哪一个设备，如：
 * 使用哪张加速卡
 * 值类型：int
 * 默认值：0
 */
static constexpr auto PREDICTOR_KEY_DEVICE_ID = "PREDICTOR_KEY_DEVICE_ID";

/**
 * @brief 设置需要同时预测的图片数量
 * 值类型: int
 * 默认值：1
 */
static constexpr auto PREDICTOR_KEY_KUNLUN_BATCH_SIZE = "PREDICTOR_KEY_KUNLUN_BATCH_SIZE";
```

PREDICTOR_KEY_DEVICE_ID：设置需要使用的加速卡的卡号。

PREDICTOR_KEY_KUNLUN_BATCH_SIZE：设置单次预测可以支持的图片数量。

使用方法：

```
int batch_size = 1;
config.set_config(easyedge::params::PREDICTOR_KEY_KUNLUN_BATCH_SIZE, batch_size);
```

模型调优 通过设置如下环境变量，可以在初始化阶段对模型调优，从而让预测的速度更快。

```
export XPU_CONV_AUTOTUNE=5
```

FAQ

1. 如何处理一些 undefined reference?

> 如：undefined reference to `curl_easy_setopt@CURL_OPENSSL_3'

方案1：通过安装libcurl3 libcurl-openssl1.0-dev来解决。 方案2：如果开发者想不想使用低版本的openssl（如Ubuntu 18.04），可以link静态库easyedge_static.a，自己指定需要的Library的版本：

示例：修改CMakeList.txt

```
find_package(CURL REQUIRED)
target_link_libraries(easyedge_demo ${OpenCV_LIBS} easyedge_static pthread ${CURL_LIBRARIES} verify_static ${其他需要的库})
```

其中，其他需要的库视具体sdk中包含的库而定。

2. EasyDL SDK与云服务效果不一致，如何处理？

后续我们会消除这部分差异，如果开发者发现差异较大，可联系我们协助处理。

3. NVIDIA GPU预测时，报错显存不足 如以下错误字样：

```
paddle.fluid.core.EnforceNotMet: Enforce failed. Expected allocating <= available, but received allocating:20998686233 > available:19587333888.
Insufficient GPU memory to allocation. at [/paddle/paddle/fluid/platform/gpu_info.cc:170]
```

请根据显存大小和模型配置。调整合适的初始 fraction_of_gpu_memory。 参数的含义参考这里。

4. 如何将我的模型运行为一个http服务？ ~~目前cpp sdk暂未集成http运行方式；~~ 0.4.7版本之后，可以通过start_http_server方法开启http服务。

5. 运行NNIE引擎报permission denied 日志显示：

```
open sys: Permission denied
open err
: Permission denied
open err
: Permission denied
```

请使用sudo在root下运行。

6. 运行SDK报错 Authorization failed

情况一：日志显示 Http perform failed: null respond 在新的硬件上首次运行，必须联网激活。

SDK 能够接受HTTP_PROXY 的环境变量通过代理处理自己的网络请求。如

```
export HTTP_PROXY="http://192.168.1.100:8888"
./easyedge_demo ...
```

情况二：日志显示failed to get/check device id(xxx)或者Device fingerprint mismatch(xxx) 此类情况一般是设备指纹发生了变更，包括（但不局限于）以下可能的情况：

- mac 地址变化
- 磁盘变更

- bios重刷

以及系统相关信息。

遇到这类情况，请确保硬件无变更，如果想更换序列号，请先删除 ~/.baidu/easyedge 目录，再重新激活。

7. 使用libcurl请求http服务时，速度明显变慢

这是因为libcurl请求continue导致server等待数据的问题，添加空的header即可

```
headers = curl_slist_append(headers, "Expect:");
```

8. 运行二进制时，提示 libverify.so cannot open shared object file

可能cmake没有正确设置rpath, 可以设置LD_LIBRARY_PATH为sdk的lib文件夹后，再运行：

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:../lib ./easyedge_demo
```

9. 运行二进制时提示 libopencv_videoio.so.4.5: cannot open shared object file: No such file or directory 同上面8的问题类似，没有正确设置动态库的查找路径，可通过设置LD_LIBRARY_PATH为sdk的thirdparty/opencv/lib文件夹解决

```
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:../thirdparty/opencv/lib
(tips: 上面冒号后面接的thirdparty/opencv/lib路径以实际项目中路径为准，比如也可能是../../thirdparty/opencv/lib)
```

10. 编译时报错：file format not recognized 可能是因为在复制SDK时文件信息丢失。请将整个压缩包复制到目标设备中，再解压缩、编译

11. 进行视频解码时，报错符号未找到、格式不支持、解析出的图片为空、无法设置抽帧 请确保安装OpenCV时，添加了-DWITH_FFMPEG=ON选项（或者GStream选项），并且检查OpenCV的安装日志中，关于Video I/O段落的说明是否为YES。

```
--   Video I/O:
--     DC1394:                      YES (ver 2.2.4)
--     FFMPEG:                      YES
--       avcodec:                   YES (ver 56.60.100)
--       avformat:                  YES (ver 56.40.101)
--       avutil:                    YES (ver 54.31.100)
--       swscale:                   YES (ver 3.1.101)
--       avresample:                NO
--     libv4l/libv4l2:              NO
--     v4l/v4l2:                    linux/videodev2.h
```

如果为NO，请搜索相关解决方案，一般为依赖没有安装，以apt为例：

```
 apt-get install yasm libjpeg-dev libjasper-dev libavcodec-dev libavformat-dev libswscale-dev libdc1394-22-dev libgstreamer0.10-dev
libgstreamer-plugins-base0.10-dev libv4l-dev python-dev python-numpy libtbb-dev libqt4-dev libgtk2.0-dev libfaac-dev libmp3lame-dev
libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev libxvidcore-dev x264 v4l-utils ffmpeg
```

12. GPU加速版运行有损压缩加速的模型，运算精度较标准模型偏低 首先请保证数据集不会太小。其次可以通过将模型目录RES中的calibrationtable移除，并通过将PREDICTOR_KEY_GTURBO_FP16设置为true，使用FP16的运算精度重新评估模型效果。若依然不理想，可将calibrationtable移除并将PREDICTOR_KEY_GTURBO_FP16设置为false,从而使用更高精度的FP32的运算精度。

视觉任务服务器端WindowsSDK集成文档

简介

本文档介绍Windows GPU SDK的使用方法。

- 网络类型支持：图像分类，物体检测，图像分割、声音分类
- 硬件支持：
  - NVIDIA GPU （普通版，加速版）
- 操作系统支持
  - 64位 Windows 7 及以上

    - 64位Windows Server 2012及以上
- 环境依赖（必须安装以下版本）
    - .NET Framework 4.5
    - Visual C++ Redistributable Packages for Visual Studio 2013
    - Visual C++ Redistributable Packages for Visual Studio 2015
- GPU非目标跟踪基础版（EasyEdge-win-x86-nvidia-gpu）依赖（必须安装以下版本）
    - CUDA 9.0.x + cuDNN 7.6.x 或者 CUDA 10.0.x + cuDNN 7.6.x 或者 CUDA 11.0.x + cuDNN 8.0.x
- GPU目标跟踪基础版（EasyEdge-win-x86-nvidia-gpu-torch）依赖（必须安装以下版本）
    - CUDA 11.0.x + cuDNN 8.0.x
- GPU加速版（EasyEdge-win-x86-nvidia-gpu-tensorrt）依赖（必须安装以下版本）
    - CUDA 9.0.x + cuDNN 7.6.x 或者 CUDA 10.0.x + cuDNN 7.6.x
    - TensorRT 7.x 必须和CUDA版本对应
- GPU加速版（EasyEdge-win-x86-nvidia-gpu-paddletrt）依赖（必须安装以下版本）
    - CUDA 11.0.x + cuDNN 8.0.x
    - TensorRT 7.1.3.4 必须和CUDA版本对应
- 协议
    - HTTP
- 更详细的环境说明可参考SDK内的README.md

Release Notes | 时间 | 版本 | 说明 | | ---------- | ------ | ---------- | | 2022-10-27 | 1.7.1 | GPU底层引擎升级，下线基础版CUDA10.0及以下版本支持 | | 2022-09-15 | 1.7.0 | 优化模型算法；GPU CUDA9.0 CUDA10.0 标记为待废弃状态 | | 2022-07-28 | 1.6.0 | 优化模型算法 | | 2022-05-27 | 1.5.1 | 新增支持BML Cloud小目标检测模型 | | 2022-05-18 | 1.5.0 | 修复各别机器下程序崩溃的问题 | | 2022-04-25 | 1.4.1 | EasyDL, BML升级支持paddle2模型 | | 2022-03-25 | 1.4.0 | 优化模型算法 | | 2021-12-22 | 1.3.5 | GPU基础版推理引擎优化升级；GPU加速版支持自定义模型文件缓存路径；demo程序优化环境依赖检测 | | 2021-10-20 | 1.3.4 | 修复已知问题| | 2021-08-19 | 1.3.2 | 新增支持EasyDL小目标检测，新增DEMO二进制文件 | | 2021-06-29 | 1.3.1 | 预测引擎升级 | | 2021-05-13 | 1.3.0 | 模型发布新增多种加速方案选择；目标追踪支持x86平台的GPU及加速版；展示已发布模型性能评估报告 | | 2021-04-08 | 1.2.3 | 支持BML平台模型仓库本地上传模型 | | 2021-03-09 | 1.2.2| 修复已知问题 | | 2021-01-27 | 1.2.1| 新增模型支持；性能优化；问题修复 | | 2020-12-18 | 1.2.0 | 推理引擎升级 | | 2020-11-26 | 1.1.20 | 新增一些模型的加速版支持 | | 2020-10-29 | 1.1.19| 修复已知问题 | | 2020-09-17 | 1.1.18| 支持更多模型 | | 2020.08.11 | 1.1.17 | 支持专业版更多模型 | | 2020.06.23 | 1.1.16 | 支持专业版更多模型 | | 2020.05.15 | 1.1.15| 更新加速版tensorrt版本，支持高精度检测 | | 2020.03.13 | 1.1.14| 支持声音分类 | | 2020.02.23 | 1.1.13| 支持多阶段模型 | | 2020.01.16 | 1.1.12 | 预测默认使用推荐阈值| | 2019.12.26 | 1.1.11 | 支持物体检测高精度算法的CPU加速版，EasyDL 专业版支持 SDK 加速版 | | 2019.12.04 | 1.1.10 | 支持图像分割 | | 2019.10.21 | 1.1.9 | 支持 EasyDL 专业版 | | 2019.08.29 | 1.1.8 | CPU 加速版支持 | | 2019.07.19 | 1.1.7 | 提供模型更新工具 | | 2019.05.16 | 1.1.3 | NVIDIA GPU 支持 | | 2019.03.15 | 1.1.0 | 架构与功能完善 | | 2019.02.28 | 1.0.6 | 引擎功能完善 | | 2019.02.13 | 1.0.5 | paddlepaddle 支持 | | 2018.11.30 | 1.0.0 | 第一版！ |

快速开始

1. 安装依赖

安装.NET Framework4.5

```
https://www.microsoft.com/zh-CN/download/details.aspx?id=42642
```

Visual C++ Redistributable Packages for Visual Studio 2013

```
https://www.microsoft.com/zh-cn/download/details.aspx?id=40784
```

Visual C++ Redistributable Packages for Visual Studio 2015

```
https://www.microsoft.com/zh-cn/download/details.aspx?id=48145
```

如果使用GPU版SDK，请安装CUDA + cuDNN

```
 https://developer.nvidia.com/cuda
https://developer.nvidia.com/cudnn
```

如果使用GPU版加速版SDK（EasyEdge-win-x86-nvidia-gpu-tensorrt），请安装TensorRT

```
 https://developer.nvidia.com/tensorrt

根据cuda版本下载：
cuda9.0: TensorRT-7.0.0.11.Windows10.x86_64.cuda-9.0.cudnn7.6
cuda10.0: TensorRT-7.0.0.11.Windows10.x86_64.cuda-10.0.cudnn7.6
cuda10.2: 不支持，请降级为cuda10.0

下载后把lib目录下的所有dll，拷贝到SDK的dll目录下
```

如果使用GPU版加速版SDK（EasyEdge-win-x86-nvidia-gpu-paddletrt），请安装TensorRT

```
 https://developer.nvidia.com/tensorrt

根据cuda版本下载：
cuda11.0: TensorRT-7.1.3.4.Windows10.x86_64.cuda-11.0.cudnn8.0

下载后把lib目录下的所有dll，拷贝到SDK的dll目录下
```

如果使用声音分类，请安装以下依赖

```
 · 安装six
打开cmd，进入sdk包所在目录。执行EasyEdge-win-mXXXX-x86-nvidia-gpu\python37\python.exe -m pip install -U  six -i
http://mirrors.aliyun.com/pypi/simple/ --trusted-host mirrors.aliyun.com

· 安装librosa
打开cmd，进入sdk包所在目录。执行EasyEdge-win-mXXXX-x86-nvidia-gpu\python37\python.exe -m pip install -U  librosa -i
http://mirrors.aliyun.com/pypi/simple/ --trusted-host mirrors.aliyun.com
```

注意事项

1. 安装目录不能包含中文
2. Windows Server 请自行开启，选择“我的电脑”——“属性”——“管理”——”添加角色和功能“——勾选”桌面体验“，点击安装，安装之后重启即可。

2. 运行离线SDK

解压下载好的SDK，SDK默认使用cuda9版本，如果需要cuda10请运行EasyEdge CUDA10.0.bat切换到cuda10版本，之后打开EasyEdge.exe，输入Serial Num 

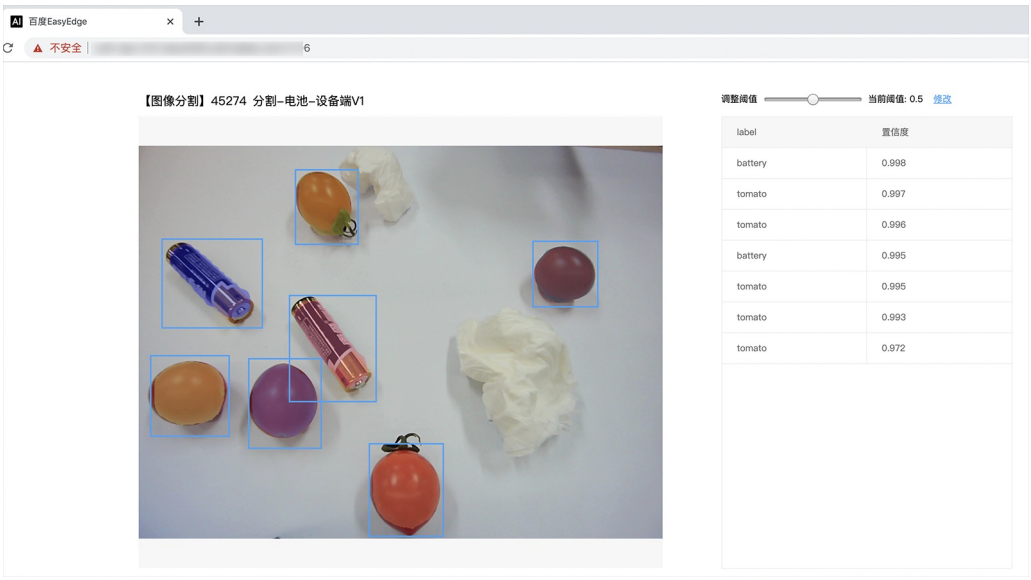
点击"启动服务"，等待数秒即可启动成功，本地服务默认运行在

```
 http://127.0.0.1:24401/
```

其他任何语言只需通过HTTP调用即可。

Demo示例(以图像服务为例)

服务运行成功，此时可直接在浏览器中输入`http://127.0.0.1:24401`，在h5中测试模型效果。



使用说明

图像服务调用说明

Python 使用示例代码如下

```
 import requests

with open('./1.jpg', 'rb') as f:
    img = f.read()

**params 为GET参数 data 为POST Body**
result = requests.post('http://127.0.0.1:24401/', params={'threshold': 0.1},
                                        data=img).json()
```

C# 使用示例代码如下

```
 FileStream fs = new FileStream("./img.jpg", FileMode.Open);
BinaryReader br = new BinaryReader(fs);
byte[] img = br.ReadBytes((int)fs.Length);
br.Close();
fs.Close();
string url = "http://127.0.0.1:8402?threshold=0.1";
HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(url);
request.Method = "POST";
Stream stream = request.GetRequestStream();
stream.Write(img, 0, img.Length);
stream.Close();

WebResponse response = request.GetResponse();
StreamReader sr = new StreamReader(response.GetResponseStream());
Console.WriteLine(sr.ReadToEnd());
sr.Close();
response.Close();
```

C++ 使用示例代码如下，需要安装curl

```
  **include <sys/stat.h>**
**include <curl/curl.h>**

**define S_ISREG(m) (((m) & 0170000) == (0100000))**
**define S_ISDIR(m) (((m) & 0170000) == (0040000))**

int main(int argc, char *argv[]) {
    const char *post_data_filename = "./img.jpg";

    FILE *fp = NULL;
    struct stat stbuf = { 0, };

    fp = fopen(post_data_filename, "rb");

    if (!fp) {
        fprintf(stderr, "Error: failed to open file \"%s\"\n", post_data_filename);
        return -1;
    }

    if (fstat(fileno(fp), &stbuf) || !S_ISREG(stbuf.st_mode)) {
        fprintf(stderr, "Error: unknown file size \"%s\"\n", post_data_filename);
        return -1;
    }

    CURL *curl;
    CURLcode res;

    curl_global_init(CURL_GLOBAL_ALL);

    curl = curl_easy_init();
    if (curl != NULL) {
        curl_easy_setopt(curl, CURLOPT_URL, "http://127.0.0.1:24401?threshold=0.1");
        curl_easy_setopt(curl, CURLOPT_POST, 1L);
        curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE_LARGE,(curl_off_t)stbuf.st_size);
        curl_easy_setopt(curl, CURLOPT_READDATA, (void *)fp);

        res = curl_easy_perform(curl);
        if (res != CURLE_OK) {
            fprintf(stderr, "curl_easy_perform() failed: %s\n", curl_easy_strerror(res));
        }
        curl_easy_cleanup(curl);
    }
    curl_global_cleanup();
    fclose(fp);

    return 0;
}
```

请求参数

字段	类型	取值	说明
threshold	float	0 ~ 1	置信度阈值

HTTP POST Body直接发送图片二进制。

返回参数 | 字段 | 类型 | 取值 | 说明 | | --- | --- | --- | --- | | confidence | float | 0~1 | 分类或检测的置信度 | | label | string | | 分类或检测的类别 | | index | number | | 分类或检测的类别 | | x1, y1 | float | 0~1 | 物体检测，矩形的左上角坐标 (相对长宽的比例值) | | x2, y2 | float | 0~1 | 物体检测，矩形的右下角坐标（相对长宽的比例值） |

关于矩形坐标

x1 * 图片宽度 = 检测框的左上角的横坐标

y1 * 图片高度 = 检测框的左上角的纵坐标

x2 * 图片宽度 = 检测框的右下角的横坐标

y2 * 图片高度 = 检测框的右下角的纵坐标

图像分割 返回结果格式参考API调用文档 代码参考 https://github.com/Baidu-AIP/EasyDL-Segmentation-Demo

声音服务调用说明

Python 使用示例代码如下

```
 import requests

with open('./1.mp3', 'rb') as f:
    audio = f.read()

**params 为GET参数 data 为POST Body**
result = requests.post('http://127.0.0.1:24401/', params={'threshold': 0.1},
                                        data=audio).json()
```

C# 使用示例代码如下

```
  FileStream fs = new FileStream("./audio.mp3", FileMode.Open);
BinaryReader br = new BinaryReader(fs);
byte[] audio = br.ReadBytes((int)fs.Length);
br.Close();
fs.Close();
string url = "http://127.0.0.1:8402?threshold=0.1";
HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(url);
request.Method = "POST";
Stream stream = request.GetRequestStream();
stream.Write(audio, 0, audio.Length);
stream.Close();

WebResponse response = request.GetResponse();
StreamReader sr = new StreamReader(response.GetResponseStream());
Console.WriteLine(sr.ReadToEnd());
sr.Close();
response.Close();
```

C++ 使用示例代码如下，需要安装curl

```
  **include <sys/stat.h>**
**include <curl/curl.h>**

int main(int argc, char *argv[]) {
    const char *post_data_filename = "./audio.mp3";

    FILE *fp = NULL;
    struct stat stbuf = { 0, };

    fp = fopen(post_data_filename, "rb");

    if (!fp) {
        fprintf(stderr, "Error: failed to open file \"%s\"\n", post_data_filename);
        return -1;
    }

    if (fstat(fileno(fp), &stbuf) || !S_ISREG(stbuf.st_mode)) {
        fprintf(stderr, "Error: unknown file size \"%s\"\n", post_data_filename);
        return -1;
    }

    CURL *curl;
    CURLcode res;

    curl_global_init(CURL_GLOBAL_ALL);

    curl = curl_easy_init();
    if (curl != NULL) {
        curl_easy_setopt(curl, CURLOPT_URL, "http://127.0.0.1:24401?threshold=0.1");
        curl_easy_setopt(curl, CURLOPT_POST, 1L);
        curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE_LARGE,(curl_off_t)stbuf.st_size);
        curl_easy_setopt(curl, CURLOPT_READDATA, (void *)fp);

        res = curl_easy_perform(curl);
        if (res != CURLE_OK) {
            fprintf(stderr, "curl_easy_perform() failed: %s\n", curl_easy_strerror(res));
        }
        curl_easy_cleanup(curl);
    }
    curl_global_cleanup();
    fclose(fp);

    return 0;
}
```

请求参数

字段	类型	取值	说明
threshold	float	0 ~ 1	置信度阈值

HTTP POST Body直接发送声音二进制。

返回参数 | 字段 | 类型 | 取值 | 说明 | | --- | --- | --- | --- | | confidence | float | 0~1 | 分类或检测的置信度 | | label | string | | 分类或检测的类别 | | index | number | | 分类或检测的类别 |

集成指南

基于HTTP集成

通过EasyEdge.exe启动服务后，参照上面的调用说明，通过HTTP请求集成到自己的服务中

基于c++ dll集成

集成前提

解压开的SDK包中包含src、lib、dll、include四个目录才支持基于c++ dll集成

集成方法

参考src目录中的CMakeLists.txt进行集成

基于c# dll集成

集成前提

解压开的SDK包中包含src\demo_serving_csharp、dll两个目录才支持基于c# dll集成

集成方法

参考src\demo_serving_csharp目录中的CMakeLists.txt进行集成

FAQ

1. 服务启动失败，怎么处理？

请确保相关依赖都安装正确，版本必须如下： *.NET Framework 4.5* Visual C++ Redistributable Packages for Visual Studio 2013 * Visual C++ Redistributable Packages for Visual Studio 2015

GPU依赖，版本必须如下： * CUDA 9.0.x + cuDNN 7.6.x 或者 CUDA 10.0.x + cuDNN 7.6.x

GPU加速版（EasyEdge-win-x86-nvidia-gpu-tensorrt）依赖，版本必须如下： *CUDA 9.0.x + cuDNN 7.6.x 或者 CUDA 10.0.x + cuDNN 7.6.x* TensorRT 7.x 必须和CUDA版本对应

GPU加速版（EasyEdge-win-x86-nvidia-gpu-paddletrt）依赖，版本必须如下： *CUDA 11.0.x + cuDNN 8.0.x* TensorRT 7.1.3.4 必须和CUDA版本对应

2. 服务调用时返回为空，怎么处理？ 调用输入的图片必须是RGB格式，请确认是否有alpha通道。

3. 多个模型怎么同时使用？ SDK设置运行不同的端口，点击运行即可。

4. JAVA、C#等其他语言怎么调用SDK?

参考 https://ai.baidu.com/forum/topic/show/943765

5. 启动失败，缺失DLL? 打开EasyEdge.log，查看日志错误，根据提示处理 缺失DLL，请使用 https://www.dependencywalker.com/ 查看相应模块依赖DLL缺失哪些，请自行下载安装

6. 启动失败，报错NotDecrypted? Windows下使用，当前用户名不能为中文，否则无法正确加载模型。

7. 启动失败，报错 SerialNum无效

日志显示`failed to get/check device id(xxx)`或者`Device fingerprint mismatch(xxx)` 此类情况一般是设备指纹发生了变更，包括（但不局限于）以下可能的情况：

- mac 地址变化
- 磁盘变更
- bios重刷

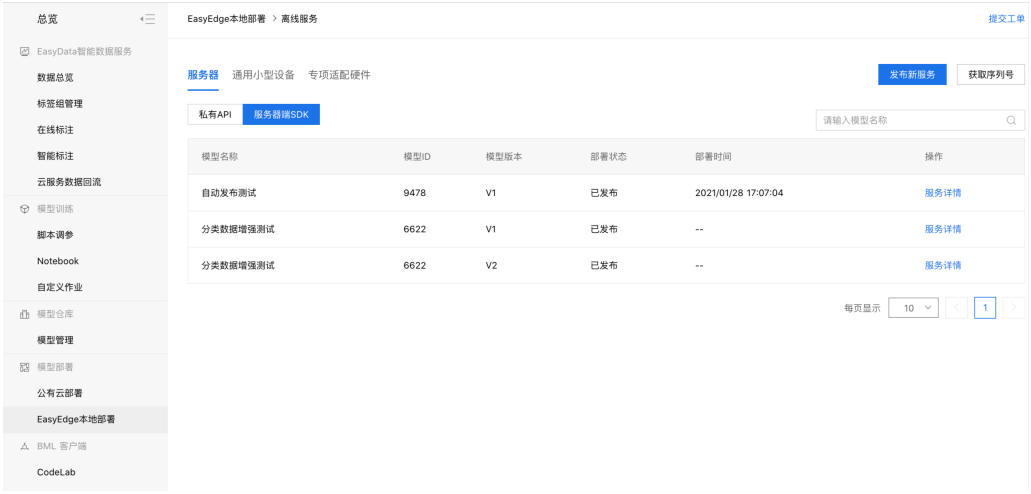
以及系统相关信息。

遇到这类情况，请确保硬件无变更，如果想更换序列号，请先删除 `C:\Users\${用户名}\.baidu\easyedge` 目录，再重新激活。

其他问题 如果无法解决，可到论坛发帖： https://ai.baidu.com/forum/topic/list/199 描述使用遇到的问题，我们将及时回复您的问题。

如何发布服务器端SDK

训练完毕后，可以在左侧导航栏中找到【EasyEdge本地部署】，选择通用【服务器】，点击【发布新服务】，进入发布界面。



step1.部署方式选择服务器，集成方式选择服务器端后，选择需要发布的模型及版本，勾选业务场景需要的操作系统与硬件芯片。设置完毕点击下一步



step2. 填写完基本信息之后，点击"提交"，即可将模型封装成可离线计算的服务器端SDK。



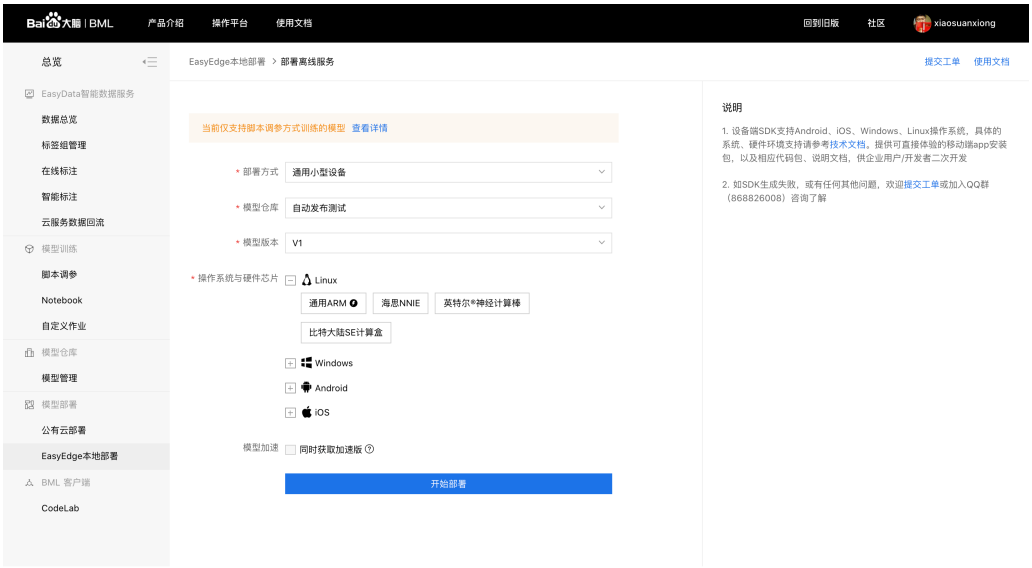
设备端SDK部署

如何发布视觉任务设备端SDK

训练完毕后，可以在左侧导航栏中找到【EasyEdge本地部署】，选择通用小型设备，点击【发布新服务】

只需勾选业务场景需要的操作系统与硬件芯片，即可将模型封装成可离线计算的设备端SDK。

部署模型界面示意：



设备端SDK目前已全面支持Android、iOS、Windows、Linux操作系统，同时提供可直接体验的移动端app安装包，以及相应代码包、说明文档，供企业用户/开发者二次开发

如SDK生成失败，或有任何其他问题，欢迎提交工单咨询了解

视觉任务设备端SDK使用说明

本文档主要说明定制化模型设备端SDK如何使用，如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入EasyDL社区交流，与其他开发者进行互动
- 加入EasyDL官方QQ群（群号:868826008）联系群管

SDK说明

目前图像分类和物体检测模型均可封装为可离线计算的设备端SDK，支持iOS、Android、Linux、Windows四种操作系统。

以下为具体的系统、硬件环境支持：

操作系统	系统支持	硬件环境要求
iOS	iOS 8.0 以上（A仿生芯片版要求11.0以上）	ARMv7 ARM64 (Starndard architectures)（暂不支持模拟器）
Android	通用ARM: Android 19以上 SNPE：Android 21以上 DDK：Android 21以上	通用ARM: 绝大部分的手机和平板 、比较耗时 SNPE：高通Soc，仅支持Qualcomm Snapdragon 450 之后发布的soc。其中 660 之后的型号可能含有Hexagon DSP模块，具体列表见snpe 高通骁龙引擎 DDK：CPU支持华为麒麟970N、980的arm-v8a的soc，支持的机型 mate10，mate10pro，P20，mate20等 支持armeabi-v7a arm-v8a CPU 架构, DDK仅支持 arm-v8a
Linux C++		CPU: AArch64 ARMv7l ASIC: Hisilicon NNIE1.1 on AArch64（Hi3559AV100/Hi3559CV100等） ASIC: Hisilicon NNIE1.2 on ARMv7l（Hi3519AV100/Hi3559V200等）
Linux Python		Intel Movidius MyRIAD2/MyRIAD X
Windows	64位 Windows7 及以上	Intel CPU x86_64 Intel Movidius MyRIAD2/MyRIAD X (仅支持Win10) 环境依赖： .NET Framework 4.5 Visual C++ Redistributable Packages for Visual Studio 2013 Visual C++ Redistributable Packages for Visual Studio 2015

说明

> 针对iOS操作系统：虽然SDK库文件很大（ipa文件很大），但最终应用在用户设备中所占用的大小会**缩小很多，如图像分类下载的ipa文件可能会100M以上，但实际集成在设备中只有20M左右**。 这与multi architechtures、bitcode和AppStore的优化有关。

单次预测耗时参考

> 根据具体设备、线程数不同，数据可能有波动，请以实测为准

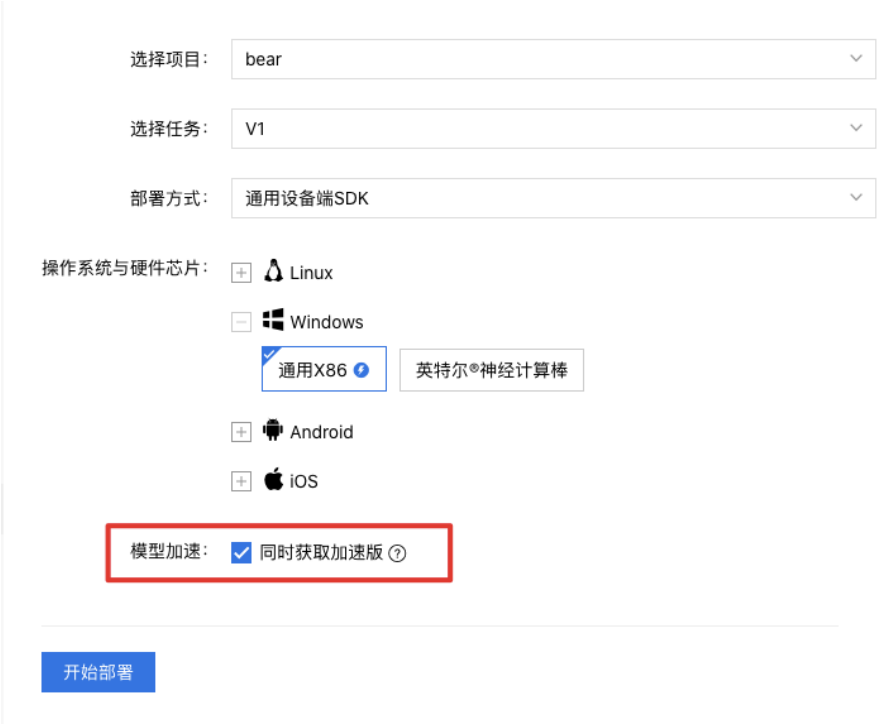
芯片类型	实测硬件	图像分类模型	物体检测模型	备注
ARM	RK3399	64-1560ms	239-4100ms	不同模型网络差异较大
Myriad 2	Intel Movidius 1	40-615ms	-	
Myriad X	Intel Movidius 2	25-238ms	-	
海思NNIE	海思Hi3559A	38-42ms	-	暂不支持SE-ResNeXt
ARM	iPhone XR	23-335ms	209ms	
ARM	HUAWEI Mate 20	29-563ms	-ms	
苹果A芯片	iPhone XR	-ms	-	
高通骁龙SNPE	XIAOMI 8	11-81ms	-	
麒麟NPU 980	HUAWEI Mate 20	-ms	-	
麒麟NPU 970	HUAWEI Mate 10 Pro	-ms	-	
x86-64	Xeon E5-2650 v4	114-4021ms	162-3600ms	

自适应芯片版SDK

发布SDK时可根据实际应用时的硬件/芯片配置选择最合适的SDK。如“华为NPU版”就是针对华为NPU芯片做了适配与加速的SDK。如实际应用时需要适配多种芯片，就可以选择“自适应芯片版”SDK，SDK被集成后会自动判断设备的芯片并运行相应的模型。

加速版SDK

发布SDK时，勾选「同时获取加速版」，就可以同时获得适配部分芯片（需选中且右侧带有加速标记）的基础版SDK和加速版SDK。



目前加速版SDK已支持部分图像分类网络的Windows X86、Linux ARM、iOS ARM、Android ARM环境，加速后的SDK普遍在包大小、内存占用、识别速度等方面表现更优，以SE-ResNeXt50为例：

模型算法	芯片类型	操作系统	实测硬件	SDK类型	单次预测耗时
SE-ResNeXt50	ARM	Linux	RK3399	基础版/加速版	564ms/408ms
	ARM	Android	HUAWEI Mate 20	基础版/加速版	323ms/167ms
	ARM	iOS	iPhone XR	基础版/加速版	308ms/90ms

加速版SDK和基础版的测试方式类似，只需在EasyDL控制台新增「加速版」测试序列号，即可获得3个月的测试期。

激活&使用SDK

SDK的激活与使用分以下四步：

① 在【我的服务】-【服务详情】内下载SDK：



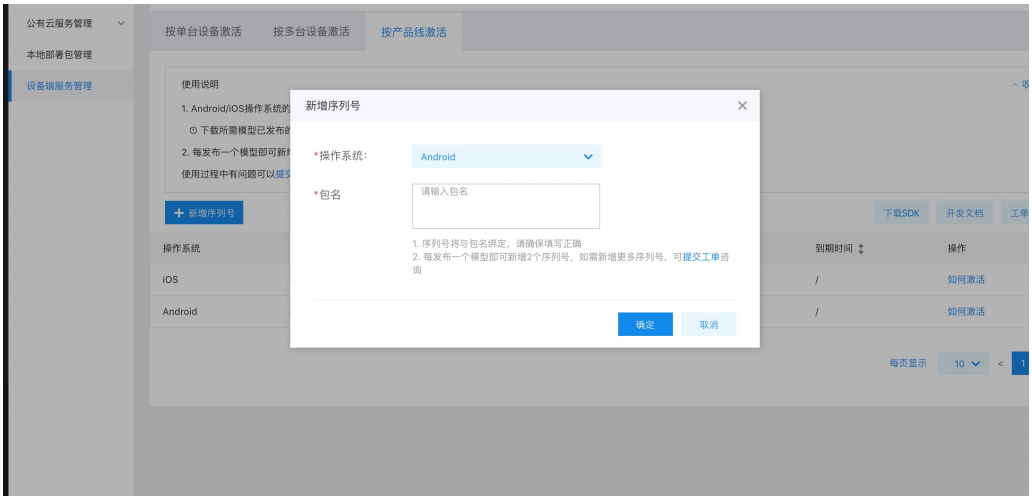


② 在BML控制台获取序列号

按设备获得授权并使用SDK时，每部署一个项目，即可新增2个序列号：



Android或iOS操作系统的SDK可以选择按产品线激活，序列号与包名（Package Name/Bundle ID）绑定：



③ 本地运行SDK，并完成首次联网激活

> 通过左侧导航栏查看不同操作系统SDK的开发文档

④ 正式使用

SDK常见问题

> 通过左侧导航栏查看不同操作系统SDK的FAQ

以下是通用FAQ，如您的问题仍未解决，欢迎加入官方QQ群813291890咨询群管

1、激活失败怎么办？

- 按设备激活时，激活失败可能由于以下几个原因造成：

①可能是当前序列号已被其他设备激活使用，请核实序列号后用未被激活的序列号重新激活

②序列号填写错误，请核实序列号后重新激活

③同一台设备绑定同一个序列号激活次数过多（超过50次），请更换序列号后重试

④首次激活需要联网，网络环境不佳或无网络环境，请检查网络环境后重试

⑤模型发布者和序列号所属账号非同一账号，如果存在这种异常建议更换账号获取有效序列号

⑥序列号已过有效期，请更换序列号后重试

- 按产品线激活时，激活失败可能由于以下几个原因造成：

①可能是包名填写错误，请核对与序列号绑定的包名是否与实际包名一致

②序列号填写错误，请核实序列号后重新激活

③首次激活需要联网，网络环境不佳或无网络环境，请检查网络环境后重试

④模型发布者和序列号所属账号非同一账号，如果存在这种异常建议更换账号获取有效序列号

⑤序列号已过有效期，请申请延期后重试

2、怎样申请序列号使用延期

序列号激活后有效期为三个月，可以在需要延期的序列号右侧「操作」找到申请延期的入口

视觉任务LinuxSDK集成文档-C++

简介

本文档介绍EasyEdge/EasyDL的Linux CPP SDK的使用方法。

- 网络类型支持： - 图像分类 - 物体检测 - 图像分割
- 硬件支持：
  - CPU: aarch64 armv7hf
  - GPU: ARM Mali G系列
  - ASIC: Hisilicon NNIE1.1 on aarch64（Hi3559AV100/Hi3559CV100等）
  - ASIC: Hisilicon NNIE1.2 on armv7l（Hi3519AV100/Hi3559V200等）
  - Intel Movidius MyRIAD2 / MyRIAD X on x86_64
  - Intel Movidius MyRIAD2 / MyRIAD X on armv7l
  - Intel Movidius MyRIAD2 / MyRIAD X on aarch64
  - Intel iGPU on x86_64
  - 比特大陆 Bitmain SE50 (BM1684)
  - 瑞芯微 RK3399Pro / RV1109 / RV1126 / RK3568
  - 华为 Atlas200
  - 晶晨 A311D
  - 寒武纪 MLU220 on aarch64
- 操作系统支持：

- Linux (Ubuntu, Centos, Debian等)
- 海思HiLinux
- 树莓派Raspbian/Debian
- 瑞芯微Firefly

性能数据参考算法性能及适配硬件

Release Notes | 时间 | 版本 | 说明 | | --- | ---- | ---- | | 2022.10.27 | 1.7.1 | 新增语义分割模型http请求示例 | | 2022.09.15 | 1.7.0 | 新增瑞芯微RK3568 支持, RK3399Pro、RV1126升级到RKNN1.7.1 | | 2022.07.28 | 1.6.0 | 引擎升级；新增英特尔 iGPU 支持 | | 2022.04.25 | 1.4.1 | EasyDL, BML升级支持paddle2模型 | | 2022.03.25 | 1.4.0 | EasyDL新增上线支持晶晨A311D NPU预测引擎；Arm CPU、Arm GPU引擎升级；atlas 200在EasyDL模型增加多个量化加速版本； | | 2021.12.22 | 1.3.5 | RK3399Pro, RV1109/RV1126 SDK扩展模型压缩加速能力，更新端上推理库版本;边缘控制台IEC功能升级，适配更多通用小型设备，NNIE 在EasyDL增加量化加速版本；Atlas200升级到Cann5.0.3 | | 2021.06.29 | 1.3.1 | 视频流解析支持调整分辨率；预测引擎升级；设备端sdk新增支持瑞芯微RV1109、RV1126 | | 2021.05.13 | 1.3.0 | 新增视频流接入支持；EasyDL模型发布新增多种加速方案选择；目标追踪支持x86平台的CPU、GPU加速版；展示已发布模型性能评估报告 | | 2021.03.09 | 1.2.0 | http server服务支持图片通过base64格式调用 | | 2021.01.27 | 1.1.0 | EasyDL经典版分类高性能模型升级；部分SDK不再需要单独安装OpenCV；新增RKNPU预测引擎支持；新增高通骁龙GPU预测引擎支持 | | 2020.12.18 | 1.0.0 | 1.0版本发布！安全加固升级、性能优化、引擎升级、接口优化等多项更新 | | 2020.10.29 | 0.5.7 | 优化多线程预测细节 | | 2020.09.17 | 0.5.6 | 支持linux aarch64架构的硬件接入intel神经计算棒预测；支持比特大陆计算盒SE50 BM1684 | | 2020.08.11 | 0.5.5 | 支持linux armv7hf架构硬件(如树莓派)接入intel神经计算棒预测 | | 2020.06.23 | 0.5.4 | arm引擎升级 | | 2020.05.15 | 0.5.3 | 支持EasyDL 专业版新增模型 ；支持树莓派(armv7hf, aarch64)| | 2020.04.16 | 0.5.2 | Jetson系列SDK支持多线程infer | | 2020.02.23 | 0.5.0 | 新增支持人脸口罩模型；Jetson SDK支持批量图片推理; ARM支持图像分割 | | 2020.01.16 | 0.4.7 | 上线海思NNIE1.2，支持EasyEdge以及EasyDL；ARM引擎升级；增加推荐阈值支持 | | 2019.12.26 | 0.4.6 | 海思NNIE支持EasyDL专业版 | | 2019.11.02 | 0.4.5 | 移除curl依赖；支持自动编译OpenCV；支持EasyDL 专业版 Yolov3; 支持EasyDL经典版高精度物体检测模型升级| | 2019.10.25 | 0.4.4 | ARM引擎升级，性能提升30%； 支持EasyDL专业版模型 | | 2019.09.23 | 0.4.3 | 增加海思NNIE加速芯片支持 | | 2019.08.30 | 0.4.2 | ARM引擎升级；支持分类高性能与高精度模型 | | 2019.07.25 | 0.4.1 | 引擎升级，性能提升 | | 2019.06.11 | 0.3.3 | paddle引擎升级；性能提升 | | 2019.05.16 | 0.3.2 | 新增armv7l支持 | | 2019.04.25 | 0.3.1 | 优化硬件支持 | | 2019.03.29 | 0.3.0 | ARM64 支持；效果提升 | | 2019.02.20 | 0.2.1 | paddle引擎支持；效果提升 | | 2018.11.30 | 0.1.0 | 第一版！|

> 【1.0 接口升级】 参数配置接口从1.0.0版本开始已升级为新接口，以前的方式被置为deprecated，并将在未来的版本中移除。请尽快考虑升级为新的接口方式，具体使用方式可以参考下文介绍以及demo工程示例。 【关于SDK包与RES模型文件夹配套使用的说明】 我们强烈建议用户使用部署tar包中配套的SDK和RES一起使用。 更新模型时，如果SDK版本号有更新，请务必同时更新SDK，旧版本的SDK可能无法正确适配新发布出来的RES。

快速开始

SDK在以下环境中测试通过

- aarch64(arm64), Ubuntu 16.04, gcc 5.3 (RK3399)
- Hi3559AV100, aarch64, Ubuntu 16.04, gcc 5.3
- Hi3519AV100, armv7l , HiLinux 4.9.37, (Hi3519AV100R001C02SPC020)
- armv7hf, Raspbian, (Raspberry 3b)
- aarch64, Raspbian, (Raspberry 4b)
- armv7hf, Raspbian, (Raspberry 3b+)
- armv7hf, Ubuntu 16.04, (RK3288)
- Bitmain se50 BM1684, Debian 9
- Rockchip rk3399pro, Ubuntu 18.04
- Rockchip rv1126, Debain 10
- Rockchip rk3568, Ubuntu 20.04
- Atlas200(华为官网指定的Ubuntu 18.04版本)
- Amlogic A311D, Ubuntu 20.04

- MLU220, aarch64, Ubuntu 18.04

安装依赖

依赖包括

- cmake 3+
- gcc 5.4 以上(需包含 GLIBCXX_3.4.22) ，gcc / glibc版本请以实际SDK ldd的结果为准
- opencv3.4.5 (可选)

依赖说明：树莓派 树莓派Raspberry默认为armv7hf系统，使用SDK包中名称中包含`armv7hf_ARM_`的tar包。如果是aarch64系统，使用SDK包中名称中包含`aarch64_ARM_`的tar包。

在安装前可通过以下命令查看是32位还是64位 ：

```
 getconf LONG_BIT
32
```

依赖说明：比特大陆SE计算盒 需要安装BMNNSDK2.2，并在CMakeList.txt中指定SDK安装地址：

```
 ###### 这里修改并填入所使用的bmnnsdk路径
set(EDGE_BMSDK_ROOT "{这里填写sdk路径}")
```

依赖说明：海思开发板 海思开发板需要根据海思SDK文档配置开运行环境和编译环境，SDK和opencv都需要在该编译环境中编译。 NNIE1.2用arm-himix200-linux交叉编译好的opencv，下载链接:https://pan.baidu.com/s/13QW0ReeWx4ZwgYg4lretyw 密码:yq0s。下载后修改SDK CMakesList.txt

依赖说明：RK3399Pro 所有用例基于 Npu driver版本1.7.1的RK3399pro开发板测试通过，SDK采用预编译模式，请务必确保板上驱动版本为1.7.1 查看RK3399Pro板上driver版本方法：`dpkg -l | grep 3399pro`

依赖说明：RV1109/RV1126 所有用例基于Rknn_server版本1.7.1的RV1126开发板测试通过，SDK采用预编译模式，请务必确保板上驱动版本为1.7.1 查看RV1109/RV1126板上Rknn_server版本方法：`strings /usr/bin/rknn_server | grep build`

依赖说明：RK3568 所有用例基于Rknn_server版本1.2.0的RK3568开发板测试通过， 查看RK3568板上Rknn_server版本方法：`strings /usr/bin/rknn_server | grep build`

依赖说明：晶晨A311D 所有用例基于晶晨A311D开发板测试通过，需要驱动版本为 6.4.4.3（下载驱动请联系开发版厂商） 查看晶晨A311D开发板驱动版本方法：`dmesg | grep Galcore`

使用序列号激活 在控制台获取的序列号请通过`global_controller()->set_licence_key("")`方法设置。

具体请参考SDK自带的`Demo.cpp`文件的使用方法。

测试Demo

模型资源文件默认已经打包在开发者下载的SDK包中。

Demo工程直接编译即可运行。

> 请先将tar包整体拷贝到具体运行的设备中，再解压缩编译； 对于硬件使用为： **-Intel Movidius MyRIAD2 / MyRIAD X / IGPU on Linux x86_64 / armv7hf / aarch64**，在编译或运行demo程序前执行以下命令: `source ${cpp_kit位置路径}/thirdparty/openvino/bin/setupvars.sh` 或者执行 `source ${cpp_kit位置路径}/thirdparty/openvino/setupvars.sh` (openvino-2022.1+) 部分SDK中已经包含预先编译的二进制, bin/easyedge_demo, bin/easyedge_serving，配置LD_LIBRARY_PATH后，可直接运行: LD_LIBRARY_PATH=../lib ./bin/easyedge_serving

请在官网获取序列号，填写在`demo.cpp`中



编译运行：

```
 cd src
mkdir build && cd build
cmake .. && make
./easyedge_image_inference {模型RES文件夹} {测试图片路径}
###### 如果是NNIE引擎，使用sudo运行
sudo ./easyedge_image_inference {模型RES文件夹} {测试图片路径}
```

如果希望SDK自动编译安装所需要的OpenCV库，修改cmake的option`EDGE_BUILD_OPENCV`为`ON`即可。 SDK会自动从网络下载opencv源码，并编译需要的module、链接。注意，此功能必须需联网。

```
cmake -DEDGE_BUILD_OPENCV=ON .. && make -j16
```

若需自定义library search path或者gcc路径，修改CMakeList.txt即可。

> 对于硬件使用为**Intel Movidius MyRIAD2 / MyRIAD X** 的，如果宿主机找不到神经计算棒Intel® Neural Compute Stick，需要执行以下命令添加USB Rules：

```
 cp ${cpp_kit位置路径}/thirdparty/openvino/deployment_tools/inference_engine/external/97-myriad-usbboot.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules
sudo udevadm trigger
sudo ldconfig
```

demo运行效果：




```
 > ./easyedge_image_inference ../../../../RES 2.jpeg
2019-02-13 16:46:12,659 INFO [EasyEdge] [easyedge.cpp:34] 140606189016192 Baidu EasyEdge Linux Development Kit
0.2.1(20190213)
2019-02-13 16:46:14,083 INFO [EasyEdge] [paddlev2_edge_predictor.cpp:60] 140606189016192 Allocate graph success.
2019-02-13 16:46:14,326 DEBUG [EasyEdge] [paddlev2_edge_predictor.cpp:143] 140606189016192 Inference costs 168 ms
1, 1:txt_frame, p:0.994905 loc: 0.168161, 0.153654, 0.920856, 0.779621
Done
```

测试Demo HTTP 服务

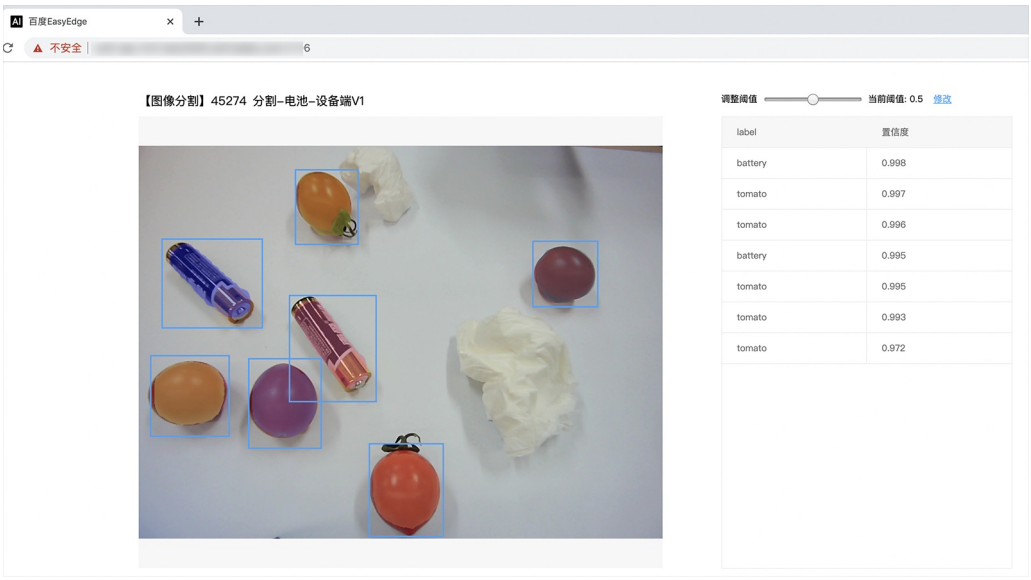
编译demo完成之后，会同时生成一个http服务 运行

```
 ###### ./easyedge_serving {res_dir} {serial_key} {host, default 0.0.0.0} {port, default 24401}
./easyedge_serving ../../../../RES "1111-1111-1111-1111" 0.0.0.0  24401
```

后，日志中会显示

```
HTTP is now serving at 0.0.0.0:24401
```

字样，此时，开发者可以打开浏览器，`http://{设备ip}:24401`，选择图片来进行测试。



同时，可以调用HTTP接口来访问服务，具体参考下文接口说明。

使用说明

使用该方式，将运行库嵌入到开发者的程序当中。

使用流程

> 请优先参考Demo的使用流程。遇到错误，请优先参考文件中的注释解释，以及日志说明。

```
    // step 1: 配置运行参数
    EdgePredictorConfig config;
    config.model_dir = {模型文件目录};

    // step 2: 创建并初始化Predictor；这这里选择合适的引擎
    auto predictor = global_controller()->CreateEdgePredictor(config);

    // step 3-1: 预测图像
    auto img = cv::imread({图片路径});
    std::vector<EdgeResultData> results;
    predictor->infer(img, results);

// step 3-2: 预测视频
std::vector<EdgeResultData> results;
FrameTensor frame_tensor;
VideoConfig video_config;
video_config.source_type = static_cast<SourceType>(video_type);  // source_type 定义参考头文件 easyedge_video.h
video_config.source_value = video_src;
/*
... more video_configs, 根据需要配置video_config的各选项
*/
auto video_decoding = CreateVideoDecoding(video_config);
while (video_decoding->next(frame_tensor) == EDGE_OK) {
    results.clear();
    if (frame_tensor.is_needed) {
        predictor->infer(frame_tensor.frame, results);
        render(frame_tensor.frame, results, predictor->model_info().kind);
    }
    //video_decoding->display(frame_tensor); // 显示当前frame，需在video_config中开启配置
    //video_decoding->save(frame_tensor); // 存储当前frame到视频，需在video_config中开启配置
}
```

对于口罩检测模型，将 EdgePredictorConfig config修改为PaddleMultiStageConfig config即可。

> 口罩检测模型请注意输入图片中人脸大小建议保持在 *88到96*96像素之间，可根据场景远近程度缩放图片后再传入SDK。

SDK参数配置 SDK的参数通过EdgePredictorConfig::set_config和global_controller()->set_config配置。 set_config的所有key在easyedge_xxxx_config.h中。其中

- PREDICTOR前缀的key是不同模型相关的配置，通过EdgePredictorConfig::set_config设置
- CONTROLLER前缀的key是整个SDK的全局配置，通过global_controller()->set_config设置

以序列号为例，KEY的说明如下：

```
/**
* @brief 序列号设置；序列号不设置留空时，SDK将会自动尝试使用本地已经激活成功的有效期内的序列号
* 值类型：string
* 默认值：空
*/
static constexpr auto PREDICTOR_KEY_SERIAL_NUM = "PREDICTOR_KEY_SERIAL_NUM";
```

使用方法如下：

```
EdgePredictorConfig config;
config.model_dir = ...;
config.set_config(params::PREDICTOR_KEY_SERIAL_NUM, "1DB7-1111-1111-D27D");
```

具体支持的运行参数可以参考开发工具包中的头文件的详细说明。

初始化

- 接口

```
auto predictor = global_controller()->CreateEdgePredictor(config);
predictor->init();
```

若返回非0，请查看输出日志排查错误原因。

预测图像

- 接口

```
/**
* @brief
* 通用接口
* @param image: must be BGR , HWC format (opencv default)
* @param result
* @return
*/
virtual int infer(
        cv::Mat& image, std::vector<EdgeResultData>& result
) = 0;
```

