ModelArts

开发环境

文档版本 01

发布日期 2024-04-29

华为云计算技术有限公司

版权所有 © 华为云计算技术有限公司 2024。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

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，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

 版权所有 © 华为云计算技术有限公司

 版权所有 © 华为云计算技术有限公司

 版权所有 © 华为云计算技术有限公司

 版权所有 © 华为云计算技术有限公司

 版权所有 © 华为云计算技术有限公司

 版权所有 © 华为云计算技术有限公司

 版权所有 © 华为云计算技术有限公司

 版权所有 © 华为云计算技术有限公司

1 开发环境介绍

软件开发的历史，就是一部降低开发者成本，提升开发体验的历史。在AI开发阶段，ModelArts也致力于提升AI开发体验，降低开发门槛。ModelArts开发环境，以云原生的资源使用和开发工具链的集成，目标为不同类型AI开发、探索、教学用户，提供更好云化AI开发体验。

ModelArts Notebook云上云下，无缝协同

- 代码开发与调测。云化JupyterLab使用，本地IDE+ModelArts插件远程开发能力，贴近开发人员使用习惯
- 云上开发环境，包含AI计算资源，云上存储，预置AI引擎
- 运行环境自定义，将开发环境直接保存成为镜像，供训练、推理使用

ModelArts CodeLab（JupyterLab），让AI探索&教学更简单

- 云原生Notebook，案例内容秒级接入与分享
- Serverless化实例管理，资源自动回收
- 免费算力，规格按需切换

亮点特性 1：远程开发 - 支持本地 IDE 远程访问 Notebook

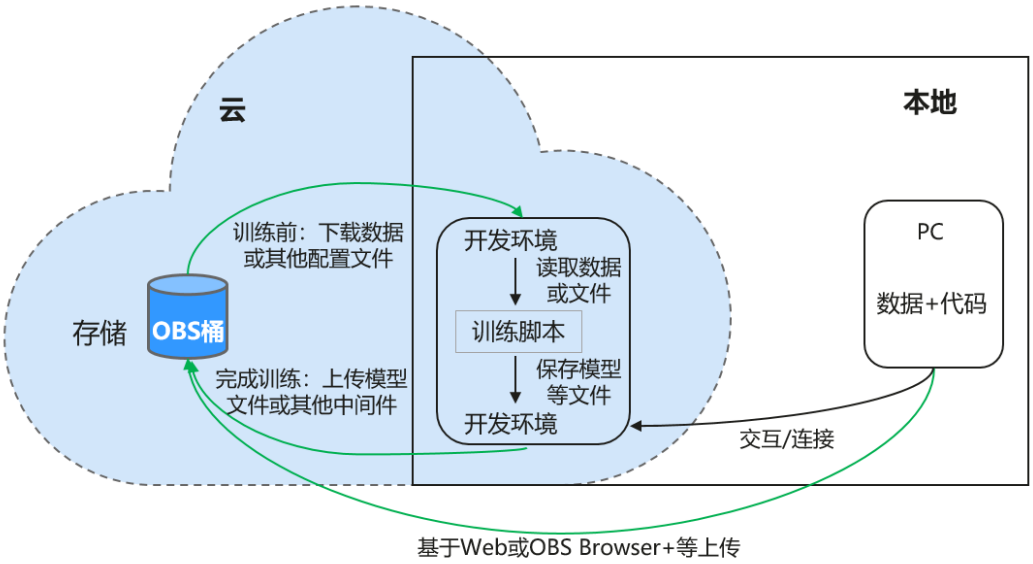
Notebook提供了远程开发功能，通过开启SSH连接，用户本地IDE可以远程连接到ModelArts的Notebook开发环境中，调试和运行代码。

对于使用本地IDE的开发者，由于本地资源限制，运行和调试环境大多使用团队公共搭建的服务器，并且是多人共用，这带来一定的环境搭建和维护成本。

而ModelArts的Notebook的优势是即开即用，它预先装好了不同的AI引擎，并且提供了非常多的可选规格，用户可以独占一个容器环境，不受其他人的干扰。只需简单配置，用户即可通过本地IDE连接到该环境进行运行和调试。

版权所有 © 华为云计算技术有限公司

图 1-1 本地 IDE 远程访问 Notebook 开发环境



ModelArts的Notebook可以视作是本地PC的延伸，均视作本地开发环境，其读取数据、训练、保存文件等操作与常规的本地训练一致。

对于习惯使用本地IDE的开发者，使用远程开发方式，不影响用户的编码习惯，并且可以方便快捷的使用云上的Notebook开发环境。

本地IDE当前支持VS Code、PyCharm、SSH工具。还有专门的插件PyCharm Toolkit和VS Code Toolkit，方便将云上资源作为本地的一个扩展。

亮点特性 2：开发环境保存 - 支持一键镜像保存

ModelArts的Notebook提供了镜像保存功能。支持一键将运行中的Notebook实例保存为镜像，将准备好的环境保存下来，可以作为自定义镜像，方便后续使用，并且方便进行分享。

保存镜像时，安装的依赖包（pip包）不丢失，VS Code远程开发场景下，在Server端安装的插件不丢失。

亮点特性 3：预置镜像 - 即开即用，优化配置，支持主流 AI 引擎

每个镜像预置的AI引擎和版本是固定的，在创建Notebook实例时明确AI引擎和版本，包括适配的芯片。

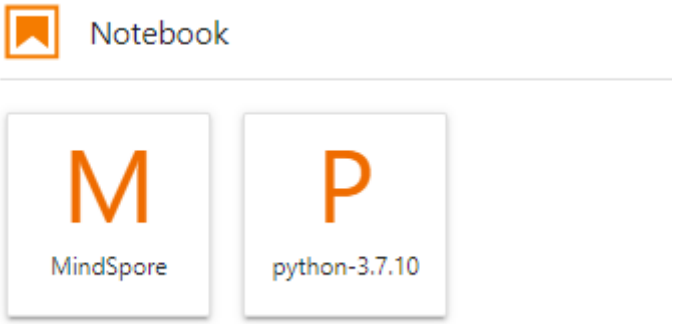
ModelArts开发环境给用户提供了一组预置镜像，主要包括PyTorch、TensorFlow、MindSpore系列。用户可以直接使用预置镜像启动Notebook实例，在实例中开发完成后，直接提交到ModelArts训练作业进行训练，而不需要做适配。

ModelArts开发环境提供的预置镜像版本是依据用户反馈和版本稳定性决定的。当用户的功能开发基于ModelArts提供的版本能够满足的时候，建议用户使用预置镜像，这些镜像经过充分的功能验证，并且已经预置了很多常用的安装包，用户无需花费过多的时间来配置环境即可使用。

ModelArts开发环境提供的预置镜像主要包含：

版权所有 © 华为云计算技术有限公司

- 常用预置包，基于标准的Conda环境，预置了常用的AI引擎，例如PyTorch、MindSpore；常用的数据分析软件包，例如Pandas、Numpy等；常用的工具软件，例如cuda、cudnn等，满足AI开发常用需求。
- 预置Conda环境：每个预置镜像都会创建一个相对应的Conda环境和一个基础Conda环境python（不包含任何AI引擎），如预置MindSpore所对应的Conda环境如下：



  用户可以根据是否使用AI引擎参与功能调试，选择不同的Conda环境。
- Notebook：是一款Web应用，能够使用户在界面编写代码，并且将代码、数学方程和可视化内容组合到一个文档中。
- JupyterLab插件：插件包括规格切换，分享案例到AI Gallery进行交流，停止实例等，提升用户体验。
- 支持SSH远程连接功能，通过SSH连接启动实例，在本地调试就可以操作实例，方便调试。
- ModelArts开发环境提供的预置镜像支持功能开发后，直接提到ModelArts训练作业中进行训练。

说明

- 为了简化操作，ModelArts的Notebook，同一个Notebook实例中支持不同引擎之间的切换。
- 不同Region支持的AI引擎不一样，请以控制台实际界面为准。

亮点特性 4：提供在线的交互式开发调试工具 JupyterLab

ModelArts集成了基于开源的JupyterLab，可为您提供在线的交互式开发调试。您无需关注安装配置，在ModelArts管理控制台直接使用Notebook，编写和调测模型训练代码，然后基于该代码进行模型的训练。

JupyterLab是一个交互式的开发环境，是Jupyter Notebook的下一代产品，可以使用它编写Notebook、操作终端、编辑MarkDown文本、打开交互模式、查看csv文件及图片等功能。

版权所有 © 华为云计算技术有限公司

2 使用场景

ModelArts提供灵活开放的开发环境，您可以根据实际情况选择。

- ModelArts提供了云化版本的Notebook，无需关注安装配置，即开即用，具体参见**JupyterLab简介及常用操作**。
- ModelArts也提供了本地IDE的方式开发模型，通过开启SSH连接，用户本地IDE可以远程连接到ModelArts的Notebook开发环境中，调试和运行代码。本地IDE方式不影响用户的编码习惯，并且可以方便快捷的使用云上的Notebook开发环境。

  本地IDE当前支持VS Code、PyCharm、SSH工具。PyCharm和VS Code还分别有专门的插件PyCharm Toolkit、VS Code Toolkit，让远程连接操作更便捷。具体参见**VS Code一键连接Notebook**。
- ModelArts还提供了可视化编排工具ML Studio，用户可以通过拖拉拽的方式开发基于ML Studio引擎的模型，具体参见**ML Studio简介**。

版权所有 © 华为云计算技术有限公司

3 管理 Notebook 实例

3.1 创建 Notebook 实例

在开始进行模型开发前，您需要创建Notebook实例，并打开Notebook进行编码。

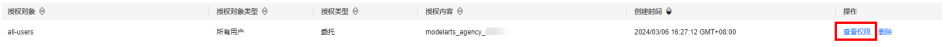
背景信息

- Notebook使用涉及到计费，具体收费项如下：
  - 处于“运行中”状态的Notebook，会消耗资源，产生费用。根据您选择的资源不同，收费标准不同，价格详情请参见**产品价格详情**。当您不需要使用Notebook时，建议停止Notebook，避免产生不必要的费用。
  - 创建Notebook时，如果选择使用云硬盘EVS存储配置，云硬盘EVS会一直收费，建议及时停止并删除Notebook，避免产品不必要的费用。
- 在创建Notebook时，默认会开启自动停止功能，在指定时间内停止运行Notebook，避免资源浪费。
- 只有处于“运行中”状态的Notebook，才可以执行打开、停止操作。
- 一个账户最多创建10个Notebook。

创建 Notebook 实例

1. 登录ModelArts管理控制台，在左侧导航栏中选择“全局配置”，检查是否配置了访问授权。如果未配置，请先配置访问授权。参考**使用委托授权**完成操作。

   图 3-1 查看委托配置信息

2. 登录ModelArts管理控制台，在左侧导航栏中选择“开发环境 > Notebook”，进入“Notebook”页面。
3. 单击右上角“创建”，进入“创建Notebook”页面，请参见如下说明填写参数。
   a. 填写Notebook基本信息，包含名称、描述、是否自动停止，详细参数请参见**表3-1**。

版权所有 © 华为云计算技术有限公司

图 3-2 Notebook 基本信息

表 3-1 基本信息的参数描述

参数名称	说明
“名称”	Notebook的名称。系统会自动生成一个名称，您可以根据业务需求重新命名，命名规则如下：只能包含数字、大小写字母、下划线和中划线，长度不能超过128位且不能为空。
“描述”	对Notebook的简要描述。
“自动停止”	默认开启，且默认值为“1小时”，表示该Notebook实例将在运行1小时之后自动停止，即1小时后停止规格资源计费。可选择“1小时”、“2小时”、“4小时”、“6小时”或“自定义”几种模式。选择“自定义”模式时，可指定1~24小时范围内任意整数。 • 定时停止：开启定时停止功能后，该Notebook实例将在运行时长超出您所选择的时长后，自动停止。 **说明** 出于对用户任务进度的保护，在您设置的自动停止时间到达后，Notebook不会立即自动停止，可能会有2-5分钟的延迟，方便您进行续约。

b. 填写Notebook详细参数，如镜像、资源规格等，详细参数请参见表3-2。

版权所有 © 华为云计算技术有限公司

图 3-3 Notebook 实例的详细参数

表 3-2 Notebook 实例的详细参数说明

参数名称	说明
“镜像”	支持公共镜像和自定义镜像。 • 公共镜像：即预置在ModelArts内部的AI引擎。可以选择界面显示的公共镜像，也可以单击“前往AI Gallery获取更多镜像”进入AI Gallery镜像页面。**AI Gallery**上发布了一些较高版本的PyTorch、MindSpore、TensorFlow镜像。进入AI Gallery镜像页面后，单击镜像名称，可查看镜像详情，复制对应局点的镜像URL，即可在ModelArts控制台“镜像管理”注册并在Notebook中使用。 如果需要使用更高版本的MindSpore类型的Ascend架构镜像，可选择**西南-贵阳一**局点。 • 自定义镜像：可以将基于公共镜像创建的实例保存下来，作为自定义镜像使用。自定义镜像的介绍及制作请参考**在Notebook中使用自定义镜像**。 一个镜像对应支持一种AI引擎，创建Notebook实例时选择好了对应AI引擎的镜像。用户可以根据需要选择镜像。在右侧搜索框中输入镜像名称关键字，可快速查找镜像。 Notebook运行停止后，可以在同一个Notebook实例中变更镜像。
“资源类型”	支持公共资源池和专属资源池。 “公共资源池”无需单独购买，即开即用，按需付费，即按您的Notebook实例运行时长进行收费。 “专属资源池”按实际情况选择已创建的专属资源池。如果没有专属资源，需要单独购买并创建。

版权所有 © 华为云计算技术有限公司

参数名称	说明
“类型”	芯片类型包括CPU、GPU和ASCEND类型。 不同的镜像支持的芯片类型不同，根据实际需要选择。 GPU性能更佳，但是相对CPU而言，费用更高。
“规格”	根据选择的芯片类型不同，可选资源规格也不同。请根据界面实际情况和需要选择。 ● CPU规格 “2核8GB”： Intel CPU通用规格，用于快速数据探索和实验 “8核32GB”： Intel CPU算力增强型，适用于密集计算场景下运算 ● GPU规格 “GPU: 1*Vnt1(32GB)|CPU: 8 核 64GB”： GPU单卡规格，32GB显存，适合深度学习场景下的算法训练和调测 “GPU: 1*Tnt004(16GB)|CPU: 8核* 32GB”： GPU单卡规格，16GB显存，推理计算最佳选择，覆盖场景包括计算机视觉、视频处理、NLP等 “GPU: 1*Pnt1(16GB)|CPU: 8核 64GB”： GPU单卡规格，16GB显存，适合深度学习场景下的算法训练和调测 ● Ascend规格 有Snt9(32GB显存)单卡、两卡、八卡等规格。配搭ARM处理器，适合深度学习场景下的模型训练和调测。

版权所有 © 华为云计算技术有限公司

参数名称	说明
“存储配置”	包括“云硬盘EVS”、“弹性文件服务SFS”、“对象存储服务OBS”和“并行文件系统PFS”。请根据界面实际情况和需要选择。 **说明** “对象存储服务OBS”和“并行文件系统PFS”是白名单功能，如果有试用需求，请提工单申请权限。 • 选择“云硬盘EVS”作为存储位置。 根据实际使用量设置磁盘规格。磁盘规格默认5GB。磁盘规格的最大值请以实际界面显示为准。 从Notebook实例创建成功开始，直至实例删除成功，磁盘每GB按照规定费用收费。 • 选择“弹性文件服务SFS”作为存储位置。 仅专属资源池支持，并需要在专属资源池对应的网络打通VPC才能生效，具体操作请参见**ModelArts网络**。 **说明** 如果需要设置SFS Turbo的文件夹权限，请参考**权限管理**文档配置。 – “弹性文件服务”：选择已创建的SFS Turbo（在**弹性文件服务控制台**创建SFS Turbo）。 – “云上挂载路径”：默认为/home/ma-user/work/。 – “子目录挂载”：选择SFS Turbo的存储位置。 – “挂载方式”：当用户配置了文件夹控制权限，则显示此参数。根据SFS Turbo存储位置的权限显示“读写”或“只读”。 • 选择“对象存储服务OBS”或“并行文件系统PFS”作为存储位置。 选择“存储位置”：设置用于存储Notebook数据的OBS路径。如果想直接使用已有的文件或数据，可将数据提前上传至对应的OBS路径下。“存储位置”不能设置为OBS桶的根目录，需设置为对应OBS桶下的具体目录。 选择“凭据”：选择已有的凭据或单击右侧的“立即创建”，跳转至数据加密控制台**创建凭据**，凭据键/值填写用户的AK、SK信息（“键”分别填写“accessKeyId”，“secretAccessKey”；“值”在控制台个人账号下“我的凭证>访问密钥”获取AK、SK）。 **图 3-4** 设置凭据值  “云硬盘EVS”和“弹性文件服务SFS”的存储路径挂载在/home/ma-user/work目录下。用户在Notebook实例中的所有文件读写操作都是针对该存储目录下的内容操作，与OBS对象存储（OBS对象桶）无关。

版权所有 © 华为云计算技术有限公司

参数名称	说明
	Notebook实例运行中，可以通过**动态挂载OBS并行文件系统**操作来增加数据存储路径。 停止或重启Notebook实例时，存储的内容会被保留，不丢失。 删除Notebook实例时，EVS存储会一起释放，存储的内容不保留。SFS可以重新挂载到新的Notebook，可以保留数据。
“扩展存储配置”	**说明** “扩展存储配置”功能是白名单功能，如果有试用需求，请提工单申请权限。 如果有多个数据存储路径，可以单击“增加扩展存储配置”，增加用户指定的存储挂载目录。支持增加的存储类型有“存储桶OBS”、“并行文件系统PFS”、“弹性文件服务SFS”。 约束限制： ● 每种存储类型最多支持挂载5个。 ● 扩展存储挂载目录不允许重复，不允许挂载到黑名单目录，允许嵌套挂载。不允许挂载的黑名单目录为以下前缀匹配的目录：/data/、/cache/、/dev/、/etc/、/bin/、/lib/、/sbin/、/modelarts/、/train-worker1-log/、/var/、/resource_info/、/usr/、/sys/、/run/、/tmp/、/infer/、/opt/ 添加扩展存储后，可进入Notebook实例详情页，单击“存储配置 > 扩展存储”，查看或编辑扩展存储信息。在存储个数未达到最大个数时，也可在右侧单击“添加扩展存储”。
“SSH远程开发”	● 开启此功能后，用户可以在本地开发环境中远程接入Notebook实例的开发环境。 ● 实例在停止状态时，用户可以在Notebook详情页中更新SSH的配置信息。 **说明** 开启此功能的实例中会预置VS Code插件（python、jupyter等）以及VS Code Server包，会占用约1G左右的持久化存储空间。
“密钥对”	开启“SSH远程开发”功能后，需要设置此参数。 可以选择已有密钥对。 也可以单击密钥对右侧的“立即创建”，跳转到数据加密控制台，在“密钥对管理 > 私有密钥对”页面，单击“创建密钥对”。 创建完Notebook后，可以在Notebook详情页中修改密钥对。 **注意** 创建好的密钥对，请下载并妥善保存，使用本地IDE远程连接云上Notebook开发环境时，需要用到密钥对进行鉴权认证。

版权所有 © 华为云计算技术有限公司

参数名称	说明
“远程访问白名单”	可选，开启“SSH远程开发”功能后，可以设置此参数。 设置为允许远程接入访问这个Notebook的IP地址（例如本地PC的IP地址或者访问机器的外网IP地址，最多配置5个，用英文逗号隔开），不设置则表示无接入IP地址限制。 如果用户使用的访问机器和ModelArts服务的网络有隔离，则访问机器的外网地址需要在主流搜索引擎中搜索“IP地址查询”获取，而不是使用**ipconfig**或**ifconfig/ip**命令在本地查询。 **图 3-5** 查询外网 IP 地址  创建完Notebook后，可以在Notebook详情页中修改白名单IP地址。

c. 可选：添加Notebook标签。在标签栏输入“标签键”和“标签值”，单击“添加”。

表 3-3 添加标签

参数名	参数说明
“标签”	ModelArts支持对接标签管理服务TMS，在ModelArts中创建资源消耗性任务（例如：创建Notebook、训练作业、推理在线服务）时，可以为这些任务配置标签，通过标签实现资源的多维分组管理。 标签详细用法请参见**ModelArts如何通过标签实现资源分组管理**。 添加标签后，可在Notebook实例详情页查看标签内容，也可进行修改、删除标签。

说明

可以在标签输入框下拉选择TMS预定义标签，也可以自己输入自定义标签。预定义标签对所有支持标签功能的服务资源可见。租户自定义标签只对自己服务可见。

4. 参数填写完成后，单击“立即创建”进行规格确认。

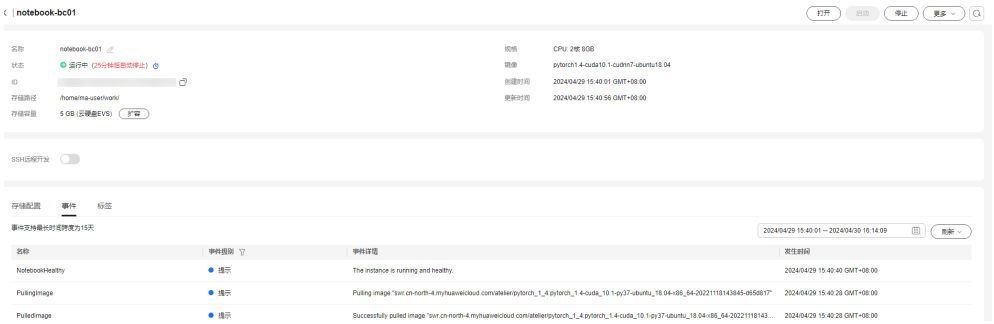
版权所有 © 华为云计算技术有限公司

5. 参数确认无误后，单击“提交”，完成Notebook的创建操作。

   进入Notebook列表，正在创建中的Notebook状态为“创建中”，创建过程需要几分钟，请耐心等待。当Notebook状态变为“运行中”时，表示Notebook已创建并启动完成。

6. 在Notebook列表，单击实例名称，进入实例详情页，查看Notebook实例配置信息。

   图 3-6 查看 Notebook 实例详情

   “SSH远程开发”功能开启时，在“白名单”右侧单击修改，可以修改允许远程访问的白名单IP地址。实例在停止状态时，在“认证”右侧单击修改，用户可以更新密钥对。

   单击“存储配置”页签的“添加数据存储”，可以挂载OBS并行文件系统，方便读取数据，具体操作参见**动态挂载OBS并行文件系统**。

   如果存储使用的是云硬盘EVS，单击存储容量右侧的“扩容”，可以动态扩充云硬盘EVS的容量，具体操作参见**动态扩充云硬盘EVS容量**。

3.2 打开 Notebook 实例

针对创建好的Notebook实例（即状态为“运行中”的实例），可以打开Notebook并在开发环境中启动编码。

基于不同AI引擎创建的Notebook实例，打开方式不一样。

pytorch、tensorflow、mindspore、tensorflow-mindspore、cylp-cbcpy、rlstudio-ray、mindquantum-mindspore镜像支持以下2种方式访问：