图片的格式务必为opencv默认的BGR, HWC格式。

- 返回格式

EdgeResultData中可以获取对应的分类信息、位置信息。

```
struct EdgeResultData {
  int index;  // 分类结果的index
  std::string label;  // 分类结果的label
  float prob;  // 置信度

  // 物体检测活图像分割时才有
  float x1, y1, x2, y2;  // (x1, y1): 左上角，  (x2, y2): 右下角； 均为0~1的长宽比例值。

  // 图像分割时才有
  cv::Mat mask;  // 0, 1 的mask
  std::string mask_rle;  // Run Length Encoding，游程编码的mask
};
```

关于矩形坐标

x1 * 图片宽度 = 检测框的左上角的横坐标

y1 * 图片高度 = 检测框的左上角的纵坐标

x2 * 图片宽度 = 检测框的右下角的横坐标

y2 * 图片高度 = 检测框的右下角的纵坐标

关于图像分割mask

```
 cv::Mat mask为图像掩码的二维数组
{
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
  {0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
  {0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
  {0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
}
其中1代表为目标区域，0代表非目标区域
```

关于图像分割mask_rle

该字段返回了mask的游程编码，解析方式可参考 http demo

以上字段可以参考demo文件中使用opencv绘制的逻辑进行解析

预测视频

SDK 提供了支持摄像头读取、视频文件和网络视频流的解析工具类`VideoDecoding`，此类提供了获取视频帧数据的便利函数。通过`VideoConfig`结构体可以控制视频/摄像头的解析策略、抽帧策略、分辨率调整、结果视频存储等功能。对于抽取到的视频帧可以直接作为SDK infer 接口的参数进行预测。

- 接口

class`VideoDecoding`：

```
    /**
     * @brief 获取输入源的下一帧
     * @param frame_tensor
     * @return
     */
    virtual int next(FrameTensor &frame_tensor) = 0;

    /**
     * @brief 显示当前frame_tensor中的视频帧
     * @param frame_tensor
     * @return
     */
    virtual int display(const FrameTensor &frame_tensor) = 0;

    /**
     * @brief 将当前frame_tensor中的视频帧写为本地视频文件
     * @param frame_tensor
     * @return
     */
    virtual int save(FrameTensor &frame_tensor) = 0;

    /**
     * @brief 获取视频的fps属性
     * @return
     */
    virtual int get_fps() = 0;
     /**
      * @brief 获取视频的width属性
      * @return
      */
    virtual int get_width() = 0;

    /**
     * @brief 获取视频的height属性
     * @return
     */
    virtual int get_height() = 0;
```

struct `VideoConfig`

```
 /**
 * @brief 视频源、抽帧策略、存储策略的设置选项
 */
struct VideoConfig {
    SourceType source_type;           // 输入源类型
    std::string source_value;         // 输入源地址，如视频文件路径、摄像头index、网络流地址
    int skip_frames{0};               // 设置跳帧，每隔skip_frames帧抽取一帧，并把该抽取帧的is_needed置为true
    int retrieve_all{false};          // 是否抽取所有frame以便于作为显示和存储，对于不满足skip_frames策略的frame，把所抽取帧的
is_needed置为false
    int input_fps{0};                 // 在采取抽帧之前设置视频的fps
    Resolution resolution{Resolution::kAuto}; // 采样分辨率，只对camera有效

    bool enable_display{false};
    std::string window_name{"EasyEdge"};
    bool display_all{false};          // 是否显示所有frame，若为false，仅显示根据skip_frames抽取的frame

    bool enable_save{false};
    std::string save_path;            // frame存储为视频文件的路径
    bool save_all{false};             // 是否存储所有frame，若为false，仅存储根据skip_frames抽取的frame

    std::map<std::string, std::string> conf;
};
```

`source_type`：输入源类型，支持视频文件、摄像头、网络视频流三种，值分别为1、2、3。 `source_value`： 若`source_type`为视频文件，该值为指向视频文件的完整路径；若`source_type`为摄像头，该值为摄像头的index，如对于/dev/video0的摄像头，则index为0；若`source_type`为网络视频流，则为该视频流的完整地址。 `skip_frames`：设置跳帧，每隔skip_frames帧抽取一帧，并把该抽取帧的is_needed置为true，标记为is_needed的帧是用来做预测的帧。反之，直接跳过该帧，不经过预测。 `retrieve_all`：若置该项为true，则无论是否设置跳帧，所有的帧都会被

抽取返回，以作为显示或存储用。 input_fps：用于抽帧前设置fps。 resolution：设置摄像头采样的分辨率，其值请参考easyedge_video.h中的定义，注意该分辨率调整仅对输入源为摄像头时有效。 conf：高级选项。部分配置会通过该map来设置。

注意： 1.如果使用VideoConfig的display功能，需要自行编译带有GTK选项的opencv，默认打包的opencv不包含此项。 2.使用摄像头抽帧时，如果通过resolution设置了分辨率调整，但是不起作用，请添加如下选项：

```
video_config.conf["backend"] = "2";
```

3.部分设备上的CSI摄像头尚未兼容，如遇到问题，可以通过工单、QQ交流群或微信交流群反馈。

具体接口调用流程，可以参考SDK中的demo_video_inference。

设置序列号

请在网页控制台中申请序列号，并在init初始化前设置。 LinuxSDK 首次使用需联网授权。

```
EdgePredictorConfig config;
config.set_config(easyedge::params::PREDICTOR_KEY_SERIAL_NUM, "this-is-serial-num");
```

日志配置

设置 EdgeLogConfig 的相关参数。具体含义参考文件中的注释说明。

```
EdgeLogConfig log_config;
log_config.enable_debug = true;
global_controller()->set_log_config(log_config);
```

http服务

1. 开启http服务 http服务的启动参考demo_serving.cpp文件。

```
/**
 * @brief 开启一个简单的demo http服务。
 * 该方法会block直到收到sigint/sigterm。
 * http服务里，图片的解码运行在cpu之上，可能会降低推理速度。
 * @tparam ConfigT
 * @param config
 * @param host
 * @param port
 * @param service_id service_id  user parameter, uri '/get/service_id' will respond this value with 'text/plain'
 * @param instance_num 实例数量，根据内存/显存/时延要求调整
 * @return
 */
template<typename ConfigT>
int start_http_server(
        const ConfigT &config,
        const std::string &host,
        int port,
        const std::string &service_id,
        int instance_num = 1);
```

2. 请求http服务

> 开发者可以打开浏览器，http://{设备ip}:24401，选择图片来进行测试。

http 请求方式一：不使用图片base64格式 URL中的get参数：

参数	说明	默认值
threshold	阈值过滤， 0~1	如不提供，则会使用模型的推荐阈值

HTTP POST Body即为图片的二进制内容(无需base64, 无需json)

Python请求示例 (针对非语义分割模型)

```
import requests

with open('./1.jpg', 'rb') as f:
    img = f.read()
    result = requests.post(
     'http://127.0.0.1:24401/',
     params={'threshold': 0.1},
     data=img).json()
```

Python请求示例 (针对语义分割模型)

```
import requests

with open('./1.jpg', 'rb') as f:
    img_data = f.read()
    res = requests.post('http://127.0.0.1:24401/',
        data=img_data)
    with open("gray_result.png", "wb") as fb:
        fb.write(res.content) # 语义分割模型是像素点级别输出，可将api返回结果保存为灰度图，每个像素值代表该像素分类结果
```

Java请求示例

http 请求方法二：使用图片base64格式 HTTP方法：POST Header如下：

参数	值
Content-Type	application/json

Body请求填写：

- 分类网络： body 中请求示例

```
{
"image": "<base64数据>"
"top_num": 5
}
```

body中参数详情

参数	是否必选	类型	可选值范围	说明
image	是	string	-	图像数据，base64编码，要求base64图片编码后大小不超过4M,最短边至少15px，最长边最大4096px，支持jpg/png/bmp格式 **注意去掉头部**
top_num	否	number	-	返回分类数量，不填该参数，则默认返回全部分类结果

- 检测和分割网络： Body请求示例：

```
{
"image": "<base64数据>"
}
```

body中参数详情：

参数	是否必选	类型	可选值范围	说明
image	是	string	-	图像数据，base64编码，要求base64图片编码后大小不超过4M,最短边至少15px，最长边最大4096px，支持jpg/png/bmp格式 **注意去掉头部**
threshold	否	number	-	默认为推荐阈值，也可自行根据需要进行设置

Python请求示例 (针对非语义分割模型)

```
 import base64
import requests


def main():
    with open("1.jpg 【图片路径】", 'rb') as f:
        result = requests.post("http://{服务ip地址}:24401/", json={
            "image": base64.b64encode(f.read()).decode("utf8")
        })
        # print(result.request.body)
        # print(result.request.headers)
        print(result.content)


if __name__ == '__main__':
    main()
```

Python 请求示例 (针对语义分割模型)

```
 import base64
import requests
def main():
    with open("1.jpg 【图片路径】", 'rb') as f:
        res = requests.post("http://{服务ip地址}:24401/", json={"image": base64.b64encode(f.read()).decode("utf8")})
        with open("gray_result.png", "wb") as fb:
            fb.write(res.content) # 语义分割模型是像素点级别输出，可将api返回结果保存为灰度图，每个像素值代表该像素分类结果
if __name__ == '__main__':
    main()
```

http 返回数据

字段	类型说明	其他
error_code	Number	0为成功,非0参考message获得具体错误信息
results	Array	内容为具体的识别结果。其中字段的具体含义请参考预测图像-返回格式一节
cost_ms	Number	预测耗时ms，不含网络交互时间

返回示例

```
{
  "cost_ms": 52,
  "error_code": 0,
  "results": [
    {
      "confidence": 0.94482421875,
      "index": 1,
      "label": "IronMan",
      "x1": 0.059185408055782318,
      "x2": 0.18795496225357056,
      "y1": 0.14762254059314728,
      "y2": 0.52510076761245728,
      "mask": "...",  // 图像分割模型字段
      "trackId": 0,  // 目标追踪模型字段
    },

  ]
}
```

其他配置

1. 日志名称、HTTP 网页标题设置

通过global_controller的set_config方法设置：

```
global_controller()->set_config(easyedge::params::KEY_LOG_BRAND, "MY_BRAND");
```

效果如下：



FAQ

1. 如何处理一些 undefined reference / error while loading shared libraries?

> 如：./easyedge_demo: error while loading shared libraries: libeasyedge.so.1: cannot open shared object file: No such file or directory 这是因为二进制运行时ld无法找到依赖的库。如果是正确cmake && make 的程序，会自动处理好链接，一般不会出现此类问题。

遇到该问题时，请找到具体的库的位置，设置LD_LIBRARY_PATH。

> 示例一：libverify.so.1: cannot open shared object file: No such file or directory 链接找不到libveirfy.so文件，一般可通过 export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:../../lib 解决(实际冒号后面添加的路径以libverify.so文件所在的路径为准)

> 示例二：libopencv_videoio.so.4.5: cannot open shared object file: No such file or directory 链接找不到libopencv_videoio.so文件，一般可通过 export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:../../thirdparty/opencv/lib 解决(实际冒号后面添加的路径以libopencv_videoio.so所在路径为准)

2. EasyDL SDK与云服务效果不一致，如何处理？

后续我们会消除这部分差异，如果开发者发现差异较大，可联系我们协助处理。

3. 如何将我的模型运行为一个http服务？ ~~目前cpp sdk暂未集成http运行方式；~~ 0.4.7版本之后，可以通过start_http_server方法开启http服务。

4. 运行NNIE引擎报permission denied 日志显示：

```
 open sys: Permission denied
open err
: Permission denied
open err
: Permission denied
```

请使用sudo在root下运行。

5. 运行SDK报错 Authorization failed

情况一：日志显示 `Http perform failed: null respond` 在新的硬件上首次运行，必须联网激活。

SDK 能够接受`HTTP_PROXY` 的环境变量通过代理处理自己的网络请求。如

```
 export HTTP_PROXY="http://192.168.1.100:8888"
./easyedge_demo ...
```

情况二：日志显示`failed to get/check device id(xxx)`或者`Device fingerprint mismatch(xxx)` 此类情况一般是设备指纹发生了变更，包括（但不局限于）以下可能的情况：

- MAC地址变化
- 磁盘变更
- BIOS重刷

以及系统相关信息。

遇到这类情况，请确保硬件无变更，如果想更换序列号，请先删除 `~/.baidu/easyedge` 目录，再重新激活。

6. 使用libcurl请求http服务时，速度明显变慢

这是因为libcurl请求continue导致server等待数据的问题，添加空的header即可

```
headers = curl_slist_append(headers, "Expect:");
```

7. 运行NNIE引擎报错 std::bad_alloc 检查开发板可用内存，一些比较大的网络占用内存较多，推荐内存500M以上

8. 运行二进制时，提示 libverify.so cannot open shared object file

可能cmake没有正确设置rpath, 可以设置LD_LIBRARY_PATH为sdk的lib文件夹后，再运行：

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:../lib ./easyedge_demo
```

9. 编译时报错：file format not recognized 可能是因为在复制SDK时文件信息丢失。请将整个压缩包复制到目标设备中，再解压缩、编译

视觉任务LinuxSDK集成文档-Python

简介

本文档介绍 EasyDL 的 Linux Python SDK 的使用方法，适用于 EasyDL 通用版和BML。

- 网络类型支持：图像分类，物体检测
- 硬件支持：
  - Intel Movidius MyRIAD2 / MyRIAD X / IGPU
  - 瑞芯微 RK3399Pro
- 语言支持： *Intel Movidius MyRIAD2 / MyRIAD X / IGPU: Python 3.5, 3.6, 3.7* 瑞芯微 RK3399Pro: Python 3.6

Release Notes

时间	版本	说明
2022.10.27	1.3.5	新增Armv7 CPU、Armv8 CPU、Jetson、华为昇腾Atlas开发板对应Python SDK，支持图像分类、物体检测、人脸检测、实例分割；新增 Intel IGPU 支持
2022.05.18	1.3.0	新增RK3399Pro NPU对应Python SDK，支持图像分类、物体检测
2021.01.27	1.2.1	EasyDL经典版分类高性能模型升级；支持更多模型
2020.12.18	1.2.0	推理引擎升级；接口升级；性能优化
2020.09.17	1.1.19	支持更多模型
2020.08.11	1.1.18	性能优化
2020.06.23	1.1.17	支持更多EasyDL专业版模型
2020.04.16	1.1.15	技术优化；升级 OpenVINO 版本
2020.01.16	1.1.12	预测函数默认使用推荐阈值
2019.12.26	1.1.11	EasyDL 专业版支持加速棒
2019.10.21	1.1.9	支持 EasyDL 专业版
2019.07.19	1.1.7	提供模型更新工具
2019.03.15	1.1.0	架构与功能完善
2019.02.28	1.0.6	引擎功能完善
2019.02.13	1.0.5	paddlepaddle 支持
2018.11.30	1.0.0	第一版！

> 2020-12-18: 【接口升级】 序列号配置接口从1.2.0版本开始已升级为新接口，以前的方式被置为deprecated，并将在未来的版本中移除。请尽快考虑升级为新的接口方式，具体使用方式可以参考下文介绍以及demo工程示例，谢谢。

快速开始

1. 安装依赖

根据引擎的不同，SDK 依赖了不同的底层引擎。根据所需自行安装。

依赖说明：Intel Movidius 加速棒 使用`Intel Movidius`加速棒 SDK、`Intel IGPU` 预测时，必须安装 OpenVINO 预测引擎，两种方式：

- 使用 OpenVINO™ toolkit 安装，请参考 OpenVINO toolkit 文档安装 2020.3.1LTS（必须）版本, 安装时可忽略`Configure the Model Optimizer`及后续部分。
- 使用源码编译安装，请参考 Openvino Inference Engine文档编译安装 2020.3.1（必须）版本。

  安装完毕，运行之前，请按照OpenVinono的文档 设置环境变量

```
source /opt/intel/openvino/bin/setupvars.sh
```

依赖说明：RK3399Pro 所有用例基于 Npu driver版本1.6.0的RK3399pro开发板测试通过 查看RK3399Pro板上driver版本方法：`dpkg -l | grep 3399pro`

2. 安装 easyedge python wheel 包 安装说明：Intel Movidius 加速棒 / Intel IGPU

```
pip3 install -U BaiduAI_EasyEdge_SDK-{版本号}-cp36-cp36m-linux_x86_64.whl
```

具体名称以 SDK 包中的 whl 为准。

安装说明：RK3399Pro

```
pip3 install -U BaiduAI_EasyEdge_SDK-{版本号}-cp36-cp36m-linux_aarch64.whl
pip3 install -U EasyEdge_Devkit_RK3399Pro-{版本号}-cp36-cp36m-linux_aarch64.whl
```

具体名称以 SDK 包中的 whl 为准，特别注意这里要同时安装两个whl包 **安装说明：ArmV7 CPU**

```
pip3 install -U BaiduAI_EasyEdge_SDK-{版本号}-cp36-cp36m-linux_armv7l.whl
pip3 install -U EasyEdge_Devkit_ARM-{版本号}-cp36-cp36m-linux_armv7l.whl
```

安装说明：ArmV8 CPU (Aarch64 CPU)

```
pip3 install -U BaiduAI_EasyEdge_SDK-{版本号}-cp36-cp36m-linux_aarch64.whl
pip3 install -U EasyEdge_Devkit_ARM-{版本号}-cp36-cp36m-linux_aarch64.whl
```

安装说明：Jetson SDK

```
pip3 install -U BaiduAI_EasyEdge_SDK-{版本号}-cp36-cp36m-linux_aarch64.whl
pip3 install -U EasyEdge_Devkit_JetPack{版本号}-{版本号}-cp36-cp36m-linux_aarch64.whl
```

安装说明：华为昇腾Atlas开发板

```
pip3 install -U BaiduAI_EasyEdge_SDK-{版本号}-cp36-cp36m-linux_aarch64.whl
EasyEdge_Devkit_Atlas200-{版本号}-cp36-cp36m-linux_aarch64.whl
```

3. 使用序列号激活

获取序列号



修改`demo.py` 填写序列号

```
pred = edge.Program()
pred.set_auth_license_key("这里填写序列号")
```

4. 测试 Demo

输入对应的模型文件夹（默认为`RES`）和测试图片路径，运行：

```
python3 demo.py {model_dir} {image_name.jpg}
```

测试效果：



使用说明

使用流程

```
import BaiduAI.EasyEdge as edge

pred = edge.Program()
pred.set_auth_license_key("这里填写序列号")
pred.init(model_dir={RES文件夹路径}, device=edge.Device.MOVIDIUS, engine=edge.Engine.OPENVINO)
pred.infer_image({numpy.ndarray的图片})
pred.close()
```

初始化

- 接口

```
    def init(self,
             model_dir,
             device=Device.LOCAL,
             engine=Engine.PADDLE_FLUID,
             config_file='conf.json',
             preprocess_file='preprocess_args.json',
             model_file='model',
             params_file='params',
             graph_file='graph.ncsmodel',
             label_file='label_list.txt',
             device_id=0
             ):
        """
        Args:
            device: Device.CPU
            engine: Engine.PADDLE_FLUID
            model_dir: str
                model dir
            preprocess_file: str
            model_file: str
            params_file: str
            graph_file: str
            label_file: str
            device_id: int

        Raises:
            RuntimeError, IOError
        Returns:
            bool: True if success

        """
```

预测图像

- 接口

```
    def infer_image(self, img,
                    threshold=0.3,
                    channel_order='HWC',
                    color_format='BGR',
                    data_type='numpy'):
        """

        Args:
            img: np.ndarray or bytes
            threshold: float
                only return result with confidence larger than threshold
            channel_order: string
                channel order HWC or CHW
            color_format: string
                color format order RGB or BGR
            data_type: string
                image data type

        Returns:
            list

        """
```

- 返回格式: [dict1, dict2, ...]

字段	类型	取值	说明
confidence	float	0~1	分类或检测的置信度
label	string		分类或检测的类别
index	number		分类或检测的类别
x1, y1	float	0~1	物体检测，矩形的左上角坐标 (相对长宽的比例值)
x2, y2	float	0~1	物体检测，矩形的右下角坐标（相对长宽的比例值）
mask	string/numpy.ndarray	图像分割的mask	

关于矩形坐标

x1 * 图片宽度 = 检测框的左上角的横坐标

y1 * 图片高度 = 检测框的左上角的纵坐标

x2 * 图片宽度 = 检测框的右下角的横坐标

y2 * 图片高度 = 检测框的右下角的纵坐标

可以参考 demo 文件中使用 opencv 绘制矩形的逻辑。

结果示例

- i) 图像分类

```
{
  "index": 736,
  "label": "table",
  "confidence": 0.9
}
```

- ii) 物体检测

```
{
  "y2": 0.91211,
  "label": "cat",
  "confidence": 1.0,
  "x2": 0.91504,
  "index": 8,
  "y1": 0.12671,
  "x1": 0.21289
}
```

- iii) 图像分割

```
{
 "name": "cat",
  "score": 1.0,
  "location": {
   "left": ...,
   "top": ...,
   "width": ...,
   "height": ...,
  },
  "mask": ...
}
```

mask字段中，data_type为`numpy`时，返回图像掩码的二维数组

```
{
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
{0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
{0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
{0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
}
其中1代表为目标区域，0代表非目标区域
```

data_type为`string`时，mask的游程编码，解析方式可参考 demo

升级模型 适用于经典版升级模型，执行`bash update_model.sh`，根据提示，输入模型路径、激活码、模型ID、模型版本，等待模型更新完毕即可。

FAQ

Q: 运行SDK报错 Authorization failed

情况一：日志显示 `Http perform failed: null respond` 在新的硬件上首次运行，必须联网激活。

SDK 能够接受`HTTP_PROXY` 的环境变量通过代理处理自己的网络请求。如

```
export HTTP_PROXY="http://192.168.1.100:8888"
./easyedge_demo ...
```

情况二：日志显示`failed to get/check device id(xxx)`或者`Device fingerprint mismatch(xxx)` 此类情况一般是设备指纹发生了变更，包括（但不局限于）以下可能的情况：

- MAC地址变化
- 磁盘变更
- BIOS重刷

以及系统相关信息。

遇到这类情况，请确保硬件无变更，如果想更换序列号，请先删除 `~/.baidu/easyedge` 目录，再重新激活。

情况三：ArmV7、ArmV8 CPU、Jetson、Atlas Python SDK日志提示`ImportError: libavformat.so.58: cannot open shared object file: No such file or`

directory 或者其他类似so找不到 可以在LD_LIBRARY_PATH环境变量加上libs和thirdpartylibs路径，例如

```
export LD_LIBRARY_PATH=/xxx/libs:/xxx/thirdpartylibs:$LD_LIBRARY_PATH  # tips: 这里/xxx需要替换为真实路径，/xxx路径查找方法如下
```

查找安装包内libs和thirdpartylibs路径的方法如下(以Atlas SDK为例，其他SDK查找方法类似)：

```
pip3 show EasyEdge-Devkit-Atlas200  # 结果中会显示 Location 路径，也就是包的安装路径
###### libs和thirdpartylibs两个路径在 Location 所指示的路径 easyedge_CANN 子文件夹下
```

视觉任务WindowsSDK集成文档

简介

本文档介绍Windows CPU SDK的使用方法。

- 网络类型支持：图像分类，物体检测，图像分割
- 硬件支持：
  - Intel CPU 普通版 * x86_64
  - CPU 加速版 - Intel Xeon with AVX2 and AVX512 - *Intel Core Processors with AVX2* - Intel Atom Processors with SSE * - AMD Core Processors with AVX2
  - Intel Movidius MyRIAD2/MyRIAD X (仅支持Win10)
- 操作系统支持
  - 普通版：64位 Windows 7 及以上，64位Windows Server2012及以上
  - 加速版：64位 Windows 10，64位Windows Server 2019及以上
- 环境依赖（必须安装以下版本）
  - .NET Framework 4.5
  - Visual C++ Redistributable Packages for Visual Studio 2013
  - Visual C++ Redistributable Packages for Visual Studio 2015-2019
- 协议
  - HTTP
- 更详细的环境说明可参考SDK内的README.md

> *intel 官方合作，拥有更好的适配与性能表现

Release Notes | 时间 | 版本 | 说明 | | ---------- | ------ | ---------- | | 2022-10-27 | 1.7.1 | 优化模型算法 | | 2022-09-15 | 1.7.0 | 新增支持表格预测 | | 2022-07-28 | 1.6.0 | 优化模型算法 | | 2022-05-27 | 1.5.1 | 新增支持BML Cloud小目标检测模型 | | 2022-05-18 | 1.5.0 | 修复各别机器下程序崩溃的问题 | | 2022-04-25 | 1.4.1 | EasyDL, BML升级支持paddle2模型 | | 2022-03-25 | 1.4.0 | 优化模型算法 | | 2021-12-22 | 1.3.5 | CPU基础版推理引擎优化升级；demo程序优化环境依赖检测 | | 2021-10-20 | 1.3.4 | CPU加速版推理引擎优化升级| | 2021-08-19 | 1.3.2 | 新增DEMO二进制文件 | | 2021-06-29 | 1.3.1 | 预测引擎升级 | | 2021-05-13 | 1.3.0 | 模型发布新增多种加速方案选择；目标追踪支持x86平台的GPU及加速版；展示已发布模型性能评估报告 | | 2021-04-08 | 1.2.3 | 支持BML平台模型仓库本地上传模型 | | 2021-03-09 | 1.2.2| CPU加速版支持int8量化模型 | | 2021-01-27 | 1.2.1| 新增模型支持；性能优化；问题修复 | | 2020.12.18 | 1.2.0 | 推理引擎升级 | | 2020-11-26 | 1.1.20 | 新增一些模型的加速版支持 | | 2020.10.29 | 1.1.20 | 修复已知问题 | | 2020.05.15 | 1.1.16 | 优化性能，修复已知问题 | | 2020-09-17 | 1.1.19| 支持更多模型 | | 2020.08.11 | 1.1.18 | 支持专业版更多模型 | | 2020.06.23 | 1.1.17 | 支持专业版更多模型 | | 2020.05.15 | 1.1.16 | 优化性能，修复已知问题 | | 2020.04.16 | 1.1.15 | 升级引擎版本 | | 2020.03.13 | 1.1.14| 支持EdgeBoardVMX | | 2020.02.23 | 1.1.13| 支持多阶段模型 | | 2020.01.16 | 1.1.12 | 预测默认使用推荐阈值| | 2019.12.26 | 1.1.11 | CPU加速版支持物体检测高精度 | | 2019.12.04 | 1.1.10 | 支持图像分割 | | 2019.10.21 | 1.1.9 | 支持 EasyDL 专业版 | | 2019.08.29 | 1.1.8 | CPU 加速版支持 | | 2019.07.19 | 1.1.7 | 提供模型更新工具 | | 2019.05.16 | 1.1.3 | NVIDIA GPU 支持 | | 2019.03.15 | 1.1.0 | 架构与功能完善 | | 2019.02.28 | 1.0.6 | 引擎功能完善 | | 2019.02.13 | 1.0.5 | paddlepaddle 支持 | | 2018.11.30 | 1.0.0 | 第一版！ |

快速开始

1. 安装依赖

必须安装：

安装.NET Framework4.5

```
https://www.microsoft.com/zh-CN/download/details.aspx?id=42642
```

Visual C++ Redistributable Packages for Visual Studio 2013

```
https://www.microsoft.com/zh-cn/download/details.aspx?id=40784
```

Visual C++ Redistributable Packages for Visual Studio 2015-2019

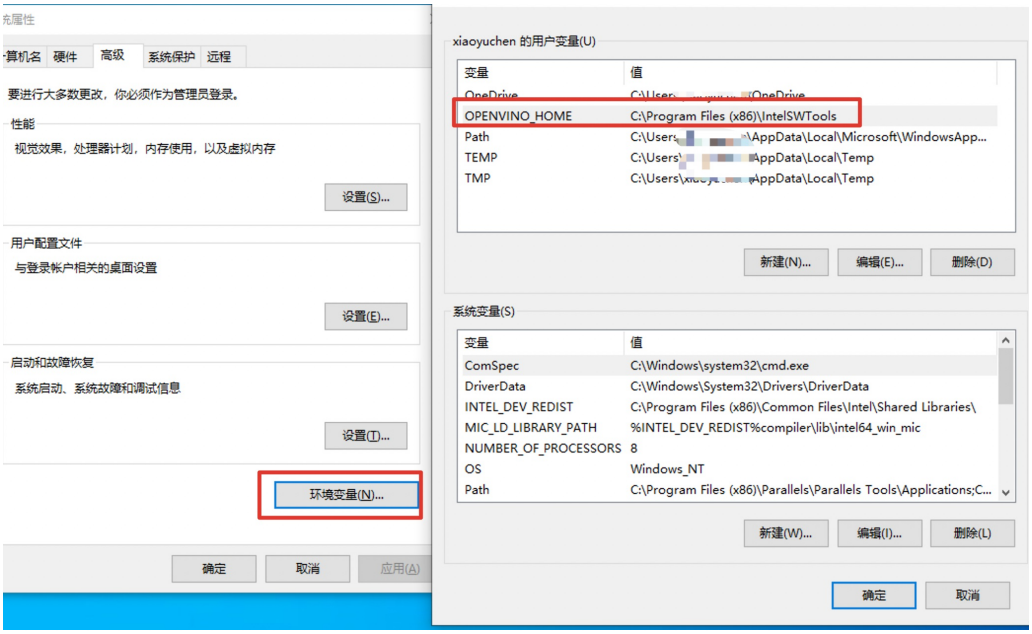
```
https://docs.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?view=msvc-160
```

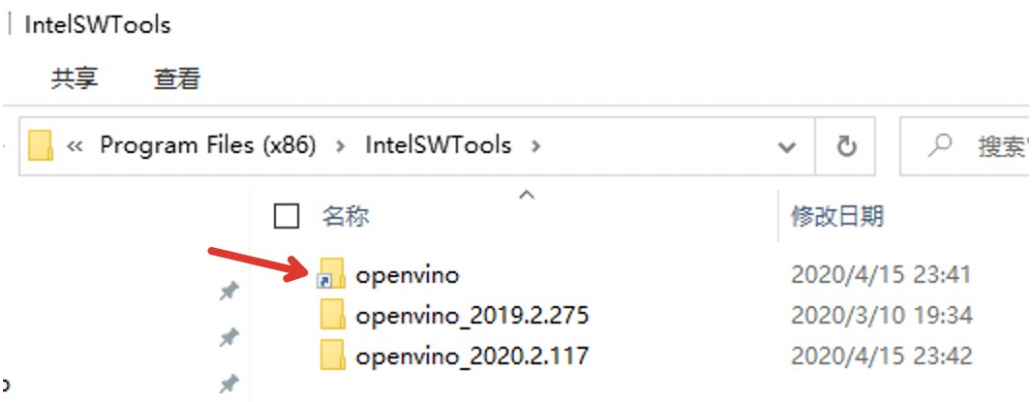
可选安装：

Openvino (使用Intel Movidius必须)

- 使用 OpenVINO™ toolkit 安装，请参考 OpenVINO toolkit 文档安装 2020.3.1LTS（必须）版本, 安装时可忽略Configure the Model Optimizer及后续部分。
- 使用源码编译安装，请参考 Openvino Inference Engine文档编译安装 2020.3.1LTS（必须）版本。

安装完成后，请设置环境变量OPENVINO_HOME为您设置的安装地址，默认是C:\Program Files (x86)\IntelSWTools，并确保文件夹下的openvino的快捷方式指到了2020.3.1LTS版本。





注意事项

1. 安装目录不能包含中文

2. Windows Server 请自行开启，选择“我的电脑”——“属性”——“管理”——”添加角色和功能“——勾选”桌面体验“，点击安装，安装之后重启即可。

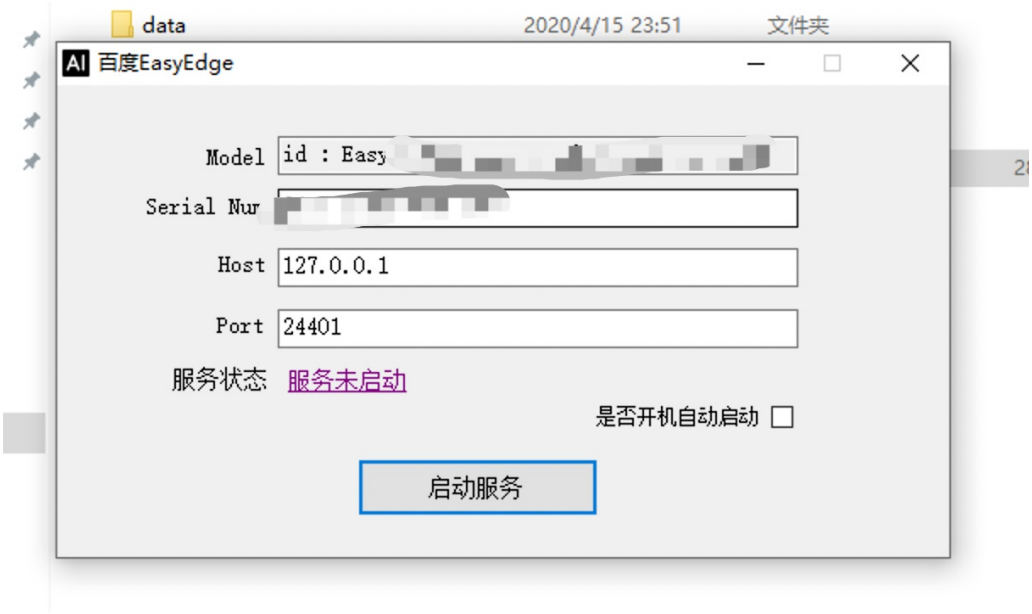
2. 运行离线SDK

解压下载好的SDK，打开EasyEdge.exe，输入Serial Num 

点击"启动服务"，等待数秒即可启动成功，本地服务默认运行在

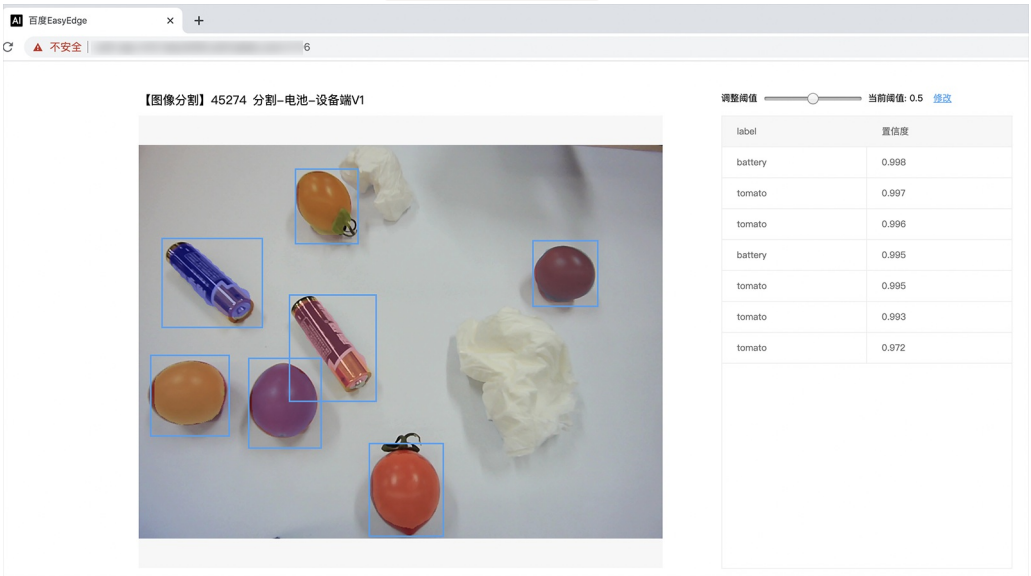
```
http://127.0.0.1:24401/
```

其他任何语言只需通过HTTP调用即可。



Demo示例(以图像服务为例)

服务运行成功，此时可直接在浏览器中输入`http://127.0.0.1:24401`，在h5中测试模型效果。



使用说明

图像服务调用说明

Python 使用示例代码如下

```
 import requests

with open('./1.jpg', 'rb') as f:
    img = f.read()

**params 为GET参数 data 为POST Body**
result = requests.post('http://127.0.0.1:24401/', params={'threshold': 0.1},
                                        data=img).json()
```

C# 使用示例代码如下

```
 FileStream fs = new FileStream("./img.jpg", FileMode.Open);
BinaryReader br = new BinaryReader(fs);
byte[] img = br.ReadBytes((int)fs.Length);
br.Close();
fs.Close();
string url = "http://127.0.0.1:8402?threshold=0.1";
HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(url);
request.Method = "POST";
Stream stream = request.GetRequestStream();
stream.Write(img, 0, img.Length);
stream.Close();

WebResponse response = request.GetResponse();
StreamReader sr = new StreamReader(response.GetResponseStream());
Console.WriteLine(sr.ReadToEnd());
sr.Close();
response.Close();
```

C++ 使用示例代码如下，需要安装curl

```
 ###### include <sys/stat.h>
###### include <curl/curl.h>
###### include <iostream>
###### include <string>
###### define S_ISREG(m) (((m) & 0170000) == (0100000))
###### define S_ISDIR(m) (((m) & 0170000) == (0040000))

size_t write_callback(void *ptr, size_t size, size_t num, void *data) {
    std::string *str = dynamic_cast<std::string *>((std::string *)data);
    str->append((char *)ptr, size*num);
    return size*num;
}

int main(int argc, char *argv[]) {
    const char *post_data_filename = "./img.jpg";
    FILE *fp = NULL;
    std::string response;
    struct stat stbuf = { 0, };
    fp = fopen(post_data_filename, "rb");
    if (!fp) {
        fprintf(stderr, "Error: failed to open file \"%s\"\n", post_data_filename);
        return -1;
    }
    if (fstat(fileno(fp), &stbuf) || !S_ISREG(stbuf.st_mode)) {
        fprintf(stderr, "Error: unknown file size \"%s\"\n", post_data_filename);
        return -1;
    }
    CURL *curl;
    CURLcode res;
    curl_global_init(CURL_GLOBAL_ALL);
    curl = curl_easy_init();
    if (curl != NULL) {
        curl_easy_setopt(curl, CURLOPT_URL, "http://127.0.0.1:24401?threshold=0.1");
        curl_easy_setopt(curl, CURLOPT_POST, 1L);
        curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE_LARGE,(curl_off_t)stbuf.st_size);
        curl_easy_setopt(curl, CURLOPT_READDATA, (void *)fp);
  curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_callback);
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, &response);
        res = curl_easy_perform(curl);

        if (res != CURLE_OK) {
            fprintf(stderr, "curl_easy_perform() failed: %s\n", curl_easy_strerror(res));
        }
        std::cout << response << std::endl; // response即为返回的json数据
        curl_easy_cleanup(curl);
    }
    curl_global_cleanup();
    fclose(fp);
    return 0;
}
```

结果 获取的结果存储在response字符串中。 **请求参数**

字段	类型	取值	说明
threshold	float	0 ~ 1	置信度阈值

HTTP POST Body直接发送图片二进制。

返回参数 | 字段 | 类型 | 取值 | 说明 | | ---------- | ------ | ---- | ---------------------- | | confidence | float | 0~1 | 分类或检测的置信度 | | label | string | | 分类或检测的类别 | | index | number | | 分类或检测的类别 | | x1, y1 | float | 0~1 | 物体检测，矩形的左上角坐标 (相对长宽的比例值) | | x2, y2 | float | 0~1 | 物体检测，矩形的右下角坐标（相对长宽的比例值） |

关于矩形坐标

x1 * 图片宽度 = 检测框的左上角的横坐标

y1 * 图片高度 = 检测框的左上角的纵坐标

x2 * 图片宽度 = 检测框的右下角的横坐标

y2 * 图片高度 = 检测框的右下角的纵坐标

图像分割 返回结果格式参考API调用文档 代码参考 https://github.com/Baidu-AIP/EasyDL-Segmentation-Demo

声音服务调用说明

Python 使用示例代码如下

```
 import requests

with open('./1.mp3', 'rb') as f:
    audio = f.read()

**params 为GET参数 data 为POST Body**
result = requests.post('http://127.0.0.1:24401/', params={'threshold': 0.1},
                                          data=audio).json()
```

C# 使用示例代码如下

```
 FileStream fs = new FileStream("./audio.mp3", FileMode.Open);
BinaryReader br = new BinaryReader(fs);
byte[] audio = br.ReadBytes((int)fs.Length);
br.Close();
fs.Close();
string url = "http://127.0.0.1:8402?threshold=0.1";
HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(url);
request.Method = "POST";
Stream stream = request.GetRequestStream();
stream.Write(audio, 0, audio.Length);
stream.Close();

WebResponse response = request.GetResponse();
StreamReader sr = new StreamReader(response.GetResponseStream());
Console.WriteLine(sr.ReadToEnd());
sr.Close();
response.Close();
```

C++ 使用示例代码如下，需要安装curl

```
 ###### include <sys/stat.h>
###### include <curl/curl.h>

###### define S_ISREG(m) (((m) & 0170000) == (0100000))
###### define S_ISDIR(m) (((m) & 0170000) == (0040000))

int main(int argc, char *argv[]) {
    const char *post_data_filename = "./audio.mp3";

    FILE *fp = NULL;
    struct stat stbuf = { 0, };

    fp = fopen(post_data_filename, "rb");

    if (!fp) {
        fprintf(stderr, "Error: failed to open file \"%s\"\n", post_data_filename);
        return -1;
    }

    if (fstat(fileno(fp), &stbuf) || !S_ISREG(stbuf.st_mode)) {
        fprintf(stderr, "Error: unknown file size \"%s\"\n", post_data_filename);
        return -1;
    }

    CURL *curl;
    CURLcode res;

    curl_global_init(CURL_GLOBAL_ALL);

    curl = curl_easy_init();
    if (curl != NULL) {
        curl_easy_setopt(curl, CURLOPT_URL, "http://127.0.0.1:24401?threshold=0.1");
        curl_easy_setopt(curl, CURLOPT_POST, 1L);
        curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE_LARGE,(curl_off_t)stbuf.st_size);
        curl_easy_setopt(curl, CURLOPT_READDATA, (void *)fp);

        res = curl_easy_perform(curl);
        if (res != CURLE_OK) {
            fprintf(stderr, "curl_easy_perform() failed: %s\n", curl_easy_strerror(res));
        }
        curl_easy_cleanup(curl);
    }
    curl_global_cleanup();
    fclose(fp);

    return 0;
}
```

请求参数

字段	类型	取值	说明
threshold	float	0 ~ 1	置信度阈值

HTTP POST Body直接发送声音二进制。

返回参数 | 字段 | 类型 | 取值 | 说明 | | --- | --- | --- | --- | | confidence | float | 0~1 | 分类或检测的置信度 | | label | string | | 分类或检测的类别 | | index | number | | 分类或检测的类别 |

集成指南

基于HTTP集成

通过EasyEdge.exe启动服务后，参照上面的调用说明，通过HTTP请求集成到自己的服务中

基于c++ dll集成

集成前提

解压开的SDK包中包含src、lib、dll、include四个目录才支持基于c++ dll集成

集成方法

参考src目录中的CMakeLists.txt进行集成

基于c# dll集成

集成前提

解压开的SDK包中包含src\demo_serving_csharp、dll两个目录才支持基于c# dll集成

集成方法

参考src\demo_serving_csharp目录中的CMakeLists.txt进行集成

FAQ

1. 服务启动失败，怎么处理？

请确保相关依赖都安装正确，版本必须如下： *.NET Framework 4.5* Visual C++ Redistributable Packages for Visual Studio 2013 * Visual C++ Redistributable Packages for Visual Studio 2015

如使用的是CPU加速版，需额外确保Openvino安装正确，版本为2020.3.1LTS版 如使用Windows Server，需确保开启桌面体验

2. 服务调用时返回为空，怎么处理？ 调用输入的图片必须是RGB格式，请确认是否有alpha通道。

3. 多个模型怎么同时使用？ SDK设置运行不同的端口，点击运行即可。

4. JAVA、C#等其他语言怎么调用SDK?

参考 https://ai.baidu.com/forum/topic/show/943765

5. 启动失败，缺失DLL? 打开EasyEdge.log，查看日志错误，根据提示处理 缺失DLL，请使用 https://www.dependencywalker.com/ 查看相应模块依赖DLL缺失哪些，请自行下载安装

6. 启动失败，报错NotDecrypted? Windows下使用，当前用户名不能为中文，否则无法正确加载模型。

7. 启动失败，报错 SerialNum无效

日志显示`failed to get/check device id(xxx)`或者`Device fingerprint mismatch(xxx)` 此类情况一般是设备指纹发生了变更，包括（但不局限于）以下可能的情况：

- mac 地址变化
- 磁盘变更
- bios重刷

以及系统相关信息。

遇到这类情况，请确保硬件无变更，如果想更换序列号，请先删除 `C:\Users\${用户名}\.baidu\easyedge` 目录，再重新激活。

8. 勾选“开机自动启动”后，程序闪退

一般是写注册表失败。

可以确认下HKEY_CURRENT_USER下Software\Microsoft\Windows\CurrentVersion\Run能否写入（如果不能写入，可能被杀毒软件等工具管制）。也可以尝试基于bin目录下的easyedge_serving.exe命令行形式的二进制，自行配置开机自启动。

其他问题 如果无法解决，可到论坛发帖： https://ai.baidu.com/forum/topic/list/199 描述使用遇到的问题，我们将及时回复您的问题。

视觉任务AndroidSDK集成文档

简介

1.1 Android SDK 硬件要求 Android 版本：支持 Android 5.0（API 21） 及以上

硬件：支持 arm64-v8a 和 armeabi-v7a，暂不支持模拟器

通常您下载的SDK只支持固定的某一类芯片。

- **通用ARM**： 支持大部分ARM 架构的手机、平板及开发板。**通常选择这个引擎进行推理**。
- 通用ARM GPU：支持骁龙、麒麟、联发科等带GPU的手机、平板及开发板。
- 高端芯片AI加速模块：
  - **高通骁龙引擎SNPE**： 高通骁龙高端SOC，利用自带的DSP加速。其中 660 之后的型号可能含有 Hexagon DSP模块，具体列表见snpe高通骁龙引擎官网。
  - **华为NPU引擎DDK**：华为麒麟980的arm-v8a的soc。 具体手机机型为mate10，mate10pro，P20，mate20，荣耀v20等。
  - **华为达芬奇NPU引擎DAVINCI**： 华为NPU的后续版本，华为麒麟810，820，990的arm-v8a的soc。具体手机机型为华为mate30，p40，nova6，荣耀v30等。

通用ARM有额外的加速版，但是有一定的精度损失。 因GPU硬件限制，通用ARM GPU物体检测模型输入尺寸较大时会运行失败，可以在训练的时候将输入尺寸设为300*300。 高端芯片AI加速模块， 一般情况下推理速度较快。 运行内存不能过小，一般大于demo的assets目录大小的3倍。

1.2 功能支持 | 引擎 | 图像分类| 物体检测| 图像分割| 文字识别
只支持EasyEdge | 姿态估计 | | :-: | :-: | :-: | :-: | :-: | :-: | | 通用ARM | √ | √ | √ | √ | √ | | 通用ARM GPU | √ | √ |||| | 高通骁龙引擎SNPE | √ | √ |||| | 华为NPU引擎DDK | √ | √ |||| | 华为达芬奇NPU引擎DAVINCI | √ | √ ||||

1.3 Release Notes

时间	版本	说明
2022.09.15	0.10.6	SNPE引擎升级；迭代优化
2022.07.28	0.10.5	迭代优化
2022.06.30	0.10.4	支持Android11；支持EasyEdge语义分割模型；迭代优化
2022.05.18	0.10.3	ARM / ARM-GPU 引擎升级；支持更多加速版模型发布；迭代优化
2022.03.25	0.10.2	ARM / ARM-GPU 引擎升级；支持更多检测模型；迭代优化
2021.12.22	0.10.1	DDK不再支持Kirin 970；迭代优化
2021.10.20	0.10.0	更新鉴权；更新达芬奇NPU、SNPE、通用ARM及ARM-GPU引擎；新增达芬奇NPU对检测模型的支持；支持更多姿态估计模型
2021.07.29	0.9.17	迭代优化
2021.06.29	0.9.16	迭代优化
2021.05.13	0.9.15	更新鉴权，更新通用arm及通用arm gpu引擎
2021.04.02	0.9.14	修正bug
2021.03.09	0.9.13	更新android arm的预处理加速
2020.12.18	0.9.12	通用ARM引擎升级；新增ARM GPU引擎
2020.10.29	0.9.10	迭代优化
2020.9.01	0.9.9	迭代优化
2020.8.11	0.9.8	更新ddk 达芬奇引擎
2020.7.14	0.9.7	支持arm版ocr模型，模型加载优化
2020.6.23	0.9.6	支持arm版fasterrcnn模型
2020.5.14	0.9.5	新增华为新的达芬奇架构npu的部分图像分类模型
2020.4.17	0.9.4	新增arm通用引擎量化模型支持
2020.1.17	0.9.3	新增arm通用引擎图像分割模型支持
2019.12.26	0.9.2	新增华为kirin麒麟芯片的物体检测支持
2019.12.04	0.9.1	使用paddleLite作为arm预测引擎
2019.08.30	0.9.0	支持EasyDL专业版
2019.08.30	0.8.2	支持华为麒麟980的物体检测模型
2019.08.29	0.8.1	修复相机在开发版调用奔溃的问题
2019.06.20	0.8.0	高通手机引擎优化
2019.05.24	0.7.0	升级引擎
2019.05.14	0.6.0	优化demo程序
2019.04.12	0.5.0	新增华为麒麟980支持
2019.03.29	0.4.0	引擎优化，支持sd卡模型读取
2019.02.28	0.3.0	引擎优化，性能与效果提升；
2018.11.30	0.2.0	第一版！

快速开始

2.1 安装软件及硬件准备

扫描模型下载SDK处的网页上的二维码，无需任何依赖，直接体验

如果需要源码方式测试：

打开AndroidStudio， 点击 "Import Project..."。在一台较新的手机上测试。

详细步骤如下：

1. 准备一台较新的手机，如果不是通用arm版本，请参见本文的“硬件要求”，确认是否符合SDK的要求

2. 安装较新版本的AndroidStudio ，下载地址

3. 新建一个HelloWorld项目， Android Studio会自动下载依赖， 在这台较新的手机上测试通过这个helloworld项目。注意不支持模拟器。

4. 解压下载的SDK。

5. 打开AndroidStudio， 点击 "Import Project..."。 即：File->New-> "Import Project...", 选择解压后的目录。

6. 此时点击运行按钮（同第3步），手机上会有新app安装完毕，运行效果和二维码扫描的一样。

7. 手机上UI界面显示后，如果点击UI界面上的“开始使用”按钮，可能会报序列号错误。请参见下文修改

2.2 使用序列号激活

如果使用的是EasyEdge的开源模型，无需序列号，可以跳过本段直接测试。

建议申请包名为"com.baidu.ai.easyaimobile.demo"的序列号用于测试。

本文假设已经获取到序列号，并且这个序列号已经绑定包名。

2.2.1 填写序列号 打开Android Studio的项目，修改MainActivity类的开头SERIAL_NUM字段。 MainActivity 位于 app\src\main\java\com\baidu\ai\edge\demo\MainActivity.java文件内。

```
 // 请替换为您的序列号
private static final String SERIAL_NUM = "XXXX-XXXX-XXXX-XXXX"; //这里填您的序列号
```

2.2.2 修改包名

如果申请的包名为"com.baidu.ai.easyaimobile.demo"，这个是demo的包名，可以不用修改

打开app/build.gradle文件，修改"com.baidu.ai.easyaimobile.demo"为申请的包名

```
  defaultConfig {
    applicationId "com.baidu.ai.easyaimobile.demo" //  修改为比如“com.xxx.xxx"
  }
```

修改序列号和包名后，可以运行测试，效果同扫描二维码的一致

2.2.3 测试精简版

对于通用ARM、高通骁龙引擎SNPE、华为NPU引擎DDK和达芬奇NPU引擎Davinci的常见功能，项目内自带精简版，可以忽略开发板不兼容的摄像头。 此外，由于实时摄像开启，会导致接口的耗时变大，此时也可以使用精简版测试。

目前以下硬件环境有精简版测试：

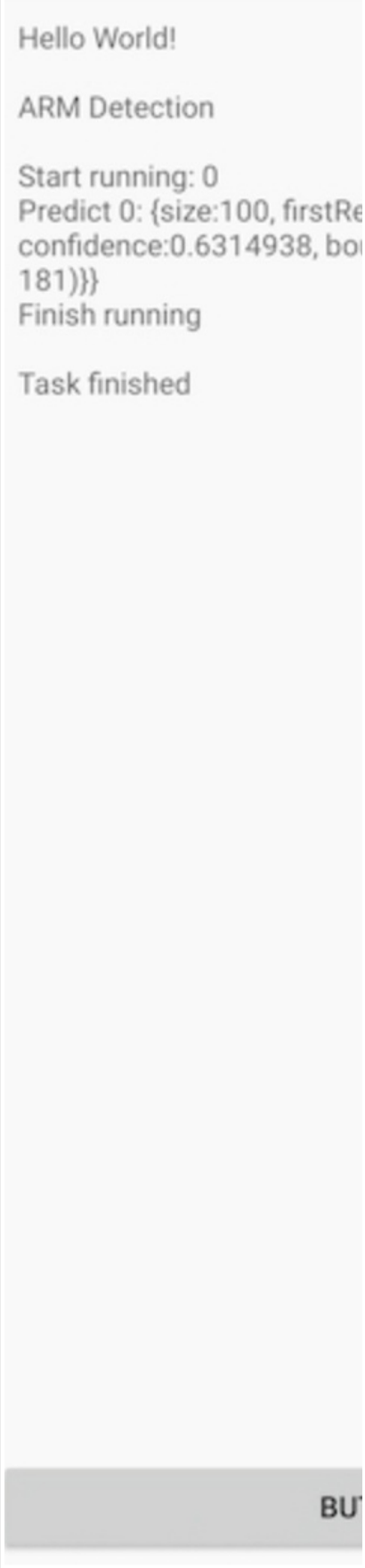
- 通用ARM：图像分类（Classify），物体检测（Detection），文字识别（OCR），图像分割（Segmentation），姿态估计（Pose）
- 高通骁龙引擎SNPE：图像分类（Classify），物体检测（Detection）
- 华为NPU引擎DDK：图像分类（Classify），物体检测（Detection）
- 华为达芬奇NPU引擎Davinci：图像分类（Classify），物体检测（Detection）

具体代码分别在infertest、snpetest、ddktest和davincitest目录下。 修改方法为（以通用ARM为例）：更改app/main/AndroidManifest.xml中的启动Activity。

```
<activity android:name=".infertest.MainActivity">  <!-- 原始的是".MainActivity" -->
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />

    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

开启后会自动选择图像分类（Classify），物体检测（Detection），文字识别（OCR），图像分割（Segmentation）或姿态估计（Pose）测试。

Demo APP 检测模型运行示例	精简版检测模型运行示例
	

使用说明

3.1 代码目录结构

集成时需要“复制到自己的项目里”的目录或者文件：

1. app/libs

2. app/src/main/assets/xxxx-xxxxx 如app/src/main/assets/infer

```
 +app  简单的设置，模拟用户的项目
|---+libs 实际使用时需要复制到自己的项目里
    |----arm64-v8a v8a的so
    |----armeabi-v7a v7a的so
    |----easyedge-sdk.jar jar库文件
|---+src/main
    |---+assets
     |----demo demo项目的配置，实际集成不需要
     |----infer 也可能是其它命名，infer表示通用arm。实际使用时可以复制到自己的项目里
 |---+java/com.baidu.ai.edge/demo
        |---+infertest 通用Arm精简版测试，里面有SDK的集成逻辑
            |--- MainActivity 通用Arm精简版启动Activity，会根据assets目录判断当前的模型类型，并运行同目录的一个Task。使用时需要修改里面的序列号
            |--- TestInferClassifyTask 通用Arm精简版分类
            |--- TestInferDetectionTask 通用Arm精简版检测
            |--- TestInferOcrTask 通用Arm精简版OCR
            |--- TestInferPoseTask 通用Arm精简版姿态
            |--- TestInferSegmentTask 通用Arm精简版分割
        |---+snpetest SNPE精简版测试
            |--- MainActivity SNPE精简版启动Activity，会根据assets目录判断当前的模型类型，并运行同目录的一个Task。使用时需要修改里面的序列号
            |--- TestSnpeClassifyTask SNPE精简版分类
            |--- TestSnpeDetectionTask SNPE精简版检测
        |---+ddktest DDK精简版测试
            |--- MainActivity DDK精简版启动Activity，会根据assets目录判断当前的模型类型，并运行同目录的一个Task。使用时需要修改里面的序列号
            |--- TestDDKClassifyTask DDK精简版分类
            |--- TestDDKDetectionTask DDK精简版检测
        |---+davincitest Davinci精简版测试
         |--- MainActivity Davinci精简版启动Activity，会根据assets目录判断当前的模型类型，并运行同目录的一个Task。使用时需要修改里面的序列号
         |--- TestDavinciClassifyTask Davinci精简版分类
         |--- TestDavinciDetectionTask Davinci精简版检测
        |----CameraActivity 摄像头扫描示例，里面有SDK的集成逻辑
        |----MainActivity 启动Activity，使用时需要修改里面的序列号
|---- build.gradle 这里修改包名
+camera_ui UI模块，集成时可以忽略
```

3.2 调用流程 以通用ARM的检测模型功能为例， 代码可以参考TestInferDetectionTask

1. 准备配置类，如InferConfig，输入：通常为一个assets目录下的文件夹，如infer。
2. 初始化Manager，比如InferManager。输入：第1步的配置类和序列号
3. 推理图片，可以多次调用 3.1 准备图片，作为Bitmap输入 3.2 调用对应的推理方法，比如detect 3.3 解析结果，结果通常是一个List，调用结果类的Get方法，通常能获取想要的结果
4. 直到长时间不再使用我们的SDK，调用Manger的destroy方法释放资源。

3.3 具体接口说明 下文的示例部分以通用ARM的检测模型功能为例 即接口为InferConfig， InferManager，InferManager.detect。 其它引擎和模型调用方法类似。

下文假设已有序列号及对应的包名

3.3.1 . 准备配置类

- INFER：通用ARM，`InferConfig`
- ARM GPU：`ArmGpuConfig`
- SNPE：高通骁龙DSP，`SnpeConfig`
- SNPE GPU：高通骁龙GPU，`SnpeGpuConfig`
- DDK：华为NPU，`DDKConfig`

- DDKDAVINCI：华为达芬奇NPU，DDKDaVinciConfig

```
InferConfig mInferConfig = new InferConfig(getAssets(),
                        "infer");
// assets 目录下的infer，infer表示通用arm
```

```
输入：assets下的配置
输出：具体的配置类
```

3.3.2. 初始化Manager类

- INFER：通用ARM，InferManager
- ARM GPU：通用ARM GPU， InferManager
- SNPE：高通骁龙DSP，SnpeManager
- SNPE GPU：高通骁龙GPU，SnpeManager
- DDK：华为NPU，DDKManager
- DDKDAVINCI：华为达芬奇NPU，DavinciManager

```
String SERIAL_NUM = "XXXX-XXXX-XXXX-XXXX";

// InferManager 为例:
new InferManager(this, config, SERIAL_NUM)); // config为上一步的InferConfig
```

注意要点

1. 同一个时刻只能有**唯一有效的**InferManager。旧的InferManager必须调用destory后，才能新建一个new InferManager() 。
2. InferManager的任何方法，都**不能在UI线程**中调用。
3. new InferManager() 及InferManager成员方法由于线程同步数据可见性问题，都**必须在一个线程中执行**。如使用android自带的**ThreadHandler类**。

```
输入： 1.配置类；2.序列号
输出： Manager类
```

3.3.3. 推理图片

- 接口可以多次调用，但是必须在一个线程里，**不能并发**
- confidence, 置信度[0-1]，小于confidence的结果不返回。 填confidence=0，返回所有结果
- confidence可以不填，默认用模型推荐的。

准备图片， 作为Bitmap输入，

- 输入为Bitmap，其中Bitmap的options为默认。如果强制指定的话，必须使用***Bitmap.Config.ARGB_8888***

调用对应的推理方法及结果解析 见下文的各个模型方法

3.3.4 分类Classify

```
public interface ClassifyInterface {
  List<ClassificationResultModel> classify(Bitmap bitmap, float confidence) throws BaseException;

  // 如InferManger 继承 ClassifyInterface
```

输入 Bitmap 默认格式或者指定Bitmap.Config.ARGB_8888
输出 ClassificationResultModel
异常：一般首次出现。可以打印出异常错误码。

ClassificationResultModel
- label：分类标签，定义在label_list.txt中
- confidence：置信度，0-1
- lableIndex：标签对应的序号

3.3.5 检测Detect

对于EasyDL口罩检测模型请注意输入图片中人脸大小建议保持在*88到96*96像素，可根据场景远近程度缩放图片后传入

```
public interface DetectInterface {
    List<DetectionResultModel> detect(Bitmap bitmap, float confidence) throws BaseException;

    // 如InferManger 继承 DetectInterface
```

输入 Bitmap 默认格式或者指定Bitmap.Config.ARGB_8888
输出 DetectionResultModel List
异常：一般首次出现。可以打印出异常错误码。

DetectionResultModel
- label：标签，定义在label_list.txt中
- confidence：置信度
- bounds：Rect，左上角和右下角坐标

3.3.6 图像分割Segmentation 暂时只支持通用ARM引擎，不支持其它引擎

```
public interface SegmentInterface {
    List<SegmentationResultModel> segment(Bitmap bitmap, float confidence) throws BaseException;
    // 如InferManger 继承 SegmentInterface
```

输入 Bitmap 默认格式或者指定Bitmap.Config.ARGB_8888
输出 SegmentationResultModel
异常：一般首次出现。可以打印出异常错误码。

SegmentationResultModel

- label：标签，定义在label_list.txt中
- confidence：置信度
- lableIndex：标签对应的序号
- box: Rect对象表示的对象框
- mask：byte[]表示的原图大小的0，1掩码，绘制1的像素即可得到当前对象区域

mask 字段说明， 如何绘制掩码也可参考demo工程
    1 0 1
image 1 1 0   =>   mask(byte[]) 101 110 011
    0 1 1

3.3.7 文字识别OCR

暂时只支持通用ARM引擎，不支持其它引擎，暂时只支持EasyEdge的开源OCR模型。

```
public interface OcrInterface {
    List<OcrResultModel> ocr(Bitmap bitmap, float confidence) throws BaseException;
    // 如InferManger 继承 OcrInterface
```

输入 Bitmap 默认格式或者指定Bitmap.Config.ARGB_8888
输出 OcrResultModel List，每个OcrResultModel对应结果里的一个四边形。
异常：一般首次出现。可以打印出异常错误码。

OcrResultModel

- label：识别出的文字
- confidence：置信度
- List<Point>：4个点构成四边形

3.3.8 姿态估计Pose

暂时只支持通用ARM引擎，不支持其它引擎

```
public interface PoseInterface {
  List<PoseResultModel> pose(Bitmap bitmap) throws BaseException;
  // 如InferManger 继承 PoseInterface
```

输入 Bitmap 默认格式或者指定Bitmap.Config.ARGB_8888
输出 PoseResultModel List
异常：一般首次出现。可以打印出异常错误码。

PoseResultModel

- label：标签，定义在label_list.txt中
- confidence：置信度
- Pair<Point, Point>：2个点构成一条线

3.3.9 释放

释放后这个对象不能再使用，如果需要使用可以重新new一个出来。

```
public void destory() throws BaseException
```

3.3.10 整体示例

以通用ARM的图像分类预测流程为例：

```
try {
// step 1: 准备配置类
InferConfig config = new InferConfig(context.getAssets(), "infer");

// step 2: 准备预测 Manager
InferManager manager = new InferManager(context, config, "");

// step 3: 准备待预测的图像，必须为 Bitmap.Config.ARGB_8888 格式，一般为默认格式
Bitmap image = getFromSomeWhere();

// step 4: 预测图像
List<ClassificationResultModel> results = manager.classify(image, 0.3f);

// step 5: 解析结果
for (ClassificationResultModel resultModel : results) {
 Log.i(TAG, "labelIndex=" + resultModel.getLabelIndex()
   + ", labelName=" + resultModel.getLabel()
   + ", confidence=" + resultModel.getConfidence());
}

// step 6: 释放资源。预测完毕请及时释放资源
manager.destroy();
} catch (Exception e) {
 Log.e(TAG, e.getMessage());
}
```

3.3.11 高通骁龙引擎的额外配置

```
    "autocheck_qcom": true, // 如果改成false， sdk跳过检查手机是否是高通的Soc，非高通的Soc会奔溃直接导致app闪退

   "snpe_runtimes_order": [],
// 不填写为自动，按照 {DSP, GPU, GPU_FLOAT16, CPU}次序尝试初始化，也可以手动指定如[2,1,3,0], 具体数字的定义见下段
```

```
public interface SnpeRuntimeInterface {
  int CPU = 0;
  int GPU = 1;
  int DSP = 2;
  int GPU_FLOAT16 = 3;
}

// SnpeManager 中，使用public static ArrayList<Integer> getAvailableRuntimes(Context context) 方法可以获取高通SOC支持的运行方式
```

集成指南

1. 复制库文件libs

2. 添加Manifest权限

3. 复制模型文件

4. 添加调用代码(见上一步具体接口说明)

4.1 复制库文件libs A. 如果项目里没有自己的jar文件和so文件:

```
 复制app/libs 至自己项目的app/libs目录。
参照demo的app/build.gradle 中添加

android {
  ....
   defaultConfig {
      ndk {
         abiFilters  'armeabi-v7a', 'arm64-v8a'
      }
   }
  sourceSets {
      main {
         jniLibs.srcDirs = ['libs']
      }
   }
}
```

B. 如果项目里有自己的jar文件，但没有so文件

```
 easyedge.jar文件同自己的jar文件放一起
arm64-v8a和armeabi-v7a放到app/src/main/jniLibs目录下

参照demo的app/build.gradle 中添加

android {
  ....
   defaultConfig {
      ndk {
         abiFilters  'armeabi-v7a', 'arm64-v8a'
      }
   }
```

C. 如果项目里有自己的jar文件和so文件

```
 easyedge.jar文件同自己的jar文件放一起
arm64-v8a和armeabi-v7a取交集和自己的so放一起，交集的意思是比如自己的项目里有x86目录，必须删除x86。

参照demo的app/build.gradle 中添加

android {
  ....
   defaultConfig {
      ndk {
         abiFilters  'armeabi-v7a', 'arm64-v8a' // abiFilter取交集，即只能少不能多
      }
   }
```

jar文件库如果没有设置成功的，编译的时候可以发现报错。

so库如果没有编译进去的话，也可以通过解压apk文件确认。运行的时候会有类似jni方法找不到的报错。

4.2 Manifest配置

参考app/src/main/AndroidManifest.xml文件，添加：

```
 <uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<!-- Android 11 支持 -->
<uses-permission
   android:name="android.permission.MANAGE_EXTERNAL_STORAGE"
   tools:ignore="ScopedStorage" />

<!-- 高版本 Android 支持 -->
<application
   android:requestLegacyExternalStorage="true"
   android:usesCleartextTraffic="true">
</application>
```

4.3 混淆规则（可选） 请不要混淆SDK里的jar文件。

```
 -keep class com.baidu.ai.edge.core.*.*{ *; }
```

4.4 Android 11支持 除Manifest中必要配置外，请参考`BaseActivity`获取所有文件访问权限，否则可能影响SDK正常使用。

SDK 默认使用 easyedge-sdk.jar，未启用 AndroidX，若您的项目使用 AndroidX，并在集成中提示 android.support 相关错误，请参考 app/build.gradle 使用 etc/easyedge-sdk-androidx.jar 以支持 AndroidX：

```
 // app/build.gradle

dependencies {
   implementation project(':camera_ui')
   implementation files('libs/easyedge-sdk-androidx.jar') // 修改 jar 包依赖
}
```

错误码 | 错误码 | 错误描述 | 详细描述及解决方法 | | ------ | ---------- | ---------- | | 1001 | assets 目录下用户指定的配置文件不存在 | SDK可以使用assets目录下config.json作为配置文件。如果传入的config.json不在assets目录下，则有此报错 | | 1002 | 用户传入的配置文件作为json解析格式不准确，如缺少某些字段 | 正常情况下，demo中的config.json不要修改 | | 19xx | Sdk内部错误 | 请与百度人员联系 | | 2001 | XxxxMANAGER 只允许一个实例 | 如已有XxxxMANAGER对象，请调用destory方法 | | 2002 | XxxxMANAGER 已经调用过destory方法 | 在一个已经调用destory方法的DETECT_MANAGER对象上，不允许再调用任何方法 | | 2003 | 传入的assets下模型文件路径为null | XxxxConfig.getModelFileAssetPath() 返回为null。由setModelFileAssetPath(null）导致 | | 2011 | libedge-xxxx.so 加载失败 | System.loadLibrary("edge-xxxx"); libedge-xxxx.so 没有在apk中。CPU架构仅支持armeabi-v7a arm-v8a | | 2012 | JNI内存错误 | heap的内存不够 | | 2103 | license过期 | license失效或者系统时间有异常 | | 2601 | assets 目录下模型文件打开失败 | 请根据报错信息检查模型文件是否存在 | | 2611 | 检测图片时，传递至引擎的图片二进制与长宽不符合 | 具体见报错信息 | | 27xx | Sdk内部错误 | 请与百度人员联系 | | 28xx | 引擎内部错误 | 请与百度人员联系 | | 29xx | Sdk内部错误 | 请与百度人员联系 | | 3000 | so加载错误 | 请确认所有so文件存在于apk中 | | 3001 | 模型加载错误 | 请确认模型放置于能被加载到的合法路径中，并确保config.json配置正确 | | 3002 | 模型卸载错误 | 请与百度人员联系 | | 3003 | 调用

模型错误 | 在模型未加载正确或者so库未加载正确的情况下调用了分类接口 | | 50xx | 在线模式调用异常 | 请与百度人员联系 |

报错日志收集 通常 Logcat 可以看见日志及崩溃信息，若设备无法获取日志信息，可使用 Demo 中的 xCrash 工具：

```
 // 1. 引入 app/build.gradle 的 xCrash 依赖
android {
 ...
 dependencies {
  implementation 'com.iqiyi.xcrash:xcrash-android-lib:2.4.5' // 可以保存崩溃信息，默认未引入
  ...
 }
}

// 2. 启用日志收集。日志将保存在 /sdcard/<包名>/xCrash
// app/src/main/java/com.baidu.ai.edge/demo/MyApplication.java
protected void attachBaseContext(Context context) {
 // 日志保存位置
 String basePath = Environment.getExternalStorageDirectory().toString() + "/" + context.getPackageName();
 // 启用
    XCrash.InitParameters params = new XCrash.InitParameters();
    params.setAppVersion(BaseManager.VERSION);
    params.setLogDir(basePath + "/xCrash");
    XCrash.init(this, params);
}
```

视觉任务iOSSDK集成文档

简介

本文档描述 EasyEdge/EasyDL iOS 离线预测SDK相关功能；

目前支持EasyEdge的功能包括：

- 图像分类
- 物体检测
- 人脸检测
- 姿态估计
- 百度OCR模型

目前支持EasyDL的功能包括：

- 图像分类
- 物体检测
- 图像分割

系统支持

系统：

- 通用arm版本：iOS 9.0 以上
- A仿生芯片版：iOS 13.0 及以上

硬件：arm64 (Starndard architectures)（暂不支持模拟器）

内存：图像分割模型需要手机内存3GB以上，并尽量减少其他程序内存占用

离线SDK包说明

根据用户的选择，下载的离线SDK，可能包括以下类型：

- EasyEdge
    - 通用ARM版：支持iPhone5s, iOS 9.0 以上所有手机。

    - A仿生芯片版：支持iPhone5s, iOS 11.0 以上手机。充分利用苹果A系列仿生芯片优势，在iPhone 8以上机型中能有显著的速度提升。

- EasyDL 通用版/全功能AI开发平台BML（原EasyDL专业版）
    - 通用ARM版：支持iPhone5s, iOS 9.0 以上所有手机。
    - A仿生芯片版：支持iPhone5s, iOS 11.0 以上手机。充分利用苹果A系列仿生芯片优势，在iPhone 8以上机型中能有显著的速度提升。
    - 自适应芯片版：同时整合了以上两种版本，自动在iOS 11以下中使用`通用ARM版`，在iOS 11以上系统中使用`A仿生芯片版`，自适应系统，但SDK体积相对较大。
- AI市场试用版SDK

SDK大小说明

SDK库的二进制与_TEXT增量约3M。

资源文件大小根据模型不同可能有所差异。

物体检测(高性能)的DemoApp在iPhone 6, iOS 11.4下占用空间实测小于40M。

> 虽然SDK库文件很大（体现为SDK包文件很大，ipa文件很大），但最终应用在用户设备中所占用的大小会缩小很多。 这与multi architechtures、bitcode和AppStore的优化有关。

获取序列号 生成SDK后，点击获取序列号进入控制台获取。EasyEdge控制台、EasyDL控制台、BML控制台。

试用版SDK在SDK的RES文件夹中的SN.txt中包含试用序列号。

更换序列号、更换设备时，首次使用需要联网激活。激活成功之后，有效期内可离线使用。

Release Notes

时间	版本	说明
2022.09.15	0.7.7	支持更多检测模型；迭代优化
2022.07.28	0.7.6	迭代优化
2022.06.29	0.7.5	支持EasyEdge语义分割模型；CoreML引擎升级，新增EasyEdge检测模型支持；迭代优化
2022.05.18	0.7.4	ARM引擎升级；支持EasyDL物体检测超高精度模型；支持更多加速版模型发布；迭代优化
2022.03.25	0.7.3	ARM引擎升级；支持更多检测模型
2021.12.22	0.7.2	支持EasyEdge更多姿态估计模型；迭代优化
2021.10.20	0.7.1	ARM引擎升级
2021.07.29	0.7.0	迭代优化
2021.04.06	0.6.1	ARM引擎升级
2021.03.09	0.6.0	支持EasyEdge人脸检测及姿态估计模型
2020.12.18	0.5.7	ARM引擎升级
2020.09.17	0.5.6	CoreML引擎升级，支持AI市场试用版SDK
2020.08.11	0.5.5	CoreML支持EasyDL专业版模型，支持EasyEdge OCR模型
2020.06.23	0.5.4	ARM引擎升级
2020.04.16	0.5.3	ARM引擎升级；支持压缩加速版模型
2020.03.13	0.5.2	ARM引擎升级；支持图像分割模型
2020.01.16	0.5.1	ARM引擎升级；增加推荐阈值支持
2019.12.04	0.5.0	ARM引擎升级；增加coreml3的支持
2019.10.24	0.4.5	支持EasyDL专业版；ARM引擎升级
2019.08.30	0.4.4	支持EasyDL经典版图像分类高性能、高精度
2019.06.20	0.4.3	引擎优化
2019.04.12	0.4.1	支持EasyDL经典版物体检测高精度、高性能模型
2019.03.29	0.4.0	引擎优化，支持CoreML;
2019.02.28	0.3.0	引擎优化，性能与效果提升；
2018.11.30	0.2.0	第一版！

快速开始 文件结构说明

```
.EasyEdge-iOS-SDK
├──── EasyDLDemo # Demo工程文件
├──── LIB  # 依赖库
├──── RES
|    ├──── easyedge # 模型资源文件夹
|    |    ├──── model
|    |    ├──── params
|    |    ├──── label_list.txt
|    |    ├──── infer_cfg.json
|    |    ├──── conf.json
└──── DOC  # 文档
```

测试Demo

按如下步骤可直接运行 SDK 体验 Demo：

步骤一：用 Xcode 打开 EasyDLDemo/EasyDLDemo.xcodeproj

步骤二：配置开发者自己的签名

步骤三：连接手机运行，不支持模拟器

检测模型运行示例：



SDK使用说明 集成指南 步骤一：依赖库集成 步骤二：`import <EasyDL/EasyDL.h>`，`import <Vision/Vision.h>`

依赖库集成

1. 复制 LIB 目录至项目合适的位置

2. 配置 Build Settings 中 Search paths: 以 SDK 中 LIB 目录路径为例

- Framework Search Paths：`${PROJECT_DIR}/../LIB/lib`
- Header Search Paths：`${PROJECT_DIR}/../LIB/include`
- Library Search Paths：`${PROJECT_DIR}/../LIB/lib`

> 集成过程如出现错误，请参考 Demo 工程对依赖库的引用

使用流程

1. 生成模型，下载SDK 开发者在官网下载的SDK已经自动为开发者配置了模型文件和相关配置，开发者直接运行即可。

2. 使用序列号激活 将前面申请的序列号填入：

```
[EasyDL setSerialNumber:@"!!!Enter Your Serial Number Here!!!"];
```

> 根据序列号类型，序列号与BundleID绑定或与BundleID+设备绑定。 请确保设备时间正确。

3. 初始化模型

```
EasyDLModel *_model = [[EasyDLModel alloc] initModelFromResourceDirectory:@"easyedge" withError:&err];
```

请注意相关资源必须以`folder reference`方式加入Xcode工程。也即默认的`easyedge`文件夹在Xcode文件列表里显示为蓝色。

4. 调用检测接口

```
 UIImage *img = .....;
NSArray *result = [model detectUIImage:img withFilterScore:0 andError:&err];

/**
 * 检测图像
 * @param image 带检测图像
 * @param score 只返回得分高于score的结果(0 ~ 1)
 * @return 成功返回识别结果，NSArray的元素为对应模型的结果类型；失败返回nil，并在err中说明错误原因
 */
- (NSArray *)detectUIImage:(UIImage *)image
          withFilterScore:(CGFloat)score
                 andError:(NSError **)err;
```

返回的数组类型如下，具体可参考 `EasyDLResultData.h` 中的定义： | 模型类型 | 类型 | | --- | ---- | | 图像-图像分类 | EasyDLClassfiData | | 图像-物体检测/人脸检测 | EasyDLObjectDetectionData | | 图像-实例分割/语义分割 | EasyDLObjSegmentationData | | 图像-姿态估计 | EasyDLPoseData | | 图像-文字识别 | EasyDLOcrData |

错误说明

SDK的方法会返回NSError错，直接返回的NSError的错误码定义在EEasyDLErrorCode中。NSError附带message（有时候会附带NSUnderlyingError），开发者可根据code和message进行错误判断和处理。

FAQ

1. 如何多线程并发预测？

SDK内部已经能充分利用多核的计算能力。不建议使用并发来预测。

如果开发者想并发使用，请务必注意`EasyDLModel`所有的方法都不是线程安全的。请初始化多个实例进行并发使用，如

```
- (void)testMultiThread {
    UIImage *img = [UIImage imageNamed:@"1.jpeg"];
    NSError *err;
    EasyDLModel * model1 = [[EasyDLModel alloc] initModelFromResourceDirectory:@"easyedge" withError:&err];
    EasyDLModel * model2 = [[EasyDLModel alloc] initModelFromResourceDirectory:@"easyedge" withError:&err];

    dispatch_queue_t queue1 = dispatch_queue_create("testQueue", DISPATCH_QUEUE_CONCURRENT);
    dispatch_queue_t queue2 = dispatch_queue_create("testQueue2", DISPATCH_QUEUE_CONCURRENT);

    dispatch_async(queue1, ^{
        NSError *detectErr;
        for(int i = 0; i < 1000; ++i) {
            NSArray * res = [model1 detectUIImage:img withFilterScore:0 andError:&detectErr];
            NSLog(@"1: %@", res[0]);
        }
    });

    dispatch_async(queue2, ^{
        NSError *detectErr;
        for(int i = 0; i < 1000; ++i) {
            NSArray * res = [model2 detectUIImage:img withFilterScore:0 andError:&detectErr];
            NSLog(@"2: %@", res[0]);
        }
    });
}
```

2. 编译时出现 Undefined symbols for architecture arm64: ...

- 出现 `cxx11, vtable` 字样：请引入 `libc++.tbd`
- 出现 `cv::Mat` 字样：请引入 `opencv2.framework`
- 出现 `CoreML`, `VNRequest` 字样：请引入`CoreML.framework` 并务必`#import <CoreML/CoreML.h>`

3. 运行时报错 Image not found: xxx ...

请Embed具体报错的库。 **4.编译时报错：Invalid bitcode version** 这个可能是开发者使用的xcode低于12导致，可以升级至12版本。

软硬一体方案部署

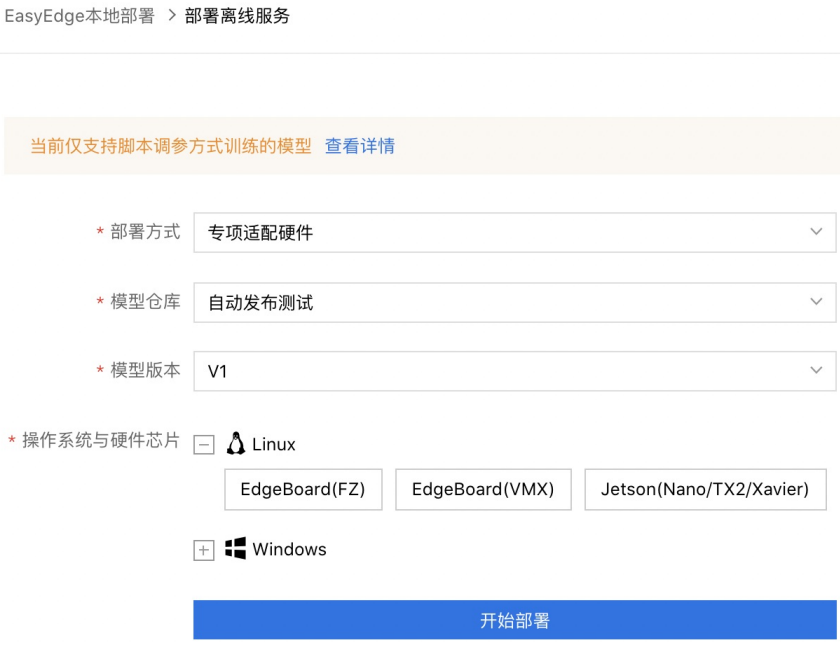
如何获取视觉任务软硬一体产品

为进一步提升前端智能计算的用户体验，BML推出了多款软硬一体方案。将高性能硬件与BML脚本调参任务训练出的图像分类/物体检测模型深度适配，可应用于工业分拣、视频监控等多种设备端离线计算场景，让离线AI落地更轻松。了解不同方案

方案获取流程如下：

Step 1：**在BML脚本调参任务中训练专项适配所选硬件的图像分类/物体检测模型**，迭代模型至效果满足业务要求。查看硬件支持的网络范围

Step 2：发布模型时选择对应硬件



Step 3：在AI市场购买方案获得硬件和用于激活专用SDK的专用序列号，参考文档集成后，即可实现离线AI预测



> 如有其他硬件方案需求，欢迎前往官方论坛交流或提交工单咨询。

视觉任务EdgeBoard(FZ)专用SDK集成文档

简介

本文档介绍 EasyEdge/EasyDL在EdgeBoard®边缘计算盒/Lite计算卡上的专用软件的使用流程。

EdgeBoard系列硬件可直接应用于AI项目研发与部署，具有高性能、易携带、通用性强、开发简单等四大优点。

详细硬件参数请在AI市场浏览。

EdgeBoard产品使用手册：https://ai.baidu.com/ai-doc/HWCE/Yk3b86gvp

软核版本

CPP-SDK版本	对应软核
1.3.2、1.3.4、1.3.5	1.8.1
1.3.0、1.3.1、1.3.2、1.3.4	1.8
0.5.7-1.2.1	1.5
0.5.2+	1.4

SDK升级需配合EdgeBoard硬件软核升级，建议升级软核为SDK对应版本，否则可能出现结果错误或者其他异常。

> 可以通过`dmesg | grep "DRIVER Version"`命令获取EdgeBoard当前的软核版本

Release Notes 注意*：升级完成相应的软核之后需要重启机器生效。

sdk对应的软核说明： **如果客户使用的软核是mobile版本的，需要使用1.4的SDK；如果不是mobile 版本，可以选择1.5+（目前最高版本更新至1.8.1）版本的SDK使用。**

1.5+版本的软核以及sdk更新情况如下表所示：

时间	版本	说明	EdgeBoard非mobile对应的软核以及特性	
2021.12.20	1.3.5	升预测引擎为PaddleLite 1.8.1,推理库支持了Ubuntu18.04文件系统	https://ai.baidu.com/ai-doc/HWCE/Fkuqounlk（含有EB升级Ubuntu18.04系统的步骤）	
2021.10.15	1.3.2、1.3.4、1.3.4	推理库支持了Ubuntu18.04文件系统	https://ai.baidu.com/ai-doc/HWCE/Fkuqounlk	
2021.06.29	1.3.1	视频流解析支持分辨率调整；预测引擎升级；	https://ai.baidu.com/ai-doc/HWCE/lkqgcqt5x	
2021.05.14	1.3.0	新增视频流接入支持；展示已发布模型性能评估报告	https://ai.baidu.com/ai-doc/HWCE/lkqgcqt5x	
2021.05.14	1.2.1	功能无更新	https://ai.baidu.com/ai-doc/HWCE/okqiwkm32	
2020.10.29	0.5.7	预测引擎切换为PaddleLite 1.5		-
2019.12.27	0.4.5	引擎升级，支持zu5/zu3，支持EasyDL 高精度检测模型		-
2019.07.25	0.4.0	EdgeBoard SDK Release!		-

mobile软核以及sdk更新情况如下表所示： | 时间 | 版本 | 说明 |EdgeBoard mobile对应的软核以及特性 | | --- | ---- | ---- |---- |---- |
|2021.05.14|1.2.1|功能无更新|https://ai.baidu.com/ai-doc/HWCE/okqiwkm32|https://ai.baidu.com/ai-doc/HWCE/Lkqiwlziw||

快速开始

开发者从EasyEdge/EasyDL下载的软件部署包中，包含了简单易用的SDK和Demo。只需简单的几个步骤，即可快速部署运行EdgeBoard计算盒。

部署包中包含多版本SDK：

- baidu*easyedge_linux_cpp_aarch64_EdgeBoardFZ1.8**：适用于EdgeBoard 1.5+软核
- baidu*easyedge_linux_cpp_aarch64_EdgeBoardFZ1.4**：适用于EdgeBoard 1.4软核

SDK文件结构

```
baidu_easyedge_linux_cpp_aarch64_EdgeBoardFZ1.5_*
├──── ReadMe.txt
├──── bin
│    ├──── easyedge_image_inference
│    ├──── easyedge_serving
│    └──── easyedge_video_inference
├──── include
│    └──── easyedge
├──── lib
│    ├──── libeasyedge.so -> libeasyedge.so.1
│    ├──── libeasyedge.so.1 -> libeasyedge.so.1.3.1
│    ├──── libeasyedge.so.1.3.1
│    ├──── libeasyedge_static.a
│    ├──── libeasyedge_videoio.so -> libeasyedge_videoio.so.1
│    ├──── libeasyedge_videoio.so.1 -> libeasyedge_videoio.so.1.3.1
│    ├──── libeasyedge_videoio.so.1.3.1
│    ├──── libeasyedge_videoio_static.a
│    ├──── libpaddle_full_api_shared.so -> libpaddle_full_api_shared.so.1.8.0
│    ├──── libpaddle_full_api_shared.so.1.8.0
│    ├──── libverify.so -> libverify.so.1
│    ├──── libverify.so.1 -> libverify.so.1.0.0
│    └──── libverify.so.1.0.0
├──── now_sre.log
├──── src
│    ├──── CMakeLists.txt
│    ├──── cmake
│    ├──── common
│    ├──── demo_image_inference
│    ├──── demo_serving
│    └──── demo_video_inference
└──── thirdparty
   └──── opencv
```

1.1.0+的SDK自带OpenCV，src编译的时候会引用thirdparty/opencv路径下的头文件和库文件。

Demo使用流程

用户在AI市场购买计算盒之后，请参考以下步骤进行集成和试用。

1. 将计算盒连接电源

指示灯亮起，等待约1分钟。

- 参考EdgeBoard使用文档配置网口或串口连接。登录EdgeBoard计算盒。
- 加载驱动（开机加载一次即可）。

```
insmod /home/root/workspace/driver/{zu9|zu5|zu3}/fpgadrv.ko
```

根据购买的版本，选择合适的驱动。若未加载驱动，可能报错：

```
Failed to to fpga device: -1
```

- 设置系统时间（系统时间必须正确）

```
date --set "2019-5-18 20:48:00"
```

2. （可选）启动HTTP服务

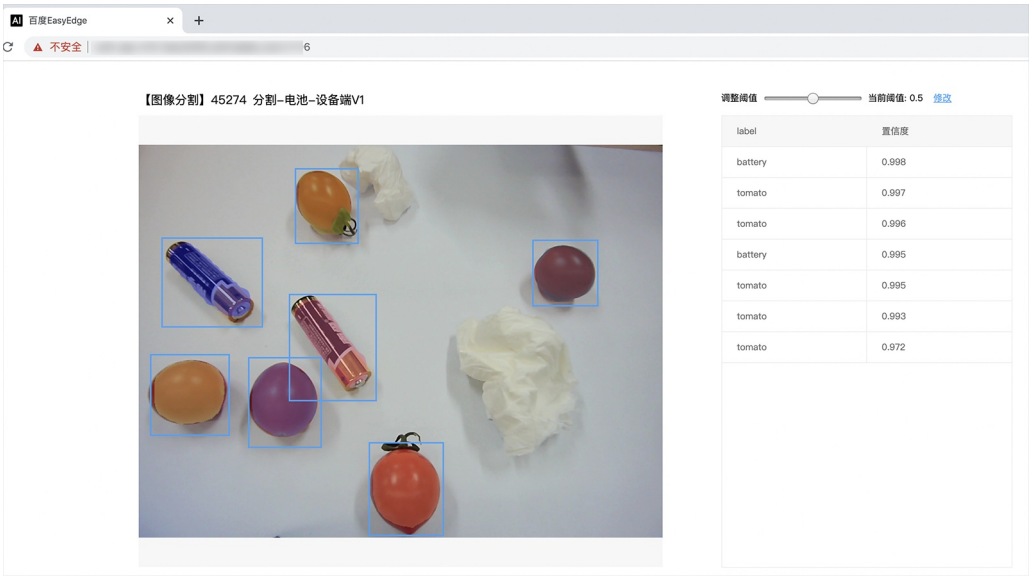
部署包中附带了HTTP服务功能，开发者可以进入SDK根目录，运行`easyedge_serving`程序启动HTTP服务。

```
###### ./easyedge_serving {RES目录} "" {绑定的host，默认0.0.0.0} {绑定的端口，默认24401}
cd ${SDK_ROOT}
export LD_LIBRARY_PATH=./lib
./demo/easyedge_serving ../../../RES ""
```

日志显示

```
2019-07-18 13:27:05,941 INFO [EasyEdge] [http_server.cpp:136] 547974369280 Serving at 0.0.0.0:24401
```

则启动成功。此时可直接在浏览器中输入`http://{EdgeBoard计算盒ip地址}:24401/`，在h5中测试模型效果。



同时，可以调用HTTP接口来访问盒子。具体参考下文接口说明。

> EdgeBoard HTTP Server 目前使用的是单线程处理请求。

3. 编译运行Demo

编译：

```
 cd src
mkdir build && cd build
cmake .. && make
```

运行

```
./easyedge_image_inference {RES资源文件夹路径} {测试图片路径}
```

便可看到识别结果。

使用说明

使用流程

激活成功之后，有效期内可离线使用。

1. 配置`PaddleFluidConfig`
2. 新建`Predictor` `:global_controller()->CreateEdgePredictor(config);`
3. 初始化 `predictor->init()`
4. 传入图片开始识别`predictor->infer(img, ...);`

> 目前EdgeBoard暂不支持并行多模型计算。

接口说明

预测图片

```
    /**
     * @brief 同步预测接口
     * inference synchronous
     * Supported by most chip and engine
     * @param image: must be BGR , HWC format (opencv default)
     * @param result
     * @param threshold
     * @return
     */
    virtual int infer(
            cv::Mat &image, std::vector<EdgeResultData> &result, float threshold = 0.1
    ) = 0;
```

识别结果说明

EdgeResultData中可以获取对应的分类信息、位置信息。

```
struct EdgeResultData {
  int index;  // 分类结果的index
  std::string label;  // 分类结果的label
  float prob;  // 置信度

  // object detection field
  float x1, y1, x2, y2;  // (x1, y1): 左上角，  (x2, y2): 右下角； 均为0~1的长宽比例值。
};
```

关于矩形坐标

x1 * 图片宽度 = 检测框的左上角的横坐标

y1 * 图片高度 = 检测框的左上角的纵坐标

x2 * 图片宽度 = 检测框的右下角的横坐标

y2 * 图片高度 = 检测框的右下角的纵坐标

可以参考demo文件中使用opencv绘制矩形的逻辑。

HTTP 私有服务请求说明

http 请求参数

URL中的get参数：

参数	说明	默认值
threshold	阈值过滤，0~1	0.1

HTTP POST Body即为图片的二进制内容(无需base64, 无需json)

Python

```
import requests

with open('./1.jpg', 'rb') as f:
    img = f.read()
    result = requests.post(
     'http://127.0.0.1:24401/',
     params={'threshold': 0.1},
     data=img).json()
```

Cpp label=C#

```
  FileStream fs = new FileStream("./img.jpg", FileMode.Open);
BinaryReader br = new BinaryReader(fs);
byte[] img = br.ReadBytes((int)fs.Length);
br.Close();
fs.Close();
string url = "http://127.0.0.1:8402?threshold=0.1";
HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(url);
request.Method = "POST";
Stream stream = request.GetRequestStream();
stream.Write(img, 0, img.Length);
stream.Close();

WebResponse response = request.GetResponse();
StreamReader sr = new StreamReader(response.GetResponseStream());
Console.WriteLine(sr.ReadToEnd());
sr.Close();
response.Close();
```

Cpp label=C++ 需要安装curl

```
#include <sys/stat.h>
  #include <curl/curl.h>
  #include <iostream>
  #include <string>
  #define S_ISREG(m) (((m) & 0170000) == (0100000))
  #define S_ISDIR(m) (((m) & 0170000) == (0040000))

  size_t write_callback(void *ptr, size_t size, size_t num, void *data) {
      std::string *str = dynamic_cast<std::string *>((std::string *)data);
      str->append((char *)ptr, size*num);
      return size*num;
  }

  int main(int argc, char *argv[]) {
      const char *post_data_filename = "./img.jpg";
      FILE *fp = NULL;
      std::string response;
      struct stat stbuf = { 0, };
      fp = fopen(post_data_filename, "rb");
      if (!fp) {
          fprintf(stderr, "Error: failed to open file "%s"
  ", post_data_filename);
          return -1;
      }
      if (fstat(fileno(fp), &stbuf) || !S_ISREG(stbuf.st_mode)) {
          fprintf(stderr, "Error: unknown file size "%s"
  ", post_data_filename);
          return -1;
      }
      CURL *curl;
      CURLcode res;
      curl_global_init(CURL_GLOBAL_ALL);
      curl = curl_easy_init();
      if (curl != NULL) {
          curl_easy_setopt(curl, CURLOPT_URL, "http://127.0.0.1:24401?threshold=0.1");
          curl_easy_setopt(curl, CURLOPT_POST, 1L);
          curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE_LARGE,(curl_off_t)stbuf.st_size);
          curl_easy_setopt(curl, CURLOPT_READDATA, (void *)fp);
    curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_callback);
          curl_easy_setopt(curl, CURLOPT_WRITEDATA, &response);
          res = curl_easy_perform(curl);

          if (res != CURLE_OK) {
              fprintf(stderr, "curl_easy_perform() failed: %s
  ", curl_easy_strerror(res));
          }
          std::cout << response << std::endl; // response即为返回的json数据
          curl_easy_cleanup(curl);
      }
      curl_global_cleanup();
      fclose(fp);
      return 0;
  }
```

Java请求示例

http 返回数据

字段	类型说明	其他
error_code	Number	0为成功,非0参考message获得具体错误信息
results	Array	内容为具体的识别结果。其中字段的具体含义请参考预测图像-返回格式一节
cost_ms	Number	预测耗时ms，不含网络交互时间

返回示例

```
{
  "cost_ms": 52,
  "error_code": 0,
  "results": [
    {
      "confidence": 0.94482421875,
      "index": 1,
      "label": "IronMan",
      "x1": 0.059185408055782318,
      "x2": 0.18795496225357056,
      "y1": 0.14762254059314728,
      "y2": 0.52510076761245728
    },
    {
      "confidence": 0.94091796875,
      "index": 1,
      "label": "IronMan",
      "x1": 0.79151463508605957,
      "x2": 0.92310667037963867,
      "y1": 0.045728668570518494,
      "y2": 0.42920106649398804
    }
  ]
}
```


预测视频

SDK 提供了支持摄像头读取、视频文件和网络视频流的解析工具类`VideoDecoding`，此类提供了获取视频帧数据的便利函数。通过`VideoConfig`结构体可以控制视频/摄像头的解析策略、抽帧策略、分辨率调整、结果视频存储等功能。对于抽取到的视频帧可以直接作为SDK infer 接口的参数进行预测。

- 接口

class`VideoDecoding`：

```
    /**
     * @brief 获取输入源的下一帧
     * @param frame_tensor
     * @return
     */
    virtual int next(FrameTensor &frame_tensor) = 0;

    /**
     * @brief 显示当前frame_tensor中的视频帧
     * @param frame_tensor
     * @return
     */
    virtual int display(const FrameTensor &frame_tensor) = 0;

    /**
     * @brief 将当前frame_tensor中的视频帧写为本地视频文件
     * @param frame_tensor
     * @return
     */
    virtual int save(FrameTensor &frame_tensor) = 0;

    /**
     * @brief 获取视频的fps属性
     * @return
     */
    virtual int get_fps() = 0;
     /**
      * @brief 获取视频的width属性
      * @return
      */
    virtual int get_width() = 0;

    /**
     * @brief 获取视频的height属性
     * @return
     */
    virtual int get_height() = 0;
```

struct `VideoConfig`

```
 /**
 * @brief 视频源、抽帧策略、存储策略的设置选项
 */
struct VideoConfig {
    SourceType source_type;           // 输入源类型
    std::string source_value;         // 输入源地址，如视频文件路径、摄像头index、网络流地址
    int skip_frames{0};               // 设置跳帧，每隔skip_frames帧抽取一帧，并把该抽取帧的is_needed置为true
    int retrieve_all{false};          // 是否抽取所有frame以便于作为显示和存储，对于不满足skip_frames策略的frame，把所抽取帧的
is_needed置为false
    int input_fps{0};                 // 在采取抽帧之前设置视频的fps
    Resolution resolution{Resolution::kAuto}; // 采样分辨率，只对camera有效

    bool enable_display{false};
    std::string window_name{"EasyEdge"};
    bool display_all{false};          // 是否显示所有frame，若为false，仅显示根据skip_frames抽取的frame

    bool enable_save{false};
    std::string save_path;            // frame存储为视频文件的路径
    bool save_all{false};             // 是否存储所有frame，若为false，仅存储根据skip_frames抽取的frame

    std::map::string, std::string> conf;
};
```

`source_type`：输入源类型，支持视频文件、摄像头、网络视频流三种，值分别为1、2、3。 `source_value`： 若`source_type`为视频文件，该值为指向视频文件的完整路径；若`source_type`为摄像头，该值为摄像头的index，如对于`/dev/video0`的摄像头，则index为0；若`source_type`为网络视频流，则为该视频流的完整地址。 `skip_frames`：设置跳帧，每隔skip_frames帧抽取一帧，并把该抽取帧的is_needed置为true，标记为is_needed的帧是用来做预测的帧。反之，直接跳过该帧，不经过预测。 `retrieve_all`：若置该项为true，则无论是否设置跳帧，所有的帧都会被

抽取返回，以作为显示或存储用。 input_fps：用于抽帧前设置fps。 resolution：设置摄像头采样的分辨率，其值请参考easyedge_video.h中的定义，注意该分辨率调整仅对输入源为摄像头时有效。 conf：高级选项。部分配置会通过该map来设置。

注意：

1. 如果使用VideoConfig的display功能，需要自行编译带有GTK选项的opencv，默认打包的opencv不包含此项。

2. 使用摄像头抽帧时，如果通过resolution设置了分辨率调整，但是不起作用，请添加如下选项：

```
video_config.conf["backend"] = "2";
```

3.部分设备上的CSI摄像头尚未兼容，如遇到问题，可以通过工单、QQ交流群或微信交流群反馈。

具体接口调用流程，可以参考SDK中的demo_video_inference。

错误说明

SDK所有主动报出的错误，均覆盖在EdgeStatus枚举中。同时SDK会有详细的错误日志，开发者可以打开Debug日志查看额外说明：

```
EdgeLogConfig log_config;
log_config.enable_debug = true;
global_controller()->set_log_config(log_config);
```

FAQ

1. 如何处理一些 undefined reference?

> 如：undefined reference to `curl_easy_setopt@CURL_OPENSSL_3'

可以通过安装libcurl3 libcurl-openssl1.0-dev来解决。 如果开发者想不想使用低版本的openssl（如Ubuntu 18.04），可以link静态库easyedge_static.a，自己指定需要的Library的版本。

示例：修改CMakeList.txt

```
find_package(CURL REQUIRED)
target_link_libraries(easyedge_demo ${OpenCV_LIBS} easyedge_static pthread ${CURL_LIBRARIES} paddle-mobile)
```

2. error while loading shared libraries: libeasyedge.so.0.4.0: cannot open shared object file: No such file or directory

类似错误包括libpaddle-mobile.so找不到。

直接运行SDK自带的二进制可能会有这个问题，设置LD_LIBRARY_PATH为SDK部署包中的lib目录即可。 开发者自行使用CMake编译的二进制可以有效管理.so的依赖。

3. 使用libcurl请求http服务时，速度明显变慢

这是因为libcurl请求continue导致server等待数据的问题，添加空的header即可

```
headers = curl_slist_append(headers, "Expect:");
```

4. 预测过程中报内存不足“Killed”

此问题仅出现在ZU5，因为FZ5A带vcu，给他预留的内存过大导致，如果用不到VCU可以把这部分改小。修改/run/media/mmcblk1p1/uEnv.txt：

```
ethaddr=00:0a:35:00:00:09
uenvcmd=fatload mmc 1 0x3000000 image.ub && bootm 0x3000000

bootargs=earlycon console=ttyPS0,115200 clk_ignore_unused cpuidle.off=1 root=/dev/mmcblk1p2 rw rootwait cma=128M
```

注意中间空行要保留。

5. 预测结果异常

如果购买的计算盒较早，驱动文件较旧，而SDK比较新（或SDK比较旧，但是计算盒较新），可能出现结果异常，如结果均为空或者nan。 请参

考“软核版本”小节更新软核和驱动版本。

6. 编译过程报错file format not recognized

```
libeasyedge.so: file format not recognized; treating as linker script
```

下载的SDK zip包需要放到板子内部后，再解压、编译。

7. 提示 driver_version(1.4.0) not match paddle_lite_version(1.5.1) 需更新驱动，否则可能导致结果异常。参考“软核版本”小节。

视觉任务EdgeBoard(VMX)专用SDK集成文档

简介

本文档旨在介绍 EasyDL在EdgeBoard USB加速卡VMX（以下简称VMX加速卡或加速卡）上的专用软件的使用流程。 EdgeBoard系列硬件适用于项目研发与部署，具有高性能、易携带、通用性强、开发简单等四大优点。 您可在AI市场了解EdgeBoard相关系列产品，同时可以在软硬一体方案了解性能数据。

注意：本型号主要面向产品集成和企业项目，未同时售卖散热片和外壳，部分情况下芯片温度较高，开发过程中，请勿用手触摸，谨防烫伤

硬件介绍

VMX加速卡，采用Intel® Movidus™ 视觉 MyriadX处理器芯片，通过 USB3.0 通讯type-c接口方式，配合外围电路即可将该模组嵌入到第三方智能化产品中，采用标准 USB通讯协议，对接简单，开发速度快，具有强大的深度学习计算功能。可通过OpenVINO™和OpenCV软件库工具链移植算法，兼容百度PaddlePaddle支持Paddle2onnx和PaddleHub并集成EasyDL，使产品应用范围广，性能更稳定，增强用户体验。

VMX加速卡适用于深度学习加速，能够解决复杂的人工智能软硬件设计挑战，它可以集成基于视觉的加速器和推理引擎来实现深度边缘学习的解决方案。（3D/2D人脸识别、人头检测、人脸属性分析（性别、年龄）、人脸特征比对、手势及姿态识别、物体检测及分类、算法移植等功能。）

硬件配置与说明

核心板模块: Intel® Movidus™ MyriadX，内置内存LP-DDR4 4GBit。

- 硬件指标

CPU

o Intel® Movidius Myriad X MA2485 Vision Processing Unit
o Total performance of over 4 trillion operations per second (TOPS) o Over 1 TOPS performance on neural network inference w/ NCE accelerator o 16 Programmable 128-bit VLIW Vector Processors o 16 Configurable MIPI Lanes w/ enhanced Vision Accelerators o 2.5 MB of Homogenous On-Chip Memory w/ 4Gbit LPDDR4

Size

o 38mm x 38mm

Interface o USB TYPE C（USB3.0）辅助接口精简设计

Boot o USB 启动模式 - 内置 switch 缺省模式设置

Power o 平均功耗0.5W~2.2W

Security o 支持 eFuse 加密

运行说明

VMX加速卡包含独立的AI运算芯片，采用 USB Type-C通讯方式，通讯协议简单可靠，可连接不同芯片架构主机，包括 X86、单片机、ARM SOC等。加速卡运行需要通过TypeC接口连接宿主机执行，宿主机目前支持的软硬件环境包括：

- Linux: x86-64, aarch64, armv7hf
- Windows: x86-64, Windows 10

> 使用过程中，请尽量避免直接触碰板卡元器件；或者使用防静电锡纸包裹板卡。

快速开始 Linux

开发者从EasyDL训练模型之后，下载的软件部署包中，包含了简单易用的SDK和Demo。只需简单的几个步骤，即可快速部署运行。

Release Notes

Python SDK

时间	版本	说明
2020.12.18	1.2.0	性能优化；接口优化升级；推理引擎升级
2020.09.17	1.1.19	支持更多模型与平台

Python SDK适用于Linux x86-64和Windows平台。

> 2020-12-18: 【接口升级】 Python SDK序列号配置接口从1.2.0版本开始已升级为新接口，以前的方式被置为deprecated，并将在未来的版本中移除。请尽快考虑升级为新的接口方式，具体使用方式可以参考下文介绍以及demo工程示例，谢谢。

C++ SDK

时间	版本	说明
2021.06.29	1.3.1	视频流解析支持分辨率调整
2021.05.14	1.3.0	新增视频流接入支持；展示已发布模型性能评估报告
2020.12.18	1.0.0	性能优化；接口优化升级；推理引擎升级
2020.09.17	0.5.6	新增C++ SDK，支持Linux aarch64和Linux armv7hf（树莓派）架构的硬件接入VMX预测

C++ SDK适用于Linux x86-64、Linux aarch64和Linux armv7hf平台。

> 2020-12-18: 【接口升级】 参数配置接口从1.0.0版本开始已升级为新接口，以前的方式被置为deprecated，并将在未来的版本中移除。请尽快考虑升级为新的接口方式，具体使用方式可以参考下文介绍以及demo工程示例，谢谢。

将加速卡连接宿主机 请使用质量合规的usb线连接。连接之后，检查设备是否被操作系统识别： Linux 通过`lsusb -v` 命令检查是否有 `Myriad`设备：

```
> sudo lsusb -v | grep -C 5 Myriad
bMaxPacketSize0        64
idVendor           0x03e7
idProduct          0x2485
bcdDevice            0.01
iManufacturer           1 Movidius Ltd.
iProduct                2 Movidius MyriadX
iSerial                 3 03e72485
```

Windows 可以在设备管理器中查询。

> 如果使用 VirtualBox 之类的虚拟机，请在虚拟机加入 03e7:24 和 03e7:f63b 两个 usb 设备。

获取并安装依赖

1) 安装依赖

宿主机与sdk为以下情况： 1☐ **Windows x86-64** 2☐ **Linux x86-64**且使用**Python SDK**时必须： 请参考 OpenVINO toolkit 文档安装 2020.3.1LTS 版本, 安装时可忽略Configure the Model Optimizer及后续部分。

> **安装完毕，运行之前，请按照OpenVino的文档 设置环境变量**

```
source /opt/intel/openvino/bin/setupvars.sh
```

2) 从EasyDL 控制台获取SDK 在任意位置解压缩。

获取序列号 从AI市场订单详情或者EasyDL控制台获取序列号。

更换序列号、更换设备时，首次使用需要联网激活。激活成功之后，有效期内可离线使用。

> 请确保激活设备时使用的 操作系统账号与后续使用时运行的账号一致，否则会造成验证失败

Python SDK

1. 安装wheel包

```
pip3 install -U BaiduAI_EasyEdge_SDK-{版本号}-cp37-cp37m-linux_x86_64.whl
```

> 注意，请根据python的版本选择对应的whl文件，其中,1.2.0是SDK版本号，cp37表示是python3.7版本

--

> 注意，pip安装时请添加-U参数

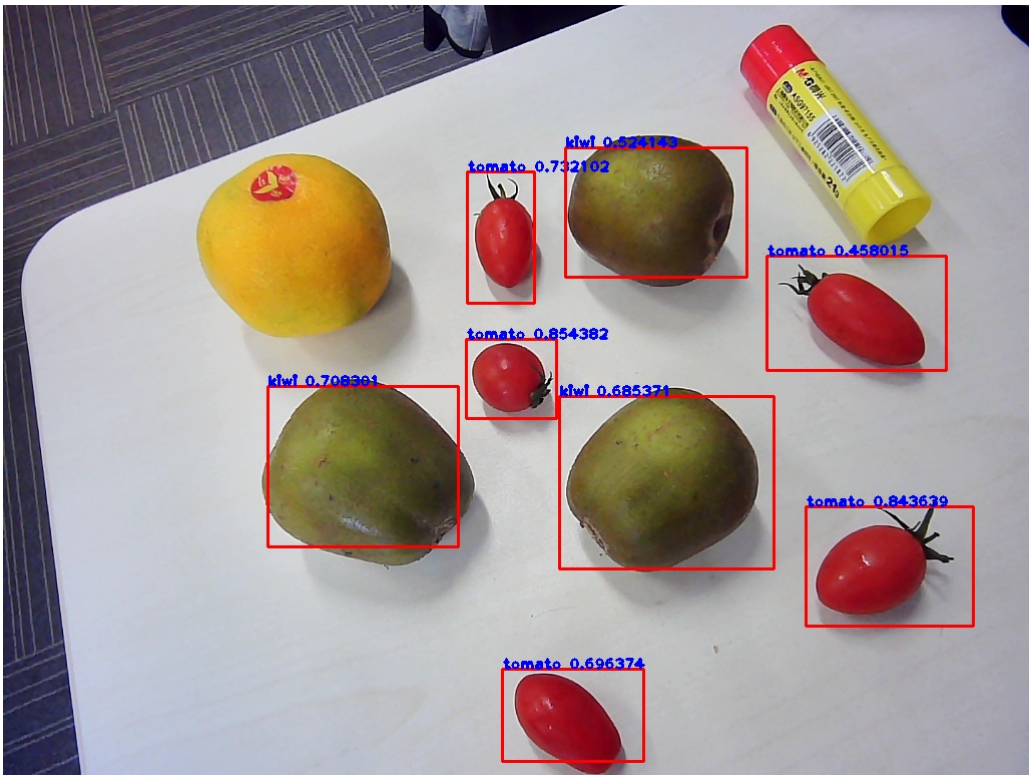
2. 将步骤2中获得的序列号 填入demo.py

```
pred = edge.Program()
pred.set_auth_license_key("这里填写序列号")
```

3. 测试demo.py

```
python3 demo.py {模型资源文件夹RES路径} {待识别的图片路径}
```

生成的样例结果图片如下：



使用流程

```
import BaiduAI.EasyEdge as edge

pred = edge.Program()
pred.set_auth_license_key("这里填写序列号")
pred.init(model_dir={RES文件夹路径}, device=edge.Device.MOVIDIUS, engine=edge.Engine.OPENVINO)
pred.infer_image({numpy.ndarray的图片})
pred.close()
```

> 接口的详细说明请主要参考 SDK 中的接口注释

接口说明

`Program`

- 初始化

```
def init(self,
         model_dir,
         device=Device.CPU,
         engine=Engine.NCSDK,
         config_file='conf.json',
         preprocess_file='preprocess_args.json',
         model_file='model',
         params_file='params',
         graph_file='graph.ncsmodel',
         label_file='label_list.txt',
         device_id=0,
         **kwargs
         ):
    """
    Args:
        model_dir: str
        device: BaiduAI.EasyEdge.Device
        engine: BaiduAI.EasyEdge.Engine
        preprocess_file: str
        model_file: str
        params_file: str
        graph_file: str ncs的模型文件 或 PaddleV2的模型文件
        label_file: str
        device_id: int 设备ID
        thread_num: int CPU的线程数

    Raises:
        RuntimeError, IOError
    Returns:
        bool: True if success

    """
```

- 预测单张图像

```
    def infer_image(self, img, threshold=None,
                    channel_order='HWC',
                    color_format='BGR',
                    data_type='numpy'
                    ):
        """

        Args:

            img: np.ndarray or bytes
            channel_order(string):
                channel order: HWC or CHW
            color_format(string):
                color format order: RGB or BGR
            threshold(float):
                only return result with confidence larger than threshold
            data_type(string): 仅在图像分割时有意义。 'numpy' or 'string'
                'numpy': 返回已解析的mask
                'string': 返回未解析的mask游程编码
        Returns:
            list

        """
```

- 返回格式: [dict1, dict2, ...]