- 本地IDE使用PyCharm/VS Code/SSH工具，远程连接访问，具体参见**VS Code一键连接Notebook**。
- 在线JupyterLab访问，具体参见**JupyterLab简介及常用操作**。

mlstudio-pyspark镜像仅支持在线JupyterLab访问，具体参见**JupyterLab简介及常用操作**

modelbox镜像仅支持在本地使用VS Code插件远程访问，具体参见**配置本地VSCode连接云上开发环境Modelbox镜像**。

创建实例，持久化存储挂载路径为/home/ma-user/work目录。

版权所有 © 华为云计算技术有限公司

```
sh-4.4$pwd
/home/ma-user
sh-4.4$cd work/
sh-4.4$pwd
/home/ma-user/work
sh-4.4$
```

存放在work目录的内容，在实例停止、重新启动后依然保留，其他目录下的内容不会保留，使用开发环境时建议将需要持久化的数据放在/home/ma-user/work目录。

3.3 查找/启动/停止/删除实例

查找实例

Notebook页面展示了所有创建的实例。如果需要查找特定的实例，可根据筛选条件快速查找。单击搜索框，可按照属性类型选择单个条件来筛选，或者同时选择多个筛选条件筛选。

图 3-7 查找实例

- 打开“查看所有”开关，可以看到IAM项目下所有子用户创建的Notebook实例。
- 按实例名称、实例ID、实例状态、使用的镜像、实例规格、实例描述、创建时间等单个筛选或组合筛选。

表格显示设置

单击设置按钮可自定义设置需要显示在表格中的列。

版权所有 © 华为云计算技术有限公司

图 3-8 设置表格列显示

启动/停止实例

由于运行中的Notebook将一直耗费资源，您可以通过停止操作，停止资源消耗。对于停止状态的Notebook，可通过启动操作重新使用Notebook。

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“开发环境 > Notebook”，进入Notebook管理页面。
2. 执行如下操作启动或停止Notebook。
   - **启动Notebook**：单击“操作”列的“启动”。只有处于“停止”状态的Notebook可以执行启动操作。
   - **停止Notebook**：单击“操作”列的“停止”。只有处于“运行中”状态的Notebook可以执行停止操作。

注意

Notebook停止后：

- /home/ma-user/work目录下的数据会保存，其余目录下内容会被清理。例如：用户在开发环境中的其他目录下安装的外部依赖包等，在Notebook停止后会被清理。
- Notebook实例将停止计费，但如有EVS盘挂载，存储部分仍会继续计费。

删除实例

针对不再使用的Notebook实例，可以删除以释放资源。

版权所有 © 华为云计算技术有限公司

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“开发环境>Notebook”，进入Notebook页面。
2. 在Notebook列表中，单击操作列的“删除”，在弹出的确认对话框中，确认信息无误，然后输入“DELETE”，单击“确定”，完成删除操作。

> **⚠ 注意**
>
> Notebook删除后不可恢复，请谨慎操作。实例删除后，挂载目录下的数据也将一并删除，请谨慎操作。

3.4 变更 Notebook 实例镜像

ModelArts允许用户在同一个Notebook实例中切换镜像，方便用户灵活调整实例的AI引擎。

约束限制

Notebook实例状态必须在“停止”中。

变更实例镜像操作

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“开发环境 > Notebook”，进入Notebook页面。
2. 在Notebook列表，单击某个Notebook实例操作栏的“更多 > 变更镜像”，在变更镜像窗口选择新的镜像，单击“确定”。

图 3-9 变更镜像

3. 在镜像窗口选择新的镜像，单击“确定”，变更成功后，在Notebook列表页的镜像栏，可以查看到变更后的镜像。

3.5 变更 Notebook 实例运行规格

ModelArts允许用户在同一个Notebook实例中切换节点运行规格，方便用户灵活调整规格资源。

版权所有 © 华为云计算技术有限公司

约束限制

只有处于“停止”、“运行中”和“启动失败”的Notebook实例才能变更规格。

变更实例规格操作

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“开发环境 > Notebook”，进入Notebook页面。
2. 在Notebook列表，单击某个Notebook实例操作列的“更多 > 变更规格”，在弹出的“变更规格”对话框中选择对应规格资源。

图 3-10 变更规格

图 3-11 选择规格

说明

规格切换需要该规格所在的集群有其他规格才可以执行，当前上线的部分规格所在集群无其他规格，切换的时候会显示为空，所以不可进行切换，如北京四、上海一的GPU: 1*T4(16GB)|CPU: 8核 32GB规格。

3.6 开发环境中如何选择存储

不同存储的实现都不同，在性能、易用性、成本的权衡中可以有不同的选择，没有一个存储可以覆盖所有场景，了解下云上开发环境中各种存储使用场景说明，更能提高使用效率。

版权所有 © 华为云计算技术有限公司

说明

仅支持挂载同一区域下的OBS并行文件系统（PFS）和OBS对象存储。

表 3-4 云上开发环境中各种存储使用场景说明

存储类型	建议使用场景	优点	缺点
云硬盘EVS	比较适合只在开发环境中做数据、算法探索，性能较好。	块存储SSD，可以理解为一个磁盘，整体IO性能比NFS要好，可以动态扩充，最大可以到4096GB。 云硬盘EVS作为持久化存储挂载在/home/ma-user/work目录下，该目录下的内容在实例停止后会被保留，存储支持在线按需扩容。	缺点是只能在单个开发环境中使用 。

版权所有 © 华为云计算技术有限公司

存储类型	建议使用场景	优点	缺点
并行文件系统PFS	**说明** 并行文件系统PFS为白名单功能，如需使用，请联系华为技术支持开通。 适合直接使用PFS桶作为持久化存储进行AI开发和探索场景。 1. 数据集的存储。将数据集直接挂载到Notebook进行浏览和数据处理，在训练时直接使用。本地数据上传请参考**如何上传文件到OBS?** 或在实例运行后，将承载数据集的OBS并行文件系统动态挂载至Notebook中，详细操作请参考**动态挂载OBS并行文件系统**。 2. 代码的存储。在Notebook调测完成，可以直接指定对应的对象存储路径作为启动训练的代码路径，方便临时修改。 3. 训练观测。可以将训练日志等输出路径进行挂载，在Notebook中实时查看和观测，特别是利用TensorBoard，Notebook功能完成对训练输出的分析。	PFS是一种经过优化的高性能对象存储文件系统，存储成本低，吞吐量大，能够快速处理高性能计算（HPC）工作负载。在需要使用对象存储服务场景下，推荐使用PFS挂载。 **说明** 建议上传时按照128MB或者64MB打包或者切分，使用时边下载边解压后在本地存储读取，以获取更好的读写与吞吐性能。	缺点是小文件频繁读写性能较差，例如直接作为存储用于模型重型训练，大文件解压等场景慎用。 **说明** PFS挂载需要用户对当前桶授权给ModelArts完整读写权限，Notebook删除后，此权限策略不会被删除。
对象存储服务OBS	**说明** OBS对象存储为白名单功能，如需使用，请联系华为技术支持开通。 在开发环境中做大规模的数据上传下载时，可以通过OBS桶做中转。	存储成本低，吞吐量大，但是小文件读写较弱。建议上传时按照128MB或者64MB打包或者切分，使用时边下载边解压后在本地读取。	对象存储语义，和Posix语义有区别，需要进一步理解。

版权所有 © 华为云计算技术有限公司

存储类型	建议使用场景	优点	缺点
弹性文件服务SFS	目前只支持在专属资源池中使用；针对探索、实验等非正式生产场景，建议使用这种。开发环境和训练环境可以同时挂载一块SFS存储，省去了每次训练作业下载数据的要求，一般来说重IO读写模型，超过32卡的大规模训练不适合。	实现为NFS，可以在多个开发环境、开发环境和训练之间共享，如果不需要重型分布式训练作业，特别是启动训练作业时，不需要额外再对数据进行下载，这种存储便利性可以作为首选。	缺点性能比EVS云硬盘块存储低。
本地存储	重型训练任务首选	运行所在虚拟机或者裸金属机器上自带的SSD高性能存储，文件读写的吞吐量大，建议对于重型训练任务先将数据准备到对应目录再启动训练。 默认在容器/cache目录下进行挂载，/cache目录可用空间请参考**开发环境中不同Notebook规格资源“/cache”目录的大小**。	缺点是存储生命周期和容器生命周期绑定，每次训练都要下载数据。

如何使用

1. 在开发环境中如何使用云硬盘EVS块存储?

   例如，在**创建Notebook实例**时选择云硬盘EVS存储小容量，Notebook运行过程中如果发现存储容量不够，可以扩容，请参考**动态扩充云硬盘EVS容量**。

2. 在开发环境中如何使用OBS并行文件系统?

   例如，在Notebook中训练时，可直接使用挂载至Notebook容器中的数据集，在运行过程中可以**动态挂载OBS并行文件系统**。

3.7 动态挂载 OBS 并行文件系统

什么是动态挂载 OBS 并行文件系统

并行文件系统（Parallel File System）是对象存储服务（Object Storage Service，OBS）提供的一种经过优化的高性能文件系统，详细介绍可以参见**并行文件系统**。

在ModelArts运行态的Notebook容器中，采用动态挂载特性，将OBS对象存储模拟成本地文件系统。其本质是通过挂载工具，将对象协议转为POSIX文件协议。挂载后应用层可以在容器中正常操作OBS对象。

版权所有 © 华为云计算技术有限公司

动态挂载适用于哪些使用场景

场景1：数据集预览和操作，将承载数据集的OBS挂载至Notebook中，可以像本地文件系统一样操作数据集。

场景2：在Notebook中训练时，可直接使用挂载至Notebook容器中的数据集。

动态挂载 OBS 并行文件系统有什么限制

OBS提供两种桶，对象存储（对象桶）和并行文件系统PFS。

ModelArts的Notebook仅支持挂载OBS的**并行文件系统**，挂载至Notebook容器“/data/”的子目录下。

动态挂载 OBS 并行文件系统操作

方式1：通过ModelArts控制台操作

1. 登录ModelArts管理控制台，在左侧导航栏中选择“开发环境 > Notebook”，进入“Notebook”页面。
2. 选择运行中的Notebook实例，单击实例名称，进入Notebook实例详情页面，在“存储配置”页签，单击“添加数据存储”，设置挂载参数。
   a. 设置本地挂载目录，在“/data/”目录下输入一个文件夹名称，例如：demo。挂载时，后台自动会在Notebook容器的“/data/”目录下创建该文件夹，用来挂载OBS文件系统。
   b. 选择存放OBS并行文件系统下的文件夹，单击“确定”。

   图 3-12 动态挂载 OBS 并行文件系统

3. 挂载成功后，可以在Notebook实例详情页查看到挂载结果。

   图 3-13 挂载成功

方式2：API模式

动态挂载API接口已发布至华北-北京四和华东-上海一站点。请参考**打开JupyterLab**、**在JupyterLab中新建ipynb文件**，新建一个ipynb文件然后执行脚本。

挂载脚本代码示例如下。更多API参数介绍请参考**动态挂载OBS**。

版权所有 © 华为云计算技术有限公司

```
import os
from json import JSONEncoder
from modelarts.config.auth import auth_by_apig
from modelarts.session import Session

session = Session()
request_url = "/v1/{}/notebooks/{}/storage".format(os.environ['PROJECT_ID'], os.environ["INSTANCE_ID"])


# 查询动态挂载列表
auth_by_apig(session, 'GET', request_url)


# 动态挂载，OBS路径obs://obs-bucket-train/dir/需要根据实际修改，挂载路径mount_path取值为“/data/xxx/”，本示例以“/data/demo/”为例，“demo”可以自定义。
body = {
    "category": "OBS",
    "uri": "obs://obs-bucket-train/dir/",
    "mount_path": "/data/demo/"
}
auth_by_apig(session, 'POST', request_url, body=JSONEncoder().encode(body))

# 动态卸载
auth_by_apig(session, 'DELETE', request_url + "/ea217c4f-3282-4af2-98ea-d3c668d2fba9")
```

3.8 动态扩充云硬盘 EVS 容量

什么是动态扩容 EVS

存储配置采用云硬盘EVS的Notebook实例， 存储盘是挂载至容器/home/ma-user/work/目录下， 可以在实例运行中的状态下，动态扩充存储盘容量，单次最大动态扩容100GB。

动态扩容 EVS 适用于哪些使用场景

在Notebook开发过程中，初期存储使用量较小时， 创建Notebook可以选择小容量EVS， 比如5G大小； 开发完成后，需要大规模数据集训练，此时再将存储容量扩容至当前阶段所需容量，可以节约成本。

动态扩容 EVS 有什么限制

- Notebook实例的存储配置采用的是云硬盘EVS。

  图 3-14 创建 Notebook 实例时选择云硬盘 EVS 存储

- 单次最大可以扩容100GB，扩容后的总容量不超过4096GB。
- 云硬盘EVS存储容量最大支持4096GB，达到4096GB时，不允许再扩容。
- 实例停止后，扩容后的容量仍然有效。计费也是按照扩容后的云硬盘EVS容量进行计费。

版权所有 © 华为云计算技术有限公司

- 云硬盘EVS只要使用就会计费，请在停止Notebook实例后，确认不使用就及时删除数据，释放资源，避免产生费用。

动态扩容 EVS 操作

1. 登录ModelArts管理控制台，在左侧导航栏中选择“开发环境 > Notebook”，进入“Notebook”页面。
2. 选择运行中的Notebook实例，单击实例名称，进入Notebook实例详情页面，单击“扩容”。

图 3-15 Notebook 实例详情页

3. 设置待扩充的存储容量大小，单击“确定”。系统显示“扩容中”，扩容成功后，可以看到扩容后的存储容量。

图 3-16 扩容

图 3-17 扩容中

版权所有 © 华为云计算技术有限公司

3.9 修改 Notebook SSH 远程连接配置

ModelArts允许用户在Notebook实例中更改SSH配置信息。

在创建Notebook实例时，如果未配置SSH远程连接，当用户需要开启远程连接时，则可以在Notebook的实例详情页打开SSH的配置信息开关；

在创建Notebook实例时，如果设置了允许远程连接Notebook的白名单IP地址，当用户需要更换一个IP地址远程连接Notebook实例时，则可以在Notebook的实例详情页修改白名单IP地址，也可更换密钥对。

约束限制

Notebook实例状态必须在“停止”中。

修改密钥对和远程连接 IP 地址

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“开发环境 > Notebook”，进入Notebook页面。
2. 在Notebook列表，进入Notebook实例详情页。打开SSH远程开发开关，更新密钥对和白名单。

说明

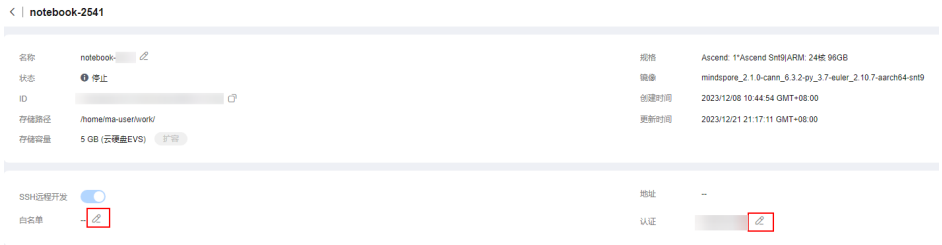
“远程SSH开发”开关可以手动打开的场景，请打开远程SSH开发开关，参考图3-18操作。SSH配置信息更新后，“远程SSH开发”开关打开后不可关闭。

“所选镜像必须配置SSH远程开发”的场景，请参考图3-19操作。

图 3-18 更新 SSH 配置信息

版权所有 © 华为云计算技术有限公司

图 3-19 修改白名单和密钥对



- 密钥对可单击 选择已有的密钥对或“立即创建”创建新的密钥对。
- 白名单IP地址的设置请参考**设置远程连接IP地址**。修改远程连接的可访问IP地址后， 原来已经建立的链接依然有效， 当链接关闭后失效；新打开建立的链接只允许当前设置的IP进行访问。

设置远程连接 IP 地址

图 3-20 设置远程连接 IP 地址

此处的IP地址，请填写外网IP地址。如果用户使用的访问机器和华为云ModelArts服务的网络有隔离，则访问机器的外网地址需要在主流搜索引擎中搜索“IP地址查询”获取，而不是使用ipconfig或ifconfig/ip命令在本地查询。

版权所有 © 华为云计算技术有限公司

图 3-21 查询外网 IP 地址

3.10 查看所有子账号的 Notebook 实例

当子用户被授予“listAllNotebooks”和“listUsers”权限时，在Notebook页面上，单击“查看所有”，可以看到IAM项目下所有子用户创建的Notebook实例 。

说明

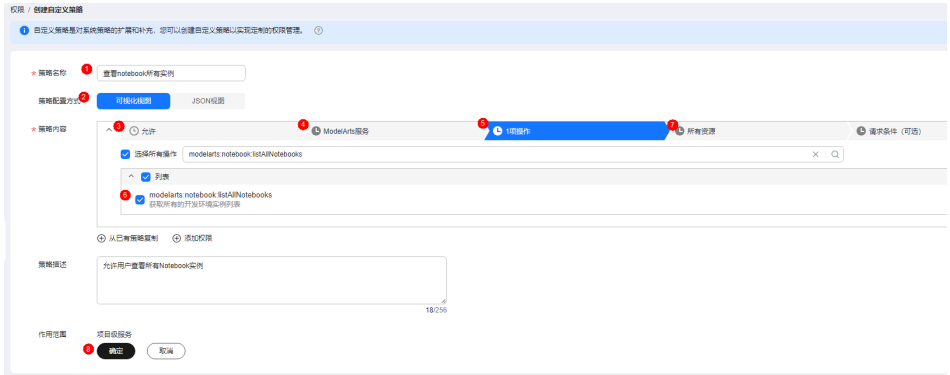
配置该权限后，可以查看子用户的Notebook，也可以在Notebook中访问子用户的OBS、SWR等。

给子账号配置查看所有 Notebook 实例的权限

1. 使用主用户账号登录ModelArts管理控制台，单击右上角用户名，在下拉框中选择“统一身份认证”，进入统一身份认证（IAM）服务。
2. 在统一身份认证服务页面的左侧导航选择“权限管理 > 权限”，单击右上角的“创建自定义策略”，需要设置两条策略。

   策略1：设置查看Notebook所有实例，如图3-22所示，单击“确定”。
   - “策略名称”：设置自定义策略名称，例如：查看Notebook所有实例。
   - “策略配置方式”：选择可视化视图。
   - “策略内容”：允许，云服务中搜索ModelArts服务并选中，操作列中搜索关键词modelarts:notebook:listAllNotebooks并选中，所有资源选择默认值。

   图 3-22 创建自定义策略

   策略2：设置查看Notebook实例创建者信息的策略。
   - “策略名称”：设置自定义策略名称，例如：查看所有子用户信息。

版权所有 © 华为云计算技术有限公司

- “策略配置方式”：选择可视化视图。
- “策略内容”：允许，云服务中搜索IAM服务并选中，操作列中搜索关键词iam:users:listUsers并选中，所有资源选择默认值。

3. 在统一身份认证服务页面的左侧导航选择“用户组”，在用户组页面查找待授权的用户组名称，在右侧的操作列单击“授权”，勾选步骤2创建的两条自定义策略，单击“下一步”，选择授权范围方案，单击“确定”。

   此时，该用户组下的所有用户均有权限查看该用户组内成员创建的所有Notebook实例。

   如果没有用户组，也可以创建一个新的用户组，并通过“用户组管理”功能添加用户，并配置授权。如果指定的子用户没有在用户组中，也可以通过“用户组管理”功能增加用户。

子用户启动其他用户的 SSH 实例

子用户可以看到所有用户的Notebook实例后，如果要通过SSH方式远程连接其他用户的Notebook实例，需要将SSH密钥对更新成自己的，否则会报错ModelArts.6786。更新密钥对具体操作请参见**修改Notebook SSH远程连接配置**。

具体的错误信息提示：ModelArts.6789: 在ECS密钥对管理中找不到指定的ssh密钥对xxx，请更新密钥对并重试。

3.11 查看 Notebook 实例事件

在Notebook的整个生命周期，包括实例的创建、启动、停止、规格变更等关键操作以及实例的运行状态等在后台都有记录，用户可以在Notebook实例详情页中查看具体的事件，通过实例的事件，从而看到实例的运行或者异常等状态详情。在右侧可以手动刷新事件，也可以设置间隔30秒，1分钟，5分钟自动刷新事件。

图 3-23 查看 Notebook 实例事件并设置自动刷新

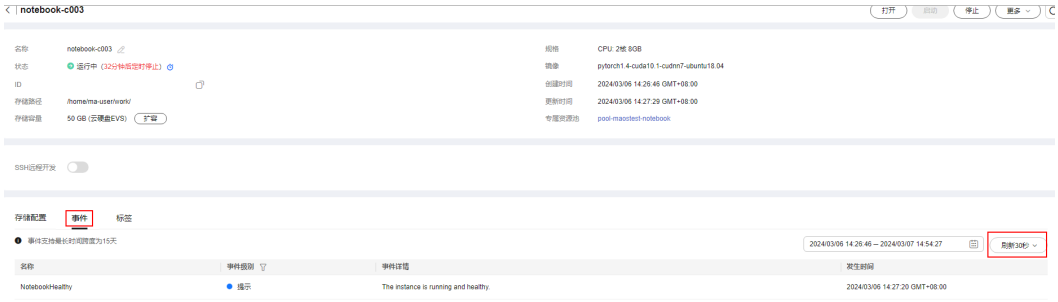


表 3-5 实例创建过程的事件列表

事件名称	事件描述	事件级别
Scheduled	实例被调度成功	提示
PullingImage	正在拉取镜像	提示
PulledImage	镜像拉取完毕	提示
NotebookHealthy	实例运行中，处于健康状态	重要