字段	类型	取值	说明
confidence	float	0~1	分类或检测的置信度
label	string		分类或检测的类别
index	number		分类或检测的类别
x1, y1	float	0~1	物体检测，矩形的左上角坐标 (相对长宽的比例值)
x2, y2	float	0~1	物体检测，矩形的右下角坐标（相对长宽的比例值）
mask	string/numpy.ndarray	图像分割的mask	

关于矩形坐标

x1 * 图片宽度 = 检测框的左上角的横坐标

y1 * 图片高度 = 检测框的左上角的纵坐标

x2 * 图片宽度 = 检测框的右下角的横坐标

y2 * 图片高度 = 检测框的右下角的纵坐标

可以参考 demo 文件中使用 opencv 绘制矩形的逻辑。

结果示例

- i) 图像分类

```
{
  "index": 736,
  "label": "table",
  "confidence": 0.9
}
```

- ii) 物体检测

```
 {
   "y2": 0.91211,
   "label": "cat",
   "confidence": 1.0,
   "x2": 0.91504,
   "index": 8,
   "y1": 0.12671,
   "x1": 0.21289
}
```

- iii) 图像分割

```
 {
  "name": "cat",
   "score": 1.0,
   "location": {
    "left": ...,
    "top": ...,
    "width": ...,
    "height": ...,
   },
   "mask": ...
}
```

mask字段中，data_type为`numpy`时，返回图像掩码的二维数组

```
 {
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
 {0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
 {0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
 {0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
}
其中1代表为目标区域，0代表非目标区域
```

data_type为`string`时，mask的游程编码，解析方式可参考 demo

C++ SDK

使用说明

模型资源文件默认已经打包在开发者下载的SDK包中。 请先将SDK包整体拷贝到具体运行的宿主机设备中，再解压缩编译；

在编译或运行demo程序前执行以下命令:

`source ${cpp_kit位置路径}/thirdparty/openvino/bin/setupvars.sh`

如果openvino预测引擎找不到设备需要执行以下命令：

```
 sudo cp ${cpp_kit位置路径}/thirdparty/openvino/deployment_tools/inference_engine/external/97-myriad-usbboot.rules
/etc/udev/rules.d/
sudo udevadm control --reload-rules
sudo udevadm trigger
sudo ldconfig ```
```

使用流程

```
    // step 1: 配置运行参数
    EdgePredictorConfig config;
    config.set_config(easyedge::params::PREDICTOR_KEY_SERIAL_NUM, "this-is-serial-num"); // 设置序列号
    config.model_dir = {模型文件目录};

    // step 2: 创建并初始化Predictor
    auto predictor = global_controller()->CreateEdgePredictor(config);
    if (predictor->init() != EDGE_OK) {
        exit(-1);
    }

    // step 3-1: 预测图像
    auto img = cv::imread({图片路径});
    std::vector<EdgeResultData> results;
    predictor->infer(img, results);

// step 3-2: 预测视频
std::vector<EdgeResultData> results;
FrameTensor frame_tensor;
    VideoConfig video_config;
    video_config.source_type = static_cast<SourceType>(video_type);  // source_type 定义参考头文件 easyedge_video.h
    video_config.source_value = video_src;
    /*
    ... more video_configs, 根据需要配置video_config的各选项
    */
    auto video_decoding = CreateVideoDecoding(video_config);
    while (video_decoding->next(frame_tensor) == EDGE_OK) {
        results.clear();
        if (frame_tensor.is_needed) {
            predictor->infer(frame_tensor.frame, results);
            render(frame_tensor.frame, results, predictor->model_info().kind);
        }
        //video_decoding->display(frame_tensor); // 显示当前frame，需在video_config中开启配置
        //video_decoding->save(frame_tensor); // 存储当前frame到视频，需在video_config中开启配置
    }
```

运行参数配置 运行参数的配置通过结构体`EdgePredictorConfig`完成，其定义如下所示：

```
 struct EdgePredictorConfig {
    /**
     * @brief 模型资源文件夹路径
     */
    std::string model_dir;

    std::map<std::string, std::string> conf;

    EdgePredictorConfig();

    template<typename T>
    T get_config(const std::string &key, const T &default_value);

    template<typename T = std::string>
    T get_config(const std::string &key);

    template<typename T>
    const T *get_config(const std::string &key, const T *default_value);

    template<typename T>
    void set_config(const std::string &key, const T &value);

    template<typename T>
    void set_config(const std::string &key, const T *value);

    static EdgePredictorConfig default_config();
};
```

运行参数选项的配置以`key`、`value`的方式存储在类型为`std::map`的`conf`中，并且键值对的设置和获取可以通

过EdgePredictorConfig的set_config和get_config函数完成。同时部分参数也支持以环境变量的方式设置键值对。 EdgePredictorConfig的具体使用方法可以参考开发工具包中的demo工程。

具体支持的运行参数可以参考开发工具包中的头文件。

初始化

- 接口

```
 auto predictor = global_controller()->CreateEdgePredictor(config);
predictor->init();
```

若返回非0，请查看输出日志排查错误原因。

预测图像

- 接口

```
 /**
 * @brief
 * 通用接口
 * @param image: must be BGR , HWC format (opencv default)
 * @param result
 * @return
 */
virtual int infer(
        cv::Mat& image, std::vector<EdgeResultData>& result
) = 0;
```

图片的格式务必为opencv默认的BGR, HWC格式。

- 返回格式

EdgeResultData中可以获取对应的分类信息、位置信息。

```
 struct EdgeResultData {
   int index;  // 分类结果的index
   std::string label;  // 分类结果的label
   float prob;  // 置信度

   // 物体检测活图像分割时才有
   float x1, y1, x2, y2;  // (x1, y1): 左上角，  (x2, y2): 右下角； 均为0~1的长宽比例值。

   // 图像分割时才有
   cv::Mat mask;  // 0, 1 的mask
   std::string mask_rle;  // Run Length Encoding，游程编码的mask
};
```

关于矩形坐标

x1 * 图片宽度 = 检测框的左上角的横坐标

y1 * 图片高度 = 检测框的左上角的纵坐标

x2 * 图片宽度 = 检测框的右下角的横坐标

y2 * 图片高度 = 检测框的右下角的纵坐标

关于图像分割mask

```
 cv::Mat mask为图像掩码的二维数组
{
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
 {0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
 {0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
 {0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
}
其中1代表为目标区域，0代表非目标区域
```

关于图像分割mask_rle

该字段返回了mask的游程编码，解析方式可参考 http demo

以上字段可以参考demo文件中使用opencv绘制的逻辑进行解析

预测视频

SDK 提供了支持摄像头读取、视频文件和网络视频流的解析工具类`VideoDecoding`，此类提供了获取视频帧数据的便利函数。通过`VideoConfig`结构体可以控制视频/摄像头的解析策略、抽帧策略、分辨率调整、结果视频存储等功能。对于抽取到的视频帧可以直接作为SDK infer 接口的参数进行预测。

- 接口

class`VideoDecoding`：

```
    /**
    * @brief 获取输入源的下一帧
    * @param frame_tensor
    * @return
    */
   virtual int next(FrameTensor &frame_tensor) = 0;

   /**
    * @brief 显示当前frame_tensor中的视频帧
    * @param frame_tensor
    * @return
    */
   virtual int display(const FrameTensor &frame_tensor) = 0;

   /**
    * @brief 将当前frame_tensor中的视频帧写为本地视频文件
    * @param frame_tensor
    * @return
    */
   virtual int save(FrameTensor &frame_tensor) = 0;

   /**
    * @brief 获取视频的fps属性
    * @return
    */
   virtual int get_fps() = 0;
    /**
     * @brief 获取视频的width属性
     * @return
     */
   virtual int get_width() = 0;

   /**
    * @brief 获取视频的height属性
    * @return
    */
   virtual int get_height() = 0;
```

struct `VideoConfig`

```
 /**
 * @brief 视频源、抽帧策略、存储策略的设置选项
 */
struct VideoConfig {
    SourceType source_type;          // 输入源类型
    std::string source_value;        // 输入源地址，如视频文件路径、摄像头index、网络流地址
    int skip_frames{0};              // 设置跳帧，每隔skip_frames帧抽取一帧，并把该抽取帧的is_needed置为true
    int retrieve_all{false};         // 是否抽取所有frame以便于作为显示和存储，对于不满足skip_frames策略的frame，把所抽取帧的
is_needed置为false
    int input_fps{0};                // 在采取抽帧之前设置视频的fps
    Resolution resolution{Resolution::kAuto}; // 采样分辨率，只对camera有效

    bool enable_display{false};
    std::string window_name{"EasyEdge"};
    bool display_all{false};         // 是否显示所有frame，若为false，仅显示根据skip_frames抽取的frame

    bool enable_save{false};
    std::string save_path;           // frame存储为视频文件的路径
    bool save_all{false};            // 是否存储所有frame，若为false，仅存储根据skip_frames抽取的frame

    std::map::string, std::string> conf;
};
```

source_type：输入源类型，支持视频文件、摄像头、网络视频流三种，值分别为1、2、3。 source_value： 若source_type为视频文件，该值为指向视频文件的完整路径；若source_type为摄像头，该值为摄像头的index，如对于/dev/video0的摄像头，则index为0；若source_type为网络视频流，则为该视频流的完整地址。 skip_frames：设置跳帧，每隔skip_frames帧抽取一帧，并把该抽取帧的is_needed置为true，标记为is_needed的帧是用来做预测的帧。反之，直接跳过该帧，不经过预测。 retrieve_all：若置该项为true，则无论是否设置跳帧，所有的帧都会被抽取返回，以作为显示或存储用。 input_fps：用于抽帧前设置fps。 resolution：设置摄像头采样的分辨率，其值请参考easyedge_video.h中的定义，注意该分辨率调整仅对输入源为摄像头时有效。 conf：高级选项。部分配置会通过该map来设置。

注意：

1. 如果使用VideoConfig的display功能，需要自行编译带有GTK选项的opencv，默认打包的opencv不包含此项。

2. 使用摄像头抽帧时，如果通过resolution设置了分辨率调整，但是不起作用，请添加如下选项：

```
video_config.conf["backend"] = "2";
```

3.部分设备上的CSI摄像头尚未兼容，如遇到问题，可以通过工单、QQ交流群或微信交流群反馈。

具体接口调用流程，可以参考SDK中的demo_video_inference。

设置序列号

请在网页控制台中申请序列号，并在init初始化前设置。 LinuxSDK 首次使用需联网授权。

```
EdgePredictorConfig config;
config.set_config(easyedge::params::PREDICTOR_KEY_SERIAL_NUM, "this-is-serial-num");
```

日志配置

设置 EdgeLogConfig 的相关参数。具体含义参考文件中的注释说明。

```
EdgeLogConfig log_config;
log_config.enable_debug = true;
global_controller()->set_log_config(log_config);
```

http服务

- 1. 开启http服务 http服务的启动参考demo_serving.cpp文件。

```
/**
  * @brief 开启一个简单的demo http服务。
  * 该方法会block直到收到sigint/sigterm。
  * http服务里，图片的解码运行在cpu之上，可能会降低推理速度。
  * @tparam ConfigT
  * @param config
  * @param host
  * @param port
  * @param service_id service_id  user parameter, uri '/get/service_id' will respond this value with 'text/plain'
  * @param instance_num 实例数量，根据内存/显存/时延要求调整
  * @return
  */
 template<typename ConfigT>
 int start_http_server(
        const ConfigT &config,
        const std::string &host,
        int port,
        const std::string &service_id,
        int instance_num = 1);
```

- 2. 请求http服务

> 开发者可以打开浏览器，`http://{设备ip}:24401`，选择图片来进行测试。

URL中的get参数：

参数	说明	默认值
threshold	阈值过滤， 0~1	如不提供，则会使用模型的推荐阈值

HTTP POST Body即为图片的二进制内容(无需base64, 无需json)

Python请求示例

```
 import requests

with open('./1.jpg', 'rb') as f:
   img = f.read()
   result = requests.post(
    'http://127.0.0.1:24401/',
    params={'threshold': 0.1},
    data=img).json()
```

Java请求示例

- http 返回数据

字段	类型说明	其他
error_code	Number	0为成功,非0参考message获得具体错误信息
results	Array	内容为具体的识别结果。其中字段的具体含义请参考`预测图像-返回格式`一节
cost_ms	Number	预测耗时ms，不含网络交互时间

返回示例

```
{
  "cost_ms": 52,
  "error_code": 0,
  "results": [
    {
      "confidence": 0.94482421875,
      "index": 1,
      "label": "IronMan",
      "x1": 0.059185408055782318,
      "x2": 0.18795496225357056,
      "y1": 0.14762254059314728,
      "y2": 0.52510076761245728
    },
    {
      "confidence": 0.94091796875,
      "index": 1,
      "label": "IronMan",
      "x1": 0.79151463508605957,
      "x2": 0.92310667037963867,
      "y1": 0.045728668570518494,
      "y2": 0.42920106649398804
    }
  ]
}
```

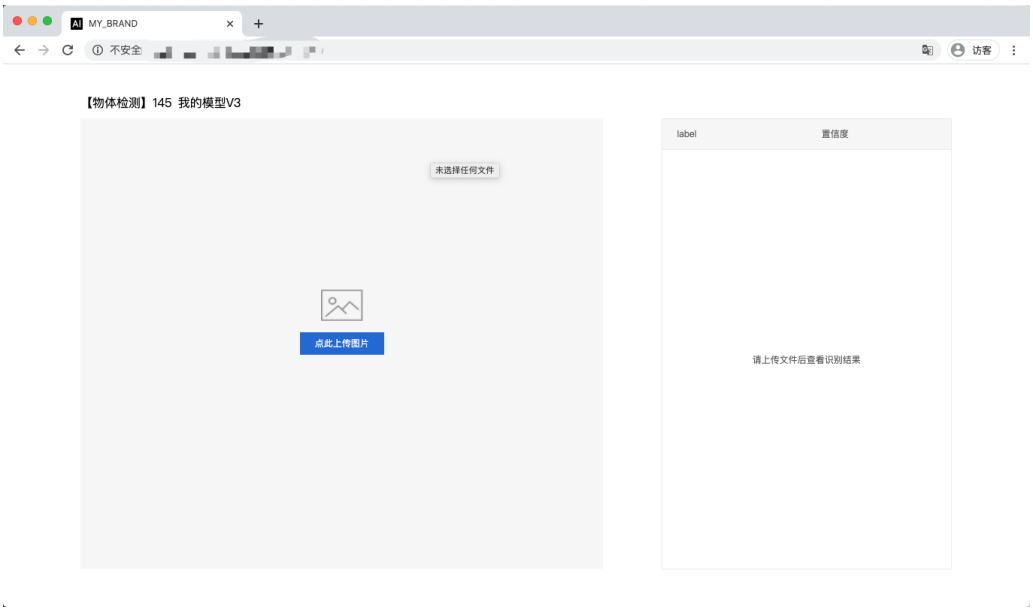

其他配置

- 1. 日志名称、HTTP 网页标题设置

通过global_controller的set_config方法设置：

```
global_controller()->set_config(easyedge::params::KEY_LOG_BRAND, "MY_BRAND");
```

效果如下：



Linux FAQ

1. EasyDL 离线 SDK 与云服务效果不一致，如何处理？ 我们会逐渐消除这部分差异，如果开发者发现差异较大，可通过工单、论坛联系我们协助处理。

2. 硬件出现问题或者出现故障怎么办？ 软件使用有问题怎么处理？

- 如果持续在静电较多的环境中使用，建议使用防静电锡纸包裹板卡
- 如果硬件无法启动等故障，您可以通过商品页联系供应商处理；其它硬件问题，您可以邮件 edgeboard-vmx.com ，我们将在0-2日内处理您的问题。为加快处理进度，您在邮件中，尽量描述清楚问题或者需求细节，避免来回沟通。
- 软件使用问题，请尽量通过工单、论坛联系我们协助处理。

3. 运行时报错：NC_ERROR

```
Can not init Myriad device: NC_ERROR
```

一般是硬件没有插上，请确保lsusb能够找到该硬件。或者等待几秒后再试。

快速开始 Windows

1. 安装依赖

将操作系统升级到Windows 10

安装.NET Framework4.5

```
https://www.microsoft.com/zh-CN/download/details.aspx?id=42642
```

Visual C++ Redistributable Packages for Visual Studio 2013

```
https://www.microsoft.com/zh-cn/download/details.aspx?id=40784
```

Visual C++ Redistributable Packages for Visual Studio 2015

```
https://www.microsoft.com/zh-cn/download/details.aspx?id=48145
```

注意事项

1. 安装目录不能包含中文
2. Windows Server 请自行开启，选择“我的电脑”——“属性”——“管理”——”添加角色和功能“——勾选”桌面体验“，点击安装，安装之后重启即可。

2. 运行离线SDK

解压下载好的SDK，打开EasyEdge.exe，输入Serial Num



点击"启动服务"，等待数秒即可启动成功，本地服务默认运行在

```
http://127.0.0.1:24401/
```

其他任何语言只需通过HTTP调用即可。

接口调用说明

Python 使用示例代码如下

```
 import requests

with open('./1.jpg', 'rb') as f:
    img = f.read()

**params 为GET参数 data 为POST Body**
result = requests.post('http://127.0.0.1:24401/', params={'threshold': 0.1},
                                        data=img).json()
```

C## 使用示例代码如下

```
 FileStream fs = new FileStream("./img.jpg", FileMode.Open);
BinaryReader br = new BinaryReader(fs);
byte[] img = br.ReadBytes((int)fs.Length);
br.Close();
fs.Close();
string url = "http://127.0.0.1:8402?threshold=0.1";
HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(url);
request.Method = "POST";
Stream stream = request.GetRequestStream();
stream.Write(img, 0, img.Length);
stream.Close();

WebResponse response = request.GetResponse();
StreamReader sr = new StreamReader(response.GetResponseStream());
Console.WriteLine(sr.ReadToEnd());
sr.Close();
response.Close();
```

请求参数

字段	类型	取值	说明
threshold	float	0 ~ 1	置信度阈值

HTTP POST Body直接发送图片二进制。

返回参数 | 字段 | 类型 | 取值 | 说明 | | ---------- | ------ | ---- | ---------------------- | | confidence | float | 0~1 | 分类或检测的置信度 | | label | string | | 分类或检测的类别 | | index | number | | 分类或检测的类别 | | x1, y1 | float | 0~1 | 物体检测，矩形的左上角坐标 (相对长宽的比例值) | | x2, y2 | float | 0~1 | 物体检测，矩形的右下角坐标（相对长宽的比例值） |

关于矩形坐标

x1 * 图片宽度 = 检测框的左上角的横坐标

y1 * 图片高度 = 检测框的左上角的纵坐标

x2 * 图片宽度 = 检测框的右下角的横坐标

y2 * 图片高度 = 检测框的右下角的纵坐标

Windows FAQ

1. 服务启动失败，怎么处理？

请确保相关依赖都安装正确，版本必须如下： *.NET Framework 4.5* Visual C++ Redistributable Packages for Visual Studio 2013 * Visual C++ Redistributable Packages for Visual Studio 2015

2. 服务调用时返回为空，怎么处理？ 调用输入的图片必须是RGB格式，请确认是否有alpha通道。

3. 多个模型怎么同时使用？ SDK设置运行不同的端口，点击运行即可。

4. JAVA、C#等其他语言怎么调用SDK?

参考 http://ai.baidu.com/forum/topic/show/943765

5. 启动失败，缺失DLL? 打开EasyEdge.log，查看日志错误，根据提示处理 缺失DLL，请使用 http://www.dependencywalker.com/ 查看相应模块依赖DLL缺失哪些，请自行下载安装

6. 启动失败，报错NotDecrypted? Windows下使用，当前用户名不能为中文，否则无法正确加载模型。

7. 其他问题 如果无法解决，可到论坛发帖： http://ai.baidu.com/forum/topic/list/199 描述使用遇到的问题，我们将及时回复您的问题。

视觉任务Jetson专用SDK集成文档

简介

本文档介绍EasyEdge/EasyDL的Jetson SDK的使用方法。Jetson SDK支持的硬件包括Jetson nano，Jetson TX2，Jetson AGX Xavier和Jetson Xavier NX。您可在AI市场了解Jetson相关系列产品，同时可以在软硬一体方案了解部署方案。

模型支持：

- EasyDL图像：图像分类高精度，图像分类高性能，物体检测高精度，物体检测均衡，物体检测高性能，目标跟踪单标签模型。
- BML：
  - 公开数据集预训练模型：SSD-MobileNetV1，YOLOv3-DarkNet，YOLOv3-MobileNetV1，ResNet50，ResNet101，SE-ResNeXt50，SE-ResNeXt101，MobileNetV2，EfficientNetB0_small，EfficientNetB4，MobileNetV3_large_x1_0，ResNet18_vd，SE_ResNet18_vd，Xception71。
  - 百度超大规模数据集预训练模型：YOLOv3-DarkNet，MobileNetV3_large_x1_0，ResNet50_vd，ResNet101_vd。
- EasyEdge：EasyEdge支持的模型较多，详见查看模型网络适配硬件。若模型不在此列表，可以尝试使用自定义网络生成端计算组件。

软件版本支持 使用EasyDL的Jetson系列SDK需要安装指定版本的JetPack和相关组件。所支持的JetPack版本会随着SDK版本的升级和新版本JetPack的推出而不断的更新。在使用SDK前请务必保证软件版本满足此处声明版本。目前所支持的JetPack版本包括：

- JetPack5.0.1
- JetPack4.6
- JetPack4.5
- JetPack4.4（deprecated，该版本SDK会在未来某个版本移除，请切换至新版本JetPack）
- JetPack4.2.2（已移除，请切换至新版本JetPack）

安装JetPack时请务必安装对应的组件：

- 使用SDK Manager安装JetPack需要勾选TensorRT、OpenCV、CUDA、cuDNN等选项。
- 使用SD Card Image方式（仅对Jetson Nano和Jetson Xavier NX有效）则无需关心组件问题，默认会全部安装。

Release Notes | 时间 | 版本 | 说明 | | --- | ---- | ---- | | 2022.07.28 | 1.6.0 | 新增支持JetPack5.0.1，新增目标追踪接入实时流的demo | | 2022.05.18 | 1.5.0 | 部分模型切换格式，max_batch_size含义变更，由输入图片数不大于该值变更为等于该值；移除适用于JetPack4.2.2的SDK；示例代码demo_stream_inference重构；示例代码移除frame_buffer，新增更安全高效的safe_queue | | 2021.12.22 | 1.3.5 | 新增支持JetPack4.6；支持在EasyEdge平台语义分割模型生成开发套件；修复缓存问题；支持自定义缓存路径 | | 2021.10.20 | 1.3.4 | 新增支持JetPack4.5；大幅提升EasyDL有损压缩加速模型的推理速度 | | 2021.06.29 | 1.3.1 | 视频流支持分辨率调整；支持将预测后的视频推流，新增推流demo | | 2021.05.13 | 1.3.0 | 新增视频流接入支持；EasyDL模型发布新增多种加速方案选择；目标追踪支持x86平台的CPU、GPU加速版；展示已发布模型性能评估报告 | | 2021.03.09 | 1.2.1 | EasyEdge新增一系列模型的支持；性能优化 | | 2021.01.27 | 1.1.0 | EasyDL经典版高性能分类模型升级；
EasyDL经典版检测模型新增均衡选项；
EasyEdge平台新增Jetson系列端计算组件的生成；
问题修复 | | 2020.12.18 | 1.0.0 | 接口升级和一些性能优化 | | 2020.08.11 | 0.5.5 | 部分模型预测速度提升 | | 2020.06.23 | 0.5.4 | 支持JetPack4.4DP，支持EasyDL专业版更多模型 | | 2020.05.15 | 0.5.3 | 专项硬件适配SDK支持Jetson系列 |

> 2022-5-18: 【接口变更】 PREDICTOR_KEY_GTURBO_MAX_BATCH_SIZE含义变更。 变更前：预测输入图片数**不大于**该值均可。 变更后：预测输入图片数需**等于**该值。SDK内部对该接口变更做了兼容处理，在输入图片数小于该值时依然可正常运行，但预测性能会和等于该值时一致。推荐根据实际输入图片数量需求修改该值，尽可能保持最小。 【版本移除】 适用于JetPack4.4版本的SDK被标记为deprecated，SDK会在未来某个版本移除，建议切换至最新版本JetPack。 适用于JetPack4.2.2版本的SDK被移除。

> 2020-12-18: 【接口升级】 参数配置接口从1.0.0版本开始已升级为新接口，以前的方式被置为deprecated，并将在未来的版本中移除。请尽快考虑升级为新的接口方式，具体使用方式可以参考下文介绍以及demo工程示例，谢谢。

> 2021-10-20: 【版本移除】 适用于JetPack4.2.2版本的SDK被标记为deprecated，该版本代码已停止更新，SDK会在未来某个版本移除，请切换至新版本JetPack

快速开始 安装依赖 本SDK适用于JetPack4.4、JetPack4.5、JetPack4.6版本，请务必安装其中之一版本，并使用对应版本的SDK。 注意在安装JetPack时，需同时安装CUDA、cuDNN、OpenCV、TensorRT等组件。

如已安装JetPack需要查询相关版本信息，请参考下文中的开发板信息查询与设置。

使用序列号激活

首先请在官网获取序列号。



将获取到的序列号填写到demo文件中或以参数形式传入。



编译并运行Demo 模型资源文件默认已经打包在开发者下载的SDK包中。Demo工程直接编译即可运行。

编译运行：

```
 cd src
mkdir build && cd build
cmake ..
make -j$(nproc)
###### make install 为可选，也可将lib所在路径添加为环境变量
sudo make install
sudo ldconfig
./demo_batch_inference/easyedge_batch_inference {模型RES文件夹} {测试图片路径或仅包含图片的文件夹路径} {序列号}
```

demo运行示例：

```
 baidu@nano:~/ljay/easydl/sdk/demo/build$ ./demo_batch_inference/easyedge_batch_inference ../../../../RES/
/ljay/images/mix008.jpeg
2020-08-06 20:56:30,665 INFO [EasyEdge] 548125646864 Compiling model for fast inference, this may take a while (Acceleration)
2020-08-06 20:57:58,427 INFO [EasyEdge] 548125646864 Optimized model saved to:
/home/baidu/.baidu/easyedge/jetson/mcache/24110044320/m_cache, Don't remove it
Results of image /ljay/images/mix008.jpeg:
2, kiwi, p:0.997594 loc: 0.352087, 0.56119, 0.625748, 0.868399
2, kiwi, p:0.993221 loc: 0.45789, 0.0730294, 0.73641, 0.399429
2, kiwi, p:0.992884 loc: 0.156876, 0.0598725, 0.3802, 0.394706
1, tomato, p:0.992125 loc: 0.523592, 0.389156, 0.657738, 0.548069
1, tomato, p:0.991821 loc: 0.665461, 0.419503, 0.805282, 0.573558
1, tomato, p:0.989883 loc: 0.297427, 0.439999, 0.432197, 0.59325
1, tomato, p:0.981654 loc: 0.383444, 0.248203, 0.506606, 0.400926
1, tomato, p:0.971682 loc: 0.183775, 0.556587, 0.286996, 0.711361
1, tomato, p:0.968722 loc: 0.379391, 0.0386965, 0.51672, 0.209681
Done
```

检测结果展示：



测试Demo HTTP 服务

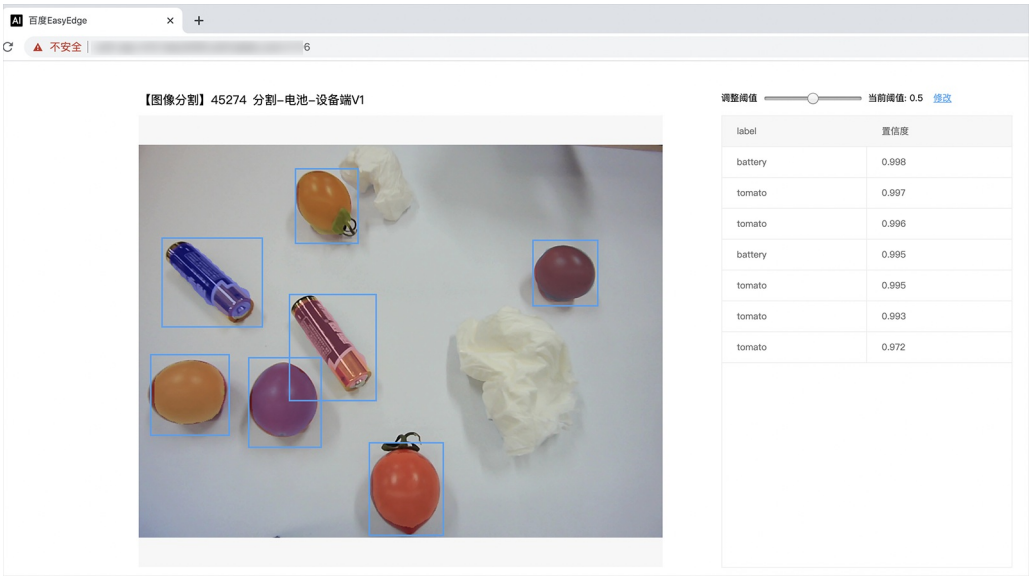
编译demo完成之后，会同时生成一个http服务，运行

```
###### ./easyedge_serving {res_dir} {serial_key} {host, default 0.0.0.0} {port, default 24401}
./easyedge_serving ../../../../RES "1111-1111-1111-1111" 0.0.0.0 24401
```

日志中会显示

```
HTTP is now serving at 0.0.0.0:24401
```

字样，此时，开发者可以打开浏览器，`http://{设备ip}:24401`，选择图片来进行测试。



同时，可以调用HTTP接口来访问服务，具体参考下文接口说明。

使用说明

使用该方式，将运行库嵌入到开发者的程序当中。

使用流程

> 请优先参考Demo的使用流程。遇到错误，请优先参考文件中的注释解释，以及日志说明。

```
 // step 1: 配置模型运行参数
EdgePredictorConfig config;
config.model_dir = model_dir;
config.set_config(params::PREDICTOR_KEY_SERIAL_NUM, serial_num);
config.set_config(params::PREDICTOR_KEY_GTURBO_MAX_BATCH_SIZE, 1);  // 优化的模型可以支持的batch_size
config.set_config(params::PREDICTOR_KEY_GTURBO_FP16, false);       // 置true开启fp16模式推理会更快，精度会略微降低，但取决于硬件是否支持fp16，不是所有模型都支持fp16，参阅文档
config.set_config(params::PREDICTOR_KEY_GTURBO_COMPILE_LEVEL, 1);   // 编译模型的策略，如果当前设置的max_batch_size与历史编译存储的不同，则重新编译模型


// step 2: 创建并初始化Predictor
auto predictor = global_controller()->CreateEdgePredictor<EdgePredictorConfig>(config);
if (predictor->init() != EDGE_OK) {
    exit(-1);
}

// step 3-1: 预测图像
auto img = cv::imread({图片路径});
std::vector<EdgeResultData> results;
predictor->infer(img, results);

// step 3-2: 预测视频
std::vector<EdgeResultData> results;
FrameTensor frame_tensor;
VideoConfig video_config;
video_config.source_type = static_cast<SourceType>(video_type);  // source_type 定义参考头文件 easyedge_video.h
video_config.source_value = video_src;
/*
... more video_configs, 根据需要配置video_config的各选项
*/
auto video_decoding = CreateVideoDecoding(video_config);
while (video_decoding->next(frame_tensor) == EDGE_OK) {
    results.clear();
    if (frame_tensor.is_needed) {
        predictor->infer(frame_tensor.frame, results);
        render(frame_tensor.frame, results, predictor->model_info().kind);
    }
    //video_decoding->display(frame_tensor); // 显示当前frame，需在video_config中开启配置
    //video_decoding->save(frame_tensor); // 存储当前frame到视频，需在video_config中开启配置
 }
```

初始化接口

```
 auto predictor = global_controller()->CreateEdgePredictor<EdgePredictorConfig>(config);
if (predictor->init() != EDGE_OK) {
        exit(-1);
}
```

若返回非0，请查看输出日志排查错误原因。

预测接口

```
 /**
 * @brief
 * 单图预测接口
 * @param image: must be BGR , HWC format (opencv default)
 * @param result
 * @return
 */
virtual int infer(
        cv::Mat& image, std::vector<EdgeResultData>& result
) = 0;

/**
 * @brief
 * 批量图片预测接口
 * @param image: must be BGR , HWC format (opencv default)
 * @param result
 * @return
 */
virtual int infer(
        std::vector<cv::Mat>& image,
        std::vector<std::vector<EdgeResultData>>& results
) = 0;

/**
  * @brief
  * 批量图片预测接口，带阈值
  * @related infer(cv::Mat & image, EdgeColorFormat origin_color_format, std::vector<EdgeResultData> &result, float threshold)
  */
virtual int infer(
            std::vector<cv::Mat> &images,
            EdgeColorFormat origin_color_format,
            std::vector<std::vector<EdgeResultData>> &results,
            float threshold
) = 0;
```

图片的格式务必为opencv默认的BGR, HWC格式。

批量图片的预测接口的使用要求在调用 `init` 接口的时候设置一个有效的 `max_batch_size`，其含义见下方参数配置接口的介绍。

参数配置接口 参数配置通过结构体`EdgePredictorConfig`完成。

```
struct EdgePredictorConfig {
  /**
   * @brief 模型资源文件夹路径
   */
  std::string model_dir;

  std::map<std::string, std::string> conf;

  EdgePredictorConfig();

  template<typename T>
  T get_config(const std::string &key, const T &default_value);

  template<typename T = std::string>
  T get_config(const std::string &key);

  template<typename T>
  const T *get_config(const std::string &key, const T *default_value);

  template<typename T>
  void set_config(const std::string &key, const T &value);

  template<typename T>
  void set_config(const std::string &key, const T *value);

  static EdgePredictorConfig default_config();
};
```

运行参数选项的配置以key、value的方式存储在类型为std::map的conf中，并且键值对的设置和获取可以通过EdgePredictorConfig的set_config和get_config函数完成。同时也支持以环境变量的方式设置键值对。 EdgePredictorConfig的具体使用方法可以参考开发工具包中的demo工程。

针对Jetson开发工具包，目前EdgePredictorConfig的运行参数所支持的Key包括如下项：

```
 /**
 * @brief 当有同类型的多个设备的时候，使用哪一个设备，如：
 * GPU: 使用哪张GPU卡
 * EdgeBoard(VMX)，Movidius NCS ：使用哪一张加速卡
 * 值类型：int
 * 默认值：0
 */
static constexpr auto PREDICTOR_KEY_DEVICE_ID = "PREDICTOR_KEY_DEVICE_ID";

/**
 * @brief 生成 batch_size 为 max_batch_size 的优化模型，单次预测图片数量可以小于或等于此值（推荐等于此值，见release notes）
 * 值类型: int
 * 默认值：4
 */
static constexpr auto PREDICTOR_KEY_GTURBO_MAX_BATCH_SIZE = "PREDICTOR_KEY_GTURBO_MAX_BATCH_SIZE";

/**
 * @brief 设置device对应的GPU卡可以支持的最大并发量
 * 实际预测的时候对应GPU卡的最大并发量不超过这里设置的范围，否则预测请求会排队等待预测执行
 * 值类型: int
 * 默认值：1
 */
static constexpr auto PREDICTOR_KEY_GTURBO_MAX_CONCURRENCY = "PREDICTOR_KEY_GTURBO_MAX_CONCURRENCY";

/**
 * @brief 是否开启fp16模式预测，开启后预测速度会更快，但精度会略有降低。并且需要硬件支持fp16
 * 值类型: bool
 * 默认值：false
 */
static constexpr auto PREDICTOR_KEY_GTURBO_FP16 = "PREDICTOR_KEY_GTURBO_FP16";

/**
 * @brief 模型编译等级
 * 1：如果当前max_batch_size与历史编译产出的max_batch_size不相等时，则重新编译模型（推荐）
```

```
 * 2：无论历史编译产出的max_batch_size为多少，均根据当前max_batch_size重新编译模型
 * 值类型: int
 * 默认值：1
 */
static constexpr auto PREDICTOR_KEY_GTURBO_COMPILE_LEVEL = "PREDICTOR_KEY_GTURBO_COMPILE_LEVEL";

/**
 * @brief GPU工作空间大小设置
 * workspace_size = workspace_prefix * (1 << workspace_offset)
 * workspace_offset: 10 = KB, 20 = MB, 30 = GB
 * 值类型: int
 * 默认值：WORKSPACE_PREFIX: 100, WORKSPACE_OFFSET: 20，即100MB
 */
static constexpr auto PREDICTOR_KEY_GTURBO_WORKSPACE_PREFIX = "PREDICTOR_KEY_GTURBO_WORKSPACE_PREFIX";
static constexpr auto PREDICTOR_KEY_GTURBO_WORKSPACE_OFFSET = "PREDICTOR_KEY_GTURBO_WORKSPACE_OFFSET";

/**
 * @brief 需要使用的dla core
 * 值类型: int
 * 默认值：-1(不使用)
 */
static constexpr auto PREDICTOR_KEY_GTURBO_DLA_CORE = "PREDICTOR_KEY_GTURBO_DLA_CORE";

/**
 * @brief 自定义缓存文件存储路径
 * 值类型: string
 * 默认值: ~/.baidu/easyedge/mcache/{model_id * 1000000 + release_id}
 */
static constexpr auto PREDICTOR_KEY_GTURBO_CACHE_DIR = "PREDICTOR_KEY_GTURBO_CACHE_DIR";

/**
 * @brief 自定义缓存文件命名，默认即可
 * 值类型: string
 * 默认值: 根据配置自动生成
 */
static constexpr auto PREDICTOR_KEY_GTURBO_CACHE_NAME = "PREDICTOR_KEY_GTURBO_CACHE_NAME";

/**
 * @brief 序列号设置；序列号不设置留空时，SDK将会自动尝试使用本地已经激活成功的有效期内的序列号
 * 值类型：string
 * 默认值：空
 */
static constexpr auto PREDICTOR_KEY_SERIAL_NUM = "PREDICTOR_KEY_SERIAL_NUM";
```

`PREDICTOR_KEY_GTURBO_CACHE_NAME`：首次加载模型会先对模型进行编译优化，通过此值可以设置优化后的产出文件名。

`PREDICTOR_KEY_GTURBO_CACHE_DIR`：首次加载模型经过编译优化后，产出的优化文件会存储在这个位置，可以按需修改。

`PREDICTOR_KEY_GTURBO_WORKSPACE_PREFIX`、`PREDICTOR_KEY_GTURBO_WORKSPACE_OFFSET`：设置运行时可以被用来使用的最大临时显存。

`PREDICTOR_KEY_GTURBO_MAX_BATCH_SIZE`：此值用来控制批量图片预测可以支持的最大图片数，实际预测的时候单次预测图片数需等于此值。

`PREDICTOR_KEY_DEVICE_ID`：设置需要使用的 GPU 卡号，对于Jetson，此值无需更改。

`PREDICTOR_KEY_GTURBO_COMPILE_LEVEL`：模型编译等级。通常模型的编译会比较慢，但编译产出是可以复用的。可以在第一次加载模型的时候设置合理的 max_batch_size 并在之后加载模型的时候直接使用历史编译产出。是否使用历史编译产出可以通过此值 compile_level 来控制，当此值为 0 时，表示忽略当前设置的 max_batch_size 而仅使用历史产出（无历史产出时则编译模型）；当此值为 1 时，会比较历史产出和当前设置的 max_batch_size 是否相等，如不等，则重新编译；当此值为 2 时，无论如何都会重新编译模型。

`PREDICTOR_KEY_GTURBO_MAX_CONCURRENCY`：通过此值设置单张 GPU 卡上可以支持的最大 infer 并发量，其上限取决于硬件限制。init 接口会根据此值预分配 GPU 资源，建议结合实际使用控制此值，使用多少则设置多少。注意：此值的增加会降低单次 infer 的速度，建议优先考虑 batch inference。

`PREDICTOR_KEY_GTURBO_FP16`：默认是 fp32 模式，置 true 可以开启 fp16 模式预测，预测速度会有所提升，但精度也会略微下降，权衡使用。注意：不是所有模型都支持 fp16 模式，也不是所有硬件都支持 fp16 模式。已知不支持fp16的模型包括：EasyDL图像分类高精度模型。

预测视频接口

SDK 提供了支持摄像头读取、视频文件和网络视频流的解析工具类`VideoDecoding`，此类提供了获取视频帧数据的便利函数。通过`VideoConfig`结构体可以控制视频/摄像头的解析策略、抽帧策略、分辨率调整、结果视频存储等功能。对于抽取到的视频帧可以直接作为SDK infer 接口的参数进行预测。

class`VideoDecoding`：

```
 /**
 * @brief 获取输入源的下一帧
 * @param frame_tensor
 * @return
 */
virtual int next(FrameTensor &frame_tensor) = 0;

/**
 * @brief 显示当前frame_tensor中的视频帧
 * @param frame_tensor
 * @return
 */
virtual int display(const FrameTensor &frame_tensor) = 0;

/**
 * @brief 将当前frame_tensor中的视频帧写为本地视频文件
 * @param frame_tensor
 * @return
 */
virtual int save(FrameTensor &frame_tensor) = 0;

/**
 * @brief 获取视频的fps属性
 * @return
 */
virtual int get_fps() = 0;
 /**
  * @brief 获取视频的width属性
  * @return
  */
virtual int get_width() = 0;

/**
 * @brief 获取视频的height属性
 * @return
 */
virtual int get_height() = 0;
```

struct `VideoConfig`

```
 /**
 * @brief 视频源、抽帧策略、存储策略的设置选项
 */
struct VideoConfig {
    SourceType source_type;           // 输入源类型
    std::string source_value;         // 输入源地址，如视频文件路径、摄像头index、网络流地址
    int skip_frames{0};               // 设置跳帧，每隔skip_frames帧抽取一帧，并把该抽取帧的is_needed置为true
    int retrieve_all{false};          // 是否抽取所有frame以便于作为显示和存储，对于不满足skip_frames策略的frame，把所抽取帧的
is_needed置为false
    int input_fps{0};                 // 在采取抽帧之前设置视频的fps
    Resolution resolution{Resolution::kAuto}; // 采样分辨率，只对camera有效

    bool enable_display{false};
    std::string window_name{"EasyEdge"};
    bool display_all{false};          // 是否显示所有frame，若为false，仅显示根据skip_frames抽取的frame

    bool enable_save{false};
    std::string save_path;            // frame存储为视频文件的路径
    bool save_all{false};             // 是否存储所有frame，若为false，仅存储根据skip_frames抽取的frame

    std::map<std::string, std::string> conf;
};
```

`source_type`：输入源类型，支持视频文件、摄像头、网络视频流三种，值分别为1、2、3。 `source_value`： 若`source_type`为视频文件，该值为指向视频文件的完整路径；若`source_type`为摄像头，该值为摄像头的index，如对于`/dev/video0`的摄像头，则index为0；若`source_type`为网络视频流，则为该视频流的完整地址。 `skip_frames`：设置跳帧，每隔skip_frames帧抽取一帧，并把该抽取帧的is_needed置为true，标记为is_needed的帧是用来做预测的帧。反之，直接跳过该帧，不经过预测。 `retrieve_all`：若置该项为true，则无论是否设置跳帧，所有的帧都会被抽取返回，以作为显示或存储用。 `input_fps`：用于抽帧前设置fps。 `resolution`：设置摄像头采样的分辨率，其值请参考`easyedge_video.h`中的定义，注意该分辨率调整仅对输入源为摄像头时有效。 `conf`：高级选项。部分配置会通过该map来设置。

注意：

1. 如果使用`VideoConfig`的`display`功能，需要自行编译带有GTK选项的opencv，默认打包的opencv不包含此项。

2. 使用摄像头抽帧时，如果通过`resolution`设置了分辨率调整，但是不起作用，请添加如下选项：

```
video_config.conf["backend"] = "2";
```

3.部分设备上的CSI摄像头尚未兼容，如遇到问题，可以通过工单、QQ交流群或微信交流群反馈。

具体接口调用流程，可以参考SDK中的`demo_video_inference`。

日志配置

设置 `EdgeLogConfig` 的相关参数。具体含义参考文件中的注释说明。

```
EdgeLogConfig log_config;
log_config.enable_debug = true;
global_controller()->set_log_config(log_config);
```

返回格式

预测成功后，从 `EdgeResultData`中可以获取对应的分类信息、位置信息。

```
struct EdgeResultData {
  int index;  // 分类结果的index
  std::string label;  // 分类结果的label
  float prob;  // 置信度

  // 物体检测或图像分割时才有
  float x1, y1, x2, y2;  // (x1, y1): 左上角，  (x2, y2): 右下角； 均为0~1的长宽比例值。

  // 图像分割时才有
  cv::Mat mask;  // 0, 1 的mask
  std::string mask_rle;  // Run Length Encoding，游程编码的mask
};
```

关于矩形坐标

x1 *图片宽度 = 检测框的左上角的横坐标 y1* 图片高度 = 检测框的左上角的纵坐标 x2 *图片宽度 = 检测框的右下角的横坐标 y2* 图片高度 = 检测框的右下角的纵坐标

以上字段可以参考demo文件中使用opencv绘制的逻辑进行解析

http服务

1. 开启http服务 http服务的启动参考`demo_serving.cpp`文件。

```
/**
 * @brief 开启一个简单的demo http服务。
 * 该方法会block直到收到sigint/sigterm。
 * http服务里，图片的解码运行在cpu之上，可能会降低推理速度。
 * @tparam ConfigT
 * @param config
 * @param host
 * @param port
 * @param service_id service_id  user parameter, uri '/get/service_id' will respond this value with 'text/plain'
 * @param instance_num 实例数量，根据内存/显存/时延要求调整
 * @return
 */
template<typename ConfigT>
int start_http_server(
        const ConfigT &config,
        const std::string &host,
        int port,
        const std::string &service_id,
        int instance_num = 1);
```

2. 请求http服务

> 开发者可以打开浏览器，`http://{设备ip}:24401`，选择图片来进行测试。

URL中的get参数：

参数	说明	默认值
threshold	阈值过滤， 0~1	如不提供，则会使用模型的推荐阈值

HTTP POST Body即为图片的二进制内容(无需base64, 无需json)

Python请求示例

```
import requests

with open('./1.jpg', 'rb') as f:
    img = f.read()
    result = requests.post(
     'http://127.0.0.1:24401/',
     params={'threshold': 0.1},
     data=img).json()
```

Java请求示例参考这里

http 返回数据

字段	类型说明	其他
error_code	Number	0为成功,非0参考message获得具体错误信息
results	Array	内容为具体的识别结果。其中字段的具体含义请参考接口使用-返回格式一节
cost_ms	Number	预测耗时ms，不含网络交互时间

返回示例

```
{
  "cost_ms": 52,
  "error_code": 0,
  "results": [
    {
      "confidence": 0.94482421875,
      "index": 1,
      "label": "IronMan",
      "x1": 0.059185408055782318,
      "x2": 0.18795496225357056,
      "y1": 0.14762254059314728,
      "y2": 0.52510076761245728
    },
    {
      "confidence": 0.94091796875,
      "index": 1,
      "label": "IronMan",
      "x1": 0.79151463508605957,
      "x2": 0.92310667037963867,
      "y1": 0.045728668570518494,
      "y2": 0.42920106649398804
    }
  ]
}
```

多线程预测 Jetson 系列 SDK 支持多线程预测，创建一个 predictor，并通过 `PREDICTOR_KEY_GTURBO_MAX_CONCURRENCY` 控制所支持的最大并发量，只需要 init 一次，多线程调用 infer 接口。需要注意的是多线程的启用会随着线程数的增加而降低单次 infer 的推理速度，建议优先使用 batch inference 或权衡考虑使用。

已知问题 1. 多线程时图片按线程分配不均 或 不同batch size的图片交叉调用infer接口时，部分结果错误

A：EasyDL图像分类高精度模型在有些显卡上可能存在此问题，可以考虑填充假图片数据到图片比较少的线程或batch以使得infer间的图片绝对平均。

2. 显存持续增长或遇到 terminate called after throwing an instance of 'std::runtime_error' what(): Failed to create object

A：如果遇到此问题，请确认没有频繁调用 init 接口，通常调用 infer 接口即可满足需求。

3. 开启 fp16 后，预测结果错误

A：不是所有模型都支持 fp16 模式。目前已知的不支持fp16的模型包括：EasyDL图像分类高精度模型。目前不支持的将会在后面的版本陆续支持。

4. 部分模型不支持序列化

A：针对JetPack4.4、4.5版本，部分模型无法使用序列化，如已知的BML的MobileNetV1-SSD和物体检测高性能模型。需要每次加载模型的时候

编译模型，过程会比较慢。此问题将在后续JetPack版本中修复。目前JetPack4.6版本SDK已修复该问题。

开发板信息查询与设置 查询L4T或JetPack版本 查询JetPack版本信息，可以通过下面这条命令先查询L4T的版本。

```
###### 在终端输入如下命令并回车
$ head -n 1 /etc/nv_tegra_release
###### 就会输出类似如下结果
$ # # R32 (release), REVISION: 4.3, GCID: 21589087, BOARD: t210ref, EABI: aarch64, DATE: Fri Jun 26 04:38:25 UTC 2020
```

从输出的结果来看，板子当前的L4T版本为R32.4.3，对应JetPack4.4。 注意，L4T的版本不是JetPack的版本，一般可以从L4T的版本唯一对应到JetPack的版本，下面列出了最近几个版本的对应关系：

```
L4T R32.6.1  --> JetPack4.6
L4T R32.5.1  --> JetPack4.5.1
L4T R32.5    --> JetPack4.5
L4T R32.4.3  --> JetPack4.4
L4T R32.4.2  --> JetPack4.4DP
L4T R32.2.1  --> JetPack4.2.2
L4T R32.2.0  --> JetPack4.2.1
```

功率模式设置与查询 不同的功率模式下，执行AI推理的速度是不一样的，如果对速度需求很高，可以把功率开到最大，但记得加上小风扇散热~

```
###### 1. 运行下面这条命令可以查询开发板当前的运行功率模式
$ sudo nvpmodel -q verbose
###### $ NV Power Mode: MAXN
###### $ 0
###### 如果输出为MAXN代表是最大功率模式

###### 2. 若需要把功率调到最大，运行下面这条命令
$ sudo nvpmodel -m 0

###### 如果你进入了桌面系统，也可以在桌面右上角有个按钮可以切换模式

###### 3. 查询资源利用率
$ sudo tegrastats
```

FAQ 1. EasyDL SDK与云服务效果不一致，如何处理？

后续我们会消除这部分差异，如果开发者发现差异较大，可联系我们协助处理。

2. 运行SDK报错 Authorization failed 日志显示 `Http perform failed: null respond` 在新的硬件上首次运行，必须联网激活。

SDK 能够接受`HTTP_PROXY` 的环境变量通过代理处理自己的网络请求。如

```
export HTTP_PROXY="http://192.168.1.100:8888"
./easyedge_demo ...
```

3. 使用libcurl请求http服务时，速度明显变慢

这是因为libcurl请求continue导致server等待数据的问题，添加空的header即可

```
headers = curl_slist_append(headers, "Expect:");
```

4. 运行demo时报找不到libeasyedge_extension.so

需要export `libeasyedge_extension.so`所在的路径，如路径为/home/work/baidu/cpp/lib，则需执行：

```
export LD_LIBRARY_PATH=/home/work/baidu/cpp/lib:${LD_LIBRARY_PATH}
```

或者在编译完后执行如下命令将lib文件安装到系统路径：

```
sudo make install
```

如不能安装，也可手动复制lib下的文件到`/usr/local/lib`下。

5. 运行demo时报如下之一错误

```
2020-12-17 16:15:07,924 INFO [EasyEdge] 547633188880 Compiling model for fast inference, this may take a while (Acceleration)
Killed

###### 或

2020-12-17 16:15:07,924 INFO [EasyEdge] 547633188880 Build graph failed
```

请适当降低`PREDICTOR_KEY_GTURBO_MAX_BATCH_SIZE`和`PREDICTOR_KEY_GTURBO_MAX_CONCURRENCY`的值后尝试。

6. 运行有损压缩加速的模型，运算精度较标准模型偏低 首先请保证数据集不会太小。其次可以通过将模型目录RES中的`calibrationtable`移除，并通过将`PREDICTOR_KEY_GTURBO_FP16`设置为`true`，使用FP16的运算精度重新评估模型效果。若依然不理想，可将`calibrationtable`移除并将`PREDICTOR_KEY_GTURBO_FP16`设置为`false`,从而使用更高精度的FP32的运算精度。

视觉任务专用辨影SDK集成开发文档

简介

本文档介绍EasyEdge/EasyDL的辨影软硬一体方案SDK的使用方法。支持的硬件包括辨影Air、辨影Pro。您可以在软硬一体方案了解部署方案。

模型支持：

- EasyDL图像：图像分类高精度，图像分类高性能，物体检测高精度，物体检测均衡，物体检测高性能
- BML：
  - 公开数据集预训练模型：SSD-MobileNetV1，YOLOv3-DarkNet，YOLOv3-MobileNetV1，ResNet50，ResNet101，SE-ResNeXt50，SE-ResNeXt101，MobileNetV2，EfficientNetB0_small，EfficientNetB4，MobileNetV3_large_x1_0，ResNet18_vd，SE_ResNet18_vd，Xception71。
  - 百度超大规模数据集预训练模型：YOLOv3-DarkNet，MobileNetV3_large_x1_0，ResNet50_vd，ResNet101_vd。
- EasyEdge：EasyEdge支持的模型较多，详见查看模型网络适配硬件。若模型不在此列表，可以尝试使用自定义网络生成端计算组件。

Release Notes | 时间 | 版本 | 说明 | | --- | ---- | ---- | | 2022.08.01 | 1.3.5 | 新增支持辨影软硬一体方案部署 |

辨影软件接入使用SDK 辨影Air/Pro自带软件预置了大量飞桨开源模型，支持EasyDL/BML模型SDK一键导入使用，详细的辨影使用说明见购买后获得的使用说明书

- 辨影推理主界面



- 辨影设置界面。在应用中可选预置模型能力，也可选择EasyDL/BML导入的模型SDK



快速开始 接下来的文档内容将会描述辨影SDK的集成开发教程，仅需要使用辨影自带软件的用户无需关注

使用序列号激活

首先请在EasyDL智能云官网获取序列号。



将获取到的序列号填写到demo文件中或以参数形式传入。



编译并运行Demo 模型资源文件默认已经打包在开发者下载的SDK包中。Demo工程直接编译即可运行。

编译运行：

```
 cd src
mkdir build && cd build
cmake ..
make
###### make install 为可选，也可将lib所在路径添加为环境变量
sudo make install
sudo ldconfig
./demo_batch_inference/easyedge_batch_inference {模型RES文件夹} {测试图片路径或仅包含图片的文件夹路径} {序列号}
```

demo运行示例：

```
 baidu@nano:~/ljay/easydl/sdk/demo/build$ ./demo_batch_inference/easyedge_batch_inference ../../../../RES/
/ljay/images/mix008.jpeg
2020-08-06 20:56:30,665 INFO [EasyEdge] 548125646864 Compiling model for fast inference, this may take a while (Acceleration)
2020-08-06 20:57:58,427 INFO [EasyEdge] 548125646864 Optimized model saved to:
/home/baidu/.baidu/easyedge/jetson/mcache/24110044320/m_cache, Don't remove it
Results of image /ljay/images/mix008.jpeg:
2, kiwi, p:0.997594 loc: 0.352087, 0.56119, 0.625748, 0.868399
2, kiwi, p:0.993221 loc: 0.45789, 0.0730294, 0.73641, 0.399429
2, kiwi, p:0.992884 loc: 0.156876, 0.0598725, 0.3802, 0.394706
1, tomato, p:0.992125 loc: 0.523592, 0.389156, 0.657738, 0.548069
1, tomato, p:0.991821 loc: 0.665461, 0.419503, 0.805282, 0.573558
1, tomato, p:0.989883 loc: 0.297427, 0.439999, 0.432197, 0.59325
1, tomato, p:0.981654 loc: 0.383444, 0.248203, 0.506606, 0.400926
1, tomato, p:0.971682 loc: 0.183775, 0.556587, 0.286996, 0.711361
1, tomato, p:0.968722 loc: 0.379391, 0.0386965, 0.51672, 0.209681
Done
```

检测结果展示：



测试Demo HTTP 服务

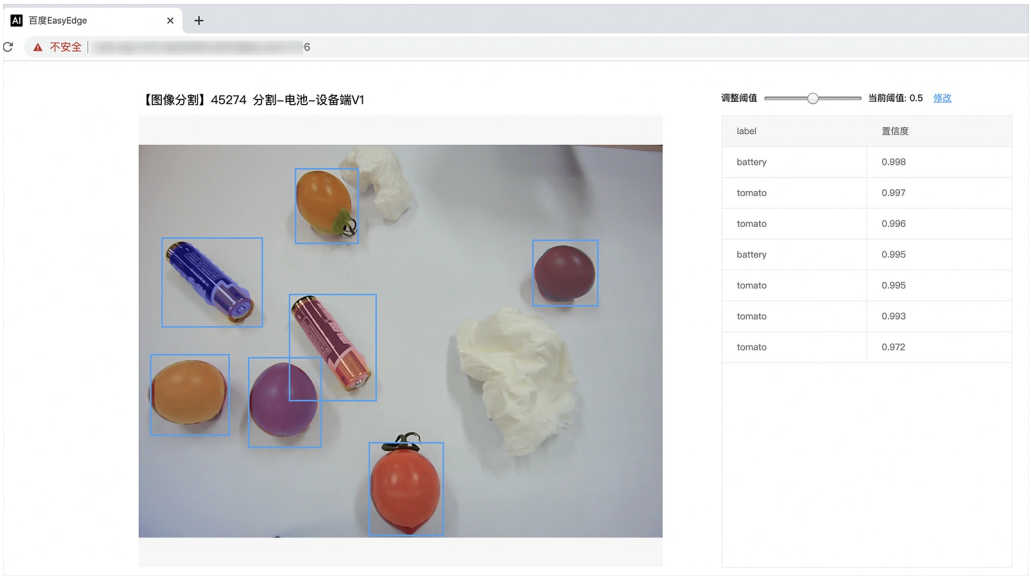
编译demo完成之后，会同时生成一个http服务，运行

```
###### ./easyedge_serving {res_dir} {serial_key} {host, default 0.0.0.0} {port, default 24401}
./easyedge_serving ../../../../RES "1111-1111-1111-1111" 0.0.0.0  24401
```

日志中会显示

```
HTTP is now serving at 0.0.0.0:24401
```

字样，此时，开发者可以打开浏览器，`http://{设备ip}:24401`，选择图片来进行测试。



同时，可以调用HTTP接口来访问服务，具体参考下文接口说明。

使用说明

使用该方式，将运行库嵌入到开发者的程序当中。

使用流程

> 请优先参考Demo的使用流程。遇到错误，请优先参考文件中的注释解释，以及日志说明。

```
 // step 1: 配置模型运行参数
EdgePredictorConfig config;
config.model_dir = model_dir;
config.set_config(params::PREDICTOR_KEY_SERIAL_NUM, serial_num);
config.set_config(params::PREDICTOR_KEY_GTURBO_MAX_BATCH_SIZE, 1);  // 优化的模型可以支持的最大batch_size，实际单次推理的图片数不能大于此值
config.set_config(params::PREDICTOR_KEY_GTURBO_FP16, false);       // 置true开启fp16模式推理会更快，精度会略微降低，但取决于硬件是否支持fp16，不是所有模型都支持fp16，参阅文档
config.set_config(params::PREDICTOR_KEY_GTURBO_COMPILE_LEVEL, 1);   // 编译模型的策略，如果当前设置的max_batch_size与历史编译存储的不同，则重新编译模型


// step 2: 创建并初始化Predictor
auto predictor = global_controller()->CreateEdgePredictor<EdgePredictorConfig>(config);
if (predictor->init() != EDGE_OK) {
    exit(-1);
}

// step 3-1: 预测图像
auto img = cv::imread({图片路径});
std::vector<EdgeResultData> results;
predictor->infer(img, results);

// step 3-2: 预测视频
std::vector<EdgeResultData> results;
FrameTensor frame_tensor;
VideoConfig video_config;
video_config.source_type = static_cast<SourceType>(video_type);  // source_type 定义参考头文件 easyedge_video.h
video_config.source_value = video_src;
/*
... more video_configs, 根据需要配置video_config的各选项
*/
auto video_decoding = CreateVideoDecoding(video_config);
while (video_decoding->next(frame_tensor) == EDGE_OK) {
    results.clear();
    if (frame_tensor.is_needed) {
        predictor->infer(frame_tensor.frame, results);
        render(frame_tensor.frame, results, predictor->model_info().kind);
    }
    //video_decoding->display(frame_tensor); // 显示当前frame，需在video_config中开启配置
    //video_decoding->save(frame_tensor); // 存储当前frame到视频，需在video_config中开启配置
 }
```

初始化接口

```
 auto predictor = global_controller()->CreateEdgePredictor<EdgePredictorConfig>(config);
if (predictor->init() != EDGE_OK) {
        exit(-1);
}
```

若返回非0，请查看输出日志排查错误原因。

预测接口

```
 /**
 * @brief
 * 单图预测接口
 * @param image: must be BGR , HWC format (opencv default)
 * @param result
 * @return
 */
virtual int infer(
        cv::Mat& image, std::vector<EdgeResultData>& result
) = 0;

/**
 * @brief
 * 批量图片预测接口
 * @param image: must be BGR , HWC format (opencv default)
 * @param result
 * @return
 */
virtual int infer(
        std::vector<cv::Mat>& image,
        std::vector<std::vector<EdgeResultData>>& results
) = 0;

/**
  * @brief
  * 批量图片预测接口，带阈值
  * @related infer(cv::Mat & image, EdgeColorFormat origin_color_format, std::vector<EdgeResultData> &result, float threshold)
  */
virtual int infer(
            std::vector<cv::Mat> &images,
            EdgeColorFormat origin_color_format,
            std::vector<std::vector<EdgeResultData>> &results,
            float threshold
) = 0;
```

图片的格式务必为opencv默认的BGR, HWC格式。

批量图片的预测接口的使用要求在调用 `init` 接口的时候设置一个有效的 `max_batch_size` ，其含义见下方参数配置接口的介绍。

参数配置接口 参数配置通过结构体`EdgePredictorConfig`完成。

```
struct EdgePredictorConfig {
  /**
   * @brief 模型资源文件夹路径
   */
  std::string model_dir;

  std::map<std::string, std::string> conf;

  EdgePredictorConfig();

  template<typename T>
  T get_config(const std::string &key, const T &default_value);

  template<typename T = std::string>
  T get_config(const std::string &key);

  template<typename T>
  const T *get_config(const std::string &key, const T *default_value);

  template<typename T>
  void set_config(const std::string &key, const T &value);

  template<typename T>
  void set_config(const std::string &key, const T *value);

  static EdgePredictorConfig default_config();
};
```

运行参数选项的配置以key、value的方式存储在类型为std::map的conf中，并且键值对的设置和获取可以通过EdgePredictorConfig的set_config和get_config函数完成。同时也支持以环境变量的方式设置键值对。 EdgePredictorConfig的具体使用方法可以参考开发工具包中的demo工程。

针对Jetson开发工具包，目前EdgePredictorConfig的运行参数所支持的Key包括如下项：

```
 /**
 * @brief 当有同类型的多个设备的时候，使用哪一个设备，如：
 * GPU: 使用哪张GPU卡
 * EdgeBoard(VMX)，Movidius NCS ：使用哪一张加速卡
 * 值类型：int
 * 默认值：0
 */
static constexpr auto PREDICTOR_KEY_DEVICE_ID = "PREDICTOR_KEY_DEVICE_ID";

/**
 * @brief 生成最大 batch_size 为 max_batch_size 的优化模型，单次预测图片数量可以小于或等于此值
 * 值类型: int
 * 默认值：4
 */
static constexpr auto PREDICTOR_KEY_GTURBO_MAX_BATCH_SIZE = "PREDICTOR_KEY_GTURBO_MAX_BATCH_SIZE";

/**
 * @brief 设置device对应的GPU卡可以支持的最大并发量
 * 实际预测的时候对应GPU卡的最大并发量不超过这里设置的范围，否则预测请求会排队等待预测执行
 * 值类型: int
 * 默认值：1
 */
static constexpr auto PREDICTOR_KEY_GTURBO_MAX_CONCURRENCY = "PREDICTOR_KEY_GTURBO_MAX_CONCURRENCY";

/**
 * @brief 是否开启fp16模式预测，开启后预测速度会更快，但精度会略有降低。并且需要硬件支持fp16
 * 值类型: bool
 * 默认值：false
 */
static constexpr auto PREDICTOR_KEY_GTURBO_FP16 = "PREDICTOR_KEY_GTURBO_FP16";

/**
 * @brief 模型编译等级
 * 1：如果当前max_batch_size与历史编译产出的max_batch_size不相等时，则重新编译模型（推荐）
```

```
     * 2：无论历史编译产出的max_batch_size为多少，均根据当前max_batch_size重新编译模型
     * 值类型: int
     * 默认值：1
     */
    static constexpr auto PREDICTOR_KEY_GTURBO_COMPILE_LEVEL = "PREDICTOR_KEY_GTURBO_COMPILE_LEVEL";

    /**
     * @brief GPU工作空间大小设置
     * workspace_size = workspace_prefix * (1 << workspace_offset)
     * workspace_offset: 10 = KB, 20 = MB, 30 = GB
     * 值类型: int
     * 默认值：WORKSPACE_PREFIX: 100, WORKSPACE_OFFSET: 20，即100MB
     */
    static constexpr auto PREDICTOR_KEY_GTURBO_WORKSPACE_PREFIX = "PREDICTOR_KEY_GTURBO_WORKSPACE_PREFIX";
    static constexpr auto PREDICTOR_KEY_GTURBO_WORKSPACE_OFFSET = "PREDICTOR_KEY_GTURBO_WORKSPACE_OFFSET";

    /**
     * @brief 需要使用的dla core
     * 值类型: int
     * 默认值：-1(不使用)
     */
    static constexpr auto PREDICTOR_KEY_GTURBO_DLA_CORE = "PREDICTOR_KEY_GTURBO_DLA_CORE";

    /**
     * @brief 自定义缓存文件存储路径
     * 值类型: string
     * 默认值: ~/.baidu/easyedge/mcache/{model_id * 1000000 + release_id}
     */
    static constexpr auto PREDICTOR_KEY_GTURBO_CACHE_DIR = "PREDICTOR_KEY_GTURBO_CACHE_DIR";

    /**
     * @brief 自定义缓存文件命名，默认即可
     * 值类型: string
     * 默认值: 根据配置自动生成
     */
    static constexpr auto PREDICTOR_KEY_GTURBO_CACHE_NAME = "PREDICTOR_KEY_GTURBO_CACHE_NAME";

    /**
     * @brief 序列号设置；序列号不设置留空时，SDK将会自动尝试使用本地已经激活成功的有效期内的序列号
     * 值类型：string
     * 默认值：空
     */
    static constexpr auto PREDICTOR_KEY_SERIAL_NUM = "PREDICTOR_KEY_SERIAL_NUM";
```

`PREDICTOR_KEY_GTURBO_CACHE_NAME`：首次加载模型会先对模型进行编译优化，通过此值可以设置优化后的产出文件名。

`PREDICTOR_KEY_GTURBO_CACHE_DIR`：首次加载模型经过编译优化后，产出的优化文件会存储在这个位置，可以按需修改。

`PREDICTOR_KEY_GTURBO_WORKSPACE_PREFIX`、`PREDICTOR_KEY_GTURBO_WORKSPACE_OFFSET`：设置运行时可以被用来使用的最大临时显存。

`PREDICTOR_KEY_GTURBO_MAX_BATCH_SIZE`：此值用来控制批量图片预测可以支持的最大图片数，实际预测的时候单次预测图片数不可大于此值，但可以是不大于此值的任意图片数。

`PREDICTOR_KEY_GTURBO_COMPILE_LEVEL`：模型编译等级。通常模型的编译会比较慢，但编译产出是可以复用的。可以在第一次加载模型的时候设置合理的 max_batch_size 并在之后加载模型的时候直接使用历史编译产出。是否使用历史编译产出可以通过此值 compile_level 来控制，当此值为 0 时，表示忽略当前设置的 max_batch_size 而仅使用历史产出（无历史产出时则编译模型）；当此值为 1 时，会比较历史产出和当前设置的 max_batch_size 是否相等，如不等，则重新编译；当此值为 2 时，无论如何都会重新编译模型。

`PREDICTOR_KEY_GTURBO_MAX_CONCURRENCY`：通过此值设置单张 GPU 卡上可以支持的最大 infer 并发量，其上限取决于硬件限制。init 接口会根据此值预分配 GPU 资源，建议结合实际使用控制此值，使用多少则设置多少。注意：此值的增加会降低单次 infer 的速度，建议优先考虑 batch inference。

`PREDICTOR_KEY_GTURBO_FP16`：默认是 fp32 模式，置 true 可以开启 fp16 模式预测，预测速度会有所提升，但精度也会略微下降，权衡使用。注意：不是所有模型都支持 fp16 模式，也不是所有硬件都支持 fp16 模式。已知不支持fp16的模型包括：EasyDL图像分类高精度模型。

预测视频接口

SDK 提供了支持摄像头读取、视频文件和网络视频流的解析工具类`VideoDecoding`，此类提供了获取视频帧数据的便利函数。通过`VideoConfig`结构体可以控制视频/摄像头的解析策略、抽帧策略、分辨率调整、结果视频存储等功能。对于抽取到的视频帧可以直接作为SDK infer 接口的参数进行预测。

class`VideoDecoding`：

```
 /**
 * @brief 获取输入源的下一帧
 * @param frame_tensor
 * @return
 */
virtual int next(FrameTensor &frame_tensor) = 0;

/**
 * @brief 显示当前frame_tensor中的视频帧
 * @param frame_tensor
 * @return
 */
virtual int display(const FrameTensor &frame_tensor) = 0;

/**
 * @brief 将当前frame_tensor中的视频帧写为本地视频文件
 * @param frame_tensor
 * @return
 */
virtual int save(FrameTensor &frame_tensor) = 0;

/**
 * @brief 获取视频的fps属性
 * @return
 */
virtual int get_fps() = 0;
 /**
  * @brief 获取视频的width属性
  * @return
  */
virtual int get_width() = 0;

/**
 * @brief 获取视频的height属性
 * @return
 */
virtual int get_height() = 0;
```

struct `VideoConfig`

```
 /**
 * @brief 视频源、抽帧策略、存储策略的设置选项
 */
struct VideoConfig {
    SourceType source_type;           // 输入源类型
    std::string source_value;         // 输入源地址，如视频文件路径、摄像头index、网络流地址
    int skip_frames{0};               // 设置跳帧，每隔skip_frames帧抽取一帧，并把该抽取帧的is_needed置为true
    int retrieve_all{false};          // 是否抽取所有frame以便于作为显示和存储，对于不满足skip_frames策略的frame，把所抽取帧的
is_needed置为false
    int input_fps{0};                 // 在采取抽帧之前设置视频的fps
    Resolution resolution{Resolution::kAuto}; // 采样分辨率，只对camera有效

    bool enable_display{false};
    std::string window_name{"EasyEdge"};
    bool display_all{false};          // 是否显示所有frame，若为false，仅显示根据skip_frames抽取的frame

    bool enable_save{false};
    std::string save_path;            // frame存储为视频文件的路径
    bool save_all{false};             // 是否存储所有frame，若为false，仅存储根据skip_frames抽取的frame

    std::map<std::string, std::string> conf;
};
```

source_type：输入源类型，支持视频文件、摄像头、网络视频流三种，值分别为1、2、3。 source_value： 若source_type为视频文件，该值为指向视频文件的完整路径；若source_type为摄像头，该值为摄像头的index，如对于/dev/video0的摄像头，则index为0；若source_type为网络视频流，则为该视频流的完整地址。 skip_frames：设置跳帧，每隔skip_frames帧抽取一帧，并把该抽取帧的is_needed置为true，标记为is_needed的帧是用来做预测的帧。反之，直接跳过该帧，不经过预测。 retrieve_all：若置该项为true，则无论是否设置跳帧，所有的帧都会被抽取返回，以作为显示或存储用。 input_fps：用于抽帧前设置fps。 resolution：设置摄像头采样的分辨率，其值请参考easyedge_video.h中的定义，注意该分辨率调整仅对输入源为摄像头时有效。 conf：高级选项。部分配置会通过该map来设置。

注意：

1. 如果使用VideoConfig的display功能，需要自行编译带有GTK选项的opencv，默认打包的opencv不包含此项。

2. 使用摄像头抽帧时，如果通过resolution设置了分辨率调整，但是不起作用，请添加如下选项：

```
 video_config.conf["backend"] = "2";
```

3.部分设备上的CSI摄像头尚未兼容，如遇到问题，可以通过工单、QQ交流群或微信交流群反馈。

具体接口调用流程，可以参考SDK中的demo_video_inference。

日志配置

设置 EdgeLogConfig 的相关参数。具体含义参考文件中的注释说明。

```
 EdgeLogConfig log_config;
log_config.enable_debug = true;
global_controller()->set_log_config(log_config);
```

返回格式

预测成功后，从 EdgeResultData中可以获取对应的分类信息、位置信息。

```
struct EdgeResultData {
  int index;  // 分类结果的index
  std::string label;  // 分类结果的label
  float prob;  // 置信度

  // 物体检测或图像分割时才有
  float x1, y1, x2, y2;  // (x1, y1): 左上角，  (x2, y2): 右下角； 均为0~1的长宽比例值。

  // 图像分割时才有
  cv::Mat mask;  // 0, 1 的mask
  std::string mask_rle;  // Run Length Encoding，游程编码的mask
};
```

关于矩形坐标

x1 *图片宽度 = 检测框的左上角的横坐标 y1* 图片高度 = 检测框的左上角的纵坐标 x2 *图片宽度 = 检测框的右下角的横坐标 y2* 图片高度 = 检测框的右下角的纵坐标

以上字段可以参考demo文件中使用opencv绘制的逻辑进行解析

http服务

1. 开启http服务 http服务的启动参考`demo_serving.cpp`文件。

```
/**
 * @brief 开启一个简单的demo http服务。
 * 该方法会block直到收到sigint/sigterm。
 * http服务里，图片的解码运行在cpu之上，可能会降低推理速度。
 * @tparam ConfigT
 * @param config
 * @param host
 * @param port
 * @param service_id service_id  user parameter, uri '/get/service_id' will respond this value with 'text/plain'
 * @param instance_num 实例数量，根据内存/显存/时延要求调整
 * @return
 */
template<typename ConfigT>
int start_http_server(
        const ConfigT &config,
        const std::string &host,
        int port,
        const std::string &service_id,
        int instance_num = 1);
```

2. 请求http服务

> 开发者可以打开浏览器，`http://{设备ip}:24401`，选择图片来进行测试。

URL中的get参数：

参数	说明	默认值
threshold	阈值过滤， 0~1	如不提供，则会使用模型的推荐阈值

HTTP POST Body即为图片的二进制内容(无需base64, 无需json)

Python请求示例

```
import requests

with open('./1.jpg', 'rb') as f:
    img = f.read()
    result = requests.post(
     'http://127.0.0.1:24401/',
     params={'threshold': 0.1},
     data=img).json()
```

Java请求示例参考这里

http 返回数据

字段	类型说明	其他
error_code	Number	0为成功,非0参考message获得具体错误信息
results	Array	内容为具体的识别结果。其中字段的具体含义请参考接口使用-返回格式一节
cost_ms	Number	预测耗时ms，不含网络交互时间

返回示例

```
{
  "cost_ms": 52,
  "error_code": 0,
  "results": [
    {
      "confidence": 0.94482421875,
      "index": 1,
      "label": "IronMan",
      "x1": 0.059185408055782318,
      "x2": 0.18795496225357056,
      "y1": 0.14762254059314728,
      "y2": 0.52510076761245728
    },
    {
      "confidence": 0.94091796875,
      "index": 1,
      "label": "IronMan",
      "x1": 0.79151463508605957,
      "x2": 0.92310667037963867,
      "y1": 0.045728668570518494,
      "y2": 0.42920106649398804
    }
  ]
}
```

多线程预测 辨影系列 SDK 支持多线程预测，创建一个 predictor，并通过 `PREDICTOR_KEY_GTURBO_MAX_CONCURRENCY` 控制所支持的最大并发量，只需要 init 一次，多线程调用 infer 接口。需要注意的是多线程的启用会随着线程数的增加而降低单次 infer 的推理速度，建议优先使用 batch inference 或权衡考虑使用。

已知问题 1. 多线程时图片按线程分配不均 或 不同batch size的图片交叉调用infer接口时，部分结果错误

A：EasyDL图像分类高精度模型在有些显卡上可能存在此问题，可以考虑填充假图片数据到图片比较少的线程或batch以使得infer间的图片绝对平均。

2. 显存持续增长或遇到 terminate called after throwing an instance of 'std::runtime_error' what(): Failed to create object

A：如果遇到此问题，请确认没有频繁调用 init 接口，通常调用 infer 接口即可满足需求。

3. 开启 fp16 后，预测结果错误

A：不是所有模型都支持 fp16 模式。目前已知的不支持fp16的模型包括：EasyDL图像分类高精度模型。目前不支持的将会在后面的版本陆续支持。

4. 部分模型不支持序列化

A：针对JetPack4.4版本，部分模型无法使用序列化，如已知的BML的MobileNetV1-SSD和物体检测高性能模型。需要每次加载模型的时候编译模

型，过程会比较慢。此问题将在后续JetPack版本中修复。

开发板信息查询与设置 查询L4T或JetPack版本 查询JetPack版本信息，可以通过下面这条命令先查询L4T的版本。

```
###### 在终端输入如下命令并回车
$ head -n 1 /etc/nv_tegra_release
###### 就会输出类似如下结果
$ # # R32 (release), REVISION: 4.3, GCID: 21589087, BOARD: t210ref, EABI: aarch64, DATE: Fri Jun 26 04:38:25 UTC 2020
```

从输出的结果来看，板子当前的L4T版本为R32.4.3，对应JetPack4.4。 注意，L4T的版本不是JetPack的版本，一般可以从L4T的版本唯一对应到JetPack的版本，下面列出了最近几个版本的对应关系：

```
L4T R32.6.1  --> JetPack4.6
L4T R32.5.1  --> JetPack4.5.1
L4T R32.5    --> JetPack4.5
L4T R32.4.3  --> JetPack4.4
L4T R32.4.2  --> JetPack4.4DP
L4T R32.2.1  --> JetPack4.2.2
L4T R32.2.0  --> JetPack4.2.1
```

功率模式设置与查询 不同的功率模式下，执行AI推理的速度是不一样的，如果对速度需求很高，可以把功率开到最大，但记得加上小风扇散热~

```
###### 1. 运行下面这条命令可以查询开发板当前的运行功率模式
$ sudo nvpmodel -q verbose
###### $ NV Power Mode: MAXN
###### $ 0
###### 如果输出为MAXN代表是最大功率模式

###### 2. 若需要把功率调到最大，运行下面这条命令
$ sudo nvpmodel -m 0

###### 如果你进入了桌面系统，也可以在桌面右上角有个按钮可以切换模式

###### 3. 查询资源利用率
$ sudo tegrastats
```

FAQ 1. EasyDL SDK与云服务效果不一致，如何处理？

后续我们会消除这部分差异，如果开发者发现差异较大，可联系我们协助处理。

2. 运行SDK报错 Authorization failed 日志显示 `Http perform failed: null respond` 在新的硬件上首次运行，必须联网激活。

SDK 能够接受`HTTP_PROXY` 的环境变量通过代理处理自己的网络请求。如

```
export HTTP_PROXY="http://192.168.1.100:8888"
./easyedge_demo ...
```

3. 使用libcurl请求http服务时，速度明显变慢

这是因为libcurl请求continue导致server等待数据的问题，添加空的header即可

```
headers = curl_slist_append(headers, "Expect:");
```

4. 运行demo时报找不到libeasyedge_extension.so

需要export `libeasyedge_extension.so`所在的路径，如路径为/home/work/baidu/cpp/lib，则需执行：

```
export LD_LIBRARY_PATH=/home/work/baidu/cpp/lib:${LD_LIBRARY_PATH}
```

或者在编译完后执行如下命令将lib文件安装到系统路径：

```
sudo make install
```

如不能安装，也可手动复制lib下的文件到/usr/local/lib下。

5. 运行demo时报如下之一错误

```
 2020-12-17 16:15:07,924 INFO [EasyEdge] 547633188880 Compiling model for fast inference, this may take a while (Acceleration)
Killed

###### 或

2020-12-17 16:15:07,924 INFO [EasyEdge] 547633188880 Build graph failed
```

请适当降低PREDICTOR_KEY_GTURBO_MAX_BATCH_SIZE和PREDICTOR_KEY_GTURBO_MAX_CONCURRENCY的值后尝试。

6. 运行有损压缩加速的模型，运算精度较标准模型偏低 首先请保证数据集不会太小。其次可以通过将模型目录RES中的calibrationtable移除，并通过将PREDICTOR_KEY_GTURBO_FP16设置为true，使用FP16的运算精度重新评估模型效果。若依然不理想，可将calibrationtable移除并将PREDICTOR_KEY_GTURBO_FP16设置为false,从而使用更高精度的FP32的运算精度。

端云协同服务部署

目录

1 端云协同服务简介
2 使用流程
2.1 发布端云协同部署包
2.2 新增设备并激活
2.2 下发部署包到设备并在本地调用
3 云端管理说明
3.1 模型部署包管理
3.1.1 发布及更新模型版本
3.2 本地设备管理

端云协同服务部署

端云协同服务简介

BML端云协同服务由EasyEdge端与边缘AI服务平台提供、基于百度智能边缘构建，能够便捷地将BML定制模型的推理能力拓展至应用现场，提供临时离线、低延时的计算服务。

「云管理，端计算」的端云协同服务，具体包括：

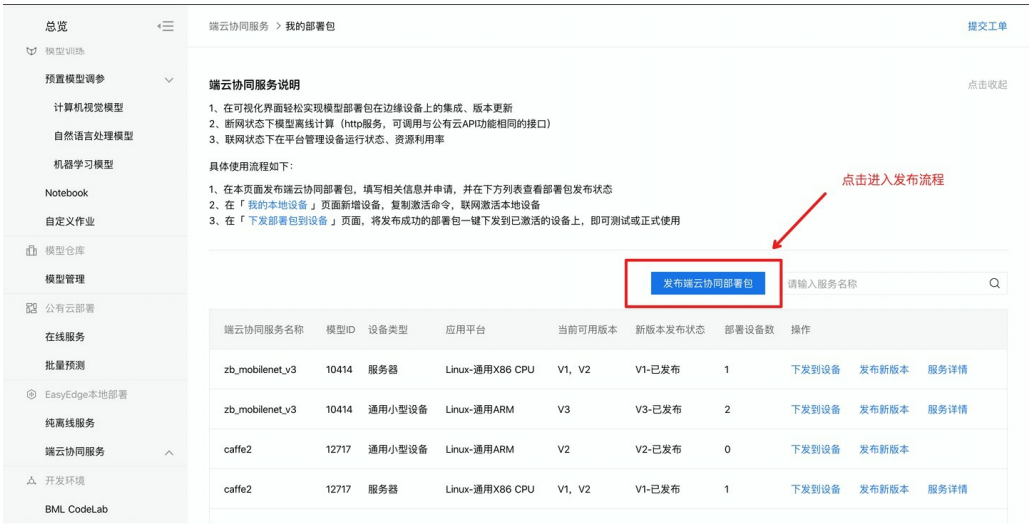
- 在可视化界面轻松实现模型部署包在边缘设备上的集成、版本更新
- 断网状态下模型离线计算（http服务，可调用与公有云API功能相同的接口）
- 联网状态下在平台管理设备运行状态、资源利用率
- 目前本地服务器的应用平台支持Linux-AMD64(x86-64)，具体使用流程请参考下方文档。

使用流程

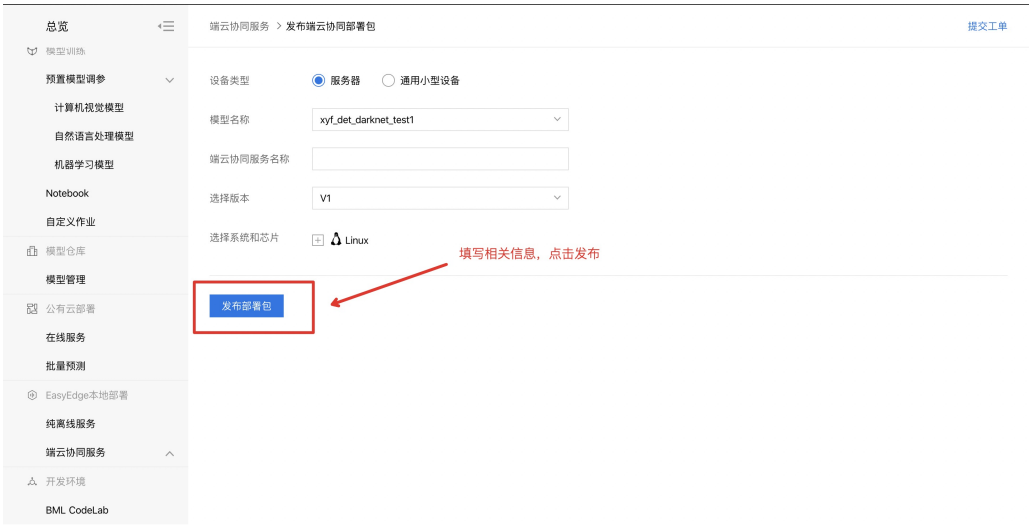
BML端云协同服务的使用流程总体可以分为发布端云协同部署包、新增设备并激活、下发部署包到设备并在本地调用三步，如下为详细介绍。

发布端云协同部署包

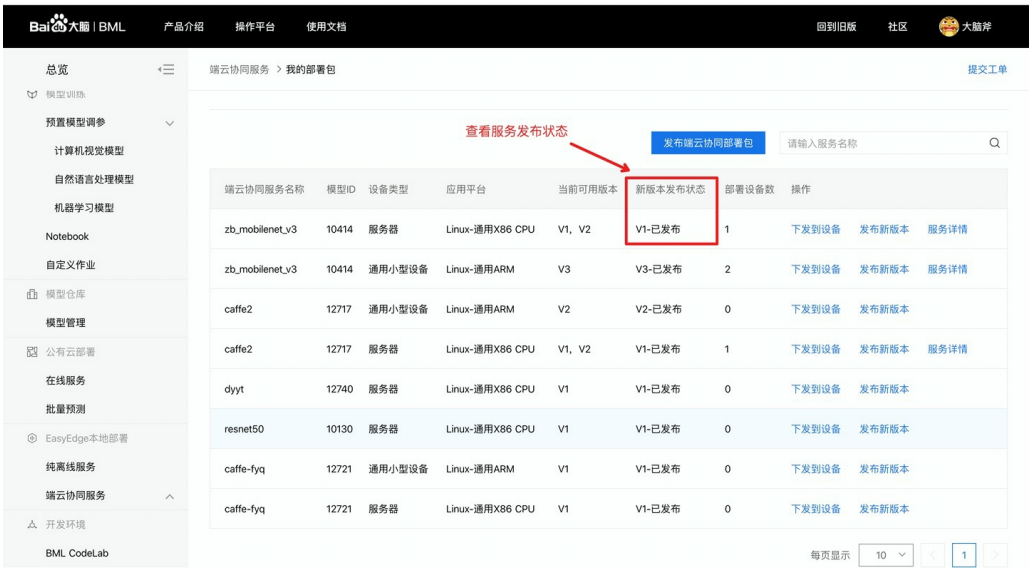
在我的部署包页面点击「发布端云协同部署包」。



填写服务名称，选择模型版本并提交发布。



在列表查看部署包发布状态。

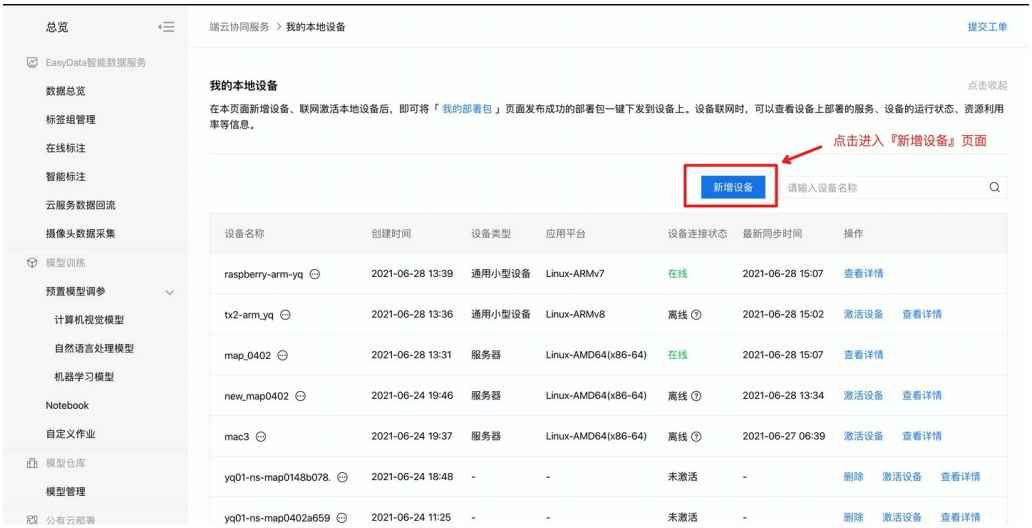


新增设备并激活

点击进入『我的本地设备』页面。



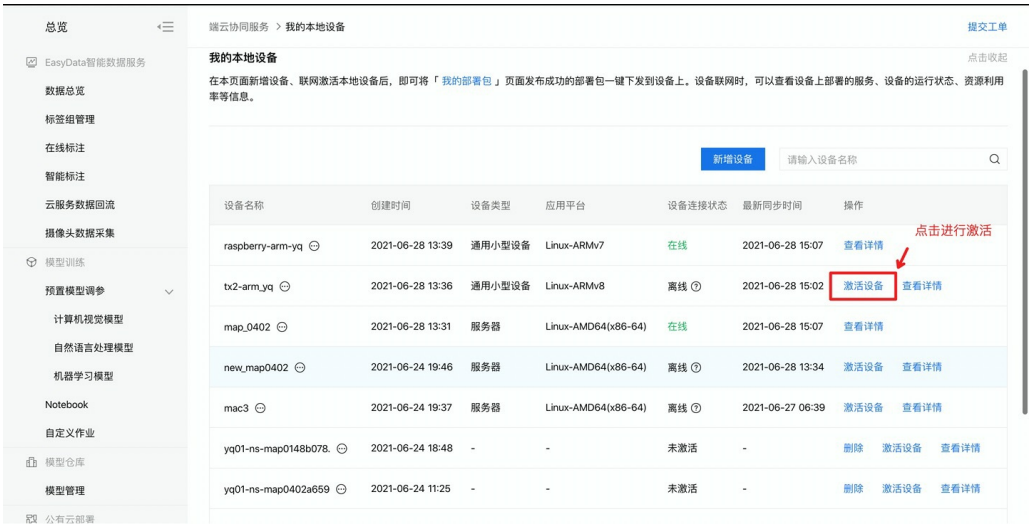
在我的本地设备页面点击新增设备。

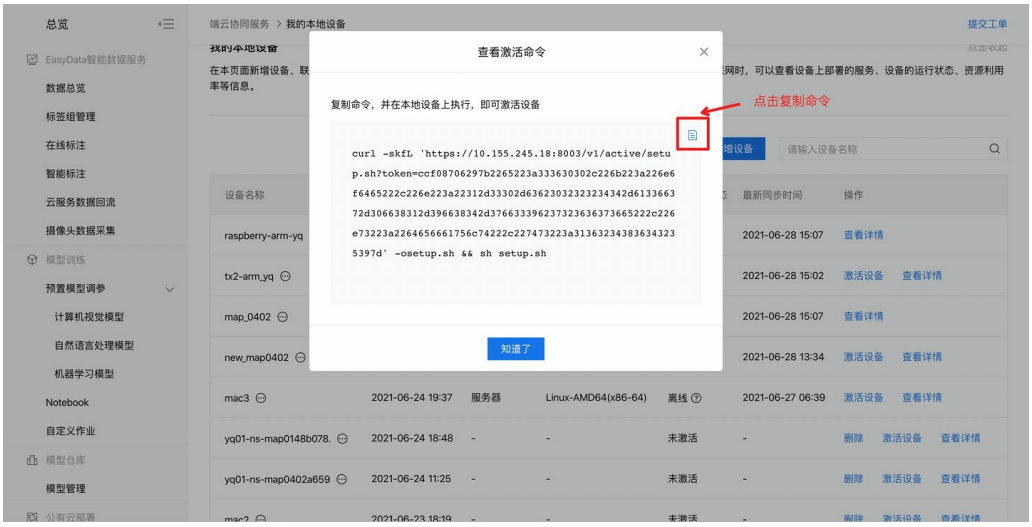


填写相关信息。



在列表中，点击设备对应的「激活设备」操作，复制激活命令并在本地设备上执行即可。

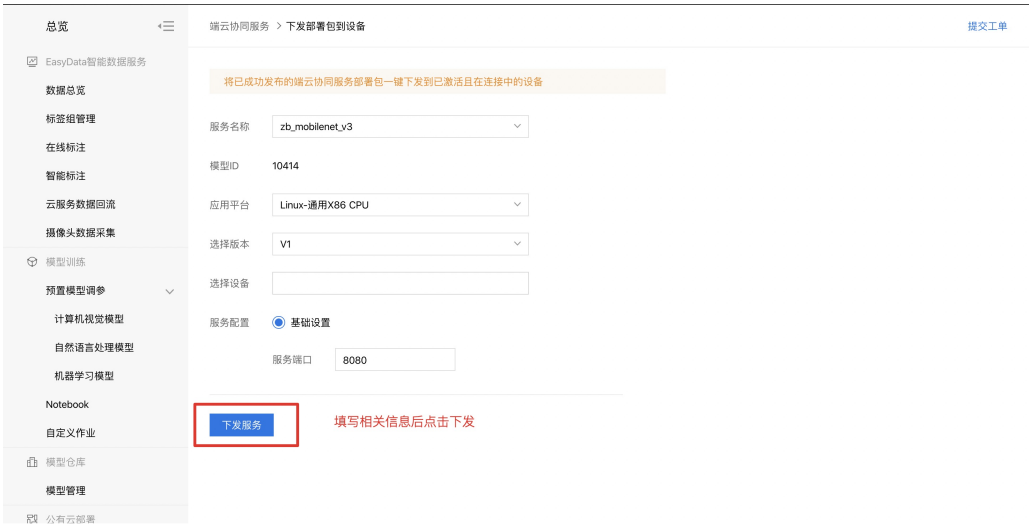




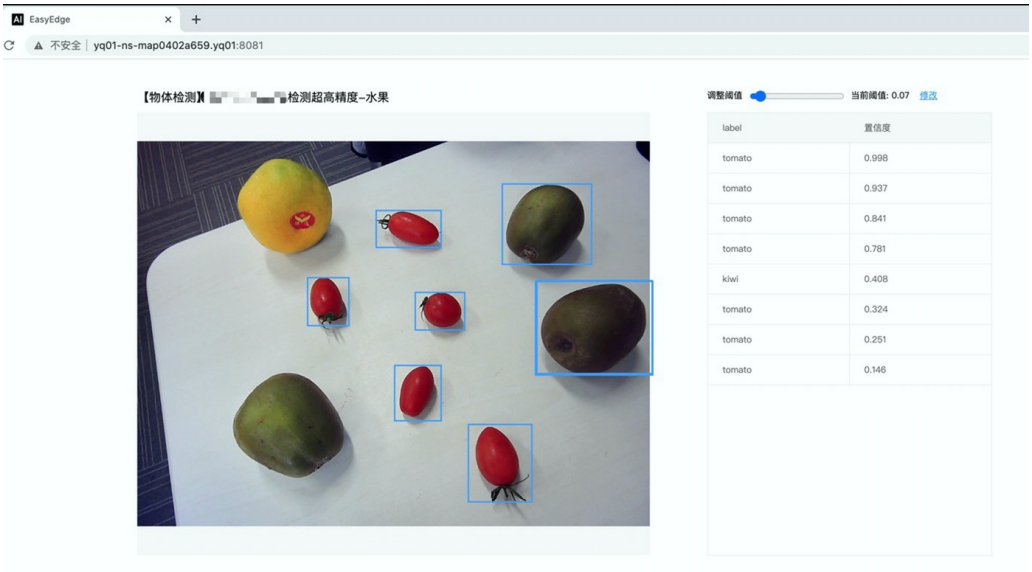
下发部署包到设备并在本地调用

在下发部署包到设备页面，将发布成功的部署包一键下发到已激活的设备上，即可测试或正式使用。





部署包下发成功之后，会在本地启动一个HTTP推理服务。在浏览器中输入"http://{设备ip}:{服务端口，默认8080}"，即可预览效果：



具体接口调用说明请参考文档 SDK - HTTP服务调用说明。

云端管理说明

BML为用户提供了云端管理端云协同服务的功能。

模型部署包管理

在我的部署包页面可以进行已发布的模型部署包的管理。

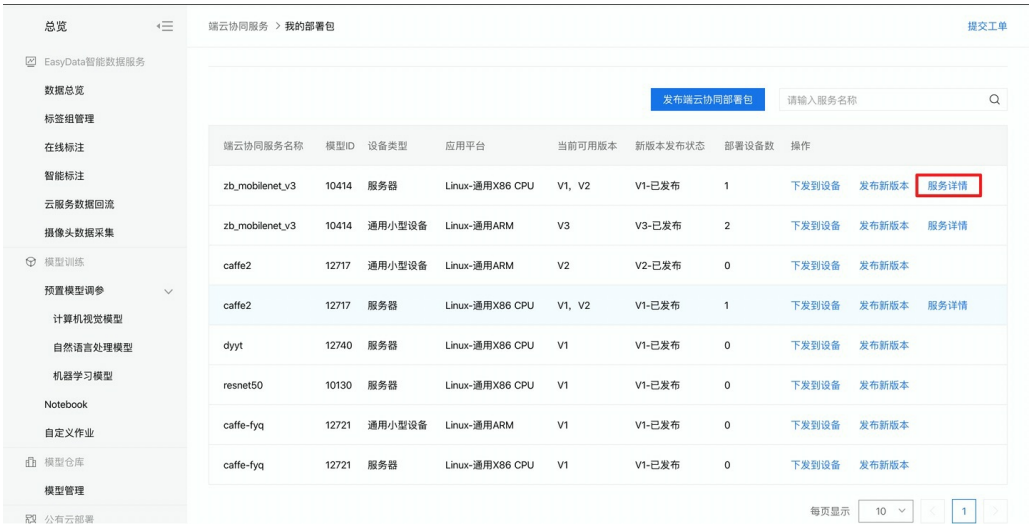
发布及更新模型版本

点击「发布新版本」操作即可快速发布对应模型ID下的新版本。同一模型ID下已发布的模型版本均会显示在列表的「当前可用版本」中。

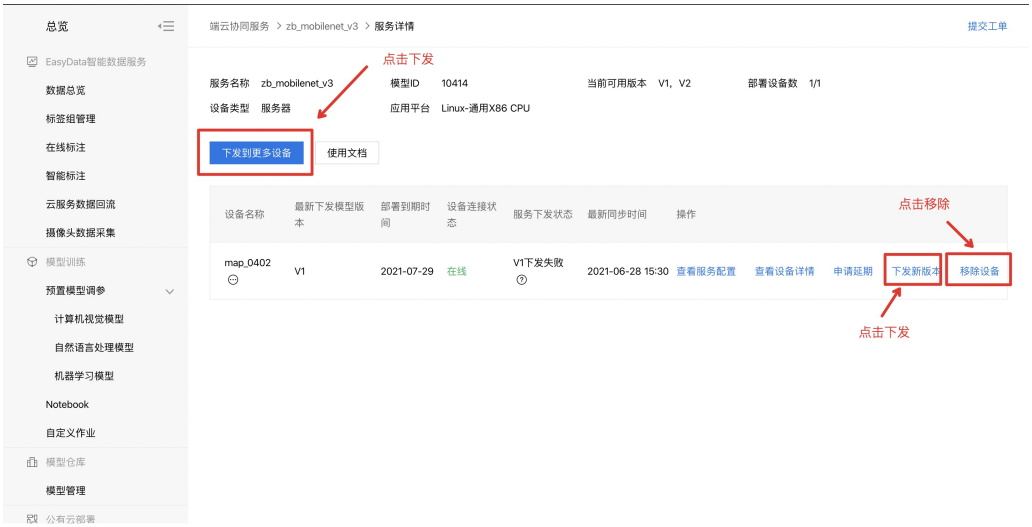




新版本发布成功后，即可点击当前服务的「服务详情」页面，将新版本下发到本地设备上。



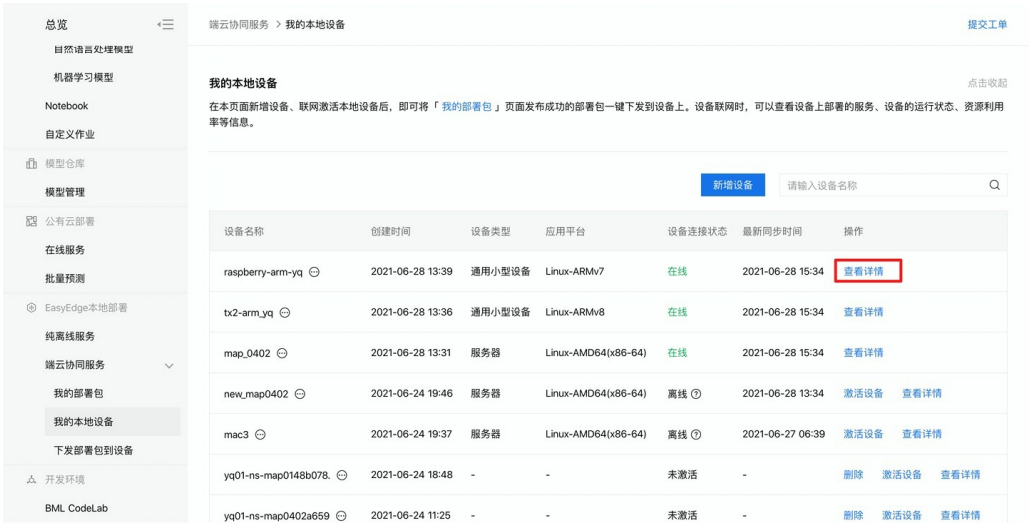
在上述的「服务详情」页面，可以查看并管理当前服务已部署的设备，包括移除设备、将服务下发到更多的设备等。

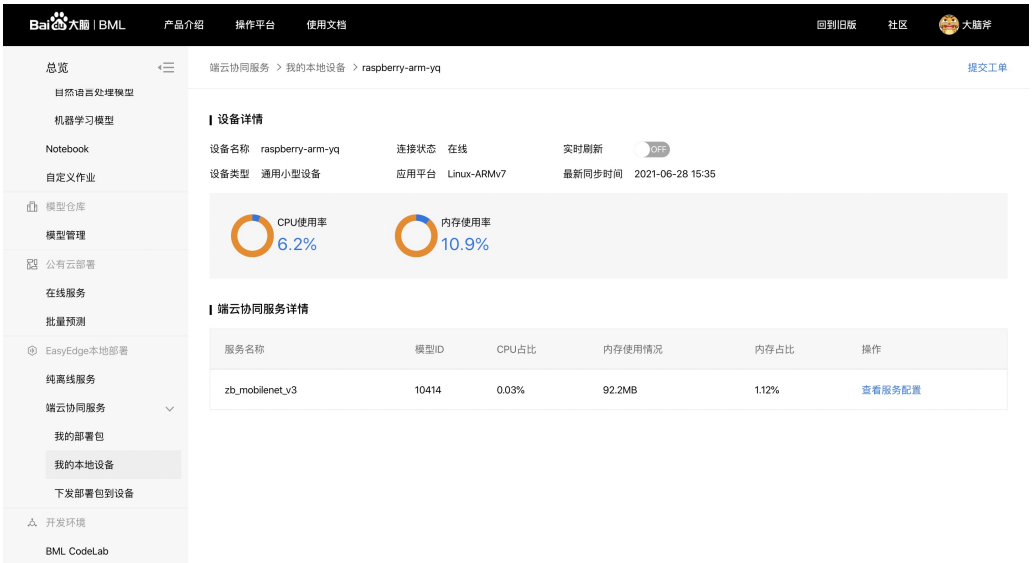


本地设备管理

在我的本地设备页面可以进行所有本地设备的管理。

点击单台设备的「服务详情」，可查看设备上运行的多个服务及设备状态：





智能边缘控制台-单节点版

EasyEdge 智能边缘控制台-单节点版 IEC

EasyEdge Intelligent Edge Console （以下简称IEC）是EasyEdge推出的边缘设备管理的本地化方案。可以运行于多种架构、多系统、多类型的终端之上。通过IEC，用户可以方便地在本地进行

- EasyDL/BML/EasyEdge的SDK的 离线 / 在线激活，服务管理

- 接入本地和远程摄像头，网页中实时预览
- 自动监控和记录相关事件
- 硬件信息的可视化查看

支持的系统+CPU架构包括：

- Windows x86_64 (Windows 7 ~ Windows 10，暂不支持Windows 11)
- Linux x86_64 / arm32 / arm64

支持各类常见的AI加速芯片，包括：

- NVIDIA GPU / Jetson 系列
- Baidu EdgeBoard FZ系列
- 比特大陆 Bitmain SC / SE 系列
- 华为 Atlas 系列
- 寒武纪 MLU 系列
- 其他EasyDL/EasyEdge/BML支持的AI芯片

完整列表可参考这里

Release Note

> 注意：2.0.0之后，默认以系统服务形式安装iec，无法兼容1.x版本的iec

版本号	发布时间	更新说明
2.2.0	2022-10-27	新增onvif/gb28181支持；完善端云通信逻辑
2.0.0	2022-03-22	支持连接中心节点IECC；支持以系统服务安装
1.0.2	2021-12-22	更新视频预览推流库；新增若干AI芯片支持；支持多种芯片温度、功耗展示；多项性能优化
1.0.0	2021-09-16	IEC 第一版！

快速开始

从这里选择您需要的操作系统和CPU架构下载：

- Windows amd64 ： intel、AMD的64位x86_84 CPU
- Linux amd64：intel、AMD的64位x86_84 CPU
- Linux arm：树莓派等32位的ARM CPU
- Linux arm64：RK3399、飞腾等64位的ARM CPU

或者从纯离线服务管理页可下载智能边缘控制台



> 您也可以通过先安装多节点版本IECC，通过中心节点来自动连接安装边缘节点。

Linux 安装 解压缩之后，目录结构如下

```
 0 EasyEdge-IEC-v2.0.0-linux-amd64 > tree .
.
├─── easyedge-iec
├─── easyedge-iec-setup.sh
├─── etc
│    ├─── easyedge-iec.service-conf.init.d
│    ├─── easyedge-iec.service-conf.systemd
│    ├─── easyedge-iec.service-conf.upstart
│    ├─── easyedge-iec.service.yml
│    └─── easyedge-iec.yml
└─── readme.txt

1 directory, 8 files
```

以系统服务形式安装（推荐） 以root用户运行`./easyedge-iec-setup.sh install`即可

```
 [setup]: sudo could not be found
[setup]: Start to install IEC...
[setup]: +  bash -c "cp easyedge-iec /usr/sbin/easyedge-iec"
[setup]: +  bash -c "chmod +x /usr/sbin/easyedge-iec"
[setup]: +  bash -c "cp etc/easyedge-iec.service.yml /etc/easyedge-iec/easyedge-iec.yml"
[setup]: +  bash -c "cp etc/easyedge-iec.service-conf.init.d /etc/init.d/easyedge-iec"
[setup]: +  bash -c "chmod +x /etc/init.d/easyedge-iec"
[setup]: Install IEC success!
[setup]: +  bash -c "service easyedge-iec start"
Starting easyedge-iec: success
[setup]: Start to check IEC status...
[setup]: +  bash -c "curl -s 127.0.0.1:8702 >/dev/null"
[setup]: IEC status: OK!
[easyedge-iec]: default configure file: /etc/easyedge-iec/easyedge-iec.yml
[easyedge-iec]: default log file: /var/log/easyedge-iec/easyedge-iec.log
[easyedge-iec]: service usage: service easyedge-iec { start | stop }
[setup]: Done!
```

- 日志：`/var/log/easyedge-iec/easyedge-iec.log`
- 系统配置：`/etc/easyedge-iec/easyedge-iec.yml`
- 服务启动/停止：`service easyedge-iec { start | stop }` (不同操作系统内可能不同，具体命令参考安装日志)

自定义安装（不推荐） 自定义安装方法仅限于 安装脚本无法识别的情况。

- 拷贝`./EasyEdge-IEC-v2.0.0/`整个目录至自定义文件夹，如`/opt/EasyEdge-IEC`
- 进入到`/opt/EasyEdge-IEC`
- 通过 nohup 等方法运行`./easyedge-iec-linux-{您的系统架构}` *amd64: intel、AMD的64位x86_84 CPU* arm：树莓派等32位的ARM CPU * arm64：RK3399、飞腾等64位的ARM CPU
- 日志：`./log/easyedge-iec.log`
- 系统配置：`./easyedge-iec.yml`

Windows 安装

解压缩之后，安装目录如下所示：

```
 0 tmp2 > tree EasyEdge-IEC-v2.0.0-windows-amd64
EasyEdge-IEC-v2.0.0-windows-amd64
├──── easyedge-iec.exe
├──── easyedge-iec-setup.bat
├──── etc
│    └──── easyedge-iec.yml
└──── readme.txt

1 directory, 4 files
```

打开命令行（非powershell）运行 `easyedge-iec-setup.bat install`。

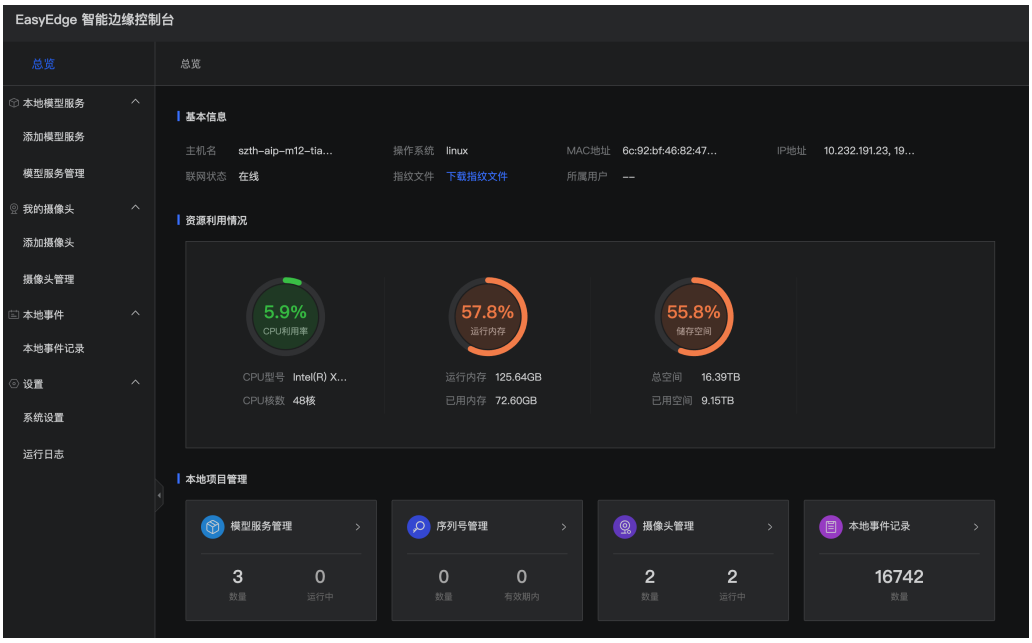
如果遇到hang住的情况，可修改命令行配置 启动之后，打开浏览器，访问 `http://{设备ip}:8702/easyedge/iec` 即可：