版权所有 © 华为云计算技术有限公司

事件名称	事件描述	事件级别
CreateNotebookFailed	创建实例失败	紧急
PullImageFailed	镜像拉取失败	紧急

表 3-6 实例启动过程的事件列表

事件名称	事件描述	事件级别
Scheduled	实例被调度成功	提示
PullingImage	正在拉取镜像	提示
PulledImage	镜像拉取完毕	提示
NotebookHealthy	实例运行中，处于健康状态	重要
RunHookScript	运行自定义脚本	提示
StartNotebookFailed	实例启动失败	紧急
PullImageFailed	镜像拉取失败	紧急
CreateKernelFailed	conda命令不可用导致创建jupyter kernel失败 （The jupyter launcher page does not contain the kernel due to conda environment issues, please ensure that {conda_env} is available and the command: {conda_cmdt} env list can be run properly）	重要
	权限问题导致创建jupyter kernel失败 （The jupyter launcher page does not contain the kernel due to permission issues, please ensure that the uid {ma_uid} have write permissions to {conda_path}）	重要
ConfigurationError	conda命令不可用导致配置modelarts sdk和ma-cli路径到conda env失败 （The modelarts sdk and cli is unavailable in the conda envs due to conda environment issues, please ensure that the {conda_env} is available and the command: {conda_cmd} env list can be run properly）	重要

版权所有 © 华为云计算技术有限公司

事件名称	事件描述	事件级别
	权限问题导致配置modelarts sdk和ma-cli路径到conda env失败 （The modelarts sdk and cli is unavailable in the conda env due to permission issues,please ensure that the uid {ma_uid} have write permissions to {conda_path}）	重要

表 3-7 实例停止过程的事件列表

事件名称	事件描述	事件级别
StopNotebook	实例停止	重要
StopNotebookResourceIdle	实例因资源空闲即将自动停止或实例因资源空闲自动停止	重要

表 3-8 更新实例过程的事件列表

事件名称	事件描述	事件级别
UpdateName	更新实例名称	提示
UpdateDescription	更新实例描述	提示
UpdateFlavor	更新实例规格	重要
UpdateImage	更新实例镜像	重要
UpdateStorageSize	实例存储正在扩容 （User %s is updating storage size from %sGB to %sGB）	重要
	实例扩容完成 （User %s updated storage size successfully）	重要
UpdateKeyPair	配置实例密钥对 （User %s updated the instance keypair to "{%s}"）	重要
	更新实例密钥对 （User %s updated the instance keypair from %s to %s）	重要
UpdateWhitelist	更新实例访问白名单	重要
UpdateHook	更新自定义脚本	重要

版权所有 © 华为云计算技术有限公司

事件名称	事件描述	事件级别
UpdateStorageSizeFailed	资源售罄引起的实例存储扩容失败 （The EVS disk is sold out）	紧急
	内部错误引起的实例扩容失败 （The EVS disk size updated failed. Operations and maintenance personnel are handling the problem）	紧急

表 3-9 镜像保存过程中的事件列表

事件名称	事件描述	事件级别
SaveImage	保存镜像成功	重要
SavedImageFailed	D进程引起的保存镜像失败 （There are processes in 'D' status, please check process status using 'ps -aux' and kill all the 'D' status processes）	紧急
	镜像大小引起的保存镜像失败 （Container size %dG is greater than threshold %dG）	紧急
	层数限制引起的保存镜像失败 （Too many layers in your image）	紧急
	任务超时引起的保存镜像失败 （Operations personnel are handling the problem）	紧急
	SWR故障引起的保存镜像失败 （Failed to save the image because the SWR service is faulty）	紧急

版权所有 © 华为云计算技术有限公司

表 3-10 实例运行过程的事件列表

事件名称	事件描述	事件级别
NotebookUnhealthy	实例处于不健康状态	紧急
OutOfMemory	实例被OOM掉了	紧急
JupyterProcessKilled	jupyter进程被killed掉了	紧急
CacheVolumeExceedQuota	/cache目录文件大小超过最大限制	紧急
NotebookHealthy	实例从不健康恢复到了健康状态	重要
EVSSoldOut	EVS存储售罄	紧急

表 3-11 OBS 动态挂载产生的事件列表

事件名称	事件描述	事件级别
DynamicMountStorage	挂载OBS存储	重要
DynamicUnmountStorage	卸载OBS存储	重要

表 3-12 用户侧触发的事件

事件名称	事件描述	事件级别
RefreshCredentialsFailed	用户鉴权失败	紧急

3.12 Notebook cache 盘告警上报

创建Notebook时，可以根据业务数据量的大小选择CPU、GPU或者Ascend资源，对GPU或Ascend类型的资源，ModelArts会挂载硬盘至“/cache”目录，用户可以使用此目录来储存临时文件。

当前开发环境的cache盘使用时，没有容量告警，在使用时很容易超过限制，并直接重启Notebook实例。重启后多种配置重置，会导致用户数据丢弃，环境丢失，造成很不好的使用体验。因此需要提供cache盘使用情况的监控和告警，并将数据上报至AOM平台。

配置流程

1. 填写告警基本信息
2. **设置告警规则**
   a. 监控对象指标配置

 版权所有 © 华为云计算技术有限公司

b. 告警触发条件设置

3. **告警通知设置**
   a. 创建主题、设置主题策略、订阅主题
   b. 创建告警行动规则
   c. 选择已创建的行动规则

告警上报配置方法

1. 登录AOM控制台。
2. 单击“告警 > 告警规则”，在“告警规则”界面，单击“添加告警”。
3. 填写告警基本信息。

4. 设置告警规则。
   “规则类型”选择“阈值规则”。
   “监控对象”：选择“选择资源对象”。单击选择资源对象，弹出新窗口。
   - 添加方式：选择“按指标维度添加”。
   - 指标名称：选择“全量指标”，搜索需要监控的cache指标名称然后选中。例如：ma_container_notebook_cache_dir_size_bytes（cache目录的总大小）、ma_container_notebook_cache_dir_util（cache目录的利用率）
   - 指标维度：根据实际需求选择相应的指标维度。例如service_id:xxx，然后单击“确定”。

   监控对象设置完成后，选择“统计方式”和“统计周期”。
   “告警条件设置”：触发条件根据实际需求设置。

版权所有 © 华为云计算技术有限公司

图 3-24 监控对象指标设置

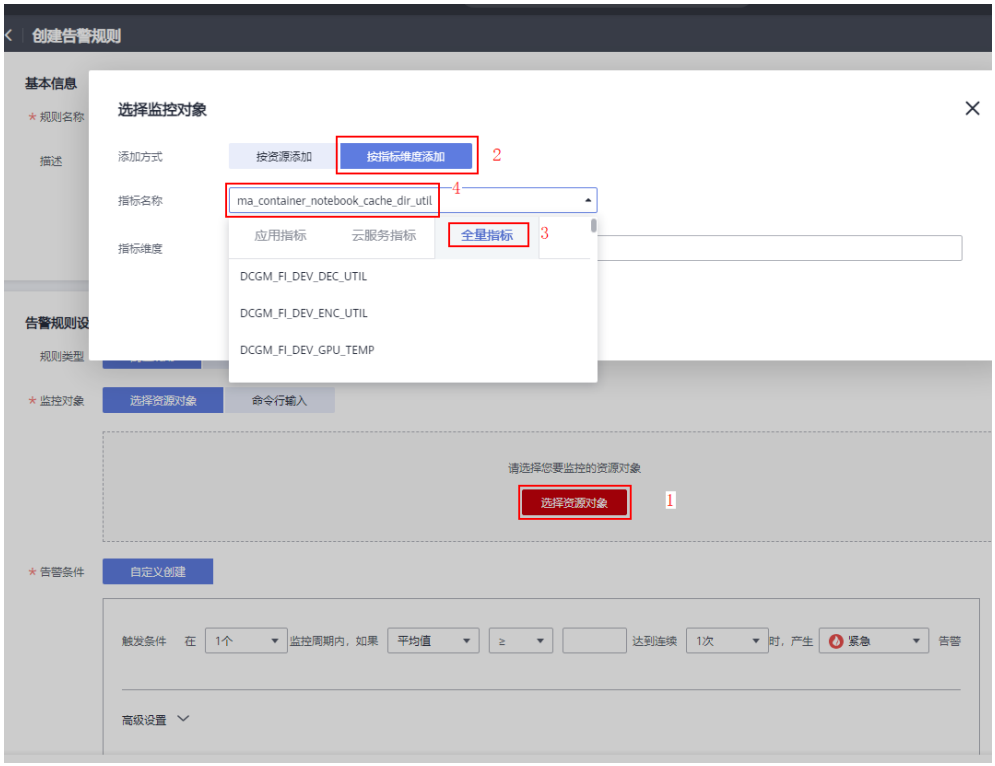


图 3-25 设置指标统计方式

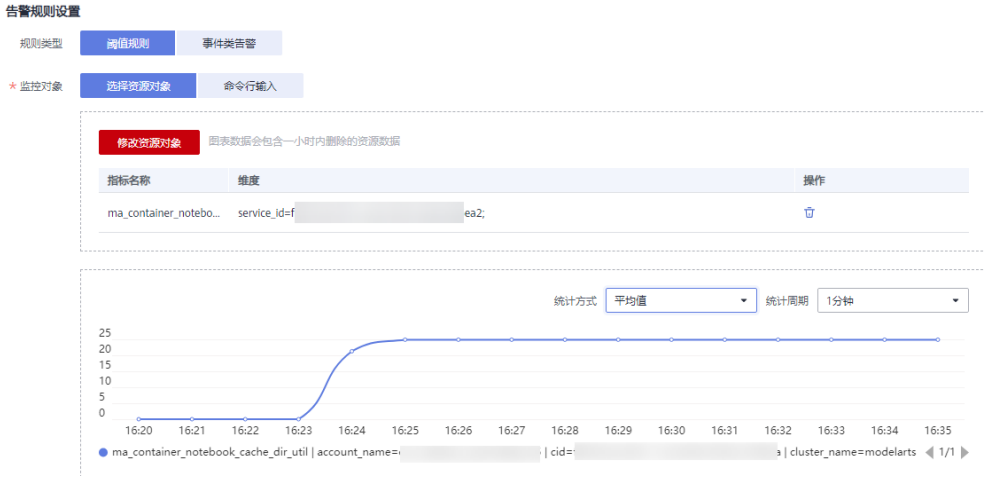
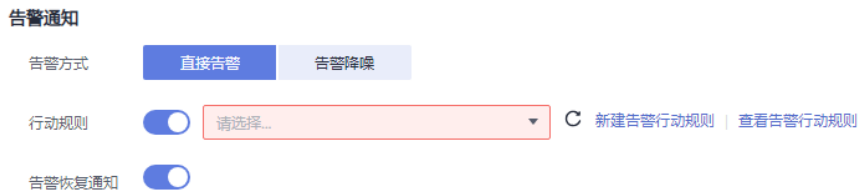


图 3-26 告警条件设置

版权所有 © 华为云计算技术有限公司

5. 设置告警通知，单击“立即创建”。
   “告警方式”：选择“直接告警”
   “行动规则”：开启开关，选择已创建的行动规则。如果现有列表中的告警行动规则无法满足需要，可单击“新建告警行动规则”添加，详细操作请参考**创建告警行动规则**。
   “告警恢复通知”：开启开关

   图 3-27 设置告警通知



   先在SMN创建一个主题，用于配置告警通知规则。更多内容请参考**消息通知服务用户指南**。

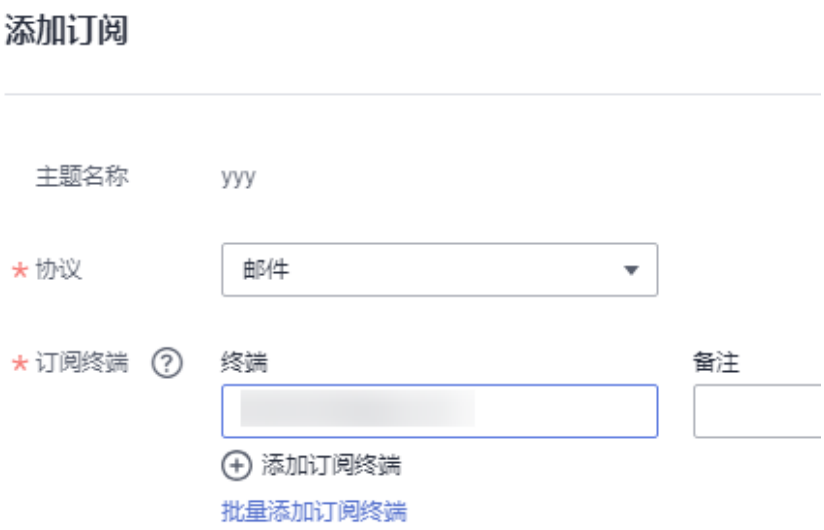
   - **创建主题**
     i. 进入“消息通知服务”控制台，单击“主题管理 > 主题”，进入“主题”页面。
     ii. 单击“创建主题”填写主题名称，选择企业项目后，单击确定即可创建一个主题。
     iii. 单击主题名称“操作”列的“更多 > 设置主题策略”。
     选择APM，即允许AOM的告警触发SMN服务。

     图 3-28 设置主题策略

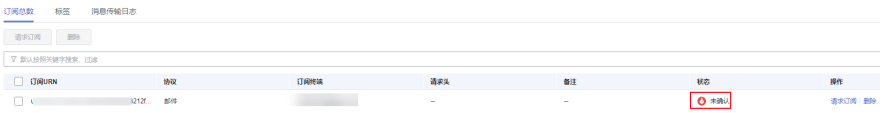
版权所有 © 华为云计算技术有限公司

iv. 单击主题名称“操作”列的“添加订阅”。订阅成功后，一旦满足告警条件，那么就会收到通知。

选择合适的协议，如邮件，短信等，并填写终端，如邮件地址，手机号等。单击确认。



此时订阅总数中会出现一条记录，但是处于未确认的状态。



收到邮件后单击“订阅确认”。

此时该订阅记录将处于已确认的状态。

版权所有 © 华为云计算技术有限公司

- **创建告警行动规则**

  行动规则即为告警触发时，AOM以怎样的方式来告知用户。启用告警行动规则后，系统根据关联SMN主题与消息模板来发送告警通知。更多详情请参考**AOM用户指南**。

  根据界面提示填写行动规则名称，选择行动规则类型，选择**上一步**创建的主题，选择消息模板，然后单击“确定”。

  图 3-29 新建告警行动规则

在之前打开的“创建告警规则”页面的**告警通知区域**，“行动规则”选择新创建的告警行动规则，单击“立即创建”。

至此，整个告警流程配置完成，一旦满足告警条件，那么就会收到邮件通知。

版权所有 © 华为云计算技术有限公司

4 CodeLab

面向众多开发者，ModelArts提供了CodeLab功能，一方面，一键进入开发环境，同时预置了免费的算力规格，可直接免费体验Notebook功能；另一方面，针对AI Gallery社区发布的Notebook样例（.ipynb格式文件），可直接在CodeLab中打开，查看他人分享的样例代码。

功能亮点

- **免费算力**

  CodeLab内置了免费算力，包含CPU和GPU两种。您可以使用免费规格，端到端体验ModelArts Notebook能力。也可使用此免费算力，在线完成您的算法开发。

- **即开即用**

  无需创建Notebook实例，打开即可编码。

- **高效分享**

  ModelArts在AI Gallery中提供的Notebook样例，可以直接通过Run in ModelArts，一键打开运行和学习，并且可将样例修改后分享到AI Gallery中直接另存用于个人开发。

  同时，您开发的代码，也可通过CodeLab快速分享到AI Gallery中给他人使用学习。

使用限制

- CodeLab默认打开，使用的是CPU计算资源。如需切换为GPU，请在右侧窗口，更换GPU规格。
- 在ModelArts控制台的“总览”界面打开CodeLab，使用的是CPU或GPU资源，无法使用Ascend资源。

  如果是AI Gallery社区的Notebook案例，本身使用的资源是Ascend的，那么“Run in ModelArts”跳转到CodeLab，就可以使用昇腾卡进行训练，也支持切换规格。
- 自启动后，免费规格默认可使用1小时，请注意右上角的剩余时长。超过1小时后，可执行续期操作，且系统每隔一段时间，将提醒确认下续期。
- 免费的CodeLab主要用于体验，72小时内未使用，将释放资源。保存在其中的代码文档将丢失，请注意备份文件以及使用时长。

 版权所有 © 华为云计算技术有限公司

CodeLab 入口

- ModelArts管理控制台的“总览”页
  在“开发工具”区域下方，展示“CodeLab”简介卡片，单击“立即体验”，即可进入。

  图 4-1 CodeLab 入口

- AI Gallery页面提供的Notebook样例
  a. 在AI Gallery页面，单击“资产集市 > 开发 > Notebook”栏目中的任意Notebook样例，进入详情页。
  b. 单击“Run in ModelArts”，进入CodeLab并打开该样例。等待右上角连接成功即可运行Notebook样例。

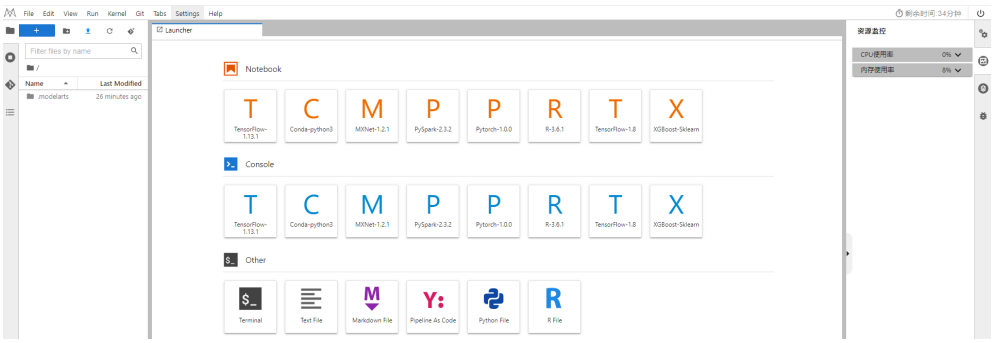
说明

- 首次进入CodeLab时，会提醒先登录ModelArts控制台，由于需要进行环境准备，需要等待1~2分钟才可顺利进入开发页面，请耐心等待。
- CodeLab的内部环境依托Notebook功能，因此其工作环境，与JupyterLab界面相似。

体验 CodeLab

1. **进入CodeLab主页。**
   从管理控制台总览页进入，展示CodeLab首页。

   图 4-2 CodeLab 首页

版权所有 © 华为云计算技术有限公司

2. **常用功能。**

   CodeLab的界面依托于JupyterLab，其相关的常见功能与JupyterLab相同。

   常用操作指导可参见JupyterLab操作指导：**JupyterLab简介及常用操作**。

   说明

   由于CodeLab的存储为系统默认路径，在使用“上传文件”或“下载文件至本地”时，只能使用JupyterLab页面提供的功能。

   如需使用大文件上传和下载的功能，建议您前往Notebook，创建一个收费的实例进行使用。

3. **切换规格。**

   CodeLab支持CPU和GPU两种规格，在右侧区域，单击切换规格，修改规格类型。

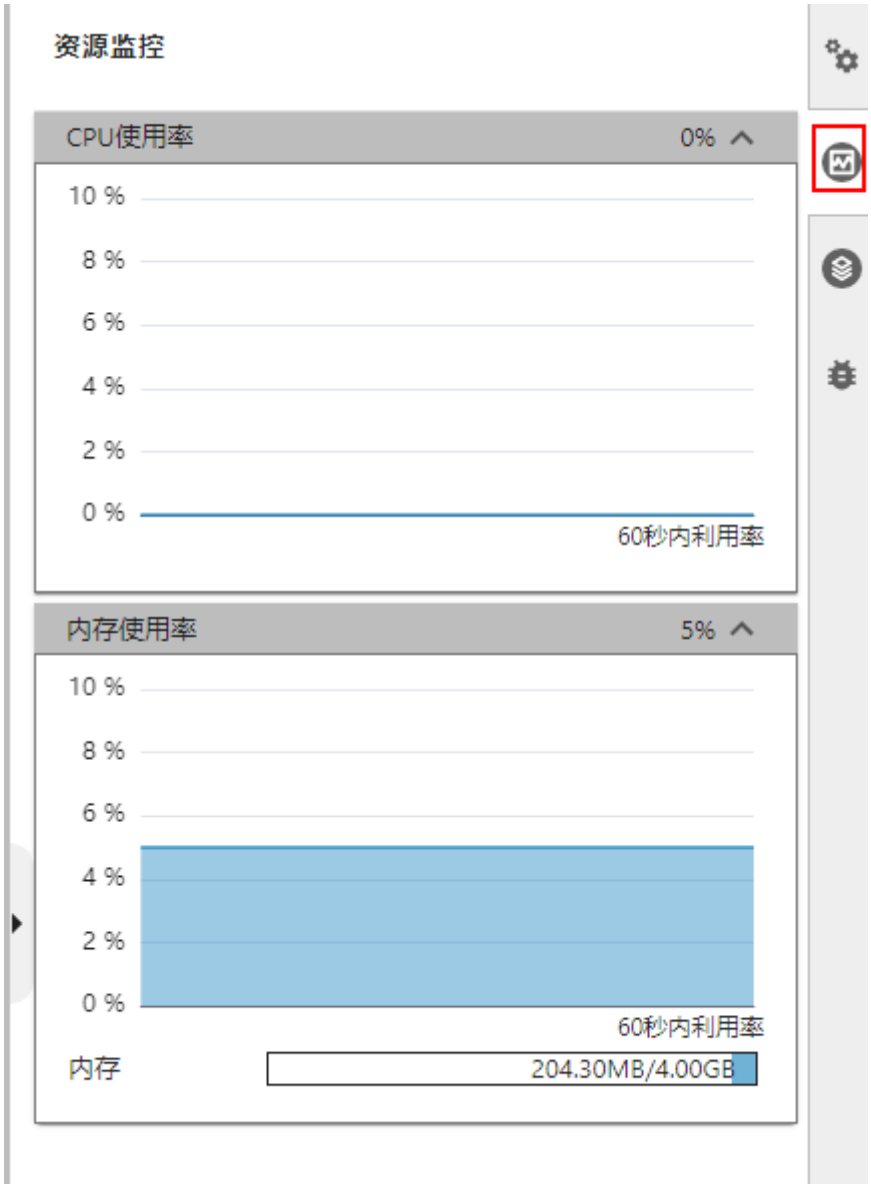
   图 4-3 切换规格

4. **资源监控。**

   在使用过程中，如果想了解资源使用情况，可在右侧区域选择“Resource Monitor”，展示“CPU使用率”和“内存使用率”。

版权所有 © 华为云计算技术有限公司

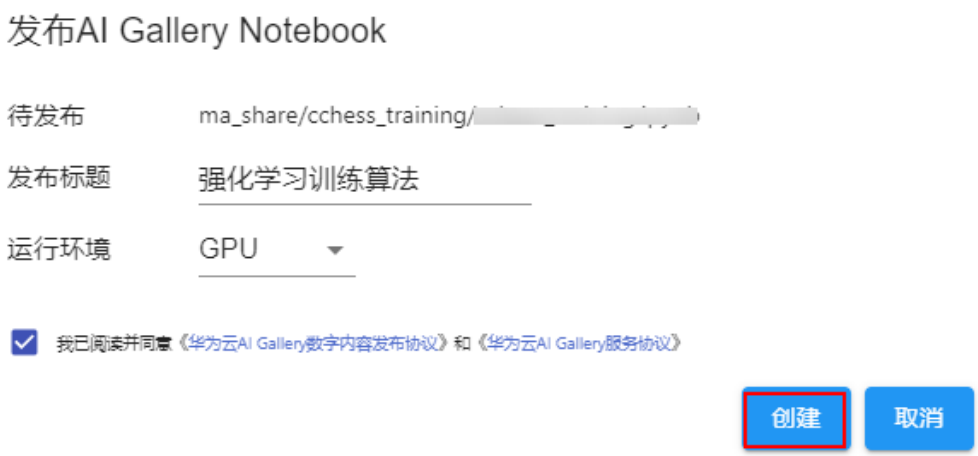
图 4-4 资源监控



5. 分享副本到AI Gallery。单击右上角的 ，将修改后的Notebook样例保存分享到AI Gallery中，供自己或他人学习使用。

版权所有 © 华为云计算技术有限公司

图 4-5 分享到 AI Gallery



分享成功后，通过分享链接可以打开分享的副本，也可以在AI Gallery中找到分享的Notebook。

图 4-6 发布成功

版权所有 © 华为云计算技术有限公司

5 JupyterLab

5.1 JupyterLab 操作流程

ModelArts支持通过JupyterLab工具在线打开Notebook，开发基于PyTorch、TensorFlow和MindSpore引擎的AI模型。具体操作流程如下图所示。