启动之后，打开浏览器，访问 `http://{设备ip}:8702/easyedge` 即可：

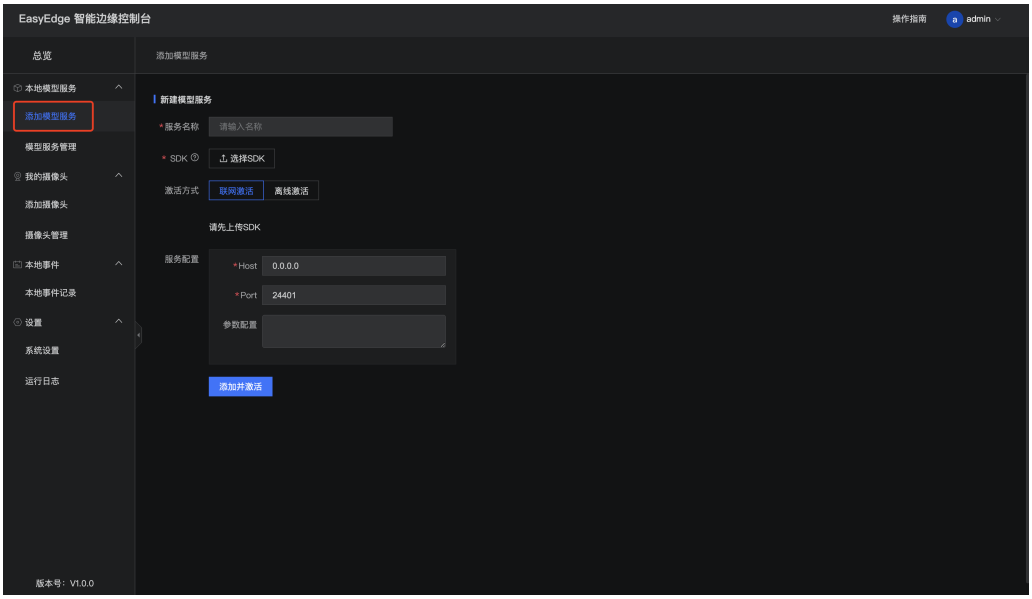


> 默认用户名密码为 admin / easyedge



功能使用说明

①添加模型服务 首先，点击导航栏的「本地模型服务」-「添加模型服务」。在页面中定义服务名称后，将已经下载好的Linux/Windows版本的SDK与IEC关联。关联完毕后可按两种激活方式，激活使用SDK。



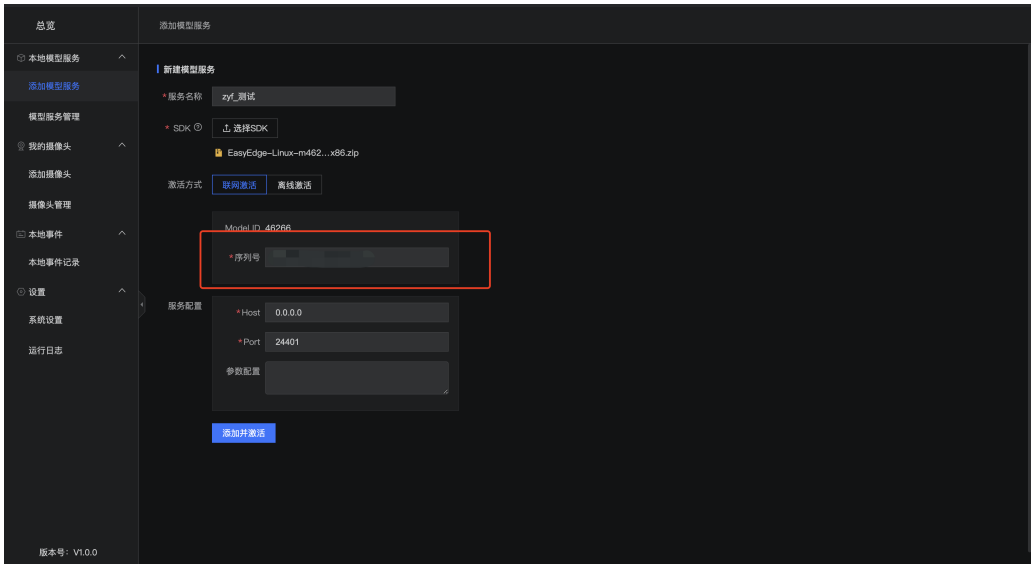
> 部分SDK需要提前安装系统依赖，如TRT等，具体请参考EasyDL/BML/EasyEdge SDK使用文件中的环境依赖安装说明

联网激活

1. 在关联SDK完成后，需要在百度智能云控制台对应部署方式管理页中新增测试序列号或购买正式序列号。（图中以服务器版SDK为例）



2. 再在IEC中填入所申请的序列号



3. 配置服务，在服务端口不冲突占用的情况下，使用默认即可

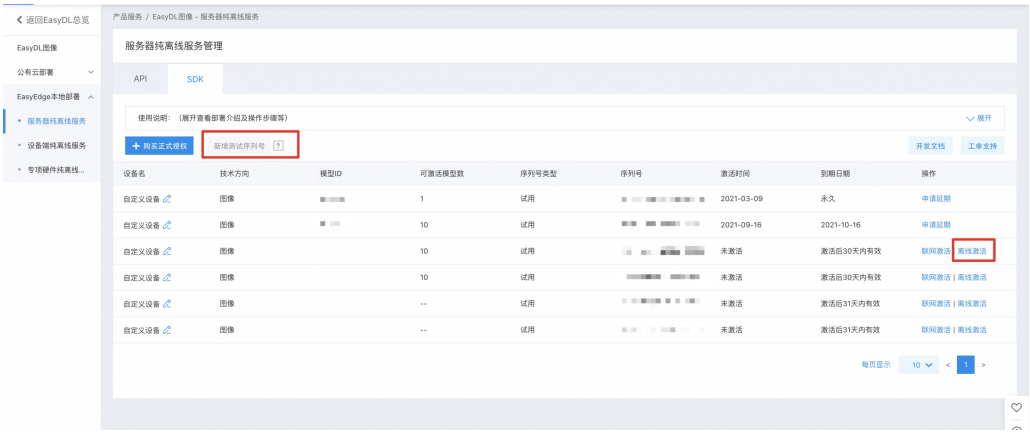
4. 添加并激活

离线激活

1. 在IEC总览页面下载「指纹文件」

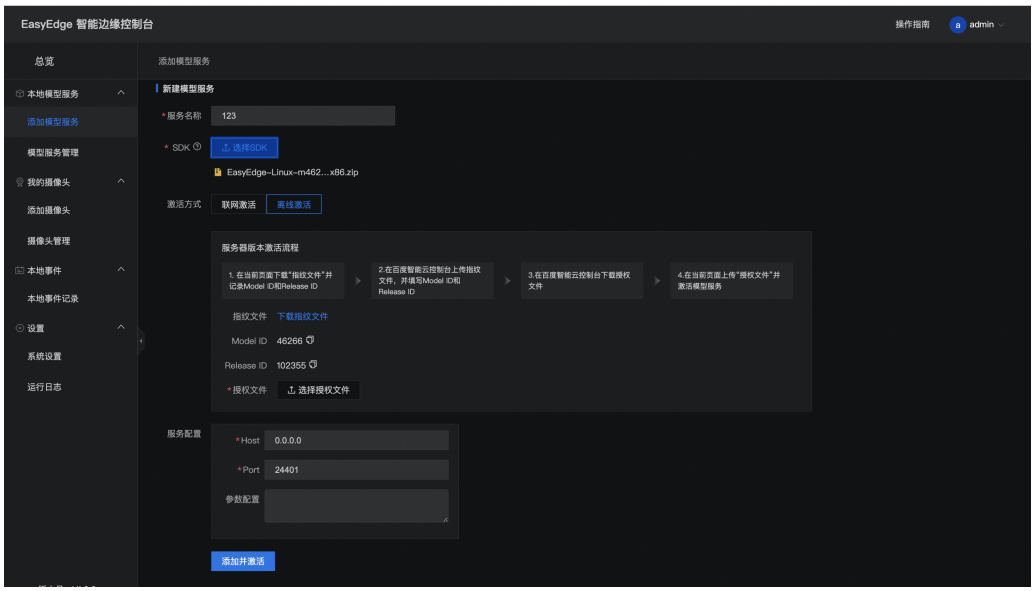


2. 在百度智能云的控制台中找到SDK对应的管理列表，图中以服务器SDK为例。申请序列号后，点击对应序列号尾部的「离线激活」操作，按指



引激活

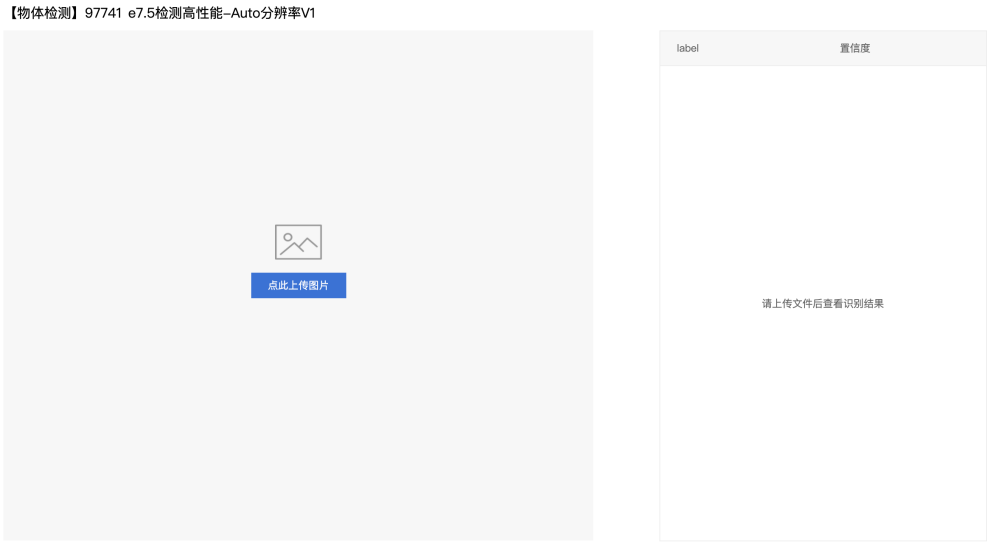
3. 在IEC的添加模型服务页面，上传下载好的授权文件，完成激活



激活完成后即可在「模型服务管理」列表中启动服务，使用后续的操作栏功能。

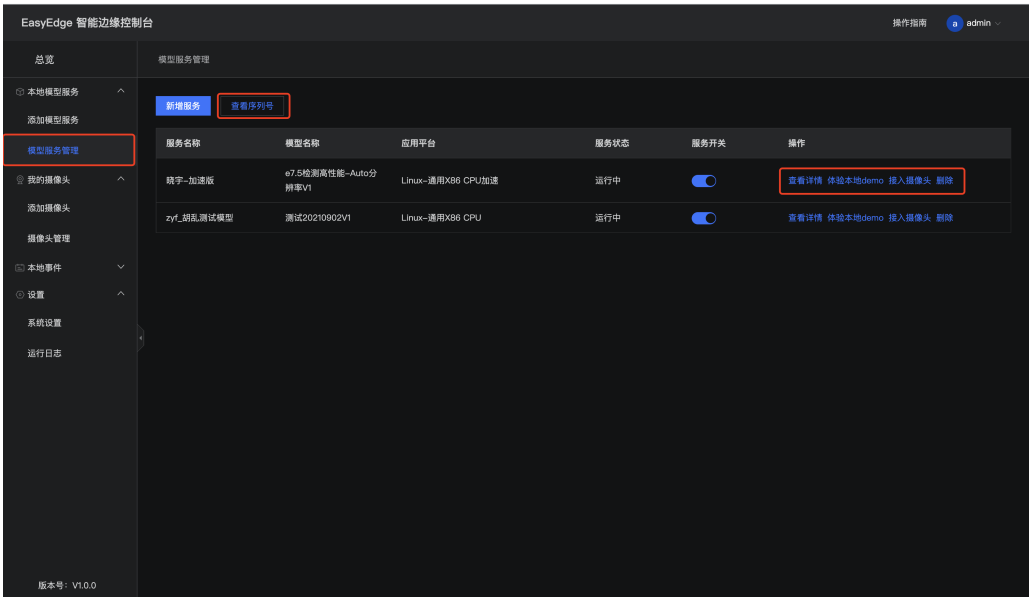
体验本地demo

点击「本地demo体验」即可在立即上传图片进行预测

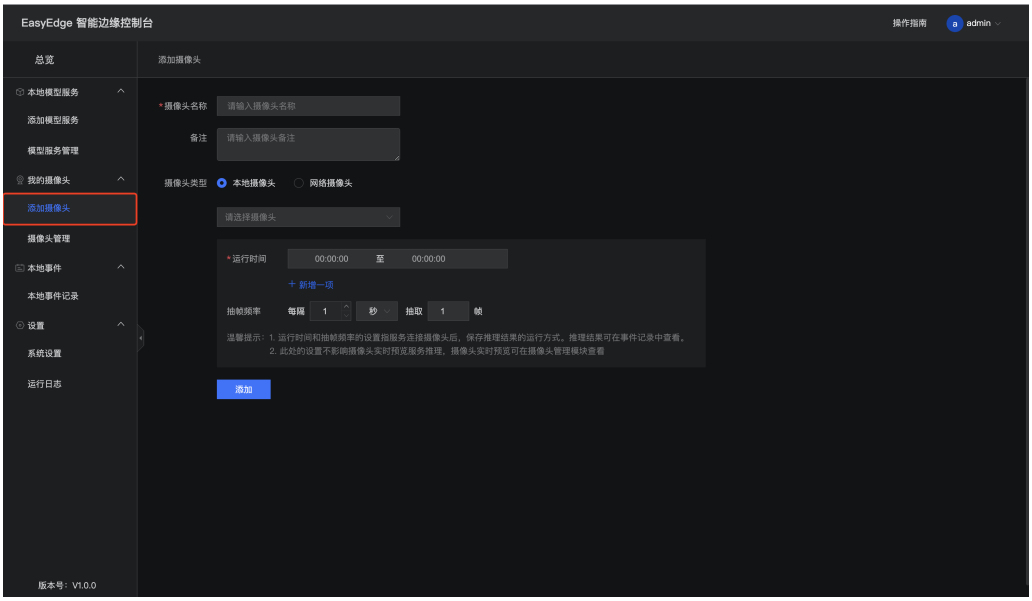


接入摄像头

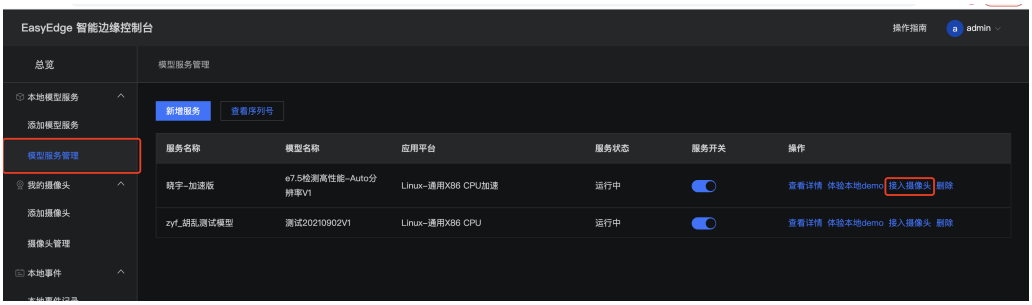
使用接入摄像头功能首先需要添加摄像头，请参考第②步，完成后按照第③步操作 **注：服务启动后也可参考「模型发布」模块的技术文档进行开发使用，本文档主要介绍IEC使用功能**



②添加摄像头 导航栏点击「我的摄像头」-「添加摄像头」，定义摄像头名称、备注后即可添加摄像头。支持本地摄像头和网络摄像头。 摄像头添加成功后即可设置摄像头的运行时间和频率



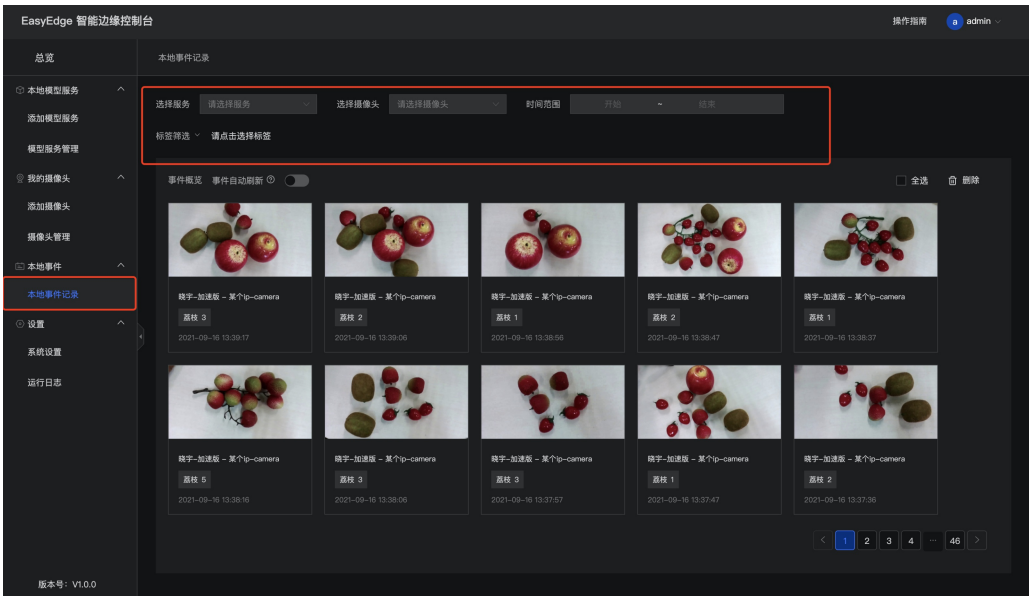
③摄像头接入模型服务预测 点击「本地模型服务」-「模型服务管理」中，所需接入预测的服务的「接入摄像头」



在弹出的弹窗中选择第②步中添加的摄像头，此时点击确认即可在「摄像头管理」中的实时预览功能中查看摄像头预测结果，识别结果默认不保存。如需保存识别结果，可设置对应的「本地事件触发条件」，根据标签和置信度，将识别结果保存至本地事件记录当中。设置多个标签条件时，IEC会以“或“的逻辑来将所有满足条件的识别结果保存



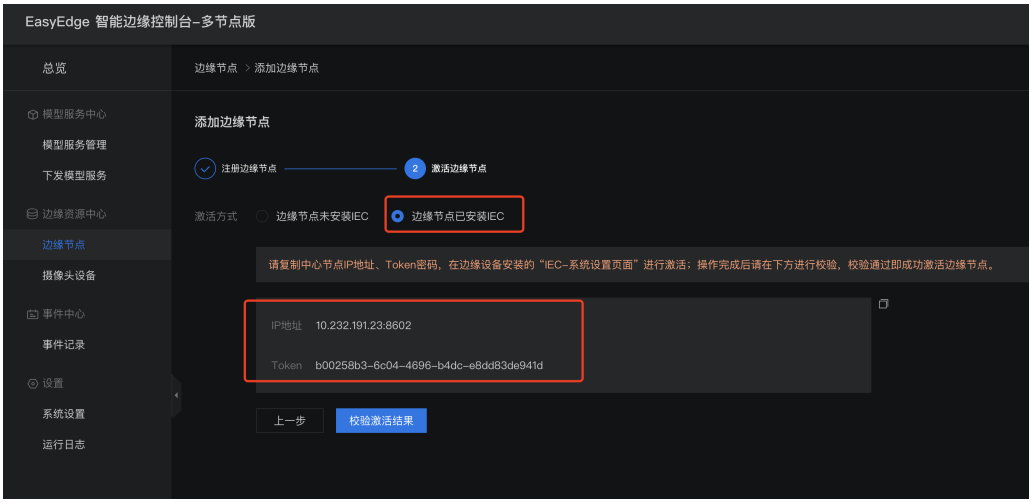
④本地事件 点击导航栏「本地事件记录」，可通过服务名称、摄像头名称、事件记录的时间、标签及置信度来筛选识别结果查看，多个标签及置信度同样也是"或"的逻辑记录。如有想要删除的事件数据可选择后删除，全选为本页全选。



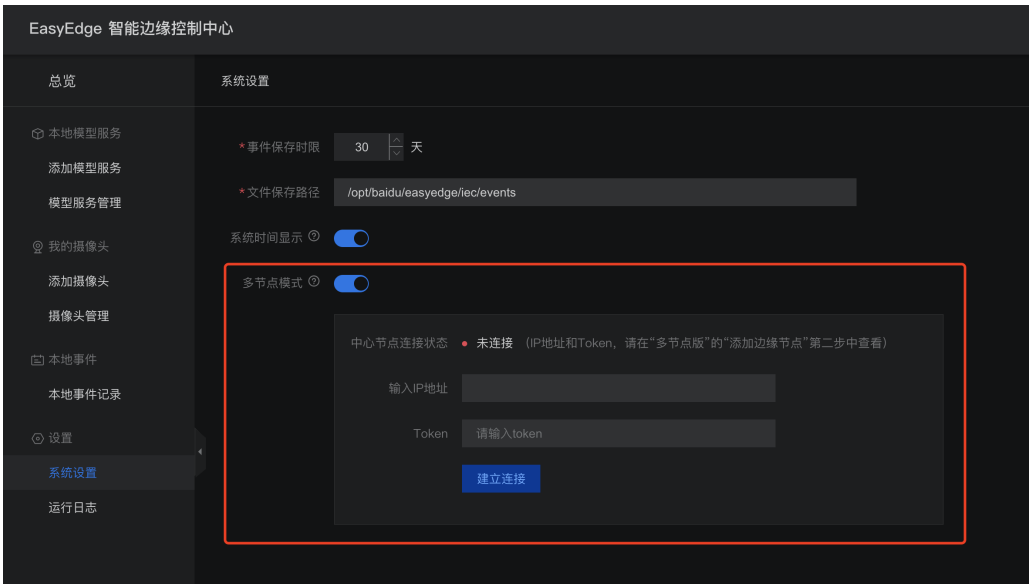
⑤ 连接到智能边缘控制台-多节点版（IECC）

> 与中心节点连接之后，边缘节点主程序版本会自动随控制中心版本升级。（>2.0.0）

- Step 1 在IECC中添加边缘节点，选择「边缘节点已安装IEC」，并记录IP地址与Token



- Step 2 在IEC的系统设置中打开多节点模式，并填入刚才记录的IP地址与Token，点击建立连接



- 连接完成后即可在中心节点IECC去监控/管理/应用在边缘节点上的IEC

配置项*

配置文件`etc/easyedge-iec.yml`中有关于IEC的各项配置说明，一般无需修改，请确保理解配置项含义之后，再做修改。

```
 ##### IEC系统配置
##### ---------- 高级配置一般无需修改 ----------
##### ！！！注意！！！ 请确保理解配置项含义后再做修改
version: 3

com:
 # hub: 作为中心节点模式启动。  edge: 作为子节点启动
 # role: edge
 # 硬件利用率刷新时间间隔： 过低的刷新间隔可能会消耗CPU资源。
 hardwareUsageRefreshSecond: 60
 # 事件监测触发扫描周期
 eventTriggerIntervalSecond: 10
 # IEC保存SDK等资源的路径：可填写 "default"，也可以直接填写绝对路径
 appDataFolder: default
 # 是否开启DEBUG模式：开启之后，将会打印大量日志，便于追踪系统问题
 debug: no
 # 是否需要用户名/密码登陆，强烈建议打开！(默认用户名/密码为 admin/easyedge)
 loginEnabled: yes
 # 登录有效时间，单位秒
 sessionMaxAge: 3600

logging:
 # 是否把日志打印到控制台
 toStd: no
 # 是否把日志输出到文件。关闭后，将无法在页面中正确显示系统日志
 toFile: yes
 loggingFile: /var/log/easyedge-iec/easyedge-iec.log
 # 0:info;  -1:debug;  -2:verbose
 level: -1

webservice:
 # WEB服务的监听端口
 listenPort: 8702
 listenHost: 0.0.0.0

sdk:
 # GPU SDK所使用的cuda版本：9 / 10 / 10.2 / 11.0 / 11.1。请安装完cuda之后，这设置正确的版本号。
 cudaVersion: 10.2
 # AI服务启动时，额外配置的 LD_LIBRARY_PATH(linux) 或者 PATH(windows)
 libPath: ./
 # AI服务启动时，额外配置的其他环境变量。
 ENVs:
   EDGE_CONTROLLER_KEY_LOG_BRAND: EasyEdge
##### EDGE_CONTROLLER_KEY_XXX: XXXX
```

```
commu:
  # 普通消息等待respond的超时时间
  respondWaitTimeoutSecond: 2

##### 数据库相关配置
db:
  sqliteDbFile: /var/lib/easyedge-iec/easyedge-iec.db
  hubDbFile: /var/lib/easyedge-iec/easyedge-iec.hub.db
  eventDbFile: /var/lib/easyedge-iec/easyedge-event.db
  fileServerDbFile: /var/lib/easyedge-iec/easyedge-fileserver.hub.db
  nodeMonitorDbFile: /var/lib/easyedge-iec/easyedge-nodemonitor.hub.db

##### 推流相关配置
mediaserver:
  flvPort: 8715
  rtmpPort: 8716

##### 视频流相关配置
edgestream:
  logLevel: -1
  listenHost: 127.0.0.1
  listenPort: 8710
  # 摄像头预览：识别结果绘制延迟消失
  renderExtendFrames: 10
  # 预测队列大小: 如果设置为60，当摄像头fps=30时，视频延迟约为2秒。 降低inferenceQueueSize可以降低预览延迟，但是根据硬件的算力情况，可能导致模型推理速度跟不上，没有识别结果，不建议设置太低
  inferenceQueueSize: 60
  videoEncodeBitRate: 400000
  # 视频采样 & 视频实时预览分辨率设置
  # 0: auto, 1: 1080p, 2: 720p, 3: 480p, 4: 360p,  5: 240p
  resolution: 0
  # 内置多媒体服务配置
  # port设为0表示关闭
  mediaServerHost: 127.0.0.1
  mediaServerFlvPort: 8713
  mediaServerRtmpPort: 8714
  mediaServerRtspPort: 0
```

FAQ

启动服务后，进程中出现两个easyedge-iec进程 这是正常现象，IEC通过守护进程的方式来完成更新等操作。

启动服务时，显示端口被占用port already been used 通过修改 easyedge-iecc.yml文件的配置后，再重新启动服务。

安装服务时，报错permission denied 请以管理员身份运行安装程序。

中心节点重启后，边缘节点IEC 一直离线 中心节点短时间的离线，边缘节点会自动重连。如果中心节点已经恢复在线，边缘节点长时间未自动连接上，可通过边缘节点iec的方法来重新连接（右上角 admin - 重启系统）

IEC 是否有Android / iOS 版本 我们将会在近期发布对Android操作系统的支持

添加SDK时，报错 SDK不支持该硬件。 SDK not supported by this device 一般是因为使用的SDK跟硬件不匹配，如 GPU的SDK，硬件没有GPU卡。 对于Jetson，也可能是Jetpack版本不支持，可以通过查看 本机Jetpack版本和SDK支持的Jetpack版本列表（cpp文件中的文件名来查看）来匹配。

智能边缘控制台-多节点版

EasyEdge 智能边缘控制台——多节点版

整体介绍

智能边缘控制台 - 多节点版（EasyEdge Intelligent EdgeConsole Center 以下简称IECC），是EasyEdge推出的边缘资源管理、服务应用与管理一站式本地化方案。

通过IECC，用户可以方便地在中心节点管理子节点：

- 边缘硬件资源的管理与监控

- EasyDL/BML/EasyEdge的SDK的 离线 / 在线激活，服务管理
- 视频流解析，接入本地和远程摄像头，网页中实时预览
- 自动监控和记录相关视频流推理事件

支持的系统+CPU架构包括：

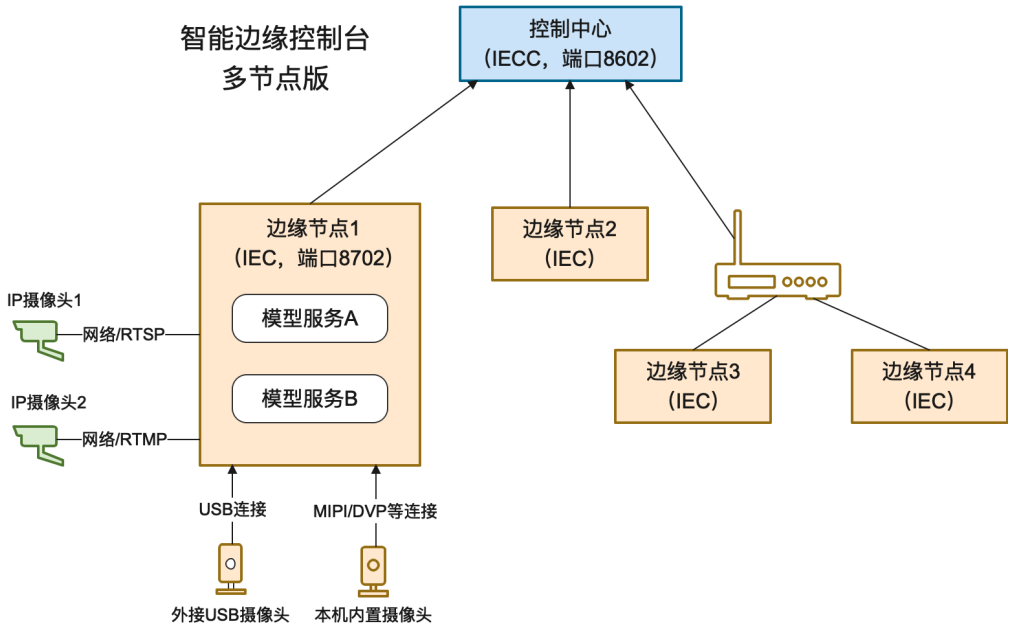
- Windows x86_64 (Windows 7 ~ Windows 10，暂不支持Windows 11)
- Linux x86_64 / arm32 / arm64

支持各类常见的AI加速芯片，包括：

- NVIDIA GPU / Jetson 系列
- Baidu EdgeBoard FZ系列
- 比特大陆 Bitmain SC / SE 系列
- 华为 Atlas 系列
- 寒武纪 MLU 系列
- 其他EasyDL/EasyEdge/BML支持的AI芯片

连接说明

以下为 中心节点（控制中心），边缘节点/子节点，摄像头的连接示意：



其中：

- 控制中心需要有固定IP，而边缘节点可以处于多级子网之下，只需IEC能够主动访问到控制中心节点即可
- 模型服务均运行于各边缘节点之上
- 摄像头均与边缘节点相连

Release Note

版本号	发布时间	更新说明
2.2.0	2022-10-27	边缘节点新增Android支持；新增onvif/gb28181支持；优化端云通信通道安全
2.0.0	2022-03-25	多节点版上线！
1.0.2	2021-12-22	更新视频预览推流库；新增若干AI芯片支持；支持多种芯片温度、功耗展示；多项性能优化
1.0.0	2021-09-16	智能边缘控制台 - 单节点版 IEC 第一版！

安装 从这里选择您需要的操作系统和CPU架构下载：

- Windows amd64 ： intel、AMD的64位x86_84 CPU
- Linux amd64：intel、AMD的64位x86_84 CPU
- Linux arm：树莓派等32位的ARM CPU
- Linux arm64：RK3399、飞腾等64位的ARM CPU

或者从纯离线服务管理页可下载智能边缘控制台



以Linux为例，解压缩后目录结构如下所示：

```
./EasyEdge-IECC-v{版本号}/
|-- easyedge-iecc
|-- easyedge-iecc-setup.sh
|-- etc/
|-- etc/easyedge-iec.yml
|-- readme.txt
```

Linux 系统

通过系统服务形式安装（推荐）

以管理员运行`bash easyedge-iecc-setup.sh install`即可。

```
 0 EasyEdge-IEC-v2.0.0 > bash ./easyedge-iecc-setup.sh install
[setup]: sudo could not be found
[setup]: Start to install IECC...
[setup]: +  bash -c "cp easyedge-iec-linux-amd64 /usr/sbin/easyedge-iecc"
[setup]: +  bash -c "chmod +x /usr/sbin/easyedge-iecc"
[setup]: +  bash -c "cp easyedge-iec-* /var/lib/easyedge-iecc/fs/tmp"
[setup]: +  bash -c "cp etc/easyedge-iecc.service.yml /etc/easyedge-iecc/easyedge-iecc.yml"
[setup]: +  bash -c "cp etc/easyedge-iecc.service-conf.init.d /etc/init.d/easyedge-iecc"
[setup]: +  bash -c "chmod +x /etc/init.d/easyedge-iecc"
[setup]: Install IECC success!
[setup]: +  bash -c "service easyedge-iecc start"
Starting easyedge-iecc: success
[setup]: Start to check IECC status...
[setup]: +  bash -c "curl -s 127.0.0.1:8702 >/dev/null"
[setup]: IECC status: OK!
[easyedge-iecc]: default configure file: /etc/easyedge-iecc/easyedge-iecc.yml
[easyedge-iecc]: default log file: /var/log/easyedge-iecc/easyedge-iecc.log
[easyedge-iecc]: service usage: service easyedge-iecc { start | stop }
[setup]: Done!
```

出现sucess字样，表示安装成功。

- 日志：`/var/log/easyedge-iecc/easyedge-iecc.log`
- 系统配置：`/etc/easyedge-iecc/easyedge-iecc.yml`
- 服务启动/停止：`service easyedge-iecc { start | stop }` (不同操作系统内可能不同，具体命令参考安装日志)
- 配置服务自启动：可根据不同操作系统参考这里进行对应配置

> 可通过 `bash easyedge-iecc-setup.sh uninstall` 来卸载，以及`bash easyedge-iecc-setup.sh upgrade`来升级为当前安装包的版本

自定义安装（不推荐）

自定义安装仅限于 安装脚本无法识别您的操作系统的情况。

- 拷贝 `./EasyEdge-IEC-v2.0.0/` 整个目录至自定义文件夹，如`/opt/EasyEdge-IEC`
- 进入到 `/opt/EasyEdge-IEC`
- 通过 nohup 等方法运行 `./easyedge-iec-linux-{您的系统架构} --com.role=hub` *amd64: intel、AMD的64位x86_84 CPU* arm：树莓派等32位的ARM CPU * arm64：RK3399、飞腾等64位的ARM CPU
- 日志： `./log/easyedge-iecc.log`
- 系统配置：`./easyedge-iecc.yml`

Windows 系统 打开命令行（非powershell）运行 `easyedge-iecc-setup.bat install`。

> 注：如果遇到hang住的情况，可修改命令行配置
>
> 
>
>
> 
>

验证安装： 启动之后，打开浏览器，访问 `http://{设备ip}:8602/easyedge` 即可：



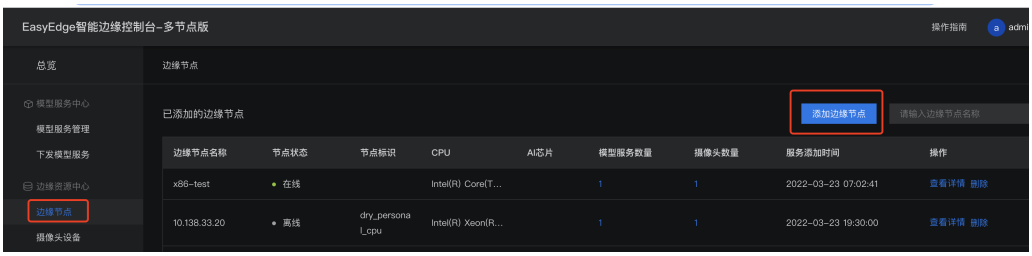
更新服务： 关闭服务，下载最新的安装包，重新执行安装流程即可。

> 注：1. 中心节点更新到新版之后，已连接的边缘节点会自动跟随中心节点，自我升级到同样的版本。
>
> 2. 报错: Text file busy. 一般是因为服务没有停止。

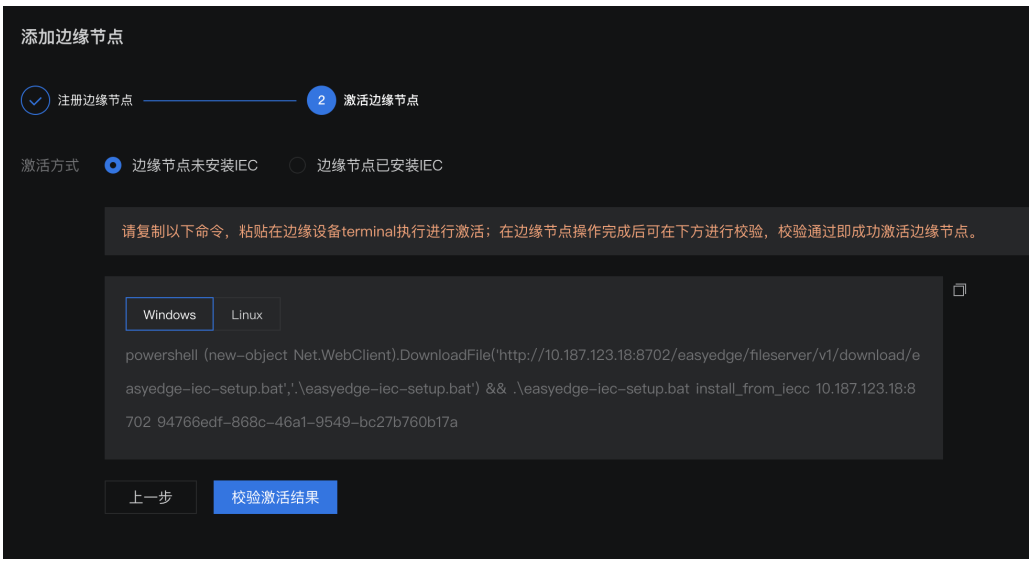
使用流程

Step 1 注册并激活边缘节点

- 在IECC导航栏中点击边缘节点，点击页面中的添加边缘节点按钮



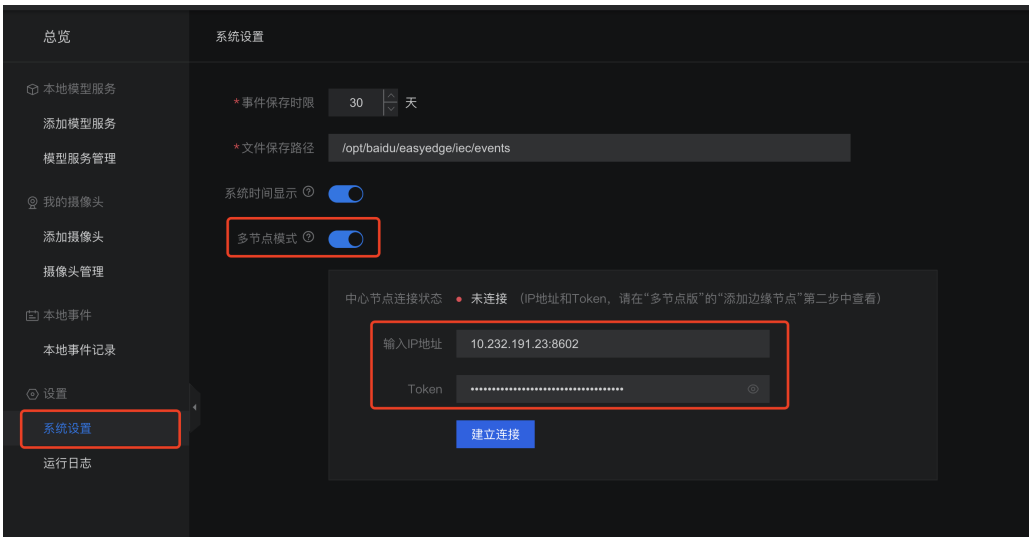
- 注册边缘节点，填写基本信息
- 激活边缘节点，根据边缘节点上是否安装智能边缘控制台-单节点版（IEC）分两种激活方式
  - 边缘节点未安装IEC：复制提供的命令，在边缘节点的终端中输入执行（命令会自动在当前目录，下载单节点版IEC并注册到控制中心）。终端命令执行完成后，在下方校验激活结果，如结果通过即可完成边缘节点的激活



  - 边缘节点已安装IEC：记录页面中提供的IP地址和Token



- 在**边缘节点的IEC**-系统设置中，打开**多节点模式**开关，将刚才记录的IP地址和Token填入其中，建立连接



- 成功激活后可在边缘节点页面中看到一行状态为**在线**的记录



Step 2 上传并下发模型服务

- 在模型服务管理-已添加的模型服务页面中点击**添加模型服务**



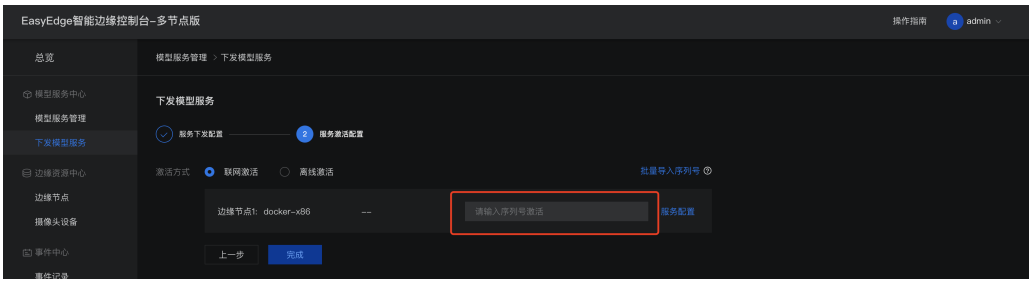
- 上传来自于EasyDL/BML的SDK，目前仅支持Windows/Linux的SDK



- 添加成功后可在已添加的模型服务页面查看添加的模型服务SDK
- 在模型服务SDK上传成功以及边缘节点也添加激活过后，即可将模型服务下发至边缘。点击导航栏-下发模型服务，选择已添加的模型服务，选择下发的目标节点（支持多节点批量下发）进行模型服务下发



- 确定下发配置后，填入模型服务在边缘节点联网激活运行的序列号（支持批量导入）即可完成模型服务下发，序列号可在智能云控制台获取。离线激活的过程可参考IECC中的具体指引



- 完成上述流程后即可在模型服务管理-已下发的模型服务列表中查看记录，并进行下一步应用功能体验

> 注：完成此步骤后即可在边缘节点进行二次集成已下发的模型服务，具体的集成方式可在文档-某图像任务类型-模型发布中查找对应的SDK开发文档进行集成开发

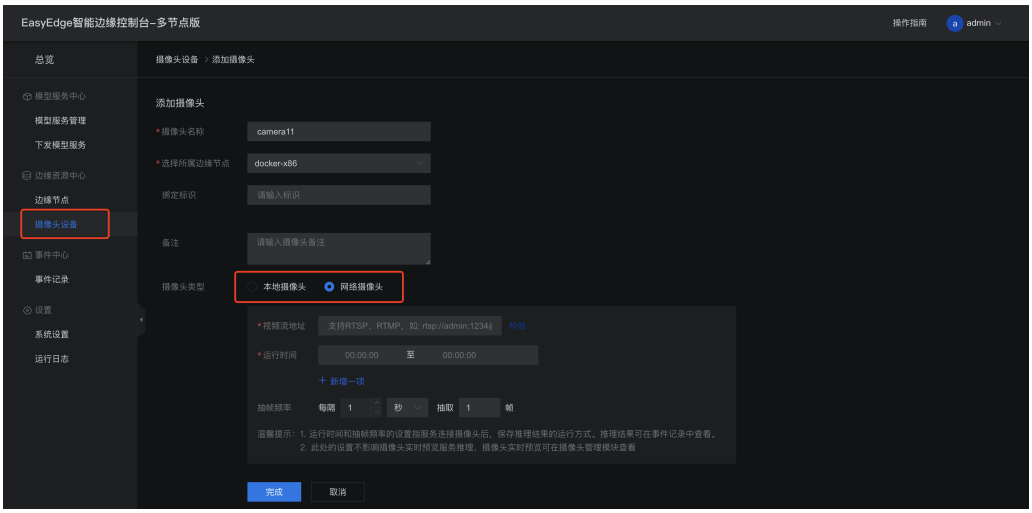


> 下发时可以通过高级配置设置服务运行的host和port。若不设置，默认host为0.0.0.0，port为系统随机分配的可用端口

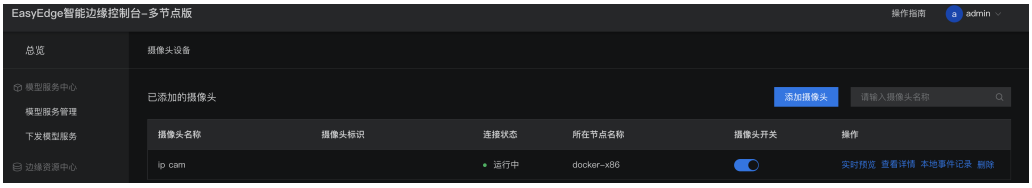
Step 3 配置摄像头

> Step 3 - 5 描述的是如何使用IECC可视化进行视频流式推理与应用，对此有需求的用户建议详细查看后续步骤内容。如仅需对下发的模型服务进行二次集成的用户无需进行后续操作，参考SDK对应的开发文档进行集成即可

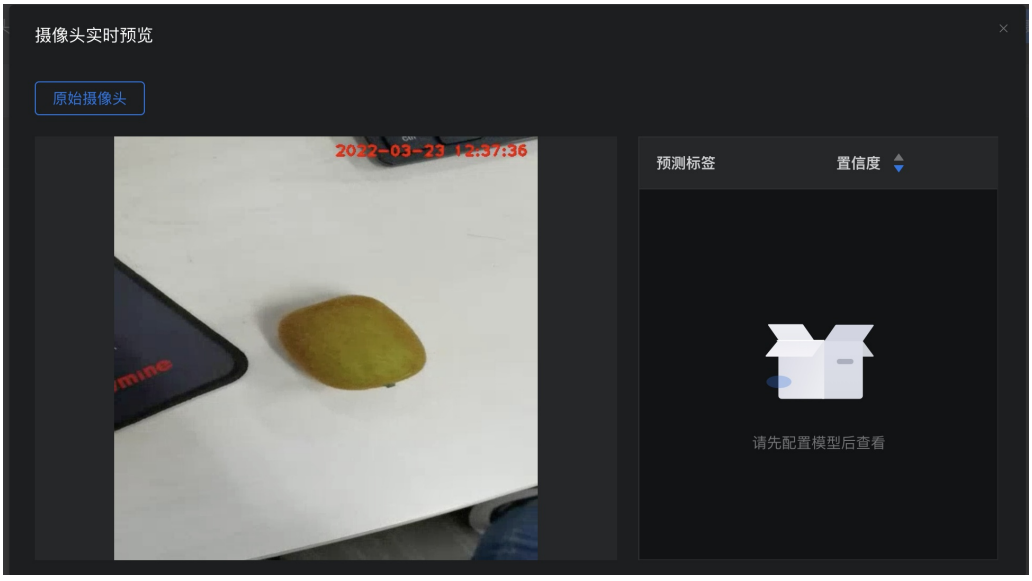
- 首先需要确定边缘节点已经接入物理摄像头，可通过USB插口接入，也可通过RTSP/RTMP流式协议接入。在摄像头设备页面点击添加摄像头按钮，填写对应的信息添加摄像头。支持设置摄像头的运行时间以及摄像头的抽帧频率



- 添加完成后可在摄像头设备页面查看记录



- 点击预览可查看摄像头预览画面



Step 4 模型服务接入视频流预测

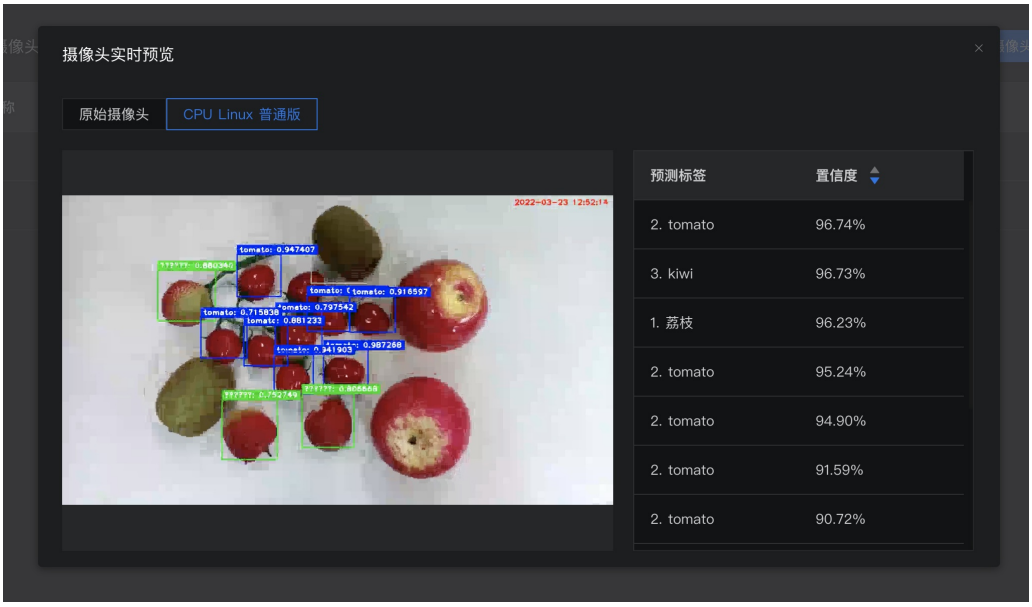
- 模型服务可接入摄像头直接进行预测，并可同时设置告警规则，出发告警条件的结果将会以事件的形式保存至IECC中。点击模型服务管理页面中对应服务的接入摄像头操作



- 将已添加至IECC的摄像头与模型服务关联，并在下方设置对应的事件告警条件。告警规则通过标签阈值的方式来建立，例如设置“猕猴桃”标签阈值80%-100%，则大于80%置信度的“猕猴桃”识别结果将会保存至事件记录中

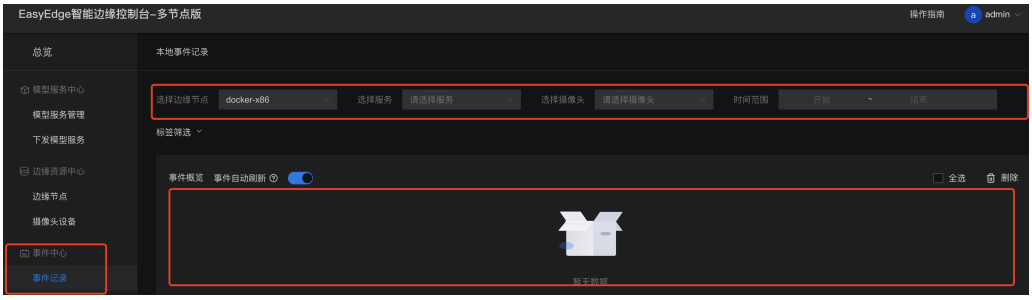


- 也可在摄像头设备页面-实时预览中查看实时的模型服务预测结果



Step 5 视频事件告警

- 可在事件中心-事件记录中查看满足时间告警条件的图片记录



高级配置说明

在**系统设置** - **高级**，可以修改控制中心的高级系统配置

```
 ##### IECc系统配置
version: 3

com:
  # hub: 作为中心节点模式启动。  edge: 作为子节点启动
  role: hub
  # 硬件利用率刷新时间间隔： 过低的刷新间隔可能会消耗CPU资源。
  hardwareUsageRefreshSecond: 60
  # IECC保存SDK等资源的路径：可填写 "default", 也可以直接填写绝对路径
  appDataFolder: default
  # 是否开启DEBUG模式：开启之后，将会打印大量日志，便于追踪系统问题
  debug: no
  # 是否需要用户名/密码登陆，强烈建议打开！(默认用户名/密码为 admin/easyedge)
  loginEnabled: yes
  # 登录有效时间，单位秒
  sessionMaxAge: 3600

logging:
  # 是否把日志打印到控制台
  toStd: no
  # 是否把日志输出到文件。关闭后，将无法在页面中正确显示系统日志
  toFile: yes
  loggingFile: /var/log/easyedge-iecc/easyedge-iecc.log
  # 0:info;  -1:debug;  -2:verbose
  level: -1

webservice:
  # WEB服务的监听端口
  listenPort: 8602
  listenHost: 0.0.0.0

commu:
  mqServer:
    host: 0.0.0.0
    port: 8632
    HTTPPort: 8620
    maxPayload: 8388608
    pingIntervalSecond: 30
  # 普通消息等待respond的超时时间
  respondWaitTimeoutSecond: 2
  nodeRefreshIntervalSecond: 30


##### ---------- 以下高级配置一般无需修改 ----------
##### ！！！注意！！！ 请确保理解配置项含义后再做修改
##### 数据库相关配置
db:
  sqliteDbFile: /var/lib/easyedge-iecc/easyedge-iecc.db
  hubDbFile: /var/lib/easyedge-iecc/easyedge-iecc.hub.db
  eventDbFile: /var/lib/easyedge-iecc/easyedge-event.db
  fileServerDbFile: /var/lib/easyedge-iecc/easyedge-fileserver.hub.db
  nodeMonitorDbFile: /var/lib/easyedge-iecc/easyedge-nodemonitor.hub.db

##### 推流相关配置
mediaserver:
  flvPort: 8613
  rtmpPort: 8614

##### 文件服务器相关配置
fileserver:
  root: /var/lib/easyedge-iecc/fs
```

FAQ

启动服务后，进程中出现两个easyedge-iec进程 这是正常现象，IEC通过守护进程的方式来完成更新等操作。

启动服务时，显示端口被占用port already been used 通过修改 easyedge-iecc.yml文件的配置后，再重新启动服务。

安装服务时，报错permission denied 请以管理员身份运行安装程序。

添加SDK时，报错 SDK不支持该硬件。 SDK not supported by this device 一般是因为使用的SDK跟硬件不匹配，如 GPU的SDK，硬件没有GPU卡。 对于Jetson，也可能是Jetpack版本不支持，可以通过查看 本机Jetpack版本和SDK支持的Jetpack版本列表（cpp文件中的文件名来查看）来匹配。