图 5-1 使用 JupyterLab 在线开发调试代码

1. 创建Notebook实例。

   在ModelArts控制台创建一个Notebook开发环境实例，选择要使用的AI框架。具体参见**创建Notebook实例**。
2. 使用JupyterLab打开Notebook实例。具体参见**打开JupyterLab**。
3. 准备训练数据和代码文件，上传到JupyterLab中。具体参见**上传本地文件至JupyterLab**。
4. 在JupyterLab中编写代码文件，并运行调试。具体参见**JupyterLab简介及常用操作**。
5. 在JupyterLab中直接调用ModelArts提供的SDK，创建训练作业，上云训练。

   调用SDK创建训练作业的操作请参见**调用SDK创建训练作业**。

5.2 JupyterLab 简介及常用操作

JupyterLab是一个交互式的开发环境，是Jupyter Notebook的下一代产品，可以使用它编写Notebook、操作终端、编辑MarkDown文本、打开交互模式、查看csv文件及图片等功能。

版权所有 © 华为云计算技术有限公司

可以说，JupyterLab是开发者们下一阶段更主流的开发环境。JupyterLab具有和Jupyter Notebook一样的组件，但支持更加灵活和更加强大的项目操作方式。

打开 JupyterLab

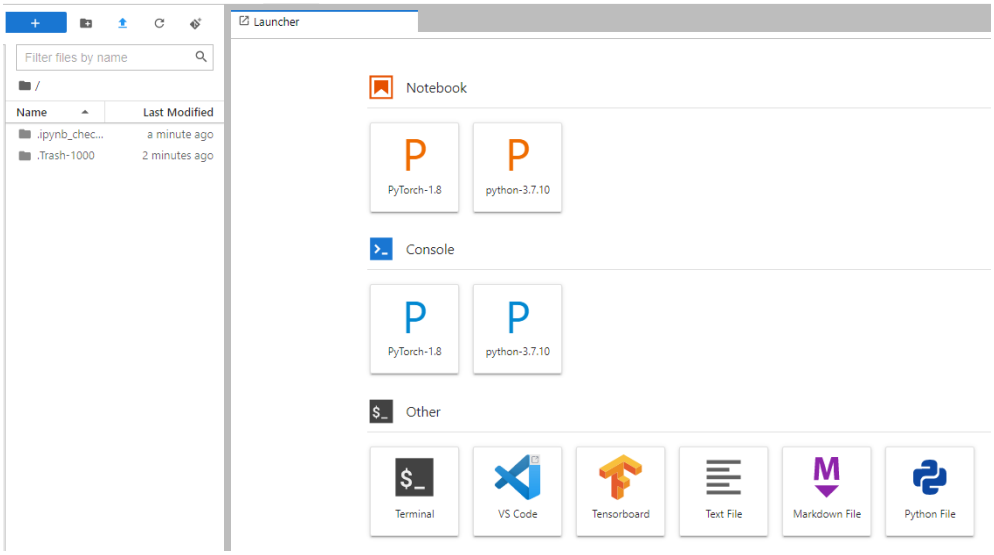
下面介绍如何从运行中的Notebook实例打开JupyterLab。

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“开发环境 > Notebook”，进入Notebook页面。
2. 选择状态为“运行中”的Notebook实例，单击操作列的“打开”，访问JupyterLab。

图 5-2 打开 Notebook 实例

3. 进入JupyterLab页面后，自动打开Launcher页面，如下图所示。您可以使用开源支持的所有功能，详细操作指导可参见**JupyterLab官网文档**。

图 5-3 JupyterLab 主页



说明

不同AI引擎的Notebook，打开后Launcher页面呈现的Notebook和Console内核及版本均不同，**图5-3**仅作为示例，请以实际控制台为准。

- Notebook：选择运行Notebook的一个内核，例如TensorFlow、python
- Console：可调出终端进行命令控制
- Other：可编辑其他文件

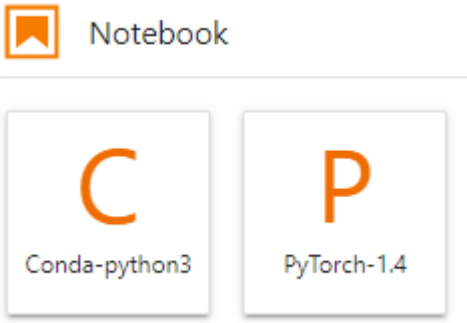
在 JupyterLab 中新建 ipynb 文件

进入JupyterLab主页后，可在“Notebook”区域下，选择适用的AI引擎，单击后将新建一个对应框架的ipynb文件。

版权所有 © 华为云计算技术有限公司

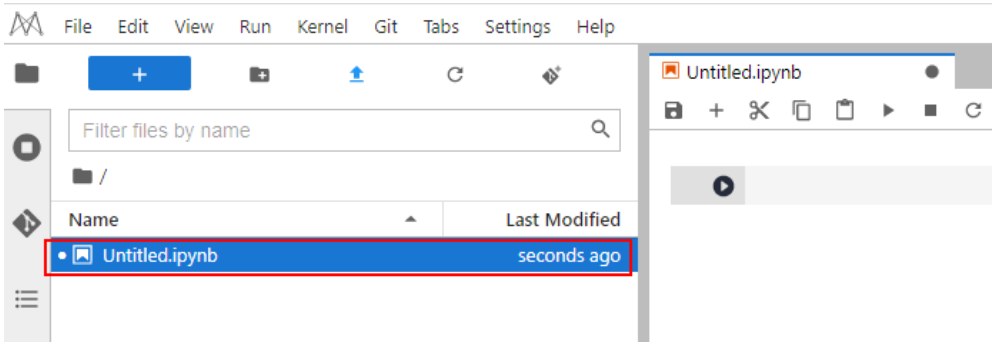
由于每个Notebook实例选择的工作环境不同，其支持的AI框架也不同，下图仅为示例，请根据实际显示界面选择AI框架。

图 5-4 选择 AI 引擎并新建一个 ipynb 文件



新建的ipynb文件将呈现在左侧菜单栏中。

图 5-5 新建文件



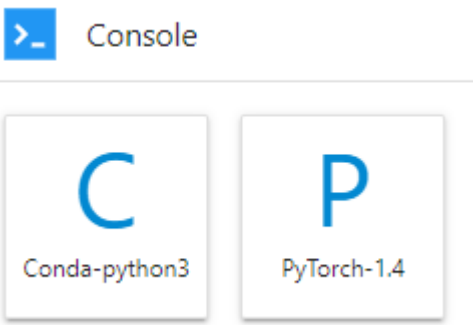
新建文件并打开 Console

Console的本质为Python终端，输入一条语句就会给出相应的输出，类似于Python原生的IDE。

进入JupyterLab主页后，可在“Console”区域下，选择适用的AI引擎，单击后将新建一个对应框架的Notebook文件。

由于每个Notebook实例选择的工作环境不同，其支持的AI框架也不同，下图仅为示例，请根据实际显示界面选择AI框架。

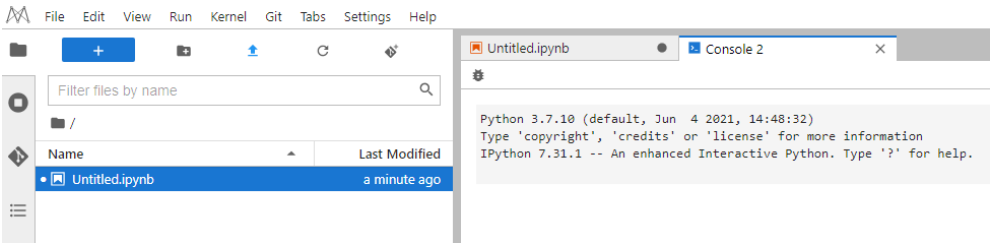
图 5-6 选择 AI 引擎并新建一个 Console



版权所有 © 华为云计算技术有限公司

文件创建成功后，将直接呈现Console页面。

图 5-7 新建文件（Console）

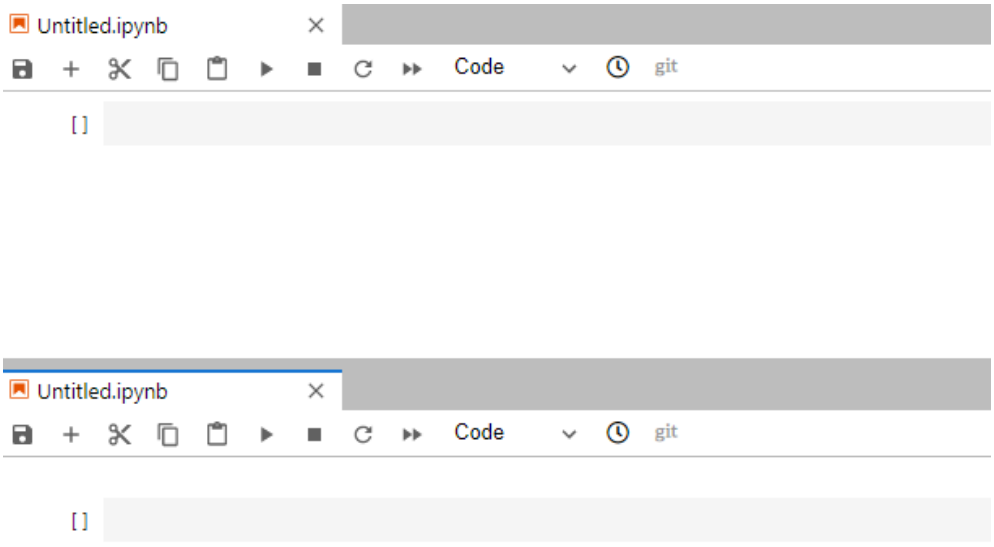


在 JupyterLab 中编辑文件

JupyterLab可以在同一个窗口同时打开几个Notebook或文件（如HTML、TXT、Markdown等），以页签形式展示。

JupyterLab的一大优点是，可以任意排版多个文件。在右侧文件展示区，您可以拖动打开文件，随意调整文件展示位置，可以同时打开多个文件。

图 5-8 多文件任意编排

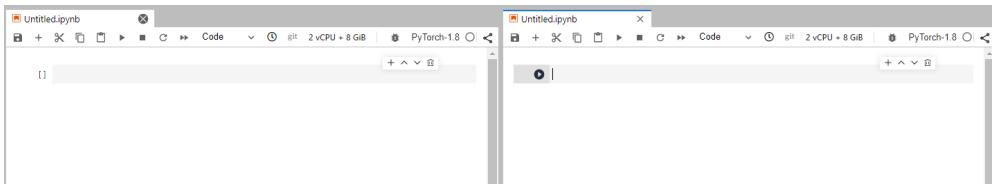


当在一个Notebook中写代码时，如果需要实时同步编辑文件并查看执行结果，可以新建该文件的多个视图。

打开ipynb文件，然后单击菜单栏“File > New View for Notebook”，即可打开多个视图。

版权所有 © 华为云计算技术有限公司

图 5-9 同一个文件的多个视图



JupyterLab的ipynb文件代码栏中输入代码，需要在代码前加!符号。

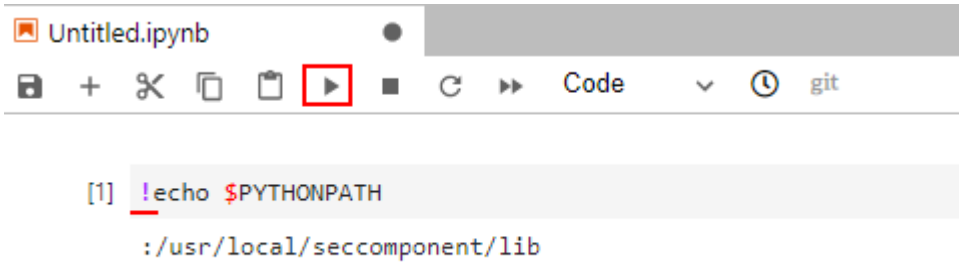
例如：安装外部库Shapely

```
!pip install Shapely
```

例如：查看PythonPath

```
!echo $PYTHONPATH
```

图 5-10 运行代码

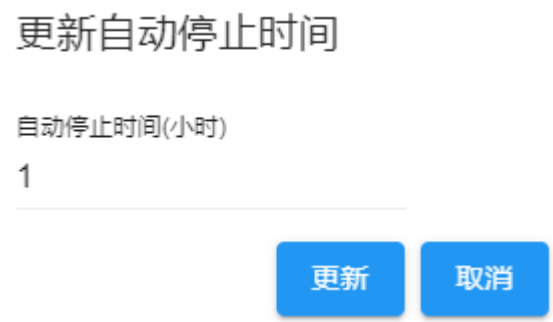


自动停止及续期

在创建或启动Notebook时，如果启用了自动停止功能，则在JupyterLab的右上角会显示当前实例停止的剩余时长，在计时结束前可以单击剩余时间进行续期。

图 5-11 自动停止

图 5-12 续期



 版权所有 © 华为云计算技术有限公司

JupyterLab 常用快捷键和插件栏

图 5-13 JupyterLab 常用快捷键和插件栏

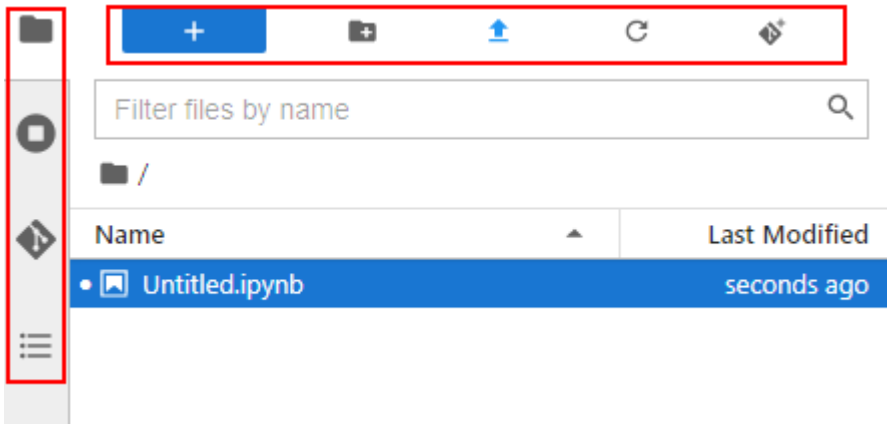


表 5-1 快捷键说明

快捷键	说明
+	快速打开Notebook、Terminal。或打开Launcher页面，可快速创建新的Notebook、Console或其他文件。
	创建文件夹。
	上传文件。
	刷新文件目录。
	Git插件，可连接此Notebook实例关联的Github代码库。

表 5-2 插件栏常用插件说明

插件	说明
	文件列表。单击此处，将展示此Notebook实例下的所有文件列表。
	当前实例中正在运行的Terminal和Kernel。
	Git插件，可以方便快捷的使用Github代码库。
	属性检查器。

版权所有 © 华为云计算技术有限公司

插件	说明
	文档结构图。

图 5-14 导航栏按钮

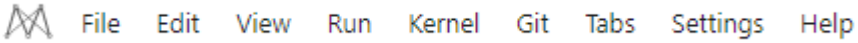


表 5-3 导航栏按钮介绍

按钮	说明
File	新建、关闭、保存、重新加载、重命名、导出、打印Notebook等功能。
Edit	编辑ipynb文件中代码块的相关操作，包括撤销、重做、剪切、复制、粘贴、选择、移动、合并、清除、查找代码块等。
View	查看视图相关操作。
Run	运行代码块相关操作，例如：运行选中代码块、一键运行所有代码块等。
Kernel	中断、重启、关闭、改变Kernel相关操作。
Git	Git插件相关操作，可以方便快捷的使用Github代码库。
Tabs	同时打开多个ipynb文件时，通过Tabs激活或选择文件。
Settings	JupyterLab工具系统设置。
Help	JupyterLab工具自带的帮助参考。

图 5-15 ipynb 文件菜单栏中的快捷键

表 5-4 ipynb 文件菜单栏中的快捷键

快捷键	说明
	保存文件。
+	添加新代码块。

版权所有 © 华为云计算技术有限公司

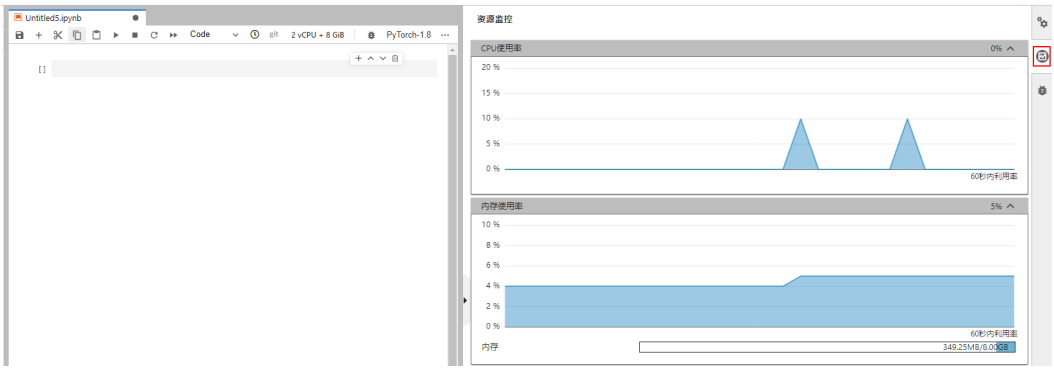
快捷键	说明
	剪切选中的代码块。
	复制选中的代码块。
	粘贴选中的代码块。
	执行选中的代码块。
	终止kernel。
	重启kernel。
	重启kernel，然后重新运行当前Notebook的所有代码。
Code	此处下拉框有4个选项，分别是： Code（写python代码），Markdown（写Markdown代码，通常用于注释），Raw（一个转换工具），-（不修改）。
	查看代码历史版本。
git	git插件，图标显示灰色表示当前Region不支持。
2 vCPU + 4 GiB	当前的资源规格。
PyTorch-1.4	单击可以选择Kernel。
	表示代码运行状态，变为实心圆  时，表示代码在运行中。
	分享到AI Gallery。

资源监控

在使用过程中，如果想了解资源使用情况，可在右侧区域选择“Resource Monitor”，展示“CPU使用率”和“内存使用率”。

版权所有 © 华为云计算技术有限公司

图 5-16 资源监控



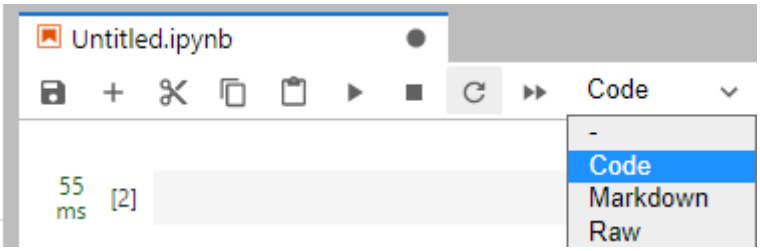
5.3 代码参数化插件

代码参数化插件可以降低Notebook案例的复杂度，用户无需感知复杂的源码，按需调整参数快速进行案例复现、模型训练等。该插件可用于定制Notebook案例，适用于比赛、教学等场景。

使用介绍

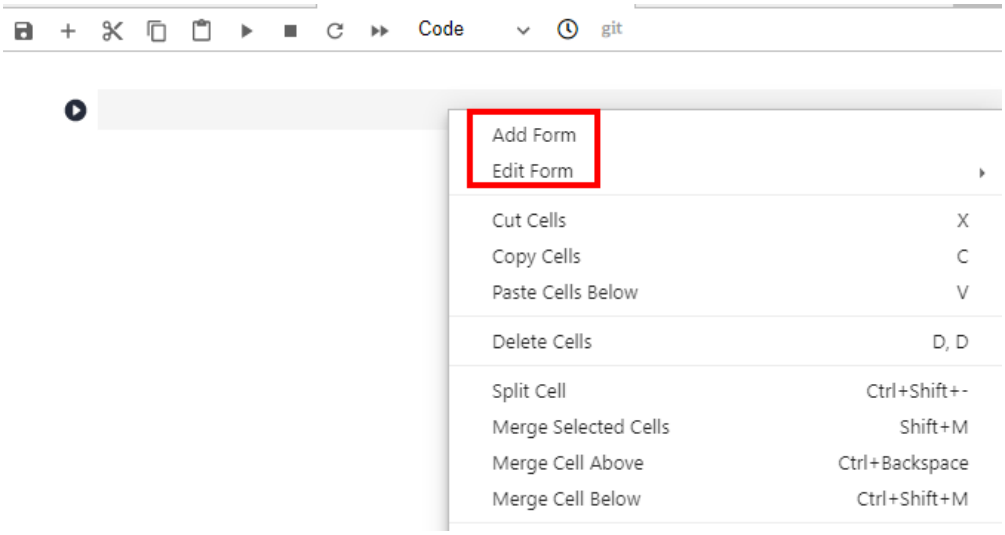
- 仅对Code cell类型新增了Edit Form和Add Form功能，如果cell类型是Markdown或者Raw类型则不支持。如下图所示：