NLP模型部署

自然语言处理任务模型部署整体说明

训练完成后，可将模型部署在公有云服务器上，通过API进行调用。

公有云API

训练完成的模型存储在云端，可通过独立Rest API调用模型，实现AI能力与业务系统或硬件设备整合

具有完善的鉴权、流控等安全机制，GPU集群稳定承载高并发请求

支持查找云端模型识别错误的数据，纠正结果并将其加入模型迭代的训练集，不断优化模型效果

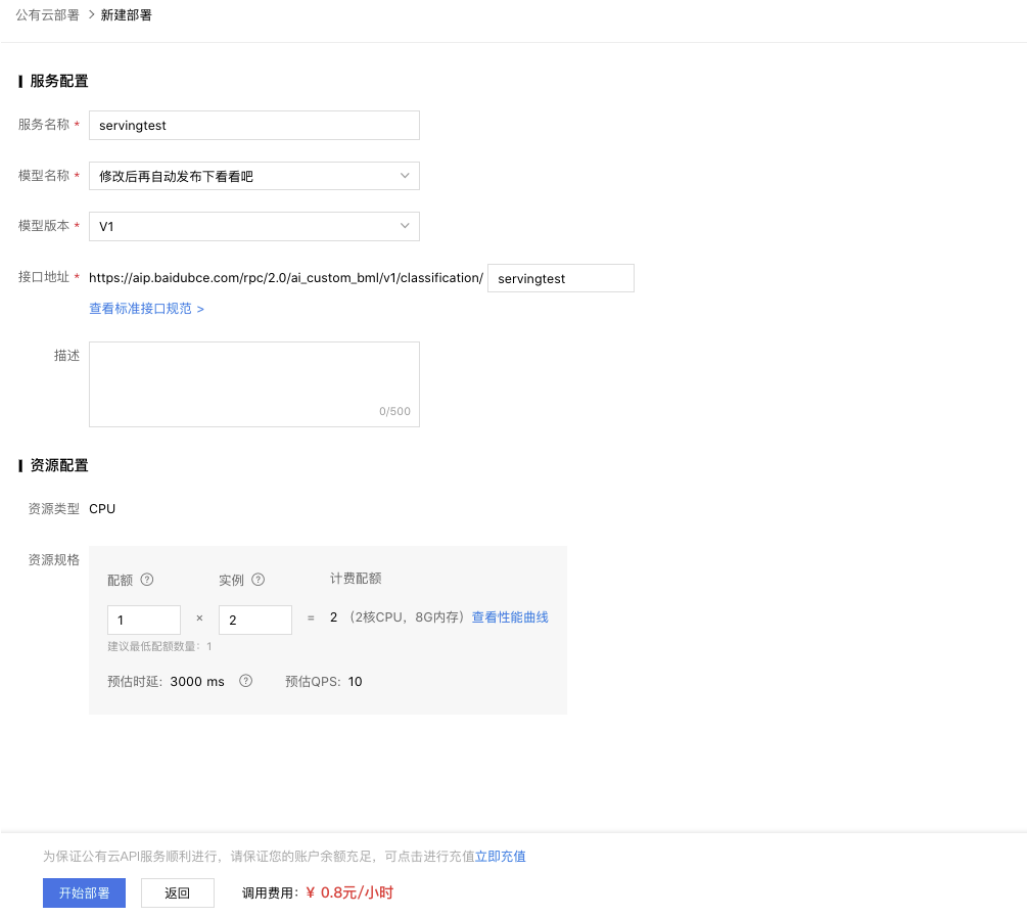
公有云部署

如何发布自然语言处理任务API

模型仓库中的NLP模型，发布为公有云部署时运行在云端，可通过独立Rest API调用模型，实现AI能力与业务系统或硬件设备整合。
BML具有完善的鉴权、流控等安全机制，并配置丰富的资源集群稳定承载高并发请求。 并且支持查找云端模型识别错误的数据，纠正结果并将其加入模型迭代的训练集，不断优化模型效果

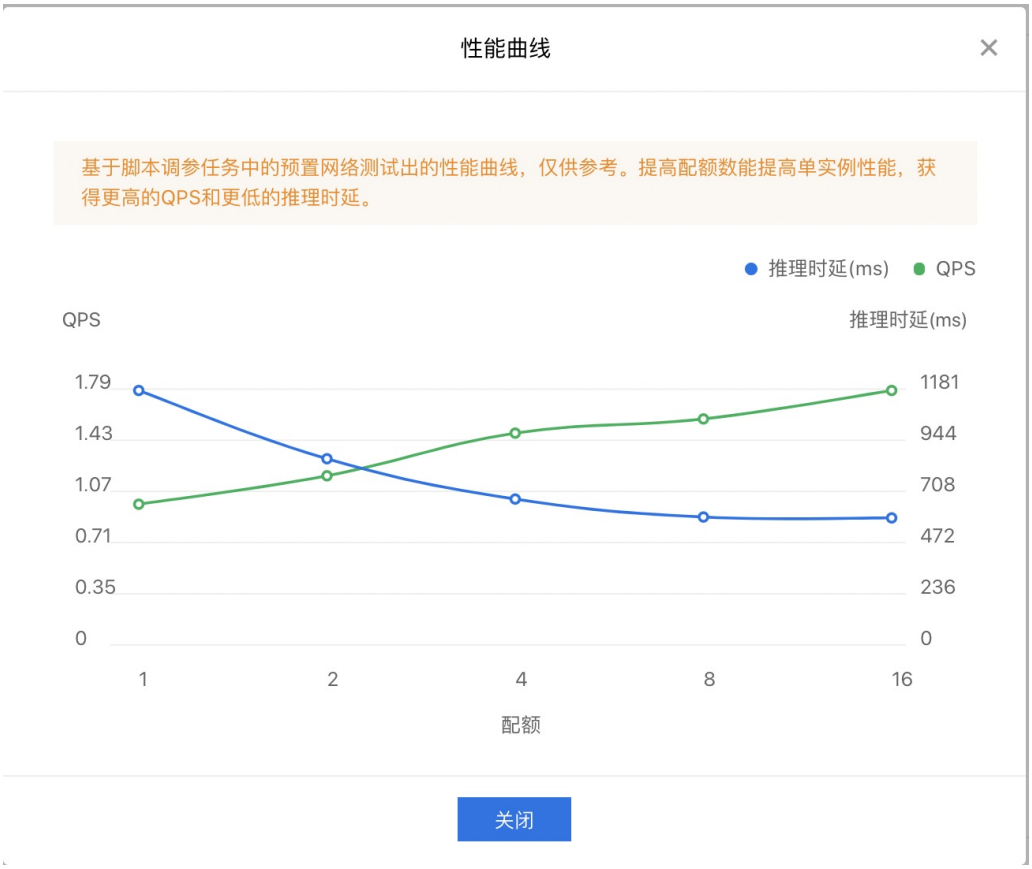
公有云部署 模型部署流程

1. 将训练完成的模型发布到模型仓库，在模型仓库中的『版本列表』中点击『公有云部署』或者点击左侧导航栏的公有云部署，进入公有云部署界面后点击『部署模型』。
2. 在部署界面，首先进行服务配置：自定义服务名称、接口地址后缀，选择需要部署的模型，
3. 对每个部署模型还需要进行资源配置，配置模型预测服务所需机器资源。目前资源开放CPU类型，支持按配额与实例进行灵活配置，计费说明请点击查看



4. 当部署模型为脚本调参任务产出的模型时，系统会自动显示该模型在当前配置下的预估时延和QPS。点击"性能曲线"，即可查看**单实例**下的推

理时延，QPS与配额的曲线关系，您可以根据性能曲线配置合理的配额。

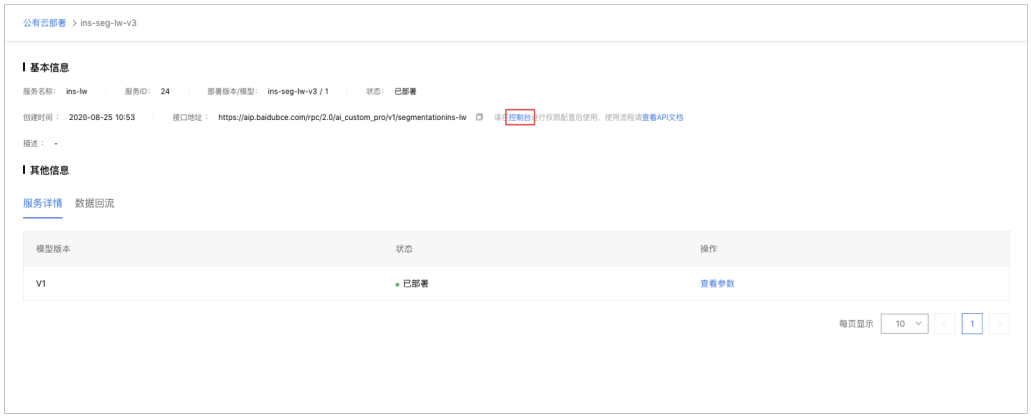


5. 配置完成后，点击开始部署即可一键部署模型，并获得专用的API地址。

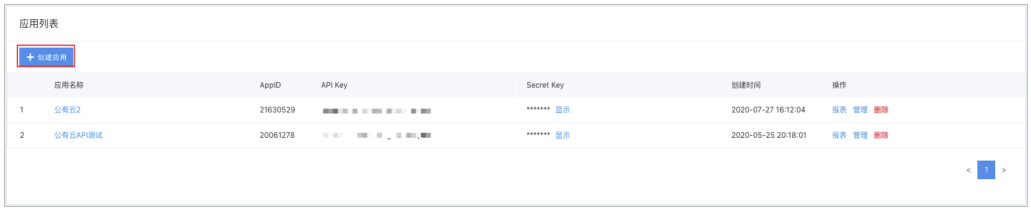
> 说明：
>
> 1、系统将运行公有云部署的机器资源1核cpu和4G内存折算为1配额，提高配额数量能提升QPS和降低推理时延
>
> 2、实例为cpu资源运行的实例数，为保证服务的稳定性，实例数至少需要配置为2。实例数与QPS成正比关系，提高实例数时，QPS能正比增加

接口赋权

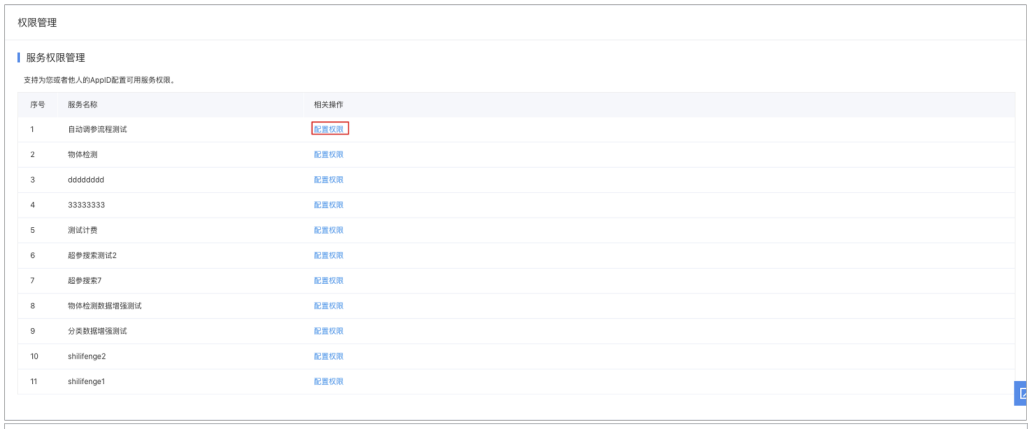
1. 正式使用还需要对接口进行赋权，在已发布的服务的详情界面，点击『控制台』进入BML控制台。

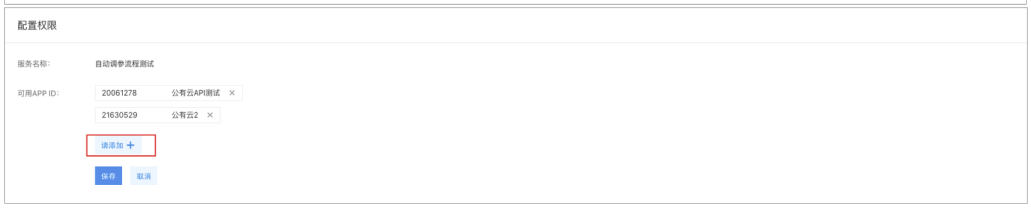


2. 在控制台中创建一个应用，获得由一串数字组成的appid，然后就可以参考接口文档正式使用了



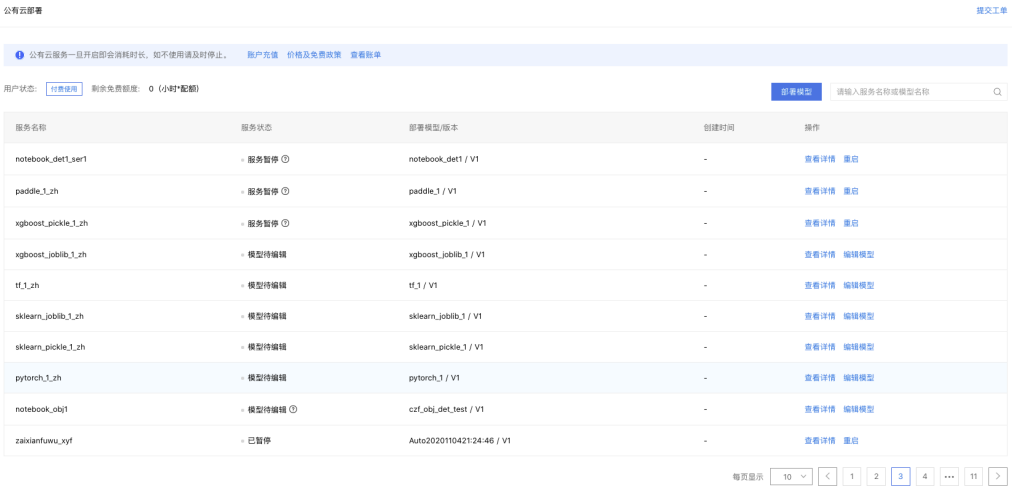
3. 如需为第三方用户配置调用接口的权限，可以前往BML控制台-公有云服务管理-权限管理配置





公有云部署列表

公有云部署列表信息由历史提交公有云服务组成，显示服务的基本信息。



如图，在预测服务列表页支持以下操作：

查看详情：点击进入该公有云服务的详情界面

重启：对已停止或部署失败的服务进行重新创建，恢复对应的服务为可用状态

停止：停止该公有云服务，用户不再使用相关服务后可进行这一操作，停止后将不进行计费

编辑模型：对于已经部署为公有云服务，但尚未配置机器资源的模型，点击可以配置机器资源

文本分类-单标签API调用文档

本文档主要说明定制化模型发布后获得的API如何使用，如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动
- 加入BML官方QQ群（群号:868826008）联系群管

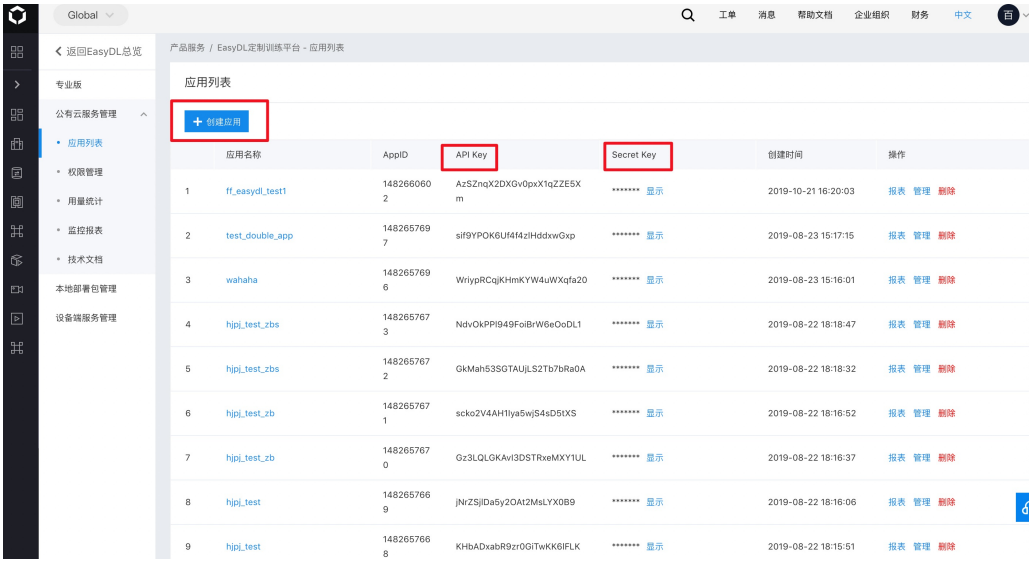
接口描述

基于自定义训练出的文本分类模型，实现个性化文本分类。模型训练完毕后发布可获得定制化文本分类API

接口鉴权

1、在BML——控制台创建应用

2、应用列表页获取AK SK



请求说明

请求示例

HTTP 方法：`POST`

请求URL： 请首先进行自定义模型训练，完成训练后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

注意：如果出现336001的错误码很可能是因为请求方式错误，需以json方式请求。

Body请求示例：

```
{
  "text": "<UTF-8编码数据>",
  "top_num": 5
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
text	是	string	-	文本数据，UTF-8编码。最大长度4096汉字
top_num	否	number	-	返回分类数量，默认为6个

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
results	否	array(object)	分类结果数组
+name	否	string	分类名称
+score	否	number	置信度

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如Access Token失效返回：

```
 {
  "error_code": 110,
  "error_msg": "Access token invalid or no longer valid"
}
```

需要重新获取新的Access Token再次请求即可。

错误码	错误信息	描述
1	Unknown error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
2	Service temporarily unavailable	服务暂不可用，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
3	Unsupported openapi method	调用的API不存在，请检查后重新尝试
4	Open api request limit reached	集群超限额
6	No permission to access data	无权限访问该用户数据
13	Get service token failed	获取token失败
14	IAM Certification failed	IAM鉴权失败
15	app not exists or create failed	应用不存在或者创建失败
17	Open api daily request limit reached	每天请求量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
18	Open api qps request limit reached	QPS超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
19	Open api total request limit reached	请求总量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
100	Invalid parameter	无效的access_token参数，请检查后重新尝试
110	Access token invalid or no longer valid	access_token无效
111	Access token expired	access token过期
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（868826008）或工单联系技术支持团队
336001	Invalid Argument	入参格式有误，比如缺少必要参数、文本编码错误等等，可检查下文本编码、代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数或者代码格式有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336004	输入文件大小不合法	文本超出大小限制，文本限4M以内，请根据接口文档检查入参格式，有疑问请通过QQ群（868826008）或工单联系技术支持团队
336005	文本解码失败	文本编码错误，请检查并修改文本格式
336006	缺失必要参数	text字段缺失（未上传文本）
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请通过QQ群（868826008）或工单联系技术支持团队

文本分类-多标签API调用文档

本文档主要说明定制化模型发布后获得的API如何使用，如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动
- 加入BML官方QQ群（群号:868826008）联系群管

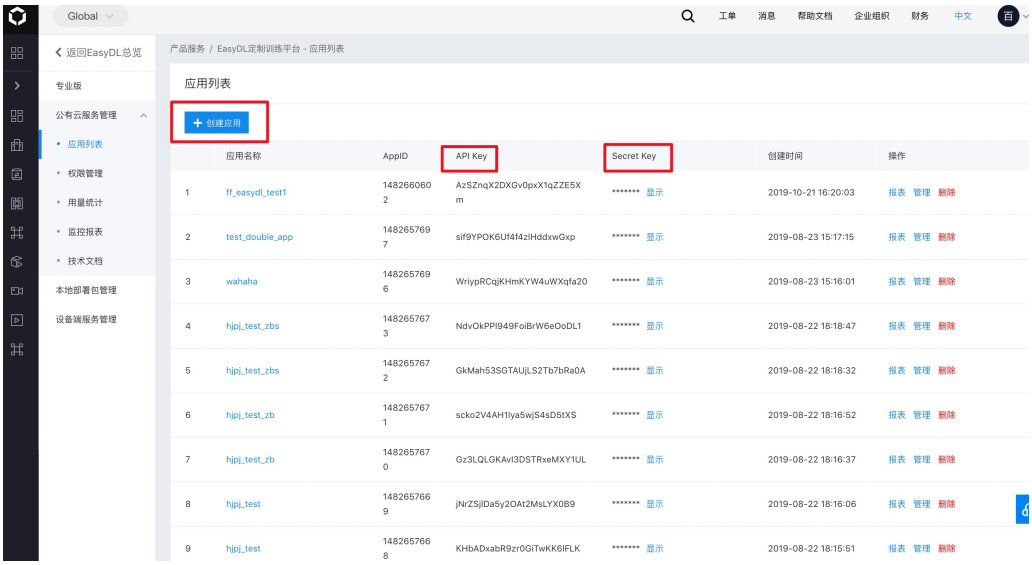
接口描述

基于自定义训练出的文本分类多标签模型，实现个性化文本分类。模型训练完毕后发布可获得定制化文本分类API

接口鉴权

1、在BML——控制台创建应用

2、应用列表页获取AK SK



请求说明

请求示例

HTTP 方法：POST

请求URL： 请首先进行自定义模型训练，完成训练后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

注意：如果出现336001的错误码很可能是因为请求方式错误，需以json方式请求。

Body请求示例：

```
{
  "text": "<UTF-8编码数据>",
  "threshold": 6
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
text	是	string	-	文本数据，UTF-8编码。最大长度4096汉字
threshold	否	number	-	对识别的文本标签进行阈值条件的筛选

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
results	否	array(object)	分类结果数组
+name	否	string	分类名称
+score	否	number	置信度

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如Access Token失效返回：

```
 {
  "error_code": 110,
  "error_msg": "Access token invalid or no longer valid"
}
```

需要重新获取新的Access Token再次请求即可。

错误码	错误信息	描述
1	Unknown error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
2	Service temporarily unavailable	服务暂不可用，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
3	Unsupported openapi method	调用的API不存在，请检查后重新尝试
4	Open api request limit reached	集群超限额
6	No permission to access data	无权限访问该用户数据
13	Get service token failed	获取token失败
14	IAM Certification failed	IAM鉴权失败
15	app not exists or create failed	应用不存在或者创建失败
17	Open api daily request limit reached	每天请求量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
18	Open api qps request limit reached	QPS超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
19	Open api total request limit reached	请求总量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
100	Invalid parameter	无效的access_token参数，请检查后重新尝试
110	Access token invalid or no longer valid	access_token无效
111	Access token expired	access token过期
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（868826008）或工单联系技术支持团队
336001	Invalid Argument	入参格式有误，比如缺少必要参数、文本编码错误等等，可检查下文本编码、代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数或者代码格式有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336004	输入文件大小不合法	文本超出大小限制，文本限4M以内，请根据接口文档检查入参格式，有疑问请通过QQ群（868826008）或工单联系技术支持团队
336005	文本解码失败	文本编码错误，请检查并修改文本格式
336006	缺失必要参数	text字段缺失（未上传文本）
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请通过QQ群（868826008）或工单联系技术支持团队

短文本匹配API调用文档

本文档主要说明定制化模型发布后获得的API如何使用，如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动
- 加入BMLL官方QQ群（群号:868826008）联系群管

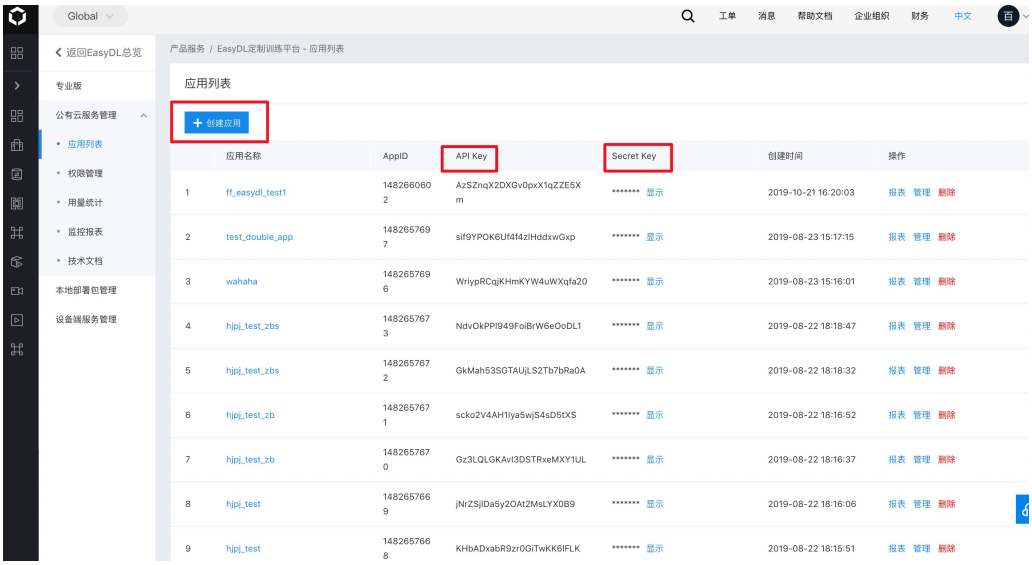
接口描述

基于自定义训练出的短文本匹配模型，实现个性化短文本相似度计算。模型训练完毕后发布可获得定制API

接口鉴权

1、在BML——控制台创建应用

2、应用列表页获取AK SK



请求说明

请求示例

HTTP 方法：`POST`

请求URL： 请首先进行自定义模型训练，完成训练后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

注意：如果出现336001的错误码很可能是因为请求方式错误，需以json方式请求。

Body请求示例：

```
{
  "text_a": "<UTF-8编码数据>",
  "text_b": "<UTF-8编码数据>"
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
text_a	是	string	-	文本数据，UTF-8编码。最大长度4096汉字
text_b	是	string	-	文本数据，UTF-8编码。最大长度4096汉字

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
score	是	number	文本相似度，从0-1，相似度递增

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如Access Token失效返回：

```
{
 "error_code": 110,
 "error_msg": "Access token invalid or no longer valid"
}
```

需要重新获取新的Access Token再次请求即可。

错误码	错误信息	描述
1	Unknown error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
2	Service temporarily unavailable	服务暂不可用，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
3	Unsupported openapi method	调用的API不存在，请检查后重新尝试
4	Open api request limit reached	集群超限额
6	No permission to access data	无权限访问该用户数据
13	Get service token failed	获取token失败
14	IAM Certification failed	IAM鉴权失败
15	app not exists or create failed	应用不存在或者创建失败
17	Open api daily request limit reached	每天请求量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
18	Open api qps request limit reached	QPS超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
19	Open api total request limit reached	请求总量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
100	Invalid parameter	无效的access_token参数，请检查后重新尝试
110	Access token invalid or no longer valid	access_token无效
111	Access token expired	access token过期
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（868826008）或工单联系技术支持团队
336001	Invalid Argument	入参格式有误，比如缺少必要参数、文本编码错误等等，可检查下文本编码、代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数或者代码格式有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336004	输入文件大小不合法	文本超出大小限制，文本限4M以内，请根据接口文档检查入参格式，有疑问请通过QQ群（868826008）或工单联系技术支持团队
336005	文本解码失败	文本编码错误，请检查并修改文本格式
336006	缺失必要参数	text字段缺失（未上传文本）
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请通过QQ群（868826008）或工单联系技术支持团队

序列标注API调用文档

本文档主要说明定制化模型发布后获得的API如何使用，如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动
- 加入BML官方QQ群（群号:868826008）联系群管

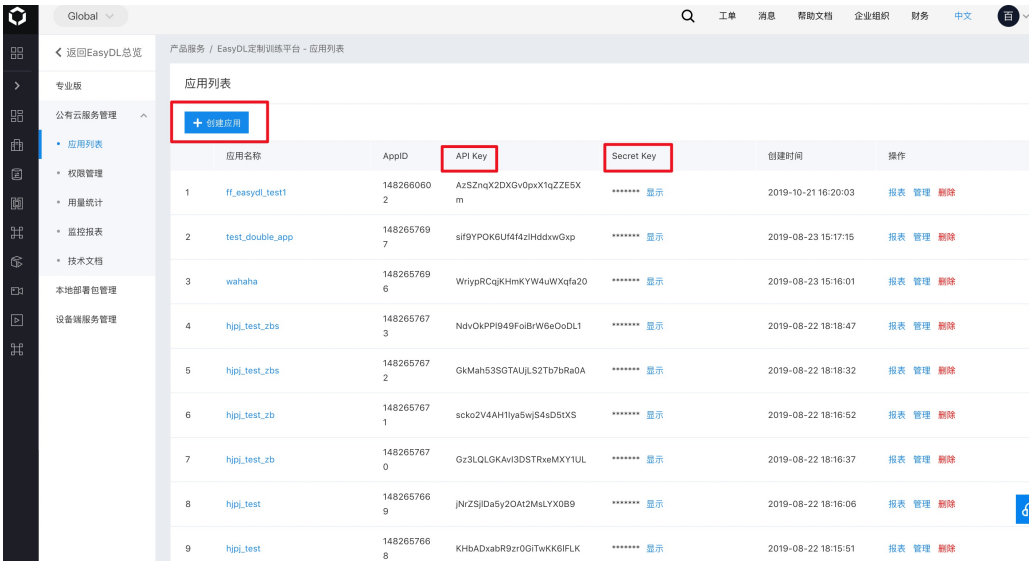
接口描述

基于自定义训练出的序列标注模型，实现定制标签的序列标注。模型训练完毕后发布可获得定制API

接口鉴权

1、在BML——控制台创建应用

2、应用列表页获取AK SK



请求说明

请求示例

HTTP 方法：POST

请求URL： 请首先进行自定义模型训练，完成训练后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

注意：如果出现336001的错误码很可能是因为请求方式错误，需以json方式请求。

Body请求示例：

```
{
  "text": "<UTF-8编码数据>"
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
text	是	string	-	文本数据，UTF-8编码。最大长度512汉字，空格分隔的字/词

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
raw_text	否	array(string)	原始输入，按空格切分
label	否	array(string)	标注结果
error_code	否	number	错误码
error_msg	否	string	错误信息

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如Access Token失效返回：

```
 {
  "error_code": 110,
  "error_msg": "Access token invalid or no longer valid"
}
```

需要重新获取新的Access Token再次请求即可。

错误码	错误信息	描述
1	Unknown error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
2	Service temporarily unavailable	服务暂不可用，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
3	Unsupported openapi method	调用的API不存在，请检查后重新尝试
4	Open api request limit reached	集群超限额
6	No permission to access data	无权限访问该用户数据
13	Get service token failed	获取token失败
14	IAM Certification failed	IAM鉴权失败
15	app not exists or create failed	应用不存在或者创建失败
17	Open api daily request limit reached	每天请求量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
18	Open api qps request limit reached	QPS超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
19	Open api total request limit reached	请求总量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
100	Invalid parameter	无效的access_token参数，请检查后重新尝试
110	Access token invalid or no longer valid	access_token无效
111	Access token expired	access token过期
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（868826008）或工单联系技术支持团队
336001	Invalid Argument	入参格式有误，比如缺少必要参数、文本编码错误等等，可检查下文本编码、代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数或者代码格式有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336004	输入文件大小不合法	文本超出大小限制，文本限4M以内，请根据接口文档检查入参格式，有疑问请通过QQ群（868826008）或工单联系技术支持团队
336005	文本解码失败	文本切词错误，请检查并修改文本格式
336006	缺失必要参数	text字段缺失（未上传文本）
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请通过QQ群（868826008）或工单联系技术支持团队

文本实体抽取私有API调用说明

接口描述

基于自定义训练出的文本分类模型，实现个性化文本识别。模型训练完毕后发布可获得定制化实体抽取API。

如有其它问题，请在百度智能云控制台内提交工单反馈。

请求说明

请求示例

HTTP 方法：`POST`

请求URL： 请首先在定制化训练平台进行自定义模型训练，完成训练后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

注意：如果出现336001的错误码很可能是因为请求方式错误，服务以json方式请求。

Body请求示例：

```
 {
   "text": "<UTF-8编码文本>"
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
text	是	string	-	文本数据，UTF-8编码。最大长度512个字符

注意：通过API接口预测时，模型仅接收512个字符（包括汉字和标点符号）内的文本，超出将被截断。

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
error_code	否	number	错误码，当请求错误时返回
error_msg	否	string	错误描述信息，当请求错误时返回
content	是	string	预测的文本内容（最多仅支持512个字）
results	否	array(object)	分类结果数组
+span	否	string	抽取的具体实体内容
+offset	否	number	具体实体内容的位置，从entity_content中，第一个字符记为0起算
+tag	否	string	识别对应span的实体类别

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如Access Token失效返回：

```
 {
  "error_code": 110,
  "error_msg": "Access token invalid or no longer valid"
}
```

需要重新获取新的Access Token再次请求即可。

错误码	错误信息	描述
1	Unknown error	服务器内部错误，请再次请求， 如果持续出现此类错误，请在百度智能云控制台内提交工单反馈。
2	Service temporarily unavailable	服务暂不可用，请再次请求， 如果持续出现此类错误，请在百度智能云控制台内提交工单反馈。
3	Unsupported openapi method	调用的API不存在，请检查后重新尝试
4	Open api request limit reached	集群超限额
6	No permission to access data	无权限访问该用户数据
13	Get service token failed	获取token失败
14	IAM Certification failed	IAM鉴权失败
15	app not exists or create failed	应用不存在或者创建失败
17	Open api daily request limit reached	每天请求量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度智能云控制台内提交工单反馈
18	Open api qps request limit reached	QPS超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度智能云控制台内提交工单反馈
19	Open api total request limit reached	请求总量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度智能云控制台内提交工单反馈
100	Invalid parameter	无效的access_token参数，请检查后重新尝试
110	Access token invalid or no longer valid	access_token无效
111	Access token expired	access token过期
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请在百度智能云控制台内提交工单反馈
336001	Invalid Argument	入参格式有误，比如缺少必要参数、图片base64编码错误等等，可检查下图片编码、代码格式是否有误。有疑问请在百度智能云控制台内提交工单反馈
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数或代码格式有误。有疑问请在百度智能云控制台内提交工单反馈
336003	Base64解码失败	文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请在百度智能云控制台内提交工单反馈
336004	输入文件大小不合法	文本大小不合法，目前支持文本文件类型为支持txt，文本文件大小限制长度最大4096 UTF-8字符。
336005	解码失败	文本编码错误（不是utf-8），目前支持文本文件类型为支持txt。如果遇到请重试，如反复失败，请在百度智能云控制台内提交工单反馈
336006	缺失必要参数	未上传文本文件
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请在百度智能云控制台内提交工单反馈

私有服务器部署

如何部署在私有服务器

发布模型页面中，可以申请私有服务器部署，将模型部署在私有服务器中。

页面中操作步骤如下：

1. 选择模型
2. 选择部署方式「私有服务器部署」
3. 选择集成方式「私有API」

私有API：将模型以Docker形式在私有服务器（仅支持Linux）上部署为http服务，可调用与公有云API功能相同的接口。可纯离线完成部署，服务调用便捷

点击「提交申请」后，前往控制台申请私有部署包。并参考文档完成集成

私有服务器部署价格说明 将定制模型部署在私有服务器上，只需在发布模型时提交私有服务器部署申请，通过审核后即可获得一个月免费试用。如需购买永久使用授权，请微信搜索“**BaiduEasyDL**”添加小助手咨询，通过线下签订合同购买使用。

私有服务API说明

私有化部署接口说明-文本分类

本文档主要说明定制化模型本地部署后，如何使用本地API。如还未训练模型，请先前往BML进行训练。

如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动
- 加入BML官方QQ群（群号:868826008）联系群管

部署包使用说明

BMLL定制化文本分类模型的本地部署通过EasyPack实现，目前提供单机一键部署的方式。

在BML控制台申请、下载部署包后，可以参考EasyPack-单机一键部署将软件包部署在本地服务器上。部署成功后，启动服务，即可调用与在线API功能类似的接口。

授权说明

本地部署包根据服务器硬件（CPU单机或GPU单卡）进行授权，只能在申请时提交的硬件指纹所属的硬件上使用。

部署包测试期为1个月，如需购买永久授权，可提交工单咨询

API参考

请求说明

请求示例

HTTP 方法：`POST`

请求URL：请首先在EasyDL专业版进行自定义模型训练，完成训练后申请本地部署，本地部署成功后拼接url。

请求URL： http://{IP}:{PORT}/{DEPLOY_NAME}/TextClassification

- IP：服务本地部署所在机器的ip地址
- PORT：服务部署后获取的端口
- DEPLOY_NAME：申请时填写的本地服务名称

Header如下：

参数	值
Content-Type	application/json

Body请求示例：

```
{
  "text": "<UTF-8编码数据>",
  "top_num": 5
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
text	是	string	-	文本数据，UTF-8编码。最大长度4096汉字
top_num	否	number	-	返回分类数量，默认为6个

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
results	否	array(object)	分类结果数组
+name	否	string	分类名称
+score	否	number	置信度

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如缺少必要出入参时返回：

```
{
 "error_code": 336001,
 "error_msg": "Invalid Argument"
}
```

错误码	错误信息	描述
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（868826008）或工单联系技术支持团队
336001	Invalid Argument	入参格式有误，比如缺少必要参数、文本的编码UTF-8等问题。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336004	输入文件大小不合法	文本超出大小限制，文本限4M以内，请根据接口文档检查入参格式，有疑问请通过QQ群（868826008）或工单联系技术支持团队
336005	文本解码失败	文本编码错误，请检查并修改文本格式
336006	缺失必要参数	文本字段内容缺失
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请通过QQ群（868826008）或工单联系技术支持团队
337000	Auth check failed	离线鉴权调用失败

私有部署说明-短文本相似度

本文档主要说明定制化模型本地部署后，如何使用本地API。如还未训练模型，请先前往BML进行训练。

如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动
- 加入BML官方QQ群（群号:868826008）联系群管

部署包使用说明

BML定制化短文本相似度模型的本地部署通过EasyPack实现，目前提供单机一键部署的方式。

在BML控制台申请、下载部署包后，可以参考EasyPack-单机一键部署将软件包部署在本地服务器上。部署成功后，启动服务，即可调用与在线API功能类似的接口。

授权说明

本地部署包根据服务器硬件（CPU单机或GPU单卡）进行授权，只能在申请时提交的硬件指纹所属的硬件上使用。

部署包测试期为1个月，如需购买永久授权，可提交工单咨询

API参考

请求说明

请求示例

HTTP 方法：`POST`

请求URL：请首先在BML进行自定义模型训练，完成训练后申请本地部署，本地部署成功后拼接url。

请求URL：http://{IP}:{PORT}/{DEPLOY_NAME}/TextMatching

- IP：服务本地部署所在机器的ip地址
- PORT：服务部署后获取的端口
- DEPLOY_NAME：申请时填写的本地服务名称

Header如下：

参数	值
Content-Type	application/json

Body请求示例：

```
 {
   "text_a": "<UTF-8编码数据>",
   "text_b": "<UTF-8编码数据>"
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
text_a	是	string	-	文本数据，UTF-8编码。最大长度4096汉字
text_b	是	string	-	文本数据，UTF-8编码。最大长度4096汉字

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
score	是	number	文本相似度，从0-1，相似度递增

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如缺少必要出入参时返回：

```
{
  "error_code": 336001,
  "error_msg": "Invalid Argument"
}
```

错误码	错误信息	描述
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（868826008）或工单联系技术支持团队
336001	Invalid Argument	入参格式有误，比如缺少必要参数、文本的编码UTF-8等问题。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336004	输入文件大小不合法	文本大小不合法，目前支持文本文件类型为支持txt，每个文本大小限制长度最大512个 UTF-8字符。
336005	解码失败	文本编码错误（不是utf-8），如果遇到请重试，如反复失败，请在百度智能云控制台内提交工单反馈
336006	缺失必要参数	未上传文本文件
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请通过QQ群（868826008）或工单联系技术支持团队
337000	Auth check failed	离线鉴权调用失败

私有部署文档-序列标注

本文档主要说明定制化模型本地部署后，如何使用本地API。如还未训练模型，请先前往BML进行训练。

如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入EasyDL社区交流 ,与其他开发者进行互动
- 加入BML官方QQ群（群号:868826008）联系群管

部署包使用说明

BML定制化序列标注模型的本地部署通过EasyPack实现，目前提供单机一键部署的方式。

在BML控制台申请、下载部署包后，可以参考EasyPack-单机一键部署将软件包部署在本地服务器上。部署成功后，启动服务，即可调用与在线API功能类似的接口。

授权说明

本地部署包根据服务器硬件（CPU单机或GPU单卡）进行授权，只能在申请时提交的硬件指纹所属的硬件上使用。

部署包测试期为1个月，如需购买永久授权，可提交工单咨询

API参考

请求说明

请求示例

HTTP 方法：`POST`

请求URL：请首先在BML进行自定义模型训练，完成训练后申请本地部署，本地部署成功后拼接url。

请求URL：http://{IP}:{PORT}/{DEPLOY_NAME}/SequenceLabel

- IP：服务本地部署所在机器的ip地址
- PORT：服务部署后获取的端口
- DEPLOY_NAME：申请时填写的本地服务名称

Header如下：

参数	值
Content-Type	application/json

Body请求示例：

```
 {
   "text": "<UTF-8编码数据>"
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
text	是	string	-	文本数据，UTF-8编码。最大长度512个汉字，注意，文本必须以空格分隔的字/词

返回说明

返回参数 | 字段 | 是否必选 | 类型 | 说明 | | ---------- | -------- | ------------ | ------------------------- | | log_id | 是 | number | 唯一的log id，用于问题定位 | | raw_text | 否 | array(string) | 原始输入，按空格切分 | | label | 否 | array(string) | 标注结果 | | error_code | 否 | number | 错误码 | | error_msg | 否 | string | 错误信息 |

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如缺少必要出入参时返回：

```
 {
  "error_code": 336001,
  "error_msg": "Invalid Argument"
}
```

错误码	错误信息	描述
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（868826008）或工单联系技术支持团队
336001	Invalid Argument	入参格式有误，比如缺少必要参数、文本的编码UTF-8等问题。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336004	输入文件大小不合法	文本超出大小限制，文本限4M以内，请根据接口文档检查入参格式，有疑问请通过QQ群（868826008）或工单联系技术支持团队
336005	文本解码失败	文本编码错误（不是utf-8）
336006	缺失必要参数	文本字段内容缺失
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请通过QQ群（868826008）或工单联系技术支持团队
337000	Auth check failed	离线鉴权调用失败

文本实体抽取API调用文档

接口描述

基于自定义训练出的文本分类模型，实现个性化文本识别。模型训练完毕后发布可获得定制化实体抽取API 。

如有其它问题，请在百度智能云控制台内提交工单反馈。

请求说明

请求示例

HTTP 方法：`POST`

请求URL： 完成训练后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

注意：如果出现336001的错误码很可能是因为请求方式错误，服务以json方式请求。

Body请求示例：

```
 {
   "text": "<UTF-8编码文本>"
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
text	是	string	-	文本数据，UTF-8编码。最大长度512个字符

注意：通过API接口预测时，模型仅接收512个字符（包括汉字和标点符号）内的文本，超出将被截断。

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
error_code	否	number	错误码，当请求错误时返回
error_msg	否	string	错误描述信息，当请求错误时返回
content	是	string	预测的文本内容（最多仅支持512个字）
results	否	array(object)	分类结果数组
+span	否	string	抽取的具体实体内容
+offset	否	number	具体实体内容的位置，从entity_content中，第一个字符记为0起算
+tag	否	string	识别对应span的实体类别

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如Access Token失效返回：

```
{
 "error_code": 110,
 "error_msg": "Access token invalid or no longer valid"
}
```

需要重新获取新的Access Token再次请求即可。

错误码	错误信息	描述
1	Unknown error	服务器内部错误，请再次请求， 如果持续出现此类错误，请在百度智能云控制台内提交工单反馈。
2	Service temporarily unavailable	服务暂不可用，请再次请求， 如果持续出现此类错误，请在百度智能云控制台内提交工单反馈。
3	Unsupported openapi method	调用的API不存在，请检查后重新尝试
4	Open api request limit reached	集群超限额
6	No permission to access data	无权限访问该用户数据
13	Get service token failed	获取token失败
14	IAM Certification failed	IAM鉴权失败
15	app not exists or create failed	应用不存在或者创建失败
17	Open api daily request limit reached	每天请求量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度智能云控制台内提交工单反馈
18	Open api qps request limit reached	QPS超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度智能云控制台内提交工单反馈
19	Open api total request limit reached	请求总量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度智能云控制台内提交工单反馈
100	Invalid parameter	无效的access_token参数，请检查后重新尝试
110	Access token invalid or no longer valid	access_token无效
111	Access token expired	access token过期
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请在百度智能云控制台内提交工单反馈
336001	Invalid Argument	入参格式有误，比如缺少必要参数、图片base64编码错误等等，可检查下图片编码、代码格式是否有误。有疑问请在百度智能云控制台内提交工单反馈
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数或代码格式有误。有疑问请在百度智能云控制台内提交工单反馈
336003	Base64解码失败	文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请在百度智能云控制台内提交工单反馈
336004	输入文件大小不合法	文本大小不合法，目前支持文本文件类型为支持txt，文本文件大小限制长度最大4096 UTF-8字符。
336005	解码失败	文本编码错误（不是utf-8），目前支持文本文件类型为支持txt。如果遇到请重试，如反复失败，请在百度智能云控制台内提交工单反馈
336006	缺失必要参数	未上传文本文件
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请在百度智能云控制台内提交工单反馈

表格预测模型部署

整体说明

训练完成后，可将模型部署在公有云服务器上，通过API进行调用。当前表格预测模型仅支持公有云部署，即将模型部署在BML提供的云端机器资源中，您可以通过访问在线API实时获取模型结果。模型部署在BML提供的云端机器资源中，您可以通过访问在线API实时获取模型结果。

公有云API

训练完成的模型存储在云端，可通过独立Rest API调用模型，实现AI能力与业务系统的无缝集成。

具有完善的鉴权、流控等安全机制，稳定承载高并发请求。

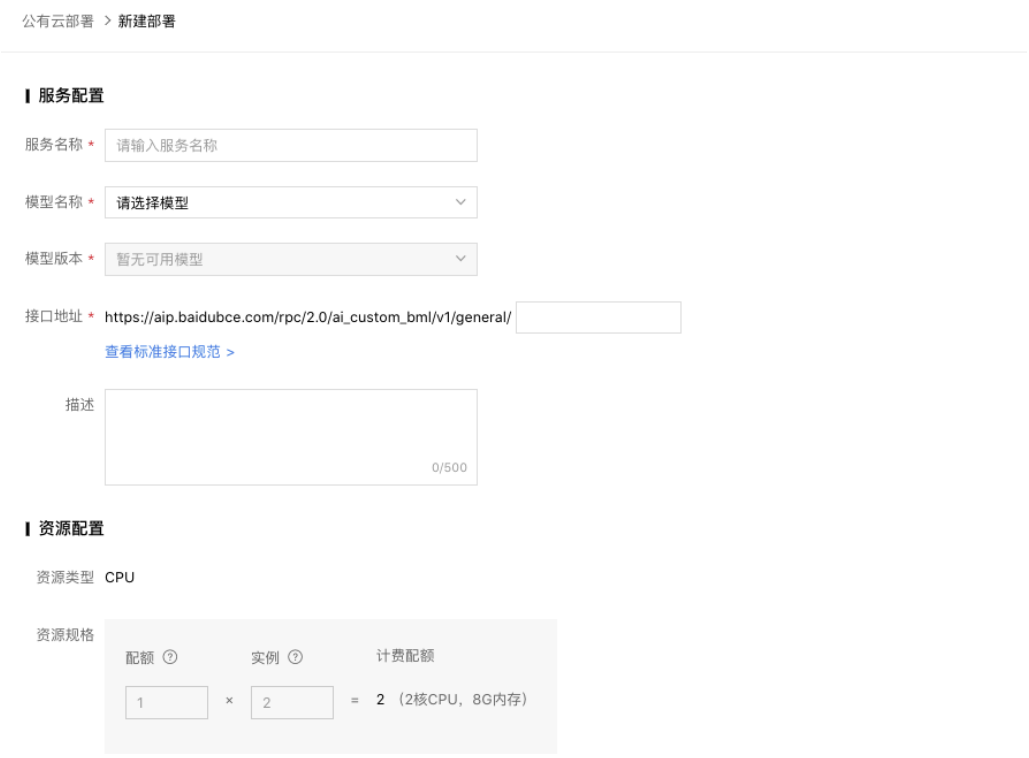
公有云部署

公有云部署简介

模型仓库中的表格数据预测模型，可以通过公有云部署为在线服务，并对外提供REST API，已有的业务系统可以通过API进行模型预测服务的集成。

模型部署流程

1. 将训练完成的表格预测模型发布到模型仓库，在模型仓库中的『版本列表』中点击『公有云部署』或者点击左侧导航栏的公有云部署，进入公有云部署界面后点击『部署模型』。
2. 在部署界面，首先进行服务配置：自定义服务名称、接口地址后缀，选择需要部署的模型，
3. 对每个部署模型还需要进行资源配置，配置模型预测服务所需机器资源。目前仅支持CPU机型，支持按配额与实例进行灵活配置，计费说明请点击查看



4. 配置完成后，点击开始部署，在完成部署后可以通过配置的API地址调用预测服务。

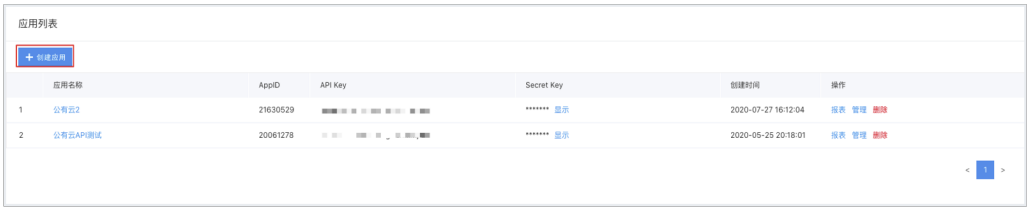
> 说明：
>
> 1、系统将运行公有云部署的机器资源1核cpu和4G内存折算为1配额，提高配额数量能提升QPS和降低推理时延
>
> 2、实例为cpu资源运行的实例数，为保证服务的稳定性，实例数至少需要配置为2。实例数与QPS成正比关系，提高实例数时，QPS能正比增加
>
> 3、表格预测模型的API可以在模型仓库中模型版本详情页面的“API参考”页签中查看。

接口赋权

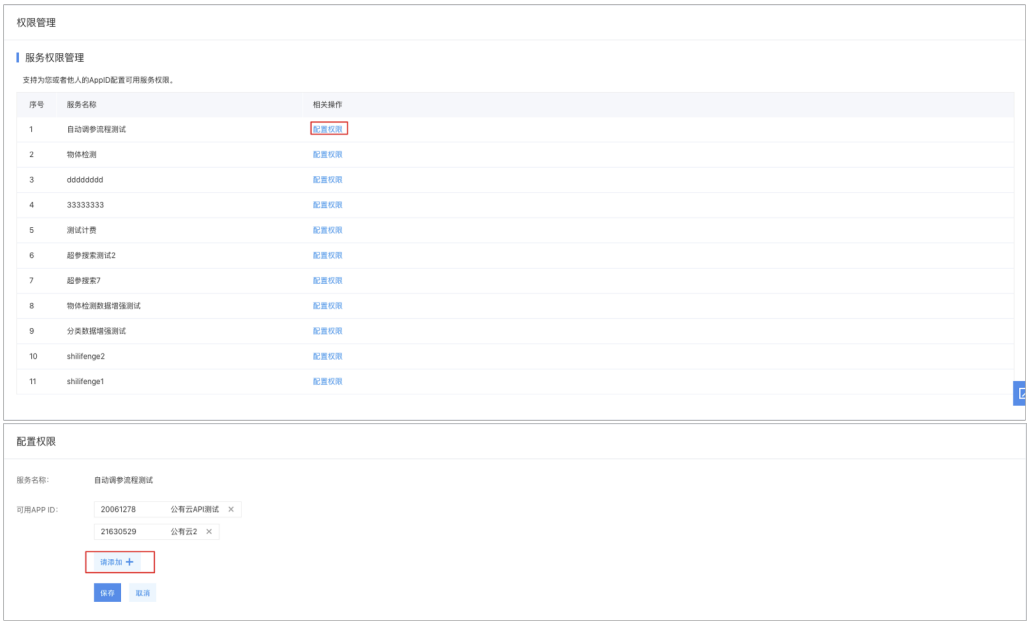
1. 正式使用还需要对接口进行赋权，在已发布的服务的详情界面，点击『控制台』进入BML控制台。



2. 在控制台中创建一个应用，获得由一串数字组成的appid、AK、SK，然后就可以参考接口文档正式使用了

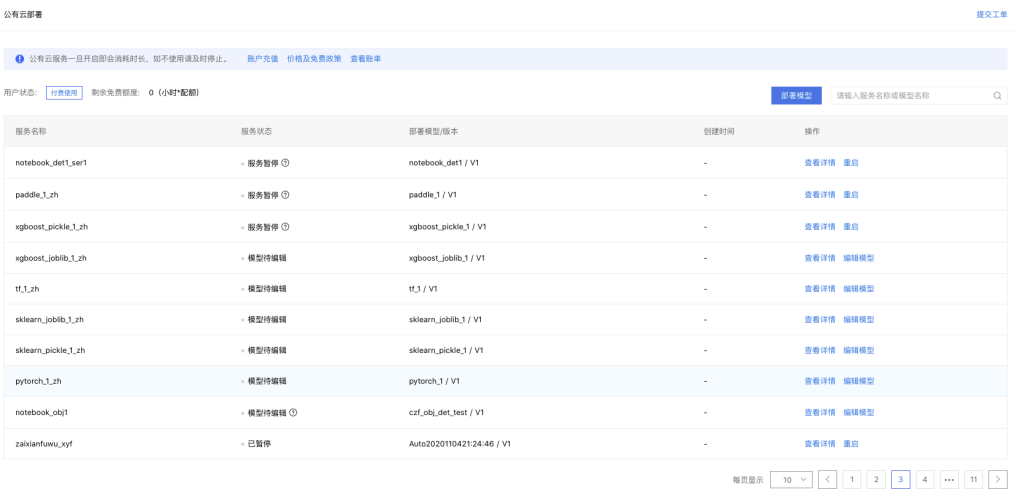


3. 如需为第三方用户配置调用接口的权限，可以前往BML控制台-公有云服务管理-权限管理配置



公有云部署列表

公有云部署列表信息由历史提交公有云服务组成，显示服务的基本信息。



如图，在预测服务列表页支持以下操作：

查看详情：点击进入该公有云服务的详情界面

重启：对已停止或部署失败的服务进行重新创建，恢复对应的服务为可用状态

停止：停止该公有云服务，用户不再使用相关服务后可进行这一操作，停止后将不进行计费

编辑模型：对于已经部署为公有云服务，但尚未配置机器资源的模型，点击可以配置机器资源

请求说明

请求示例

HTTP 方法：`POST`

请求URL： 请首先进行自定义模型训练，完成训练后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

注意：如果出现336001的错误码很可能是因为请求方式错误，与其他图像识别服务不同的是定制化图像识别服务以json方式请求。

Body请求接口：

```
{
  "include_req": false,
  "data": <待预测数组>
}
```

具体示例如下：

```
{
"include_req": false,
"data": [{
  "sepal_length": 5.1,
  "sepal_width": 3.5,
  "petal_length": 1.4,
  "petal_width": 0.2
 },
 {
  "sepal_length": 5.3,
  "sepal_width": 3.4,
  "petal_length": 1.2,
  "petal_width": 0.3
 }
]
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
data	是	array	-	待预测数据，每条待预测数据是由各个特征及其取值构成的键值对的集合

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
error_code	否	number	错误码，当请求错误时返回
error_msg	否	string	错误描述信息，当请求错误时返回
batch_results	否	array(object)	预测结果数组

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如Access Token失效返回：

```
{
 "error_code": 110,
 "error_msg": "Access token invalid or no longer valid"
}
```

需要重新获取新的Access Token再次请求即可。

错误码	错误信息	描述
1	Unknown error	服务器内部错误，请再次请求， 如果持续出现此类错误，请在百度云控制台内提交工单反馈。
2	Service temporarily unavailable	服务暂不可用，请再次请求， 如果持续出现此类错误，请在百度云控制台内提交工单反馈。
3	Unsupported openapi method	调用的API不存在，请检查后重新尝试
4	Open api request limit reached	集群超限额
6	No permission to access data	无权限访问该用户数据
13	Get service token failed	获取token失败
14	IAM Certification failed	IAM鉴权失败
15	app not exists or create failed	应用不存在或者创建失败
17	Open api daily request limit reached	每天请求量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度云控制台内提交工单反馈
18	Open api qps request limit reached	QPS超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度云控制台内提交工单反馈
19	Open api total request limit reached	请求总量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度云控制台内提交工单反馈
100	Invalid parameter	无效的access_token参数，请检查后重新尝试
110	Access token invalid or no longer valid	access_token无效
111	Access token expired	access token过期
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请在百度云控制台内提交工单反馈
336001	Invalid Argument	入参格式有误，比如缺少必要参数、data格式错误等等，可检查下请求数据格式是否有误。有疑问请在百度云控制台内提交工单反馈
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数或者代码格式有误。有疑问请在百度云控制台内提交工单反馈
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请在百度云控制台内提交工单反馈