  图 5-17 查看 Code cell

- 打开新的代码后，需先Add Form，再Edit Form。

  图 5-18 Code 类型的 cell 右键选项

版权所有 © 华为云计算技术有限公司

Add Form 介绍

“Add Form”按钮会将Code cell水平拆分为两种编辑区域，左侧为代码区域，右侧为表单区域。单击表单右侧的“Edit”可修改默认标题。

图 5-19 两种编辑区域

Edit Form 介绍

“Edit Form”按钮有四个子选项，分别是“Add new form field”、“Hide code”、“Hide form”和“show all”四个按钮，下文介绍这四个选项的功能。

- “Add new form field”按钮支持新增“dropdown”、“input”和“slider”类型的表单。如图5-20所示。每新增一个字段，会分别在代码和表单区域中增加对应的变量，修改表单区域的值也会同时修改代码变量值。

  说明

  创建dropdown类型的表单时，“ADD Item”至少创建2项。如图5-21所示。

图 5-20 “dropdown”，“input”，“slider”的表单样式

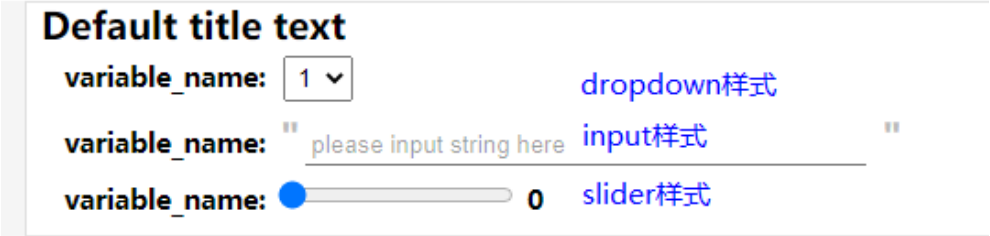
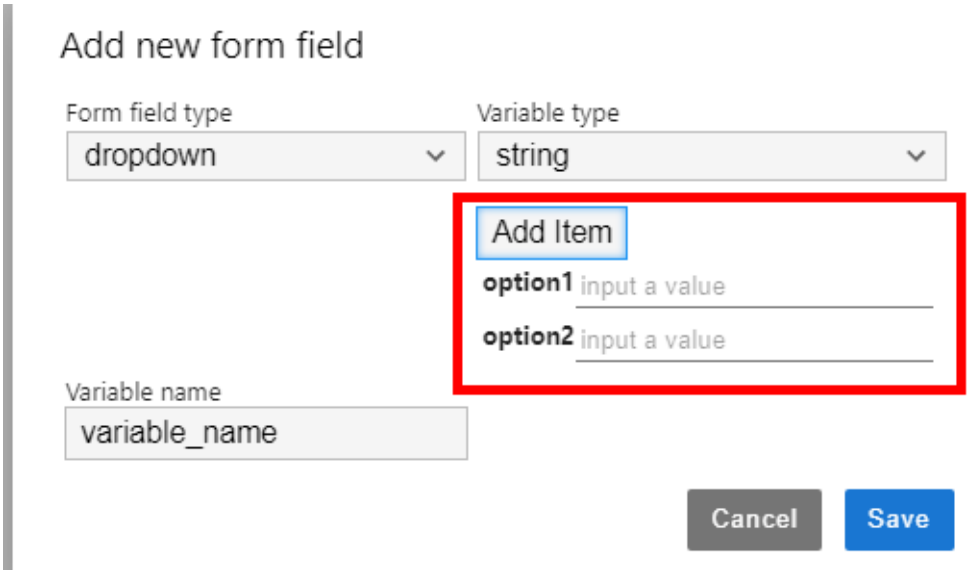


图 5-21 创建“dropdown”类型的表单



版权所有 © 华为云计算技术有限公司

图 5-22 删除表单

  - 表单字段类型为“dropdown”时，支持的变量类型为“raw”和“string”。
  - 表单字段类型为“input”时，支持的变量类型有“boolean”、“date”、“integer”、“number”、“raw”和“string”。
  - 表单字段类型为“slider”时，支持输入滑动条的最小值、最大值和步长。
- “Hide code”按钮会隐藏代码区域。
- “Hide form”按钮会隐藏表单区域。
- “Show all”按钮会同时展示code和form区域。

5.4 使用 ModelArts SDK

在Notebook中，通过使用ModelArts SDK，可以完成OBS管理、训练作业管理、模型管理以及在线服务管理。

ModelArts SDK使用请参见《**ModelArts SDK参考**》。

在Notebook中，已承载了登录用户的鉴权信息（AK/SK）和区域信息，因此SDK session鉴权时，无需输入参数即可完成session鉴权。

示例代码

- 创建训练作业

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(
                    modelarts_session=session,
                    framework_type='PyTorch',                                   # AI引擎名称
                    framework_version='PyTorch-1.0.0-python3.6',                # AI引擎版本
                    code_dir='/obs-bucket-name/src/',                           # 训练脚本目录
                    boot_file='/obs-bucket-name/src/pytorch_sentiment.py',      # 训练启动脚本目录
                    log_url='/obs-bucket-name/log/',                            # 训练日志目录
                    hyperparameters=[
                                {"label":"classes",
                                 "value": "10"},
                                {"label":"lr",
                                 "value": "0.001"}
                                ],
                    output_path='/obs-bucket-name/output/',                     # 训练输出目录
                    train_instance_type='modelarts.vm.xxx.xxx',                 # 训练环境规格
                    train_instance_count=1,                                     # 训练节点个数
                    job_description='pytorch-sentiment with ModelArts SDK')     # 训练作业描述
job_instance = estimator.fit(inputs='/obs-bucket-name/data/train/', wait=False,
job_name='my_training_job')
```

- 查询模型列表

```
from modelarts.session import Session
from modelarts.model import Model
session = Session()
```


版权所有 © 华为云计算技术有限公司

```
model_list_resp = Model.get_model_list(session, model_status="published", model_name="digit",
order="desc")
```

- 查询服务详情

```
from modelarts.session import Session
from modelarts.model import Predictor
session = Session()
predictor_instance = Predictor(session, service_id="input your service_id")
predictor_info_resp = predictor_instance.get_service_info()
```

5.5 使用 Git 插件

在JupyterLab中使用Git插件可以克隆GitHub开源代码仓库，快速查看及编辑内容，并提交修改后的内容。

前提条件

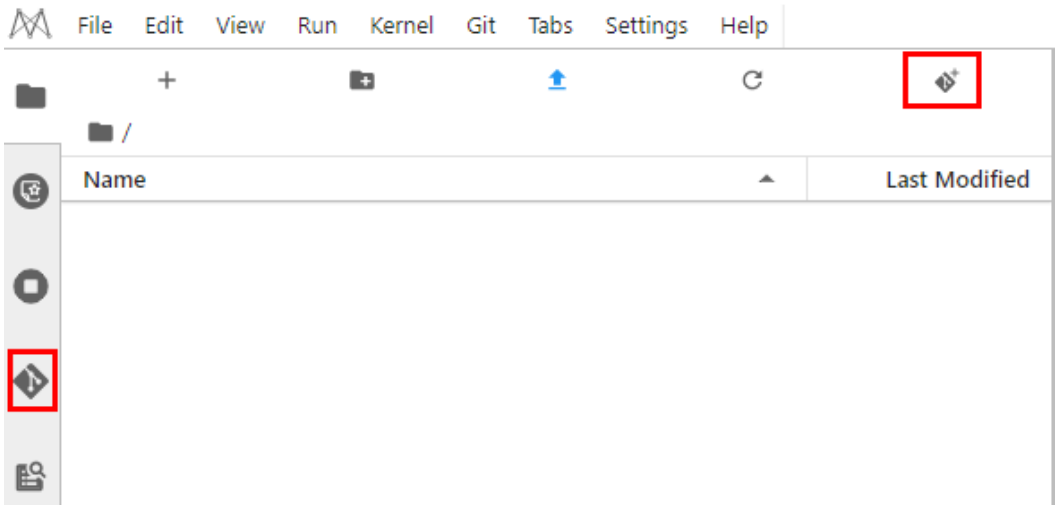
Notebook处于运行中状态。

打开 JupyterLab 的 git 插件

在Notebook列表中，选择一个实例，单击右侧的打开进入“JupyterLab”页面。

图5-23所示图标，为JupyterLab的Git插件。

图 5-23 Git 插件

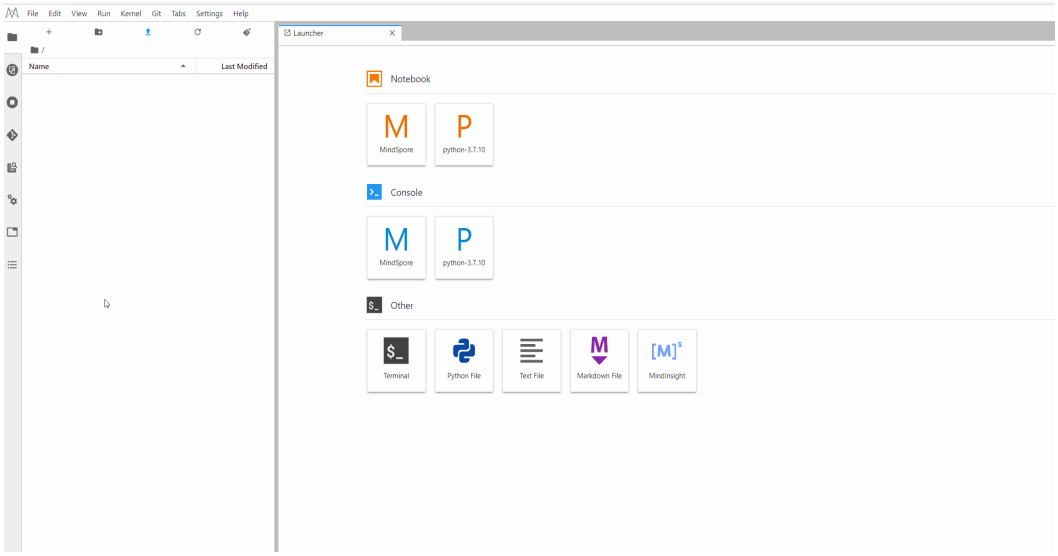


克隆 GitHub 的开源代码仓库

GitHub开源仓库地址：https://github.com/jupyterlab/extension-examplesitHub，单击 ，输入仓库地址，单击确定后即开始克隆，克隆完成后，JupyterLab左侧导航出现代码库文件夹。

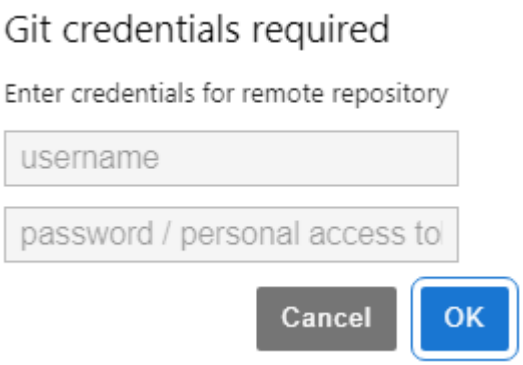
版权所有 © 华为云计算技术有限公司

图 5-24 使用 git 插件克隆 GitHub 的开源代码仓库



克隆 GitHub 的私有仓库

克隆GitHub私有仓库时，会弹出输入个人凭证的对话框，如下图。此时需要输入GitHub中Personal Access Token信息。



查看Personal Access Token步骤如下：

1. 登录**Github**，打开设置页面。
2. 单击“Developer settings”。
3. 单击“Personal access tokens > Generate new token”。
4. 验证登录账号。
5. 填写Token描述并选择权限，选择私有仓库访问权限，单击“Generate token”生成Token。
6. 复制生成的Token到编译构建服务即可。

版权所有 © 华为云计算技术有限公司

须知

- Token生成后，请及时保存，下次刷新页面将无法读取，需要重新生成新Token。
- 注意填写有效的Token描述信息，避免误删除导致构建失败。
- 无需使用时及时删除Token，避免信息泄露。

图 5-25 克隆 GitHub 的私有仓库（目前只支持 Personal Access Token 授权）

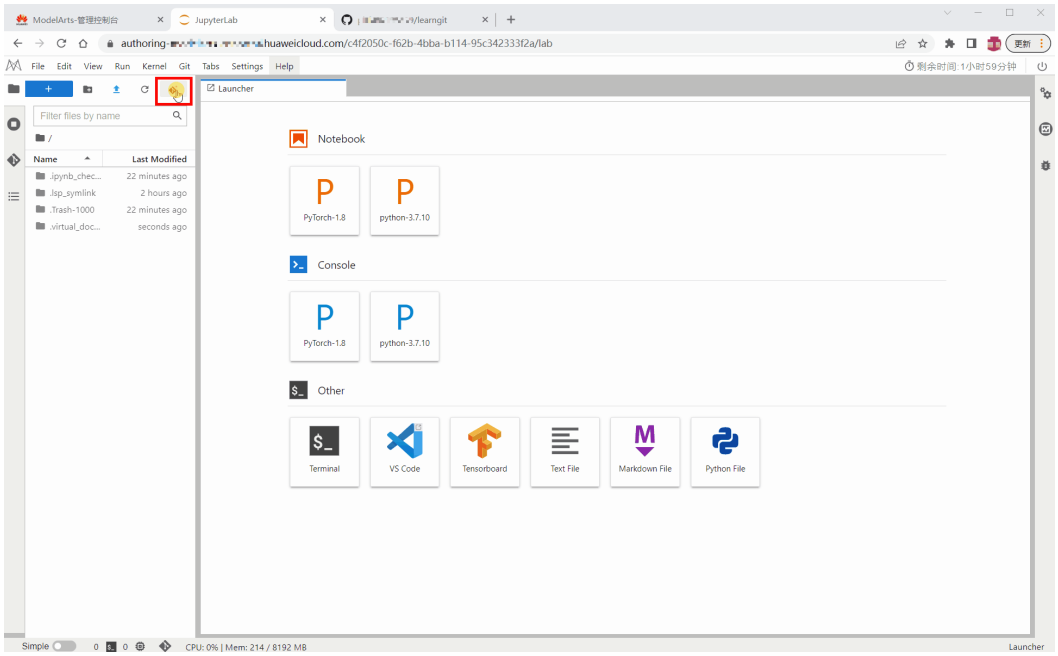
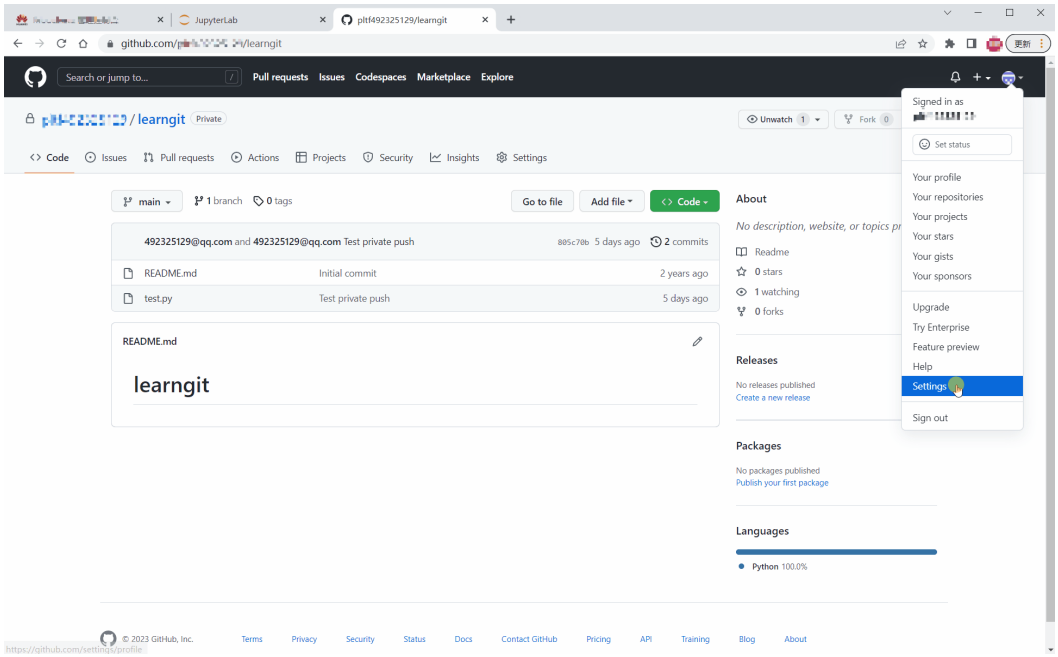


图 5-26 获取 Personal Access Token

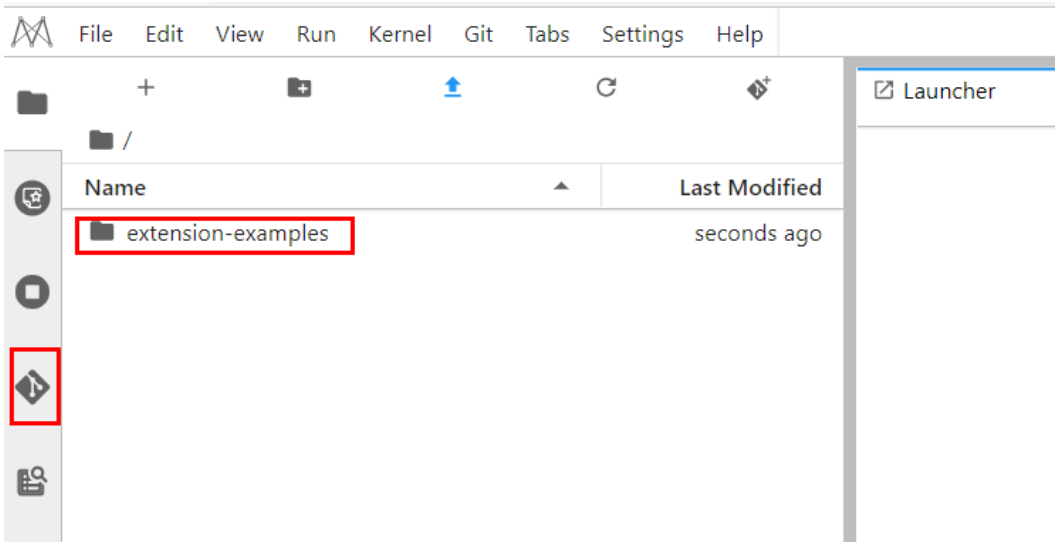


版权所有 © 华为云计算技术有限公司

查看代码库信息

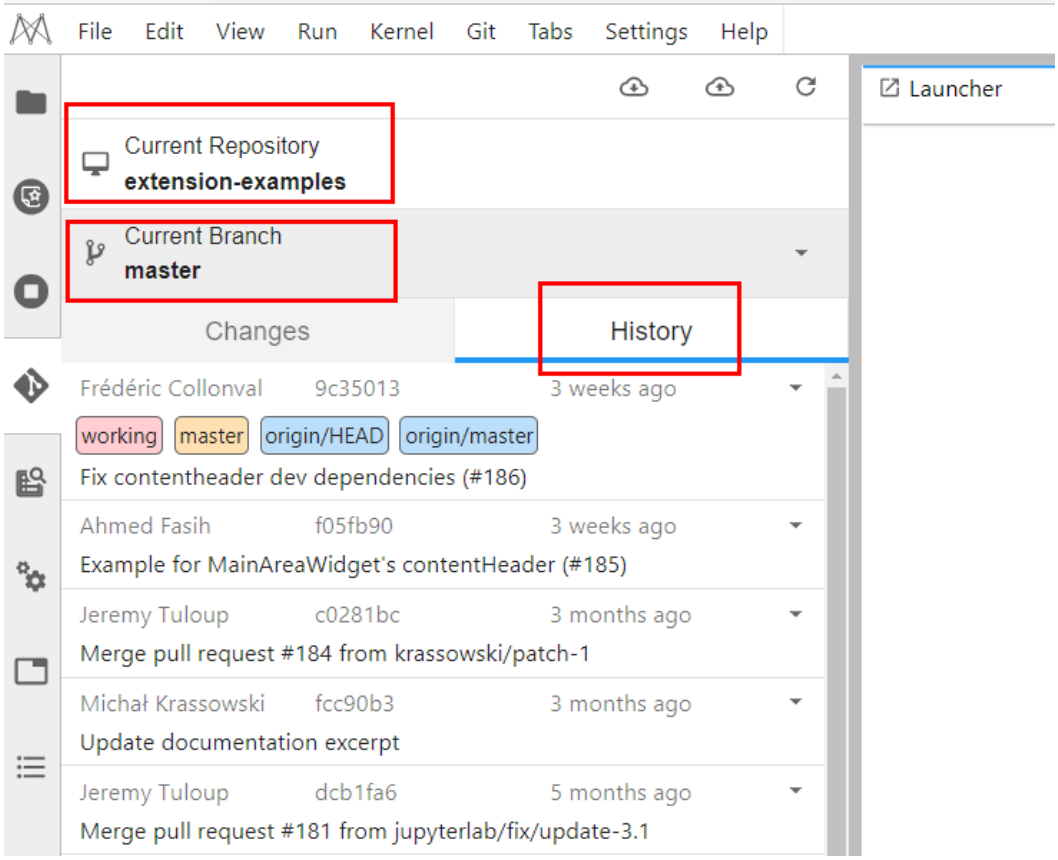
在Name下方列表中，选中您希望使用的文件夹，双击打开，然后单击左侧git插件图标进入此文件夹对应的代码库。

图 5-27 打开文件夹后打开 git 插件



即可看到当前代码库的信息，如仓库名称、分支、历史提交记录等。

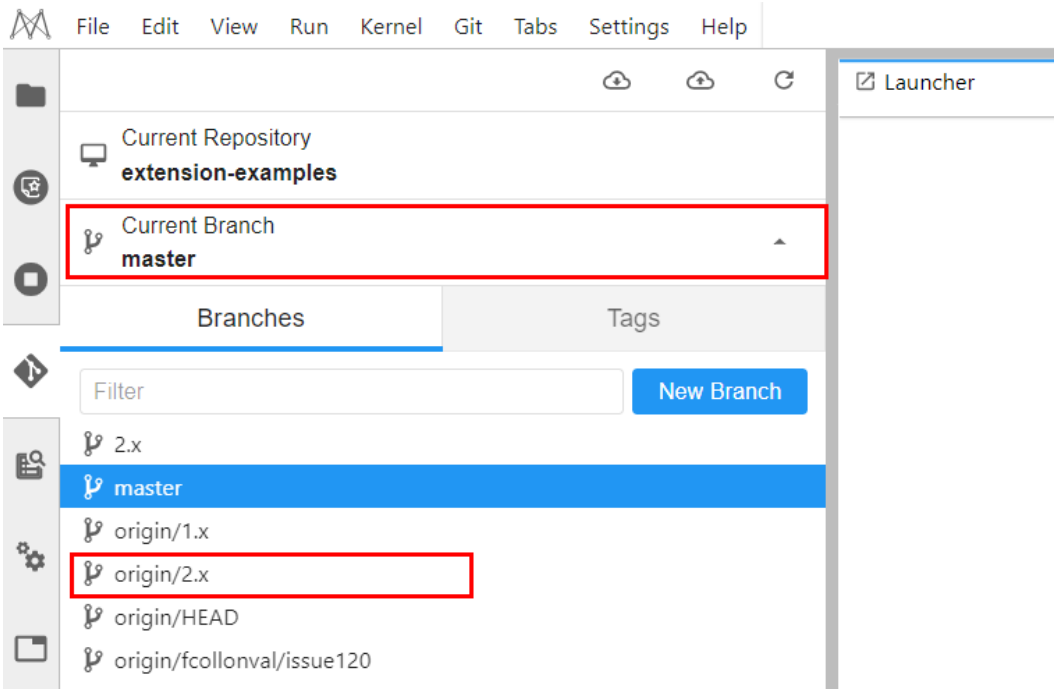
图 5-28 查看代码库信息



 版权所有 © 华为云计算技术有限公司

说明

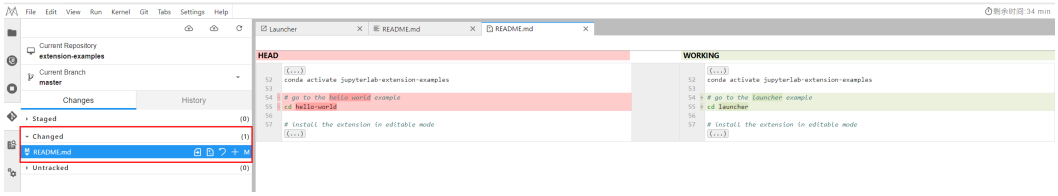
Git插件一般默认克隆master分支，如果要切换分支可单击Current Branch展开所有分支，单击相应分支名称可完成切换。



查看修改的内容

如果修改代码库中的某个文件，在“Changes”页签的“Changed”下可以看到修改的文件，并单击修改文件名称右侧的“Diff this file”，可以看到修改的内容。

图 5-29 查看修改的内容

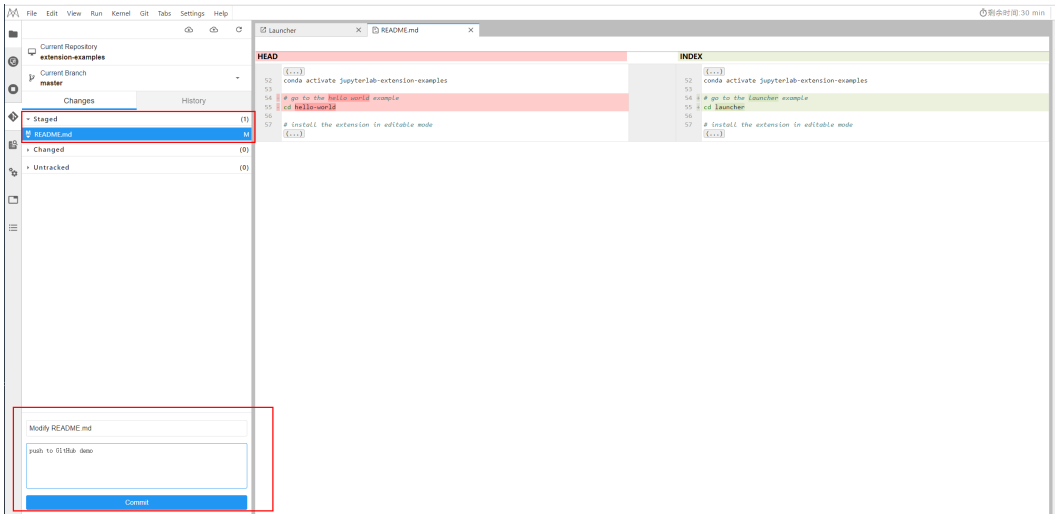


提交修改的内容

确认修改无误后，单击修改文件名称右侧的“Stage this change”，文件将进入Staged状态，相当于执行了git add命令。在左下方输入本次提交的Message，单击“Commit”，相当于执行了git commit命令。

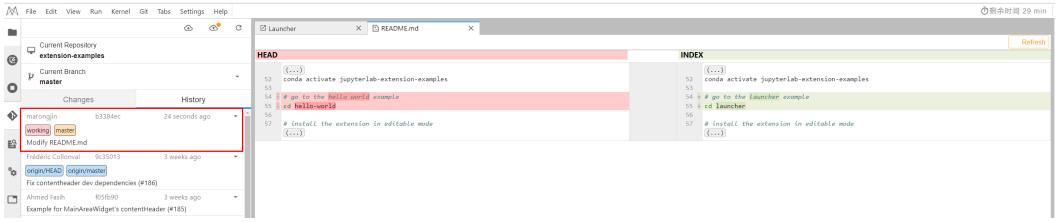
版权所有 © 华为云计算技术有限公司

图 5-30 提交修改内容



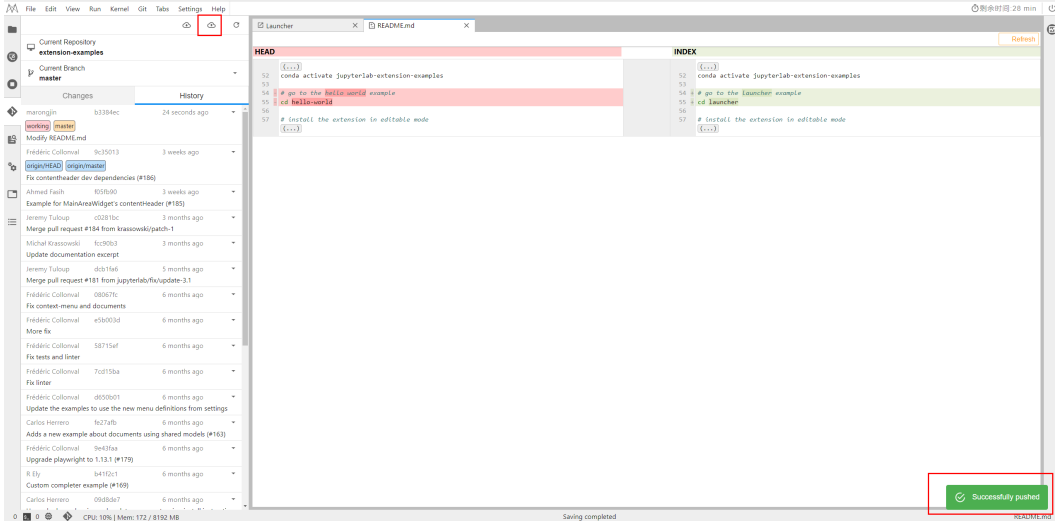
此时，可以在“History”页签下看到本地提交已成功。

图 5-31 查看是否提交成功



单击“push”按钮，相当于执行git push命令，即可提交代码到GitHub仓库中。提交成功后会提示“Successfully completed ”。如果OAuth鉴权的token过期，则此时再push会弹框让输入用户的token或者账户信息，按照提示输入即可。这里推荐使用Personal Access Token授权方式，如果出现密码失效报错请参考**git插件密码失效如何解决?**

图 5-32 提交代码至 GitHub 仓库



版权所有 © 华为云计算技术有限公司

完成上述操作后，可以在JupyterLab的git插件页面的History页签，看到“origin/HEAD”和“orgin/master”已指向最新一次的提交。同时在GitHub对应仓库的commit记录中也可以查找到对应的信息。

5.6 可视化训练作业

5.6.1 可视化训练作业介绍

ModelArts支持在开发环境中开启TensorBoard和MindInsight可视化工具。在开发环境中通过小数据集训练调试算法，主要目的是验证算法收敛性、检查是否有训练过程中的问题，方便用户调测。

ModelArts可视化作业支持创建TensorBoard类型和MindInsight两种类型。

TensorBoard和MindInsight能够有效地展示训练作业在运行过程中的变化趋势以及训练中使用到的数据信息。

- TensorBoard

  TensorBoard是一个可视化工具，能够有效地展示TensorFlow在运行过程中的计算图、各种指标随着时间的变化趋势以及训练中使用到的数据信息。TensorBoard相关概念请参考**TensorBoard官网**。

  TensorBoard可视化训练作业，当前仅支持基于TensorFlow2.1、Pytorch1.4/1.8版本镜像，CPU/GPU规格的资源类型。请根据实际局点支持的镜像和资源规格选择使用。

- MindInsight

  MindInsight能可视化展现出训练过程中的标量、图像、计算图以及模型超参等信息，同时提供训练看板、模型溯源、数据溯源、性能调试等功能，帮助您在更高效地训练调试模型。MindInsight当前支持基于MindSpore引擎的训练作业。MindInsight相关概念请参考**MindSpore官网**。

  MindInsight可视化训练作业，当前支持的镜像如下，请根据实际局点支持的镜像和资源规格选择使用。

  - mindspore1.2.0版本，CPU/GPU规格的资源类型。
  - mindspore1.5.x以上版本，Ascend规格的资源类型。

您可以使用模型训练时产生的Summary文件在开发环境Notebook中创建可视化作业。

- 在开发环境中创建MindInsight可视化作业，请参见**MindInsight可视化作业**。
- 在开发环境中创建TensorBoard可视化作业，请参见**TensorBoard可视化作业**。

5.6.2 MindInsight 可视化作业

ModelArts支持在开发环境中开启MindInsight可视化工具。在开发环境中通过小数据集训练调试算法，主要目的是验证算法收敛性、检查是否有训练过程中的问题，方便用户调测。

MindInsight能可视化展现出训练过程中的标量、图像、计算图以及模型超参等信息，同时提供训练看板、模型溯源、数据溯源、性能调试等功能，帮助您在更高效地训练调试模型。MindInsight当前支持基于MindSpore引擎的训练作业。MindInsight相关概念请参考**MindSpore官网**。

版权所有 © 华为云计算技术有限公司

MindSpore支持将数据信息保存到Summary日志文件中，并通过可视化界面MindInsight进行展示。

前提条件

使用MindSpore引擎编写训练脚本时，为了保证训练结果中输出Summary文件，您需要在脚本中添加收集Summary相关代码。

将数据记录到Summary日志文件中的具体方式请参考**收集Summary数据**。

注意事项

- 在开发环境跑训练任务，在开发环境使用MindInsight，要求先启动MindInsight，后启动训练进程。
- 仅支持单机单卡训练。
- 运行中的可视化作业不单独计费，当停止Notebook实例时，计费停止。
- Summary文件如果存放在OBS中，由OBS单独收费。任务完成后请及时停止Notebook实例，清理OBS数据，避免产生不必要的费用。

在开发环境中创建 MindInsight 可视化作业流程

Step1 创建开发环境并在线打开

Step2 上传Summary数据

Step3 启动MindInsight

Step4 查看训练看板中的可视化数据

Step1 创建开发环境并在线打开

在ModelArts控制台，进入“开发环境> Notebook”页面，创建MindSpore引擎的开发环境实例。创建成功后，单击开发环境实例操作栏右侧的“打开 ”，在线打开运行中的开发环境。

MindInsight可视化训练作业，当前支持的镜像如下，请根据实际局点支持的镜像和资源规格选择使用。

- mindspore1.2.0版本，CPU/GPU规格的资源类型。
- mindspore1.5.x以上版本，Ascend规格的资源类型。

Step2 上传 Summary 数据

在开发环境中使用MindInsight可视化功能，需要用到Summary数据。

Summary数据可以直接传到开发环境的这个路径下/home/ma-user/work/，也可以放到OBS并行文件系统中。

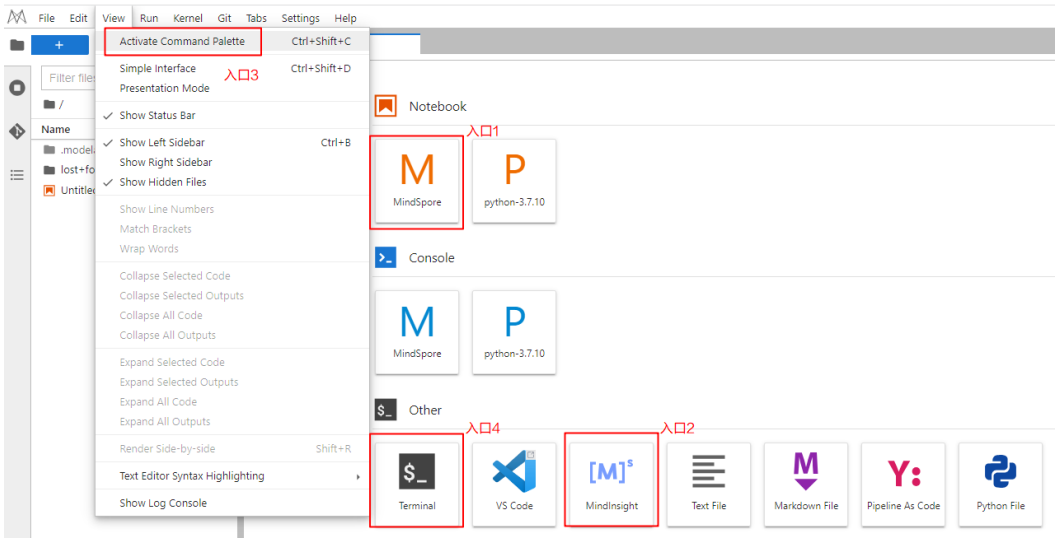
- Summary数据上传到Notebook路径/home/ma-user/work/下的方式，请参见**上传数据至Notebook**。
- Summary数据如果是通过OBS并行文件系统挂载到Notebook中，请将模型训练时产生的Summary文件先上传到OBS并行文件系统，并确保OBS并行文件系统与ModelArts在同一区域。在Notebook中启动MindInsight时，Notebook会自动从挂载的OBS并行文件系统目录中读取Summary数据。

 版权所有 © 华为云计算技术有限公司

Step3 启动 MindInsight

在开发环境的JupyterLab中打开MindInsight有多种方法。可根据使用习惯选择。

图 5-33 JupyterLab 中打开 MindInsight 的方法



方式1：

1. 单击入口1 MindSpore ，进入JupyterLab开发环境，并自动创建“.ipynb”文件。
2. 在对话框中输入MindInsight相应命令，即可展示界面。

```
%reload_ext mindinsight
%mindinsight --port {PORT} --summary-base-dir {SUMMARY_BASE_DIR}
```

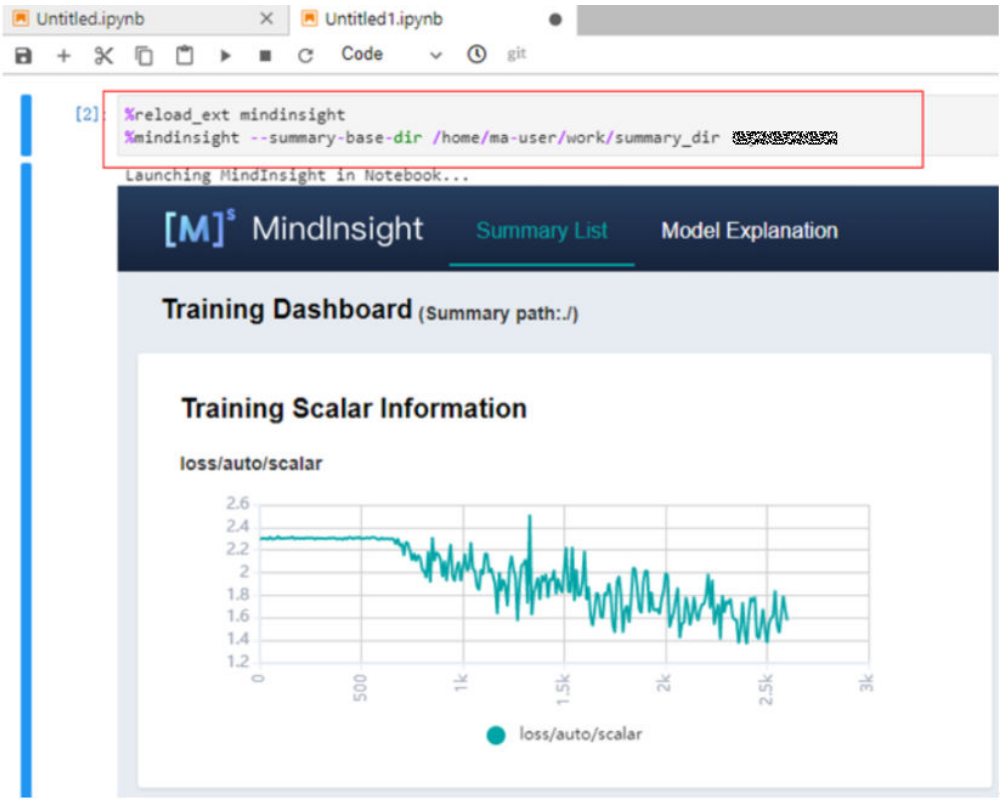
参数解释：

- --port {PORT}：指定Web可视化服务端口。可以不设置，默认使用8080端口。如果8080端口被占用了，需要在1~65535任意指定一个端口。
- --summary-base-dir {SUMMARY_BASE_DIR}：表示数据在开发环境中的存储路径。
  - 开发环境本地路径：“./work/xxx”（相对路径）或/home/ma-user/work/xxx（绝对路径）
  - OBS并行文件系统桶的路径：obs://xxx/

```
例如：
#Summary数据如果是在开发环境的这个路径下/home/ma-user/work/，执行下面这条命令
%mindinsight --summary-base-dir /home/ma-user/work/xxx
或者
#Summary数据如果是存在OBS并行文件系统中，执行下面这条命令，开发环境会自动挂载该OBS并行文件系统路径并读取数据
%mindinsight --summary-base-dir obs://xxx/
```


版权所有 © 华为云计算技术有限公司

图 5-34 MindInsight 界面（1）



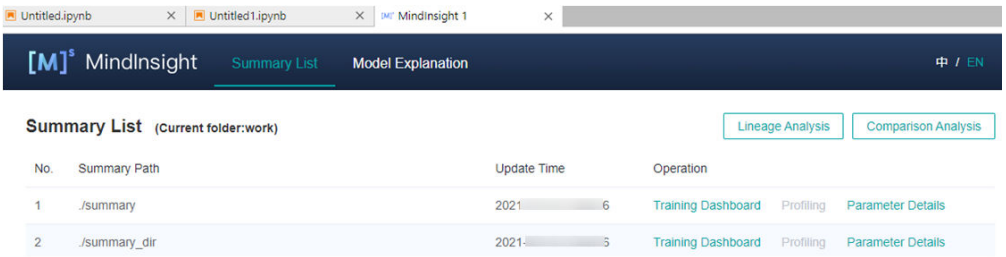
方式2：

单击入口2，直接进入MindInsight可视化界面。

默认读取路径/home/ma-user/work/

当存在两个以及以上工程的log时，界面如下。通过Runs下选择查看相对应的log。

图 5-35 MindInsight 界面（2）

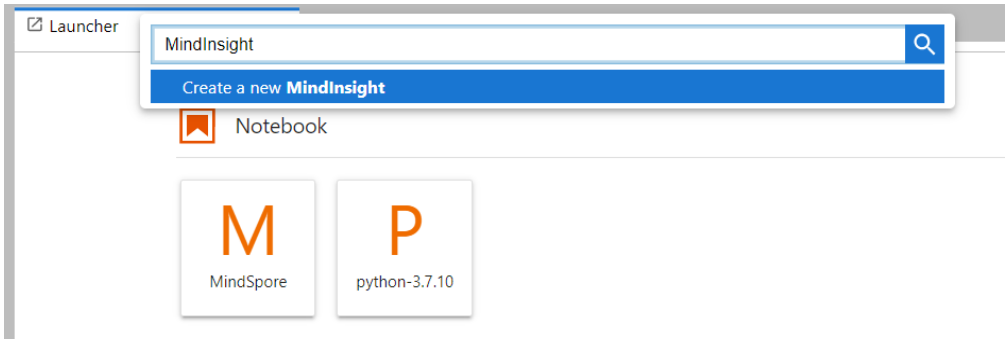


方式3：

1. 单击入口3 View -> Activate Command Palette，在搜索框中输入“MindInsight”，再单击“Create a new MindInsight”。

版权所有 © 华为云计算技术有限公司

图 5-36 Create a new MindInsight



2. 填写需要查看的Summary数据路径，或者OBS并行文件系统桶的路径，单击CREATE。
   - 开发环境本地路径：./summary（相对路径）或/home/ma-user/work/summary（绝对路径）
   - OBS并行文件系统的路径：obs://xxx/

图 5-37 输入 Summary 数据路径

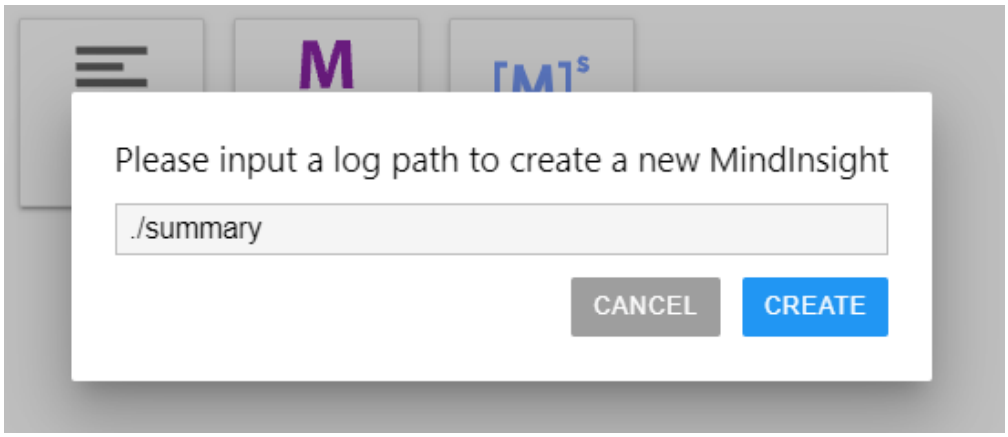
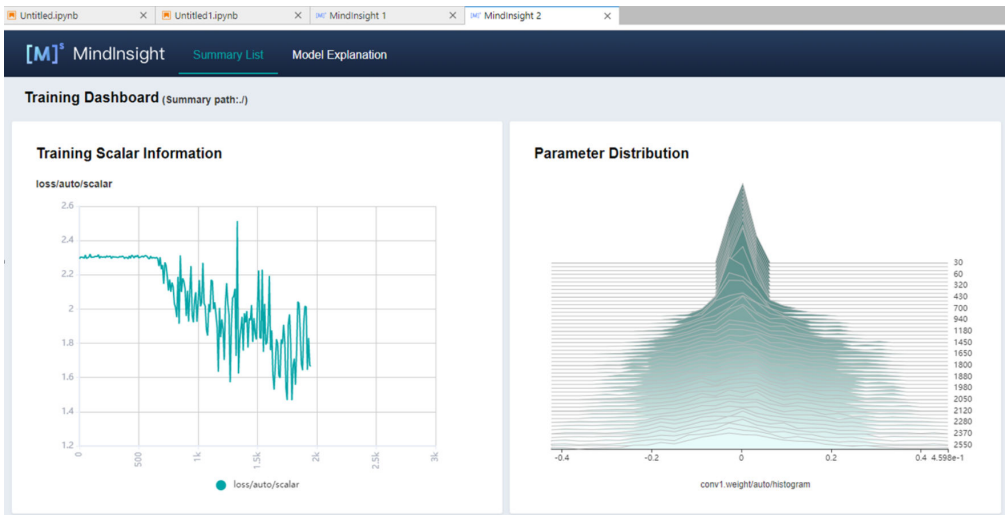


图 5-38 MindInsight 界面（3）



版权所有 © 华为云计算技术有限公司

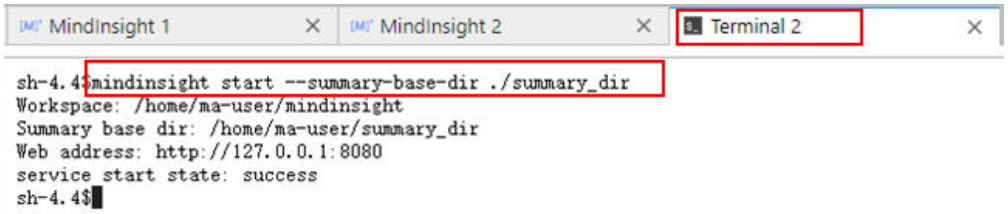
> **说明**
>
> 方式2及方式3最多可以启动10个MindInsight实例。

方式4：

单击入口4 Terminal ，输入并执行以下命令，但启动后不能显示UI界面。