通用模型部署

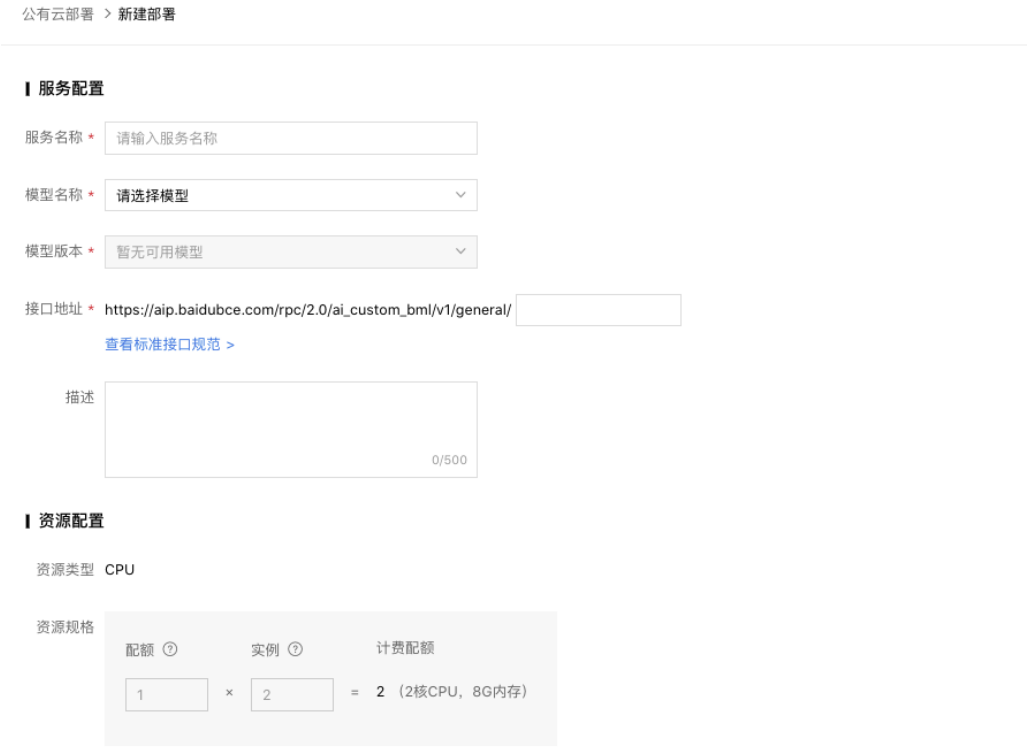
公有云部署

公有云部署简介

模型仓库中的模型，发布为公有云部署时储在云端，可通过独立Rest API调用模型，实现AI能力与业务系统或硬件设备整合。
BML具有完善的鉴权、流控等安全机制，并配置丰富的资源集群稳定承载高并发请求。 并且支持查找云端模型识别错误的数据，纠正结果并将其加入模型迭代的训练集，不断优化模型效果

模型部署流程

1. 将训练完成的模型发布到模型仓库，在模型仓库中的『版本列表』中点击『公有云部署』或者点击左侧导航栏的公有云部署，进入公有云部署界面后点击『部署模型』。
2. 在部署界面，首先进行服务配置：自定义服务名称、接口地址后缀，选择需要部署的模型，
3. 对每个部署模型还需要进行资源配置，配置模型预测服务所需机器资源。目前资源开放CPU类型，支持按配额与实例进行灵活配置，计费说明请点击查看

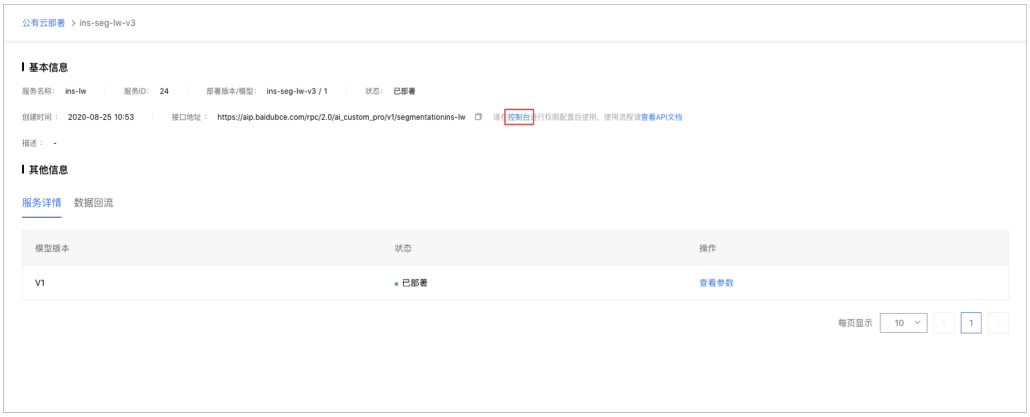


4. 配置完成后，点击开始部署即可一键部署模型，并获得专用的API地址。

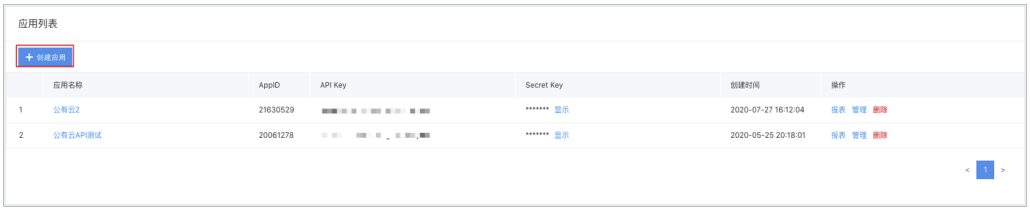
> 说明：
>
> 1、系统将运行公有云部署的机器资源1核cpu和4G内存折算为1配额，提高配额数量能提升QPS和降低推理时延
>
> 2、实例为cpu资源运行的实例数，为保证服务的稳定性，实例数至少需要配置为2。实例数与QPS成正比关系，提高实例数时，QPS能正比增加

接口赋权

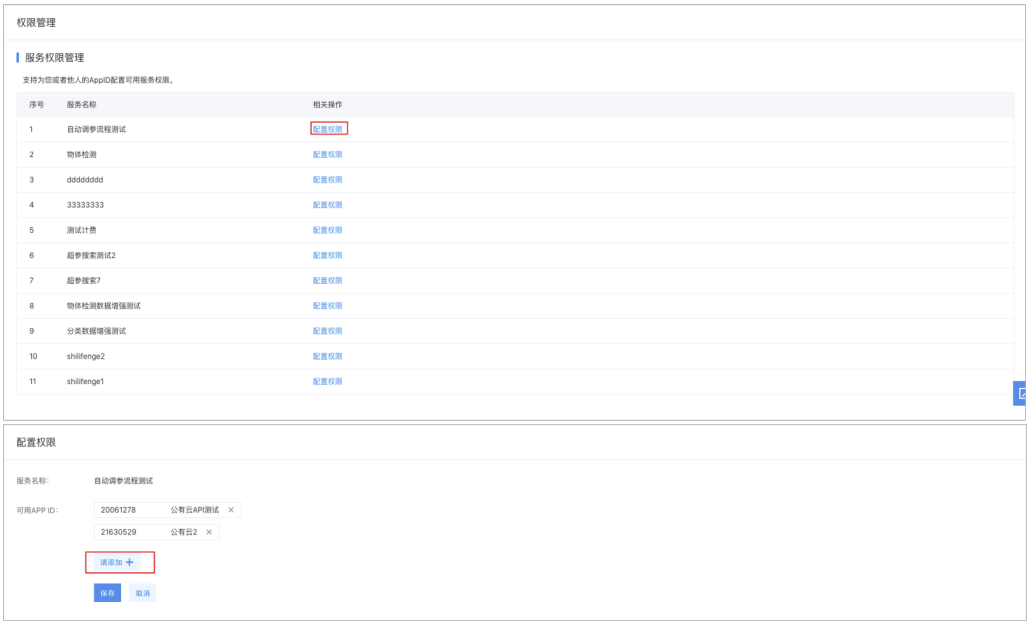
1. 正式使用还需要对接口进行赋权，在已发布的服务的详情界面，点击『控制台』进入BML控制台。



2. 在控制台中创建一个应用，获得由一串数字组成的appid，然后就可以参考接口文档正式使用了

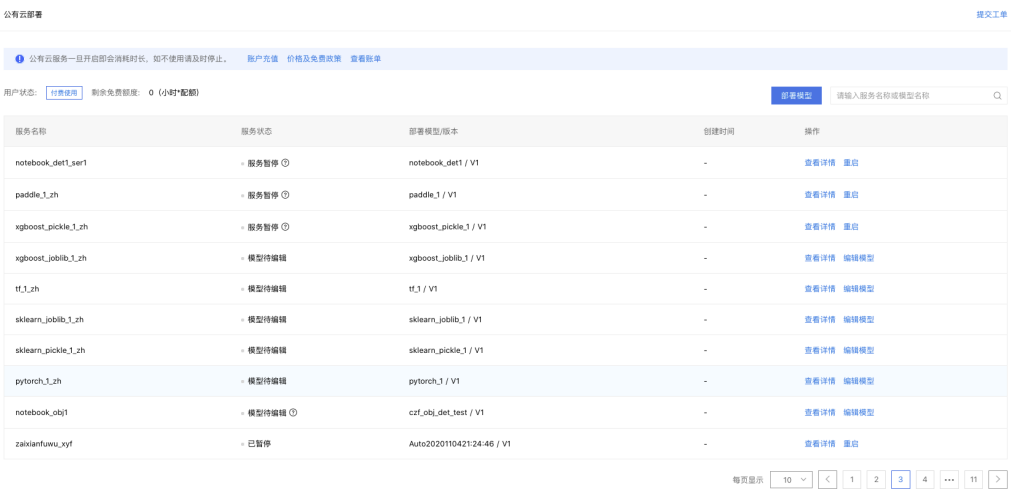


3. 如需为第三方用户配置调用接口的权限，可以前往BML控制台-公有云服务管理-权限管理配置



公有云部署列表

公有云部署列表信息由历史提交公有云服务组成，显示服务的基本信息。



如图，在预测服务列表页支持以下操作：

查看详情：点击进入该公有云服务的详情界面

重启：对已停止或部署失败的服务进行重新创建，恢复对应的服务为可用状态

停止：停止该公有云服务，用户不再使用相关服务后可进行这一操作，停止后将不进行计费

编辑模型：对于已经部署为公有云服务，但尚未配置机器资源的模型，点击可以配置机器资源

Paddle框架API调用文档

本文档主要说明使用Paddle框架训练的模型部署到公有云后获得的API如何使用，如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动

接口描述

基于Paddle深度学习框架训练出的模型，模型训练完毕，发布到模型仓库并部署到公有云后可获得定制API

接口鉴权

1. 在BML-控制台创建应用
2. 应用列表获取AK/SK

请求说明

HTTP 方法：`POST`

请求URL： 请首先进行自定义模型训练，完成训练后申请上线，上线成功后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body请求示例：

```
{
 "instances": [{
  "tensors": [{
    "name": "", // tensor name
    "type": "FLOAT/INT64/INT32",
    // xxx_data必须与"type"相对应，一个tensor中只能出现一种数据类型的data.
    "float_data": [],
    "int64_data": [],
    "int32_data": [],
    "shape": []
  }]
 }]
}
```

Body中放置请求参数，参数想详情如下：

请求参数：

参数	是否必选	类型	可选范围值	说明
instances	是	array(object)	-	用于预测的数据实例信息
+tensors	是	array(object)	-	模型输入对应的tensor数据
++name	是	string	-	tensor名称
++shape	是	array(number)	-	tensor的shape
++type	是	string	`FLOAT`/`INT64`/`INT32`or的数据类型	
++float_data	否	array(number)	-	浮点型数据；必须与"type"相对应，当type为FLOAT时传入
++int64_data	否	array(number)	-	64位整形数据；必须与"type"相对应，当type为FLOAT时传入
++int32_data	否	array(number)	-	32位整形数据；必须与"type"相对应，当type为FLOAT时传入
++lod_vector	否	array(object)	-	tensor的LoD向量信息。常在变长序列输入模型(如文本类)中使用，详情可参考LoD-Tensor使用说明，模型不存在lod时无须指定
+++lod	否	array(number)	-	各个level的LoD信息

返回说明

返回参数

参数	是否必选	类型	可选值范围	说明
log_id	是	number	-	用于排查问题的log id
result	否	object	-	模型的预测结果
+predictions	否	array(object)	-	paddle模型预测结果
++tensors	是	array(object)	-	模型预测输出对应的tensor信息
+++name	是	string	-	tensor名称
+++shape	是	array(number)	-	tensor的shape
+++type	是	string	`FLOAT`/`INT64`/`INT32`	tensor的数据类型
+++float_data	否	array(number)	-	浮点型预测值；与"type"相对应，当type为FLOAT时返回
+++int64_data	否	array(number)	-	64位整形数据；与"type"相对应，当type为INT64时返回
+++int32_data	否	array(number)	-	32位整形数据；与"type"相对应，当type为INT32时返回
++lod_vector	否	array(object)	-	输出tensor的LoD向量信息。详情可参考LoD-Tensor使用说明
+++lod	否	array(number)	-	各个level的LoD信息

Tensorflow框架API调用文档

本文档主要说明使用TensorFlow框架训练的模型部署到公有云后获得的API如何使用，如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动

接口描述

基于TensorFlow深度学习框架训练出的模型，模型训练完毕，发布到模型仓库并部署到公有云后可获得定制API

接口鉴权

1. 在BML-控制台创建应用

2. 应用列表获取AK/SK

请求说明

HTTP 方法：POST

请求URL： 请首先进行自定义模型训练，完成训练后申请上线，上线成功后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body请求示例：

```
{
  "instances": [{
    # key为保存SavedModel时设置的inputs的key，即模型接收的输入，对应的value须和定义的类型和shape相同，
    "inputs": []
    "keep_prob": 1
  }],
  "signature_name": "predict"
}
```

Body中放置请求参数，参数详情如下：

参数	是否必选	类型	可选值范围	说明
instances	是	array(object)	-	用于预测的数据实例信息
+{key}	是	array/number	-	key为保存SavedModel时设置的inputs的key，对应的value须和定义的类型和shape相同，否则预测服务可能返回失败
+{key2}	是	array/number	-	key为保存SavedModel时设置的inputs的key，对应的value须和定义的类型和shape相同，否则预测服务可能返回失败
signature_name	否	string	-	保存SavedModel时设置的签名；若保存时未设置，请求时可不传

返回说明

返回body示例：

```
{
  "log_id": xxx,
  "result": {
    "predictions": [
      [2.13782158e-07, 7.51928155e-08, ... ]
    ]
  }
}
```

返回参数：

参数	是否必选	类型	可选值范围	说明
log_id	是	number	-	用于排查问题的log id
result	否	object	-	模型的预测结果
+predictions	否	array	-	ensorflow模型预测结果,具体结构因模型不同而存在差异
error_code	否	object	-	错误码
error_msg	否	object	-	错误信息

Pytorch框架API调用文档

本文档主要说明使用Pytorch框架训练的模型部署到公有云后获得的API如何使用，如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动

接口描述

基于Pytorch深度学习框架训练出的模型，模型训练完毕，发布到模型仓库并部署到公有云后可获得定制API

接口鉴权

1. 在BML-控制台创建应用
2. 应用列表获取AK/SK

请求说明

HTTP 方法：`POST`

请求URL： 请首先进行自定义模型训练，完成训练后申请上线，上线成功后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

无特定请求参数格式，与提交的自定义代码中的预处理`input_fn`相关

返回说明

返回参数：

参数	是否必选	类型	可选值范围	说明
log_id	是	number	-	用于排查问题的log id
result	否	object	-	模型的预测结果,具体结构因模型不同和自定义后处理代码output_fn而存在差异
error_code	否	object	-	错误码
error_msg	否	object	-	错误信息

sklearn框架API调用文档

本文档主要说明使用sklearn框架训练的模型部署到公有云后获得的API如何使用，如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动

接口描述

基于sklearn机器学习框架训练出的模型，模型训练完毕，发布到模型仓库并部署到公有云后可获得定制API

接口鉴权

1. 在BML-控制台创建应用

2. 应用列表获取AK/SK

请求说明

HTTP 方法：`POST`

请求URL： 请首先进行自定义模型训练，完成训练后申请上线，上线成功后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body请求示例：

```
{
  "instances": [
    {
      "features": [10, 20, 30, 20]
    }
  ]
}
```

请求参数：

参数	是否必选	类型	可选值范围	说明
instances	是	array(object)	-	用于预测的数据实例信息
+features	是	array(number)	-	单条特征数据，长度需和模型输入的特征总数相同

返回说明

返回body示例：

```
{
  "log_id": xxx,
  "result": {
    "labels": [
      {
        "categories":[1.0]
      }
    ]
  }
}
```

返回参数：

参数	是否必选	类型	可选值范围	说明
log_id	是	number	-	用于排查问题的log id
result	否	object	-	模型的预测结果
+labels	否	array(object)	-	各个instances对应的分类标签或者回归预测值
++categories	否	array(number)	-	单个instance对应的分类标签或者回归预测值
error_code 否	object	-	错误码	
error_msg	否	object	-	错误信息

XGBoost框架API调用文档

本文档主要说明使用XGBoost框架训练的模型部署到公有云后获得的API如何使用，如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动

接口描述

基于XGBoost机器学习框架训练出的模型，模型训练完毕，发布到模型仓库并部署到公有云后可获得定制API

接口鉴权

1. 在BML-控制台创建应用
2. 应用列表获取AK/SK

请求说明

HTTP 方法：`POST`

请求URL： 请首先进行自定义模型训练，完成训练后申请上线，上线成功后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body请求示例：

```
{
  "instances": [
    {
      "features": [10, 2, 0, 3 , 5]
    }
  ]
}
```

请求参数：

参数	是否必选	类型	可选值范围	说明
instances	是	array(object)	-	用于预测的数据实例信息
+features	是	array(number)	-	单条特征数据，长度需和模型输入的特征总数相同

返回说明

返回body示例：

```
{
  "log_id": xxx,
  "result": {
   "labels": [
      {
          "categories": [14.95627212524414]
      }
   ]
}
}
```

返回参数：

参数	是否必选	类型	可选值范围	说明
log_id	是	number	-	用于排查问题的log id
result	否	object	-	模型的预测结果
+labels	否	array(object)	-	各个instances对应的分类标签或者回归预测值
++categories	否	array(number)	-	单个instance对应的分类标签或者回归预测值
error_code	否	object	-	错误码
error_msg	否	object	-	错误信息

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如Access Token失效返回：

```
{
 "error_code": 110,
 "error_msg": "Access token invalid or no longer valid"
}
```

需要重新获取新的Access Token再次请求即可。

错误码	错误信息	描述
1	Unknown error	服务器内部错误，请再次请求， 如果持续出现此类错误，请在百度智能云控制台内提交工单反馈。
2	Service temporarily unavailable	服务暂不可用，请再次请求， 如果持续出现此类错误，请在百度智能云控制台内提交工单反馈。
3	Unsupported openapi method	调用的API不存在，请检查后重新尝试
4	Open api request limit reached	集群超限额
6	No permission to access data	无权限访问该用户数据
13	Get service token failed	获取token失败
14	IAM Certification failed	IAM鉴权失败
15	app not exists or create failed	应用不存在或者创建失败
17	Open api daily request limit reached	每天请求量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度智能云控制台内提交工单反馈
18	Open api qps request limit reached	QPS超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度智能云控制台内提交工单反馈
19	Open api total request limit reached	请求总量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度智能云控制台内提交工单反馈
100	Invalid parameter	无效的access_token参数，请检查后重新尝试
110	Access token invalid or no longer valid	access_token无效
111	Access token expired	access token过期
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请在百度智能云控制台内提交工单反馈
336001	Invalid Argument	入参格式有误，比如缺少必要参数、图片base64编码错误等等，可检查下图片编码、代码格式是否有误。有疑问请在百度智能云控制台内提交工单反馈
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数或代码格式有误。有疑问请在百度智能云控制台内提交工单反馈
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请在百度智能云控制台内提交工单反馈
336004	输入文件大小不合法	图片超出大小限制，图片限4M以内，请根据接口文档检查入参格式，有疑问请在百度智能云控制台内提交工单反馈
336005	图片解码失败	图片编码错误（非jpg,bmp,png等常见图片格式），请检查并修改图片格式
336006	缺失必要参数	image字段缺失（未上传图片）
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请在百度智能云控制台内提交工单反馈

通用类模型API参考

通用接口规范参考

自定义作业产出的模型进行公有云部署时，不具有统一的标准接口规范，根基不同的算法框架，提供通用接口规范参考。

Paddle：请在《Paddle框架API调用文档》中查看Paddle框架模型的API参考
TensorFlow：请在《Tensorflow框架API调用文档》中查看TensorFlow框架模型的API参考
Pytorch：请在《Pytorch框架API调用文档》中查看Pytorch框架模型的API参考
sklearn：请在《sklearn框架API调用文档》中查看sklearn框架模型的API参考
XGBoost：请在《XGBoost框架API调用文档》中查看XGBoost框架模型的API参考

标准接口规范参考

脚本调参和Notebook训练方式产出的模型部署到公有云时，不同任务类型的应用接口的请求与响应均满足响应的标准规范。

图像分类-单图单标签

标准接口请求参考说明：

字段名称	必须	类型	说明
image	是	string	图像数据，base64编码，要求base64编码后大小不超过4M，最短边至少15px，最长边最大4096px,支持jpg/png/bmp格式

标准接口响应字段说明：

字段名称	必须	类型	说明
log_id	是	number	唯一的log id，用于问题定位
error_code	否	number	错误码，当请求错误时返回
error_msg	否	string	错误描述信息，当请求错误时返回
results	否	array(object)	分类结果数组
+ name	否	string	分类名称
+ score	否	number	置信度

图像分类-单图多标签

标准接口请求参考说明：

字段名称	必须	类型	说明
image	是	string	图像数据，base64编码，要求base64编码后大小不超过4M，最短边至少15px，最长边最大4096px,支持jpg/png/bmp格式
threshold	否	number	阈值，默认为当前模型推荐阈值（0-1之间），具体值可以在我的训练任务列表-模型效果查看

标准接口响应字段说明：

字段名称	必须	类型	说明
log_id	是	number	唯一的log id，用于问题定位
error_code	否	number	错误码，当请求错误时返回
error_msg	否	string	错误描述信息，当请求错误时返回
results	否	array(object)	分类结果数组
+ name	否	string	分类名称
+ score	否	number	置信度

物体检测

标准接口请求参考说明：

字段名称	必须	类型	说明
image	是	string	图像数据，base64编码，要求base64编码后大小不超过4M，最短边至少15px，最长边最大4096px,支持jpg/png/bmp格式
threshold	否	number	阈值，默认为当前模型推荐阈值（0-1之间），具体值可以在我的训练任务列表-模型效果查看

标准接口响应字段说明：

字段名称	必须	类型	说明
log_id	是	number	唯一的log id，用于问题定位
error_code	否	number	错误码，当请求错误时返回
error_msg	否	string	错误描述信息，当请求错误时返回
results	否	array(object)	分类结果数组
+ name	否	string	分类名称
+ score	否	number	置信度
+ location	否	object	
++ left	否	number	目标定位位置的长方形左上顶点的水平坐标
++ top	否	number	目标定位位置的长方形左上顶点的垂直坐标
++ width	否	number	目标定位位置的长方形的宽度
++ height	否	number	目标定位位置的长方形的高度

实例分割

标准接口请求参考说明：

字段名称	必须	类型	说明
image	是	string	图像数据，base64编码，要求base64编码后大小不超过4M，最短边至少15px，最长边最大4096px,支持jpg/png/bmp格式
threshold	否	number	阈值，默认为当前模型推荐阈值（0-1之间），具体值可以在我的训练任务列表-模型效果查看

标准接口响应字段说明：

字段名称	必须	类型	说明
log_id	是	number	唯一的log id，用于问题定位
error_code	否	number	错误码，当请求错误时返回
error_msg	否	string	错误描述信息，当请求错误时返回
results	否	array(object)	分类结果数组
+ name	否	string	分类名称
+ score	否	number	置信度
+ location	否	object	
++ left	否	number	目标定位位置的长方形左上顶点的水平坐标
++ top	否	number	目标定位位置的长方形左上顶点的垂直坐标
++ width	否	number	目标定位位置的长方形的宽度
++ height	否	number	目标定位位置的长方形的高度
+ mask	否	array	基于游程编码的字符串，编码内容为和原图宽高相同的布尔数组：若数组值为0，代表原图此位置像素点不属于检测目标，若为1，代表原图此位置像素点属于检测目标。查看解码示例

文本分类

标准接口请求参考说明：

字段名称	必须	类型	说明
text	是	string	文本数据，UTF-8编码。最大长度4096汉字
top_num	否	number	返回分类数量，默认为6个

标准接口响应字段说明：

字段名称	必须	类型	说明
log_id	是	number	唯一的log id，用于问题定位
error_code	否	number	错误码，当请求错误时返回
error_msg	否	string	错误描述信息，当请求错误时返回
results	否	array(object)	分类结果数组
+ name	否	string	分类名称
+ score	否	number	置信度

序列标注

标准接口请求参考说明：

字段名称	必须	类型	说明
text	是	string	文本数据，UTF-8编码。最大长度4096汉字，空格分隔的字/词

标准接口响应字段说明：

字段名称	必须	类型	说明
log_id	是	number	唯一的log id，用于问题定位
error_code	否	number	错误码，当请求错误时返回
error_msg	否	string	错误描述信息，当请求错误时返回
raw_text	是	array(string)	原始输入，按空格切分
label	是	array(string)	标注结果

文本实体抽取

标准接口请求参考说明：

字段名称	必须	类型	说明
text	是	string	文本数据，UTF-8编码。最大长度512个字符（包括汉字和标点符号）内的文本，超出将被截断

标准接口响应字段说明：

字段名称	必须	类型	说明
log_id	是	number	唯一的log id，用于问题定位
error_code	否	number	错误码，当请求错误时返回
error_msg	否	string	错误描述信息，当请求错误时返回
results	否	array(object)	实体类别识别结果数组
+ content	是	string	预测的文本内容（最多仅支持512个字）
++ span	否	string	抽取的具体实体内容
++ offset	否	number	具体实体内容的位置，从entity_content中，第一个字符记为0起算
++ tag	否	string	识别对应span的实体类别

短文本相似度

标准接口请求参考说明：

字段名称	必须	类型	说明
text_a	是	string	文本数据，UTF-8编码。最大长度4096汉字
text_b	是	string	文本数据，UTF-8编码。最大长度4096汉字

标准接口响应字段说明：

字段名称	必须	类型	说明
log_id	是	number	唯一的log id，用于问题定位
error_code	否	number	错误码，当请求错误时返回
error_msg	否	string	错误描述信息，当请求错误时返回
score	是	number	文本相似度

文字识别模型部署

文字识别任务公有云部署

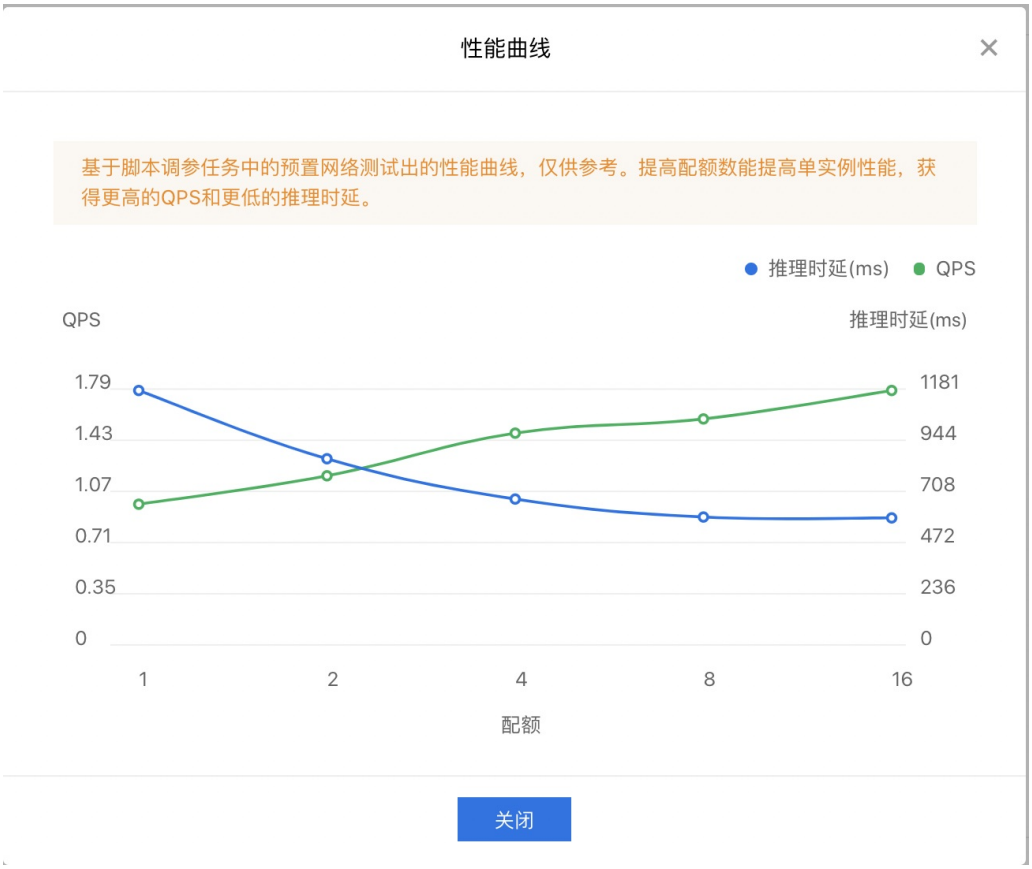
模型仓库中的文字识别模型，发布为公有云部署时储在云端，可通过独立Rest API调用模型，实现AI能力与业务系统或硬件设备整合。
BML具有完善的鉴权、流控等安全机制，并配置丰富的资源集群稳定承载高并发请求。 并且支持查找云端模型识别错误的数据，纠正结果并将其加入模型迭代的训练集，不断优化模型效果

公有云部署 模型部署流程

1. 将训练完成的模型发布到模型仓库，在模型仓库中的『版本列表』中点击『公有云部署』或者点击左侧导航栏的公有云部署，进入公有云部署界面后点击『部署模型』。
2. 在部署界面，首先进行服务配置：自定义服务名称、接口地址后缀，选择需要部署的模型，
3. 对每个部署模型还需要进行资源配置，配置模型预测服务所需机器资源。目前资源开放CPU类型，支持按配额与实例进行灵活配置，计费说明请点击查看



4. 当部署模型为脚本调参任务产出的模型时，系统会自动显示该模型在当前配置下的预估时延和QPS。点击"性能曲线"，即可查看**单实例**下的推理时延，QPS与配额的曲线关系，您可以根据性能曲线配置合理的配额。

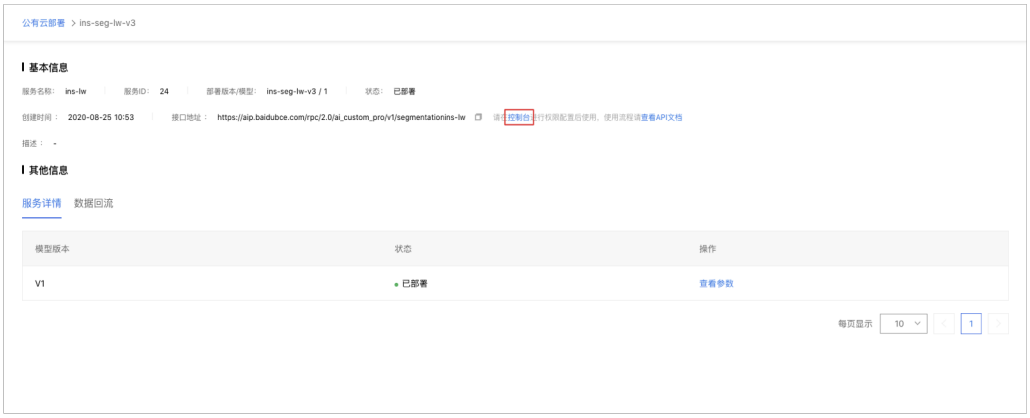


5. 配置完成后，点击开始部署即可一键部署模型，并获得专用的API地址。

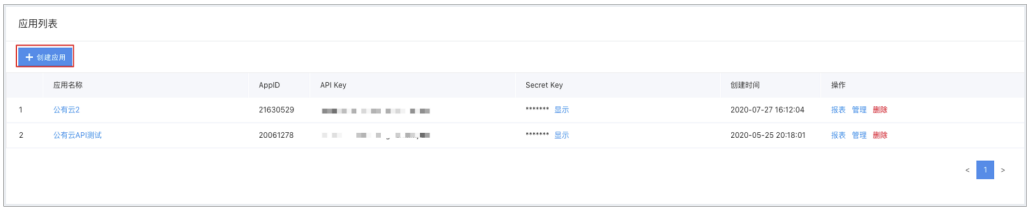
> 说明：
>
> 1、系统将运行公有云部署的机器资源1核cpu和4G内存折算为1配额，提高配额数量能提升QPS和降低推理时延
>
> 2、实例为cpu资源运行的实例数，为保证服务的稳定性，实例数至少需要配置为2。实例数与QPS成正比关系，提高实例数时，QPS能正比增加

接口赋权

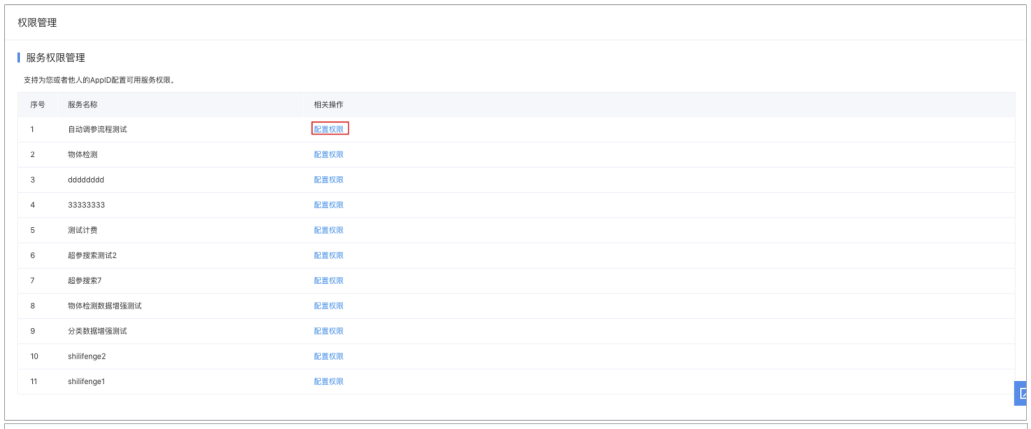
1. 正式使用还需要对接口进行赋权，在已发布的服务的详情界面，点击『控制台』进入BML控制台。

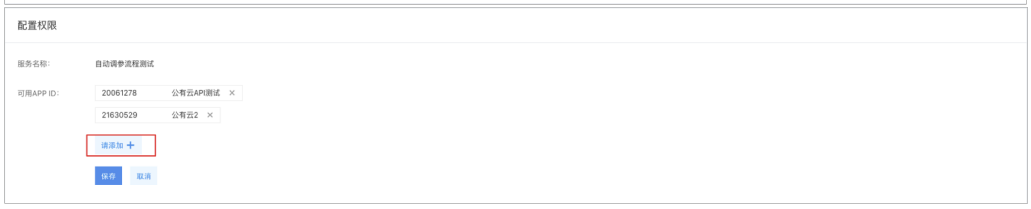


2. 在控制台中创建一个应用，获得由一串数字组成的appid，然后就可以参考接口文档正式使用了



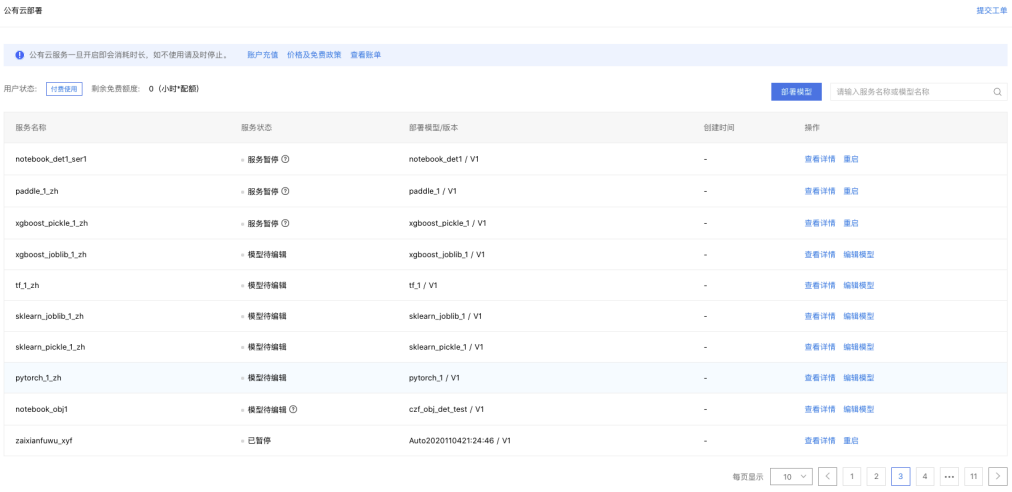
3. 如需为第三方用户配置调用接口的权限，可以前往BML控制台-公有云服务管理-权限管理配置





公有云部署列表

公有云部署列表信息由历史提交公有云服务组成，显示服务的基本信息。



如图，在预测服务列表页支持以下操作：

查看详情：点击进入该公有云服务的详情界面

重启：对已停止或部署失败的服务进行重新创建，恢复对应的服务为可用状态

停止：停止该公有云服务，用户不再使用相关服务后可进行这一操作，停止后将不进行计费

编辑模型：对于已经部署为公有云服务，但尚未配置机器资源的模型，点击可以配置机器资源

文字识别任务API参考文档

本文档主要说明定制化模型发布后获得的API如何使用，如有疑问可以通过以下方式联系我们：

- 在百度智能云控制台内提交工单
- 进入BML社区交流 ,与其他开发者进行互动
- 加入BML官方QQ群（群号:868826008）联系群管

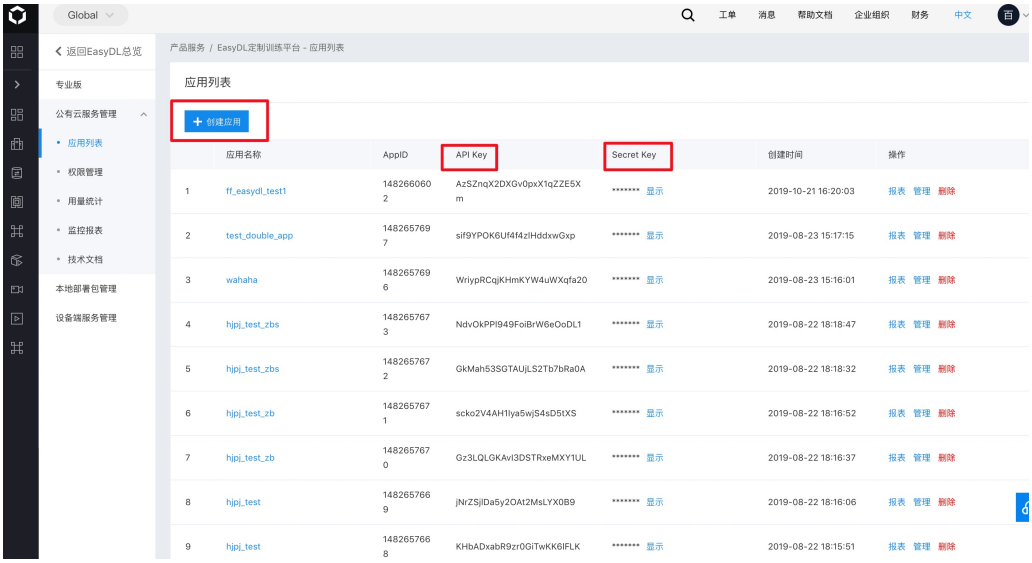
接口描述

基于自定义训练出的文字识别模型，训练完毕后发布可获得定制API。

接口鉴权

1、在BML——控制台创建应用

2、应用列表页获取AK SK



请求说明

请求示例

HTTP 方法：`POST`

请求URL： 请首先进行自定义模型训练，完成训练后申请上线，上线成功后可在服务列表中查看并获取url。

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

注意：如果出现336001的错误码很可能是因为请求方式错误，与其他图像识别服务不同的是定制化图像识别服务以json方式请求。

Body请求示例：

```
{
  "image": "<base64数据>"
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
image	是	string	-	图像数据，base64编码，要求base64编码后大小不超过4M，最短边至少15px，最长边最大4096px,支持jpg/png/bmp格式 **注意请去掉头部**
threshold	否	number	-	阈值，默认值为推荐阈值（0-1之间），具体值可在我的训练任务列表-模型效果查看

请求代码示例

提示一：使用示例代码前，请记得替换其中的示例Token、图片地址或Base64信息。

提示二：部分语言依赖的类或库，请在代码注释中查看下载地址。

PHP

Java

Python2

Python3

C++

```php
<?php
/**
 * 发起http post请求(REST API), 并获取REST请求的结果
 * @param string $url
 * @param string $param
 * @return - http response body if succeeds, else false.
 */
function request_post($url = '', $param = '')
{
    if (empty($url) || empty($param)) {
        return false;
    }

    $postUrl = $url;
    $curlPost = $param;
    // 初始化curl
    $curl = curl_init();
    curl_setopt($curl, CURLOPT_URL, $postUrl);
```

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
results	否	array(object)	识别结果数组
+text	否	string	识别文本内容
+score	否	number	置信度
+name	否	string	分类名称
+score	否	number	置信度
+location	否	object	目标物体所在位置
++points	否	list(object)	目标物体所在四边形的顶点信息
+++x	否	number	顶点横坐标
+++y	否	number	顶点纵坐标
error_code	否	number	错误码，当请求错误时返回
error_msg	否	string	错误描述信息，当请求错误时返回

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如Access Token失效返回：

```
{
 "error_code": 110,
 "error_msg": "Access token invalid or no longer valid"
}
```

需要重新获取新的Access Token再次请求即可。

错误码	错误信息	描述
1	Unknown error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
2	Service temporarily unavailable	服务暂不可用，请再次请求， 如果持续出现此类错误，请通过QQ群（649285136）或工单联系技术支持团队。
3	Unsupported openapi method	调用的API不存在，请检查后重新尝试
4	Open api request limit reached	集群超限额
6	No permission to access data	无权限访问该用户数据
13	Get service token failed	获取token失败
14	IAM Certification failed	IAM鉴权失败
15	app not exsits or create failed	应用不存在或者创建失败
17	Open api daily request limit reached	每天请求量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
18	Open api qps request limit reached	QPS超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
19	Open api total request limit reached	请求总量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请通过QQ群（649285136）联系群管手动提额
100	Invalid parameter	无效的access_token参数，请检查后重新尝试
110	Access token invalid or no longer valid	access_token无效
111	Access token expired	access token过期
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请通过QQ群（868826008）或工单联系技术支持团队
336001	Invalid Argument	入参格式有误，比如缺少必要参数、图片base64编码错误等等，可检查下图片编码、代码格式是否有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数或代码格式有误。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请通过QQ群（868826008）或工单联系技术支持团队
336004	输入文件大小不合法	图片超出大小限制，图片限4M以内，请根据接口文档检查入参格式，有疑问请通过QQ群（868826008）或工单联系技术支持团队
336005	图片解码失败	图片编码错误（非jpg,bmp,png等常见图片格式），请检查并修改图片格式
336006	缺失必要参数	image字段缺失（未上传图片）
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请通过QQ群（868826008）或工单联系技术支持团队

批量预测(用户资源池)API

语音技术模型部署

声音分类API调用文档

接口描述

声音分类模型完成模型发布后可以获得定制的声音分类API，实现基于定制的声音分类模型，调用API输入15s以内的音频数据，返回自定义的分类结果。

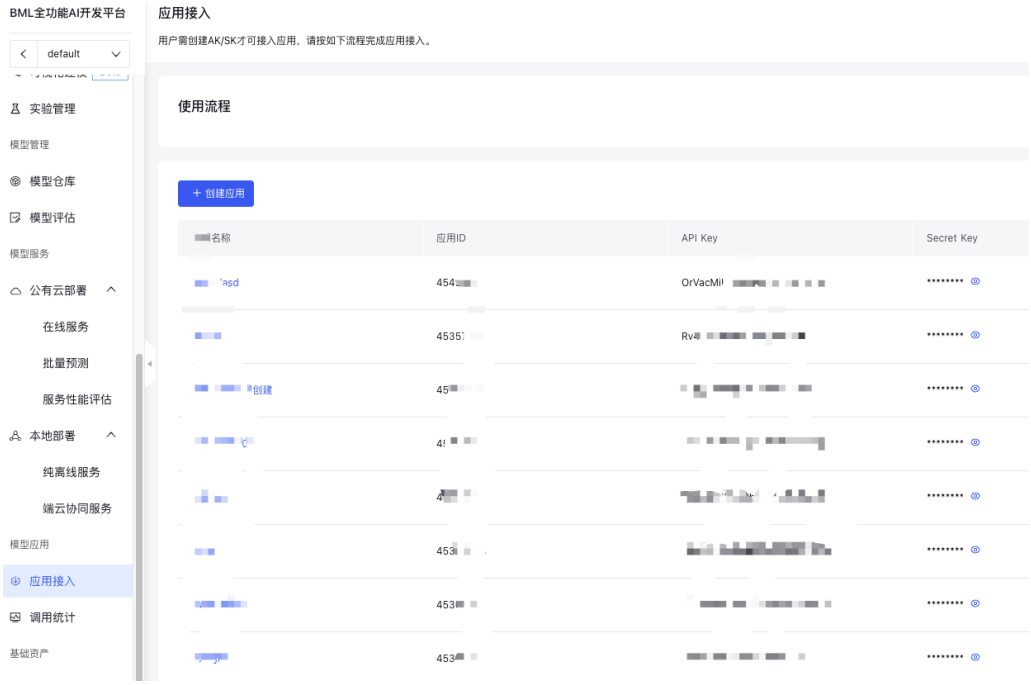
接口鉴权

创建应用

进入BML控制台概览页，在已上线的定制接口下，可以看到已发布上线的声音分类API。在正式调用之前，首先点击**创建应用**，定义应用名称、应用类型、应用描述等信息，完成应用创建。



应用创建后，在应用详情页获取API Key、Secret Key。



请求说明

请求示例

HTTP 方法：`POST`

请求URL： 请在应用详情页面获取接口地址，与发布模型时定义的url一致。



URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/json

Body请求示例：

```
{
  "sound": "<base64数据>",
  "top_num": 6
}
```

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	可选值范围	说明
sound	是	string	-	音频，base64编码，要求base64编码后大小不超过4M，支持mp3、m4a、wav格式，单个文件时间长度不超过15s **注意需要去掉编码头后再进行urlencode。**
top_num	否	number	-	返回分类数量，默认为6个

请求代码示例

提示一：使用示例代码前，请记得替换其中的示例Token、音频地址或Base64信息。

提示二：部分语言依赖的类或库，请在代码注释中查看下载地址。

PHP代码示例

```php
 <?php
/**
 * 发起http post请求(REST API), 并获取REST请求的结果
 * @param string $url
 * @param string $param
 * @return - http response body if succeeds, else false.
 */
function request_post($url = '', $param = '')
{
    if (empty($url) || empty($param)) {
        return false;
    }

    $postUrl = $url;
    $curlPost = $param;
    // 初始化curl
    $curl = curl_init();
    curl_setopt($curl, CURLOPT_URL, $postUrl);
    curl_setopt($curl, CURLOPT_HEADER, 0);
    // 要求结果为字符串且输出到屏幕上
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl, CURLOPT_SSL_VERIFYPEER, false);
    // post提交方式
    curl_setopt($curl, CURLOPT_POST, 1);
    curl_setopt($curl, CURLOPT_POSTFIELDS, $curlPost);
    // 运行curl
    $data = curl_exec($curl);
    curl_close($curl);

    return $data;
}

$token = '[调用鉴权接口获取的token]';
$url = '【接口地址】?access_token=' . $token;
$bodys = "{\"sound\":\"sfasq35sadvsvqwr5q...\"}"
$res = request_post($url, $bodys);

var_dump($res);
```

Python3代码示例

```
 """
EasyDL 声音分类 调用模型公有云API Python3实现
"""

import json
import base64
import requests
"""
使用 requests 库发送请求
使用 pip（或者 pip3）检查我的 python3 环境是否安装了该库，执行命令
  pip freeze | grep requests
若返回值为空，则安装该库
  pip install requests
"""


##### 目标音频的 本地文件路径，支持mp3/m4a/wav格式
SOUND_FILEPATH = "【您的测试音频地址，例如./example.mp3】"

##### 可选的请求参数
##### top_num: 返回的分类数量，不声明的话默认为 6 个
PARAMS = {"top_num": 3}

##### 服务详情 中的 接口地址
MODEL_API_URL = "【您的API地址】"

##### 调用 API 需要 ACCESS_TOKEN。若已有 ACCESS_TOKEN 则于下方填入该字符串
##### 否则，留空 ACCESS_TOKEN，于下方填入 该模型部署的 API_KEY 以及 SECRET_KEY，会自动申请并显示新 ACCESS_TOKEN
ACCESS_TOKEN = "【您的ACCESS_TOKEN】"
API_KEY = "【您的API_KEY】"
SECRET_KEY = "【您的SECRET_KEY】"


print("1. 读取目标音频 '{}'".format(SOUND_FILEPATH))
with open(SOUND_FILEPATH, 'rb') as f:
    base64_data = base64.b64encode(f.read())
    base64_str = base64_data.decode('UTF8')
print("将 BASE64 编码后音频的字符串填入 PARAMS 的 'sound' 字段")
PARAMS["sound"] = base64_str


if not ACCESS_TOKEN:
    print("2. ACCESS_TOKEN 为空，调用鉴权接口获取TOKEN")
    auth_url = "https://aip.baidubce.com/oauth/2.0/token?grant_type=client_credentials"\
            "&client_id={}&client_secret={}".format(API_KEY, SECRET_KEY)
    auth_resp = requests.get(auth_url)
    auth_resp_json = auth_resp.json()
    ACCESS_TOKEN = auth_resp_json["access_token"]
    print("新 ACCESS_TOKEN: {}".format(ACCESS_TOKEN))
else:
    print("2. 使用已有 ACCESS_TOKEN")


print("3. 向模型接口 'MODEL_API_URL' 发送请求")
request_url = "{}?access_token={}".format(MODEL_API_URL, ACCESS_TOKEN)
response = requests.post(url=request_url, json=PARAMS)
response_json = response.json()
response_str = json.dumps(response_json, indent=4, ensure_ascii=False)
print("结果:\n{}".format(response_str))
```

JAVA代码示例

```
 package com.baidu.ai.aip;

import com.baidu.ai.aip.utils.HttpUtil;
import com.baidu.ai.aip.utils.GsonUtils;

import java.util.*;

/**
* easydl声音分类
*/
public class EasydlSoundClassify {

    /**
    * 重要提示代码中所需工具类
    * FileUtil,Base64Util,HttpUtil,GsonUtils请从
    * https://ai.baidu.com/file/658A35ABAB2D404FBF903F64D47C1F72
    * https://ai.baidu.com/file/C8D81F3301E24D2892968F09AE1AD6E2
    * https://ai.baidu.com/file/544D677F5D4E4F17B4122FBD60DB82B3
    * https://ai.baidu.com/file/470B3ACCA3FE43788B5A963BF0B625F3
    * 下载
    */
    public static String easydlSoundClassify() {
        // 请求url
        String url = "【接口地址】";
        try {
            Map<String, Object> map = new HashMap<>();
            map.put("sound", "sfasq35sadvsvqwr5q...");

            String param = GsonUtils.toJson(map);

            // 注意这里仅为了简化编码每一次请求都去获取access_token，线上环境access_token有过期时间， 客户端可自行缓存，过期后重新获取。
            String accessToken = "[调用鉴权接口获取的token]";

            String result = HttpUtil.post(url, accessToken, "application/json", param);
            System.out.println(result);
            return result;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    public static void main(String[] args) {
        EasydlSoundClassify.easydlSoundClassify();
    }
}
```

C++代码示例

```
 ##### include <iostream>
##### include <curl/curl.h>

// libcurl库下载链接：https://curl.haxx.se/download.html
// jsoncpp库下载链接：https://github.com/open-source-parsers/jsoncpp/
const static std::string request_url = "【接口地址】";
static std::string easydlSoundClassify_result;
/**
 * curl发送http请求调用的回调函数，回调函数中对返回的json格式的body进行了解析，解析结果储存在全局的静态变量当中
 * @param 参数定义见libcurl文档
 * @return 返回值定义见libcurl文档
 */
static size_t callback(void *ptr, size_t size, size_t nmemb, void *stream) {
    // 获取到的body存放在ptr中，先将其转换为string格式
    easydlSoundClassify_result = std::string((char *) ptr, size * nmemb);
    return size * nmemb;
}
/**
 * easydl声音分类
 * @return 调用成功返回0，发生错误返回其他错误码
 */
int easydlSoundClassify(std::string &json_result, const std::string &access_token) {
    std::string url = request_url + "?access_token=" + access_token;
    CURL *curl = NULL;
    CURLcode result_code;
    int is_success;
    curl = curl_easy_init();
    if (curl) {
        curl_easy_setopt(curl, CURLOPT_URL, url.data());
        curl_easy_setopt(curl, CURLOPT_POST, 1);
        curl_slist *headers = NULL;
        headers = curl_slist_append(headers, "Content-Type:application/json;charset=UTF-8");
        curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);
        curl_easy_setopt(curl, CURLOPT_POSTFIELDS, "{\"sound\":\"sfasq35sadvsvqwr5q...\"}");
        result_code = curl_easy_perform(curl);
        if (result_code != CURLE_OK) {
            fprintf(stderr, "curl_easy_perform() failed: %s\n",
                    curl_easy_strerror(result_code));
            is_success = 1;
            return is_success;
        }
        json_result = easydlSoundClassify_result;
        curl_easy_cleanup(curl);
        is_success = 0;
    } else {
        fprintf(stderr, "curl_easy_init() failed.");
        is_success = 1;
    }
    return is_success;
}
```

返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	number	唯一的log id，用于问题定位
results	否	array(object)	分类结果数组
+name	否	string	分类名称
+score	否	number	置信度

在线调试

EasyDL零基础开发平台提供了 示例代码中心(API调试平台)-示例代码 ，用于帮助开发者在线调试接口，查看在线调用的请求内容和返回结果、复制和下载示例代码等功能，简单易用。

错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- **error_code**：错误码。
- **error_msg**：错误描述信息，帮助理解和解决发生的错误。

例如Access Token失效返回：

```
{
 "error_code": 110,
 "error_msg": "Access token invalid or no longer valid"
}
```

需要重新获取新的Access Token再次请求即可。

错误码	错误信息	描述
1	Unknown error	服务器内部错误，请再次请求， 如果持续出现此类错误，请在百度云控制台内提交工单反馈。
2	Service temporarily unavailable	服务暂不可用，请再次请求， 如果持续出现此类错误，请在百度云控制台内提交工单反馈。
3	Unsupported openapi method	调用的API不存在，请检查后重新尝试
4	Open api request limit reached	集群超限额
6	No permission to access data	无权限访问该用户数据
13	Get service token failed	获取token失败
14	IAM Certification failed	IAM鉴权失败
15	app not exists or create failed	应用不存在或者创建失败
17	Open api daily request limit reached	每天请求量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度云控制台内提交工单反馈
18	Open api qps request limit reached	QPS超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度云控制台内提交工单反馈
19	Open api total request limit reached	请求总量超限额，已上线计费的接口，请直接在控制台开通计费，调用量不受限制，按调用量阶梯计费；未上线计费的接口，请在百度云控制台内提交工单反馈
100	Invalid parameter	无效的access_token参数，请检查后重新尝试
110	Access token invalid or no longer valid	access_token无效
111	Access token expired	access token过期
336000	Internal error	服务器内部错误，请再次请求， 如果持续出现此类错误，请在百度云控制台内提交工单反馈
336001	Invalid Argument	入参格式有误，比如缺少必要参数、图片base64编码错误等等，可检查下图片编码、代码格式是否有误。有疑问请在百度云控制台内提交工单反馈
336002	JSON不合法	入参格式或调用方式有误，比如缺少必要参数或代码格式有误。有疑问请在百度云控制台内提交工单反馈
336003	Base64解码失败	图片/音频/文本格式有误或base64编码有误，请根据接口文档检查格式，base64编码请求时注意要去掉头部。有疑问请在百度云控制台内提交工单反馈
336004	输入文件大小不合法	音频大小超出限制，请根据接口文档检查入参格式，音频文件大小应控制在4M以内，有疑问请在百度云控制台内提交工单反馈
336005	音频解码失败	音频解码失败失败，音频文件非所限定的mp3, m4a, wav格式
336006	缺失必要参数	未上传音频文件，请补充必要参数后重新请求
336100	model temporarily unavailable	遇到该错误码请等待1分钟后再次请求，可恢复正常，若反复重试依然报错或有疑问请在百度云控制台内提交工单反馈