```
mindinsight start --summary-base-dir ./summary_dir
```

图 5-39 Terminal 方式打开 MindInsight



Step4 查看训练看板中的可视化数据

训练看板是MindInsight的可视化组件的重要组成部分，而训练看板的标签包含：标量可视化、参数分布图可视化、计算图可视化、数据图可视化、图像可视化和张量可视化等。

更多功能介绍请参见MindSpore官网资料：**查看训练看板中可视的数据**。

相关操作

关闭MindInsight方式如下：

- 方式1：在开发环境JupyterLab中的“.ipynb”文件窗口中输入命令，关闭MindInsight。端口号在**启动MindInsight**中设置，默认使用8080，需要替换为实际开启MindInsight时的端口。
  ```
  !mindinsight stop --port 8080
  ```
- 方式2：单击下方 按钮进入MindInsight实例管理界面，该界面记录了所有启动的MindInsight实例，单击对应实例后面的SHUT DOWN即可停止该实例。

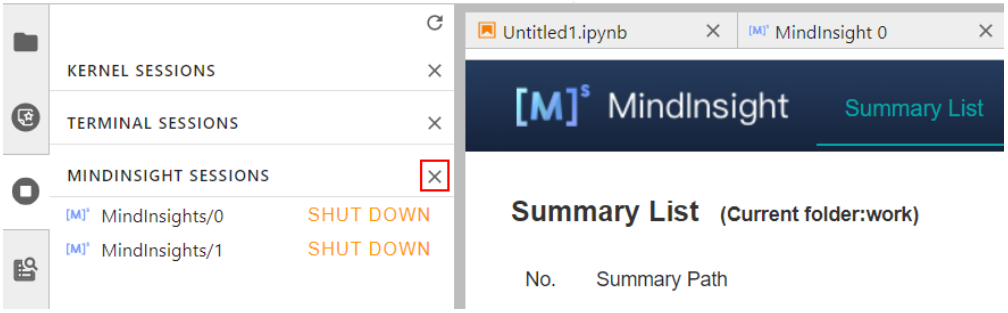
  图 5-40 单击 SHUT DOWN 停止实例

 版权所有 © 华为云计算技术有限公司

- 方式3：单击下方红框中的  按钮可以关闭所有启动的MindInsight实例。

  图 5-41 关闭所有启动的 MindInsight 实例

- 方式4（不推荐）：直接在JupyterLab中上关闭MindInsight窗口，此方式仅是关闭MindInsight可视化窗口，并未关闭后台。

5.6.3 TensorBoard 可视化作业

ModelArts支持在开发环境中开启TensorBoard可视化工具。TensorBoard是TensorFlow的可视化工具包，提供机器学习实验所需的可视化功能和工具。

TensorBoard能够有效地展示TensorFlow在运行过程中的计算图、各种指标随着时间的变化趋势以及训练中使用到的数据信息。

前提条件

为了保证训练结果中输出Summary文件，在编写训练脚本时，您需要在脚本中添加收集Summary相关代码。

TensorFlow引擎的训练脚本中添加Summary代码，具体方式请参见**TensorFlow官方网站**。

注意事项

- 运行中的可视化作业不单独计费，当停止Notebook实例时，计费停止。
- Summary文件数据如果存放在OBS中，由OBS单独收费。任务完成后请及时停止Notebook实例，清理OBS数据，避免产生不必要的费用。

在开发环境中创建 TensorBoard 可视化作业流程

Step1 创建开发环境并在线打开

Step2 上传Summary数据

Step3 启动TensorBoard

Step4 查看训练看板中的可视化数据

Step1 创建开发环境并在线打开

在ModelArts控制台，进入“开发环境 > Notebook”页面，创建TensorFlow或者PyTorch镜像的开发环境实例。创建成功后，单击开发环境实例操作栏右侧的“打开”，在线打开运行中的开发环境。

版权所有 © 华为云计算技术有限公司

TensorBoard可视化训练作业，当前仅支持基于TensorFlow2.1、Pytorch1.4/1.8以上版本镜像，CPU/GPU规格的资源类型。请根据实际局点支持的镜像和资源规格选择使用。

Step2 上传 Summary 数据

在开发环境中使用TensorBoard可视化功能，需要用到Summary数据。

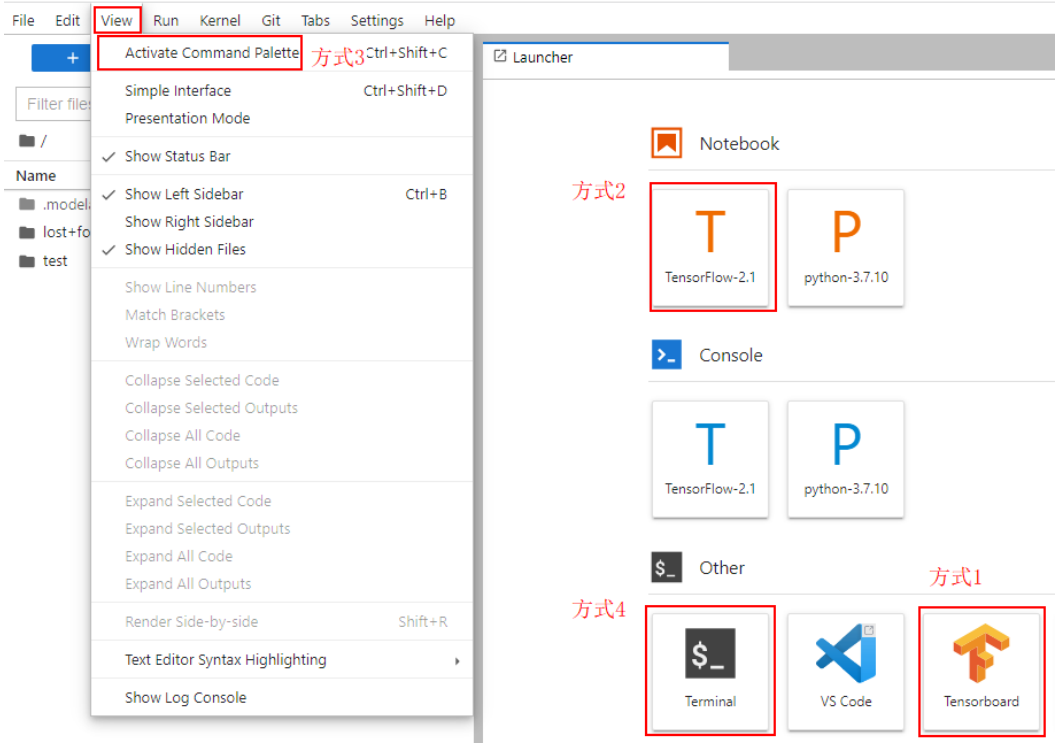
Summary数据可以直接传到开发环境的这个路径下/home/ma-user/work/，也可以放到OBS并行文件系统中。

- Summary数据上传到Notebook路径/home/ma-user/work/下的方式，请参见**上传数据至Notebook**。
- Summary数据如果是通过OBS并行文件系统挂载到Notebook中，请将模型训练时产生的Summary文件先上传到OBS并行文件系统，并确保OBS并行文件系统与ModelArts在同一区域。在Notebook中启动TensorBoard时，Notebook会自动从挂载的OBS并行文件系统目录中读取Summary数据。

Step3 启动 TensorBoard

在开发环境的JupyterLab中打开TensorBoard有多种方法。可根据使用习惯选择。

图 5-42 JupyterLab 中打开 TensorBoard 的方法



方式1（推荐）：

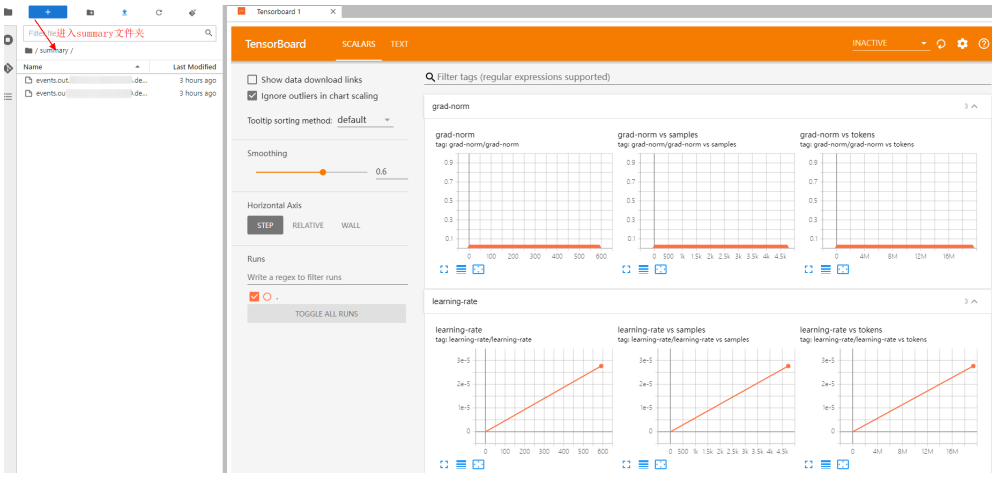
1. 在JupyterLab左侧导航创建名为“summary”的文件夹，将数据上传到“/home/ma-user/work/summary”路径。注：文件夹命名只能为summary否则无法使用。

版权所有 © 华为云计算技术有限公司

2. 进入“summary”文件夹，单击方式1 

，直接进入TensorBoard可视化界面。如图5-43所示。

图 5-43 TensorBoard 界面（1）



方式2：

须知

用户可以自行升级除2.4.0之外的TensorBoard，但需注意升级后只有方式2使用新的TensorBoard，其余方式保持TensorBoard2.1.1不变。

1. 单击方式2 

，进入JupyterLab开发环境中，并自动创建“.ipynb”文件。
2. 在对话框中输入TensorBoard相应命令，即可展示界面。

```
%reload_ext ma_tensorboard
%ma_tensorboard  --port {PORT} --logdir {BASE_DIR}
```

参数解释：

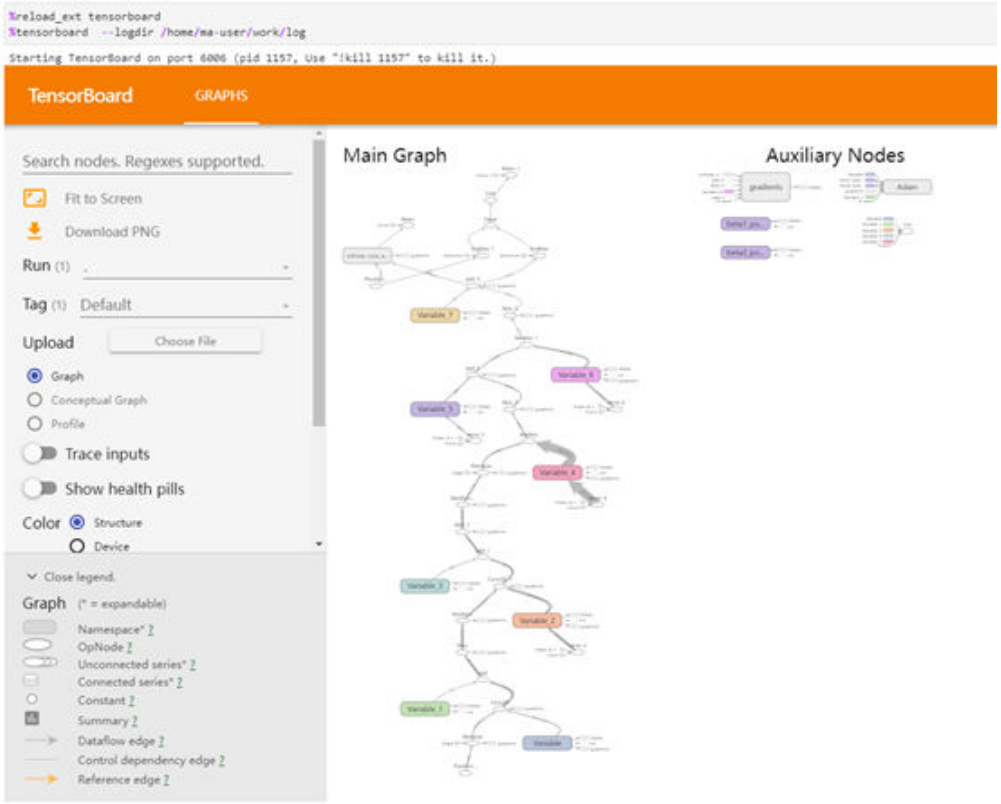
- --port {PORT}：指定Web可视化服务端口。可以不设置，默认使用8080端口。如果8080端口被占用了，需要在1~65535任意指定一个端口。
- --logdir {BASE_DIR}：表示数据在开发环境中的存储路径
  - 开发环境本地路径：“./work/xxx”（相对路径）或/home/ma-user/work/xxx（绝对路径）

版权所有 © 华为云计算技术有限公司

- OBS并行文件系统的路径：obs://xxx/

例如：
#Summary数据如果是在开发环境的这个路径下/home/ma-user/work/，执行下面这条命令。
%ma_tensorboard --port {PORT} --logdir /home/ma-user/work/xxx
或者
#Summary数据如果是存在OBS并行文件系统中，执行下面这条命令，开发环境会自动挂载该OBS并行文件系统路径并读取数据。
%ma_tensorboard --port {PORT} --logdir obs://xxx/

图 5-44 TensorBoard 界面（2）

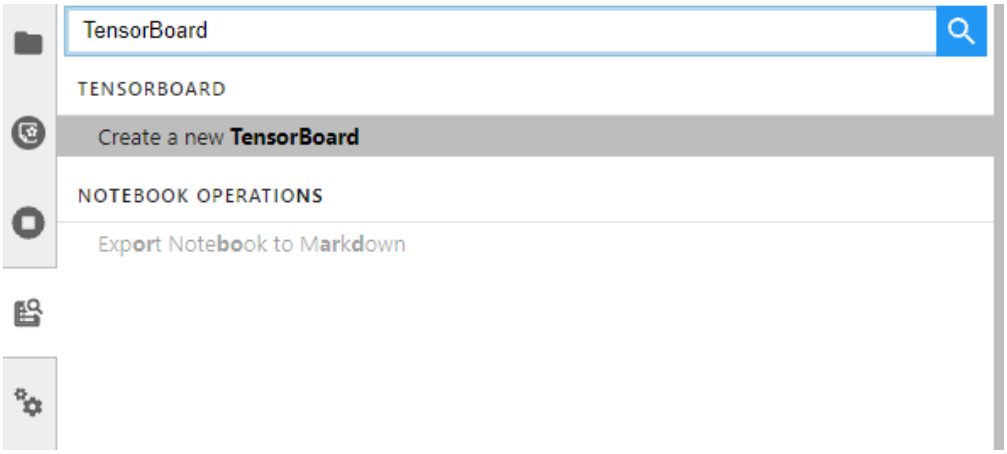


方式3：

1. 单击方式3 View -> Activate Command Palette，在搜索框中输入“TensorBoard”，再单击“Create a new TensorBoard”。

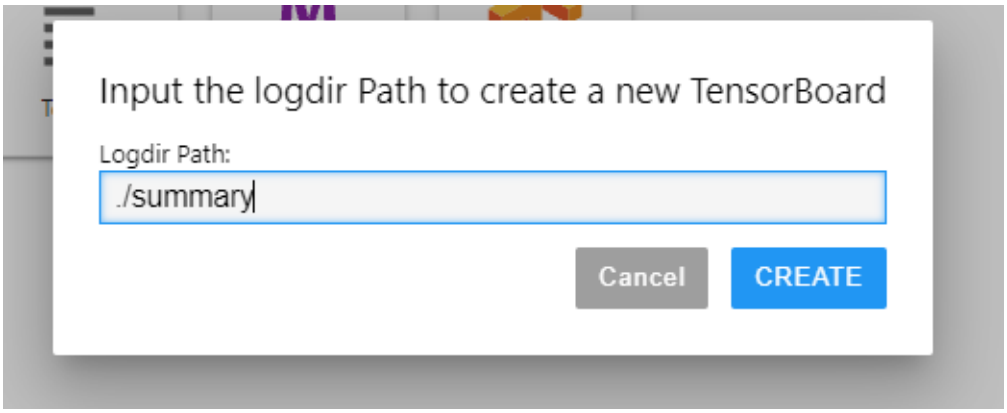
版权所有 © 华为云计算技术有限公司

图 5-45 Create a new TensorBoard



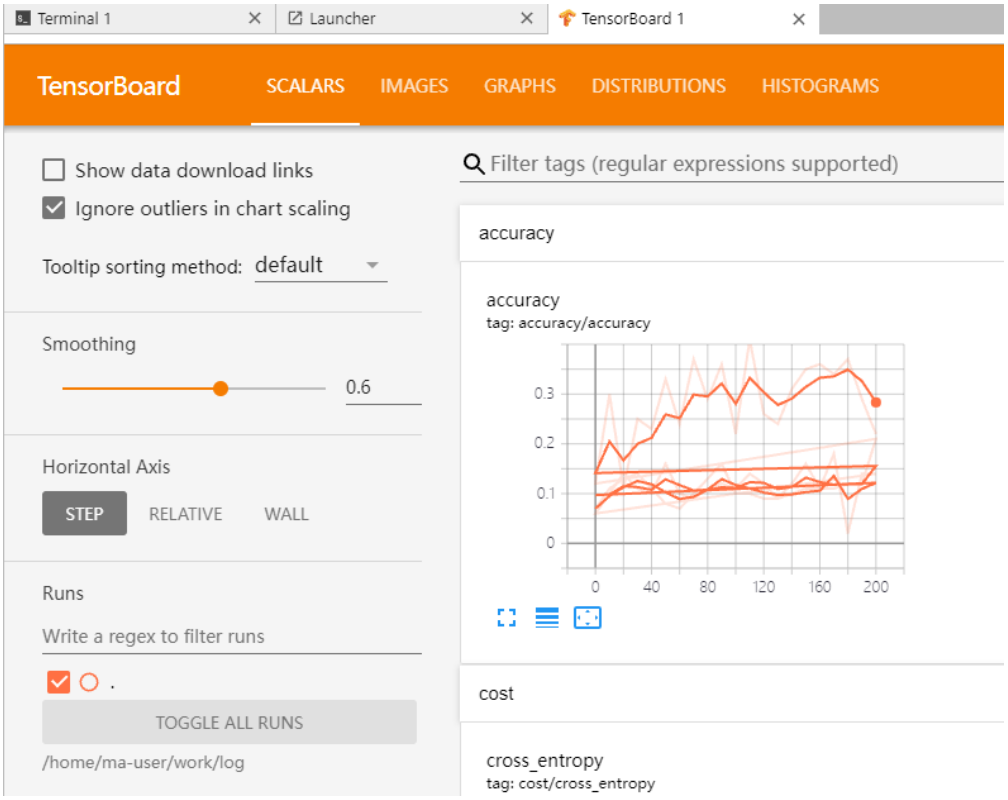
2. 填写需要查看的Summary数据路径，或者OBS并行文件系统桶的路径。
   - 开发环境本地路径：./summary（相对路径）或/home/ma-user/work/summary（绝对路径）
   - OBS并行文件系统桶的路径：obs://xxx/

图 5-46 输入 Summary 数据路径



版权所有 © 华为云计算技术有限公司

图 5-47 TensorBoard 界面（3）



方式4：

单击方式4 Terminal ，输入命令执行，但启动后不能显示UI界面。

```
tensorboard --logdir ./log
```

图 5-48 Terminal 方式打开 TensorBoard

Step4 查看训练看板中的可视化数据

训练看板是TensorBoard的可视化组件的重要组成部分，而训练看板的标签包含：标量可视化、图像可视化和计算图可视化等。

更多功能介绍请参见**TensorBoard官网资料**。

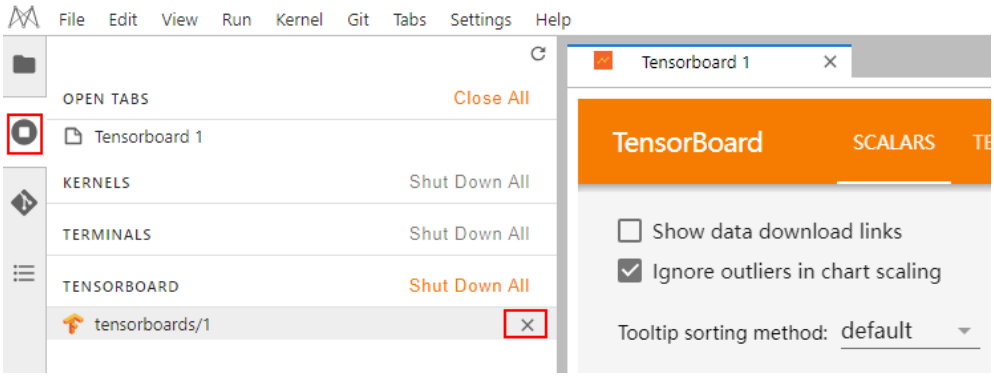
相关操作

关闭TensorBoard方式如下：

 版权所有 © 华为云计算技术有限公司

- 方式1：单击下图所示的，进入TensorBoard实例管理界面，该界面记录了所有启动的TensorBoard实例，单击对应实例后面的SHUT DOWN即可停止该实例。

图 5-49 单击 SHUT DOWN 停该实例

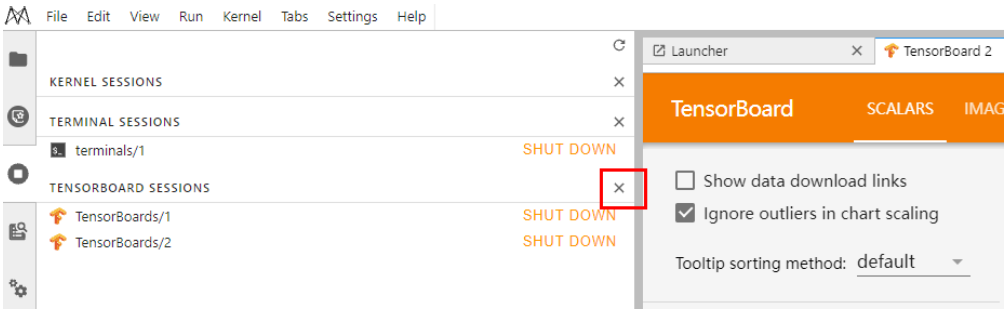


- 方式2：在开发环境JupyterLab中的“.ipynb”文件窗口中输入命令，关闭TensorBoard。PID在启动界面有提示或者通过ps -ef | grep tensorboard查看。

```
!kill PID
```

- 方式3：单击下方红框中的 按钮可以关闭所有启动的TensorBoard实例。

图 5-50 关闭所有启动的 TensorBoard 实例



- 方式4（不推荐）：直接在JupyterLab中上关闭TensorBoard窗口，此方式仅关闭可视化窗口，并未关闭后台。

5.7 Notebook 中的数据上传下载

5.7.1 上传文件至 JupyterLab

5.7.1.1 场景介绍

在AI开发过程中，如何将文件方便快速地上传到Notebook几乎是每个开发者都会遇到的问题。

ModelArts之前对文件直接上传到Notebook的大小限制是100MB，超过限制的文件无法直接上传；其次需要上传的文件并不都在本地，可能是GitHub的开源仓库，可能是类似开源数据集（**https://nodejs.org/dist/v12.4.0/node-v12.4.0-linux-x64.tar.xz**）

版权所有 © 华为云计算技术有限公司

这样的远端文件，也可能是存放在OBS中的文件，ModelArts之前无法将这些文件直接上传到Notebook中；在文件上传过程中，用户无法获得更多的信息，例如上传进度和速度。

ModelArts上传文件特性主要解决了以上三个问题，不仅提供了更多上传文件的功能满足用户需求，而且展示了更多文件上传的细节，提升了用户的体验。

当前的文件上传功能：

- 支持上传本地文件；
- 支持Clone GitHub开源仓库；
- 支持上传OBS文件；
- 支持上传远端文件；
- 将文件上传详情可视化。

5.7.1.2 上传本地文件至 JupyterLab

5.7.1.2.1 上传场景和入口介绍

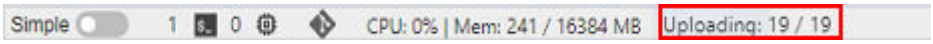
Notebook的JupyterLab中提供了多种方式上传文件。

上传文件要求

- 对于大小不超过100MB的文件直接上传，并展示文件大小、上传进度及速度等详细信息。
- 对于大小超过100MB不超过5GB的文件可以使用OBS中转，系统先将文件上传OBS（对象桶或并行文件系统），然后从OBS下载到Notebook，上传完成后，会将文件从OBS中删除。
- 5GB以上的文件上传通过调用ModelArts SDK或者Moxing完成。
- 对于Notebook当前目录下已经有同文件名称的文件，可以覆盖继续上传，也可以取消。
- 支持10个文件同时上传，其余文件显示“等待上传”。不支持上传文件夹，可以将文件夹压缩成压缩包上传至Notebook后， 在Terminal中解压压缩包。

  ```
  unzip xxx.zip  #在xxx.zip压缩包所在路径直接解压
  ```

  解压命令的更多使用说明可以在主流搜索引擎中查找Linux解压命令操作。
- 多个文件同时上传时，JupyterLab窗口最下面会显示上传文件总数和已上传文件数。

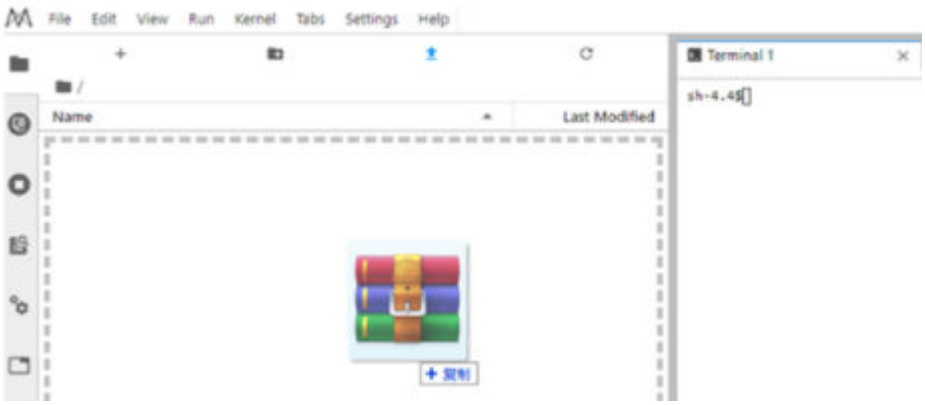
前提条件

使用JupyterLab打开一个运行中的Notebook环境。

上传文件入口 1：将文件直接拖拽至浏览器窗口中

直接将文件拖拽到JupyterLab窗口左边的空白处上传。

版权所有 © 华为云计算技术有限公司



上传文件入口 2：通过上传图标传文件

单击窗口上方导航栏的 ModelArts Upload Files按钮，打开文件上传界面，拖拽或者选择本地文件上传。

图 5-51 上传文件图标

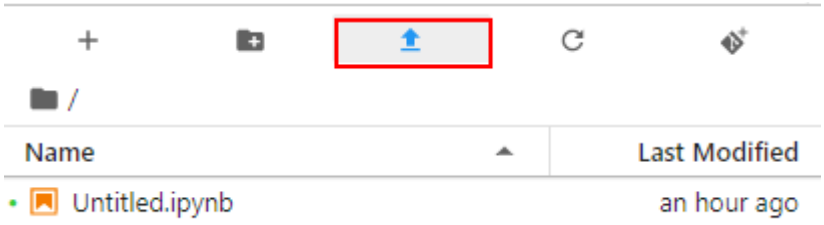


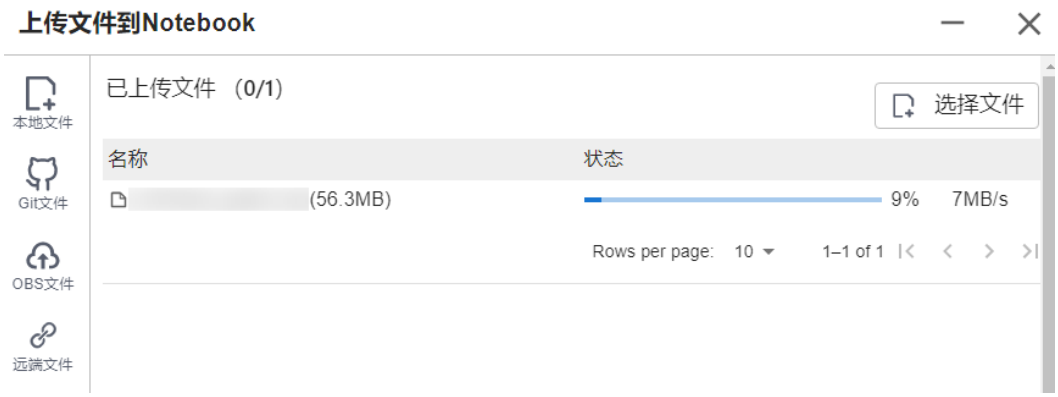
图 5-52 上传文件窗口

版权所有 © 华为云计算技术有限公司

5.7.1.2.2 上传本地小文件（100MB 以内）至 JupyterLab

对于大小不超过100MB的文件直接上传，并展示文件大小、上传进度及速度等详细信息。

图 5-53 上传 100MB 以下小文件



文件上传完成后给出提示。

图 5-54 上传成功

5.7.1.2.3 上传本地大文件（100MB~5GB）至 JupyterLab

对于大小超过100MB不超过5GB的文件可以使用OBS中转，系统先将文件上传至OBS（对象桶或并行文件系统），然后从OBS下载到Notebook。下载完成后，ModelArts会将文件自动从OBS中删除。

例如，对于下面这种情况，可以通过“OBS中转”上传。

版权所有 © 华为云计算技术有限公司

图 5-55 通过 OBS 中转上传大文件

如果使用OBS中转需要提供一个OBS中转路径，可以通过以下三种方式提供：

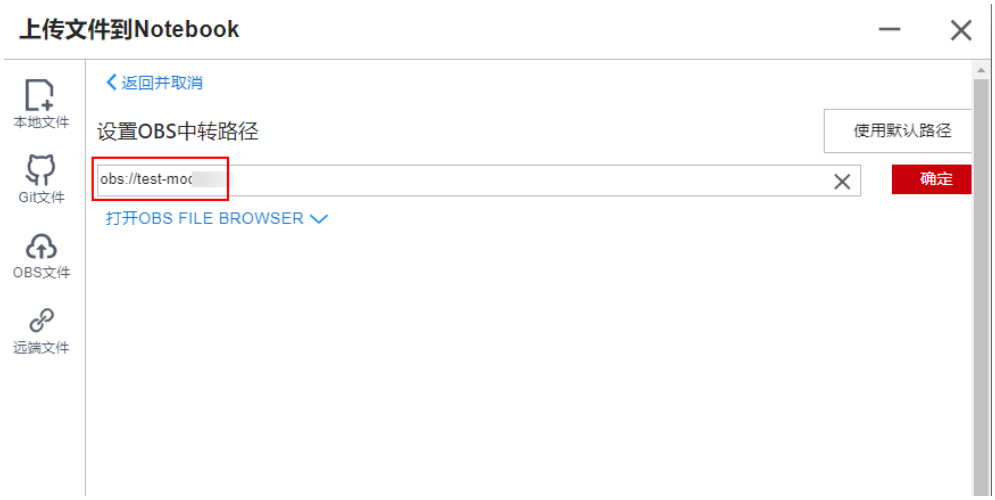
图 5-56 通过 OBS 中转路径上传

说明

仅第一次单击“OBS中转”需要提供OBS中转路径，以后默认使用该路径直接上传，可以通过上传文件窗口左下角的设置按钮 更新OBS中转路径。如图5-60所示。

- 方式一：在输入框中直接输入有效的OBS中转路径，然后单击“确定”完成。

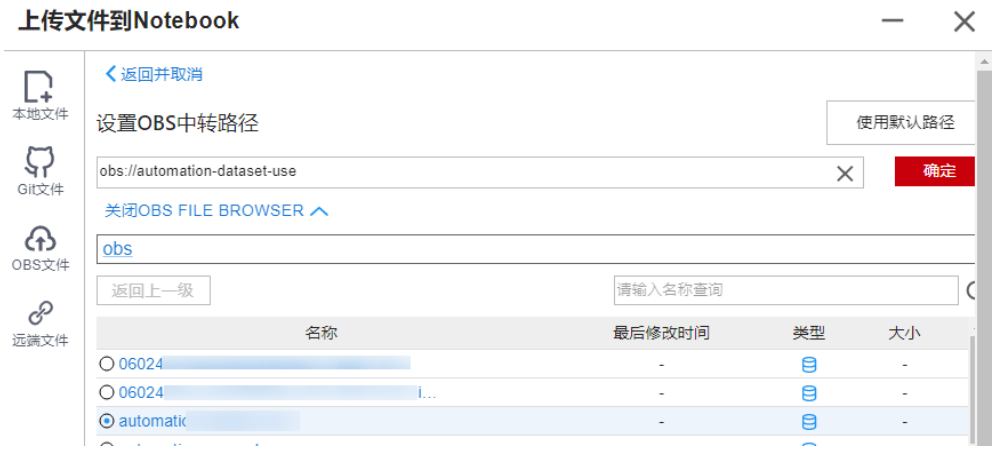
  图 5-57 输入有效的 OBS 中转路径

版权所有 © 华为云计算技术有限公司

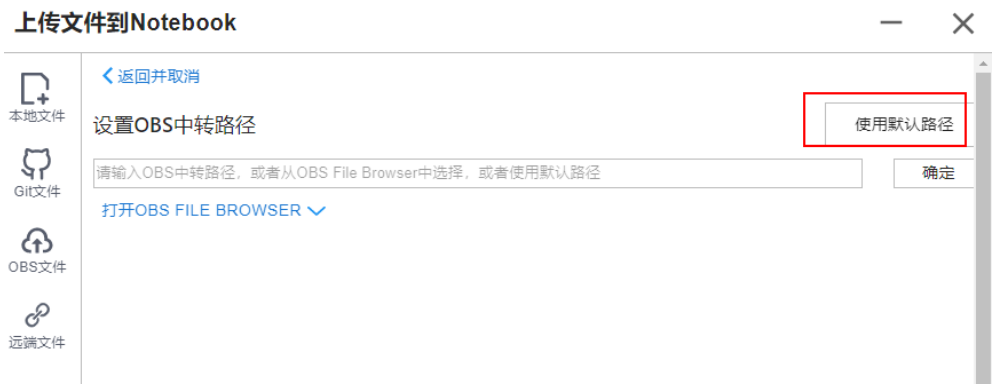
- 方式二：打开OBS File Browser选择一个OBS中转路径，然后单击“确定”完成。

图 5-58 打开 OBS File Browser



- 方式三：单击“使用默认路径”完成。

图 5-59 使用默认路径上传文件

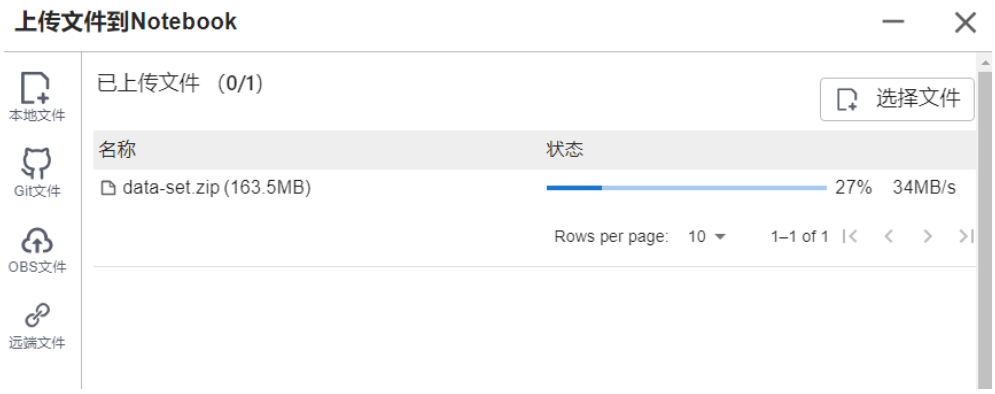


版权所有 © 华为云计算技术有限公司

图 5-60 设置本地文件 OBS 中转路径

完成OBS中转路径设置后，开始上传文件。

图 5-61 上传文件



解压缩文件包

将文件以压缩包形式上传至Notebook JupyterLab后， 可在Terminal中解压缩文件包。

```
unzip xxx.zip  #在xxx.zip压缩包所在路径直接解压
```

解压命令的更多使用说明可以在主流搜索引擎中查找Linux解压命令操作。

5.7.1.2.4 上传本地超大文件（5GB 以上）至 JupyterLab

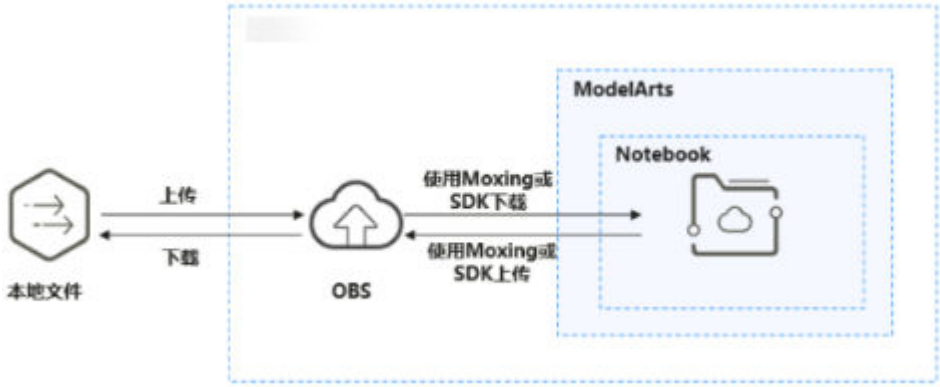
不支持在Notebook的JupyterLab中直接上传大小超过5GB的文件。

版权所有 © 华为云计算技术有限公司

图 5-62 不支持直接上传大小超过 5GB 的文件

5GB以上的文件需要先从本地上传到OBS中，再在Notebook中调用ModelArts的Moxing接口或者SDK接口读写OBS中的文件。

图 5-63 在 Notebook 中上传下载大文件



具体操作如下：

1. 从本地上传文件至OBS。具体操作请参见**上传文件至OBS桶**。
2. 将OBS中的文件下载到Notebook，可以通过在Notebook中运行代码的方式完成数据下载，具体方式有2种，ModelArts的SDK接口或者调用MoXing接口。
   - 方法一：使用ModelArts SDK接口将OBS中的文件下载到Notebook后进行操作。

     示例代码：

     ```
     from modelarts.session import Session
     session = Session()
     session.obs.copy("obs://bucket-name/obs_file.txt","/home/ma-user/work/")
     ```

   - 方法二：使用**Moxing操作OBS文件**将OBS中的文件同步到Notebook后进行操作。

     ```
     import moxing as mox

     #下载一个OBS文件夹sub_dir_0，从OBS下载至Notebook
     mox.file.copy_parallel('obs://bucket_name/sub_dir_0', '/home/ma-user/work/sub_dir_0')
     #下载一个OBS文件obs_file.txt，从OBS下载至Notebook
     mox.file.copy('obs://bucket_name/obs_file.txt', '/home/ma-user/work/obs_file.txt')
     ```

     如果下载到Notebook中的是zip文件，在Terminal中执行下列命令，解压压缩包。

     ```
     unzip xxx.zip  #在xxx.zip压缩包所在路径直接解压
     ```

     代码执行完成后，参考**图5-64**打开Terminal后执行**ls /home/ma-user/work**命令查看下载到Notebook中的文件。或者在Jupyter左侧导航中显示下载的文件，如果没有显示，请刷新后查看，如**图5-65**所示。

版权所有 © 华为云计算技术有限公司

图 5-64 打开 Terminal

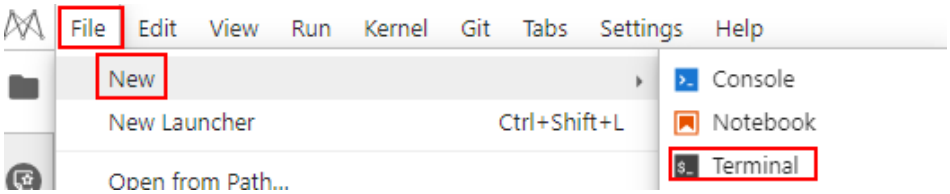
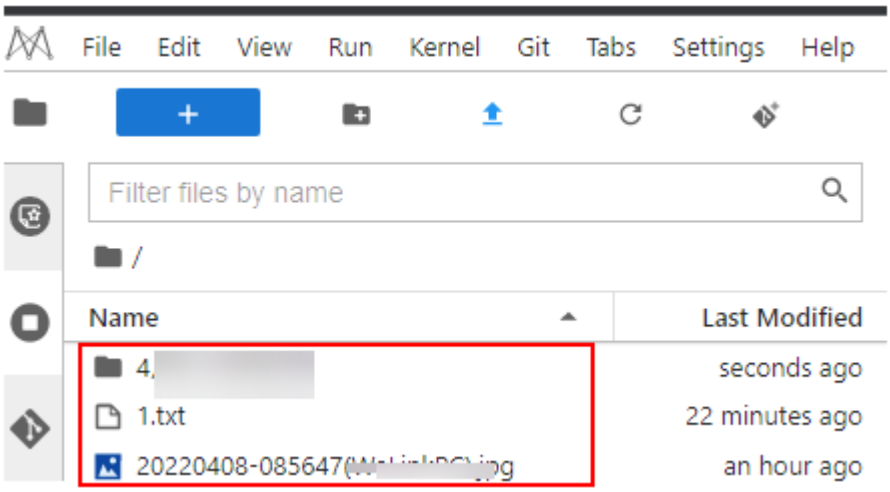


图 5-65 查看下载到 Notebook 中的文件



异常处理

通过OBS下载文件到Notebook中时，提示Permission denied。请依次排查：

- 请确保读取的OBS桶和Notebook处于同一站点区域，例如：都在华北-北京四站点。不支持跨站点访问OBS桶。
- 请确认操作Notebook的账号有权限读取OBS桶中的数据。

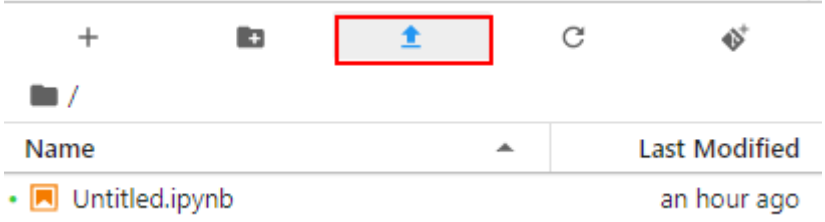
具体请参见**ModelArts中提示OBS路径错误**。

5.7.1.3 GitHub 开源仓库 Clone

在Notebook的JupyterLab中，支持从GitHub开源仓库Clone文件。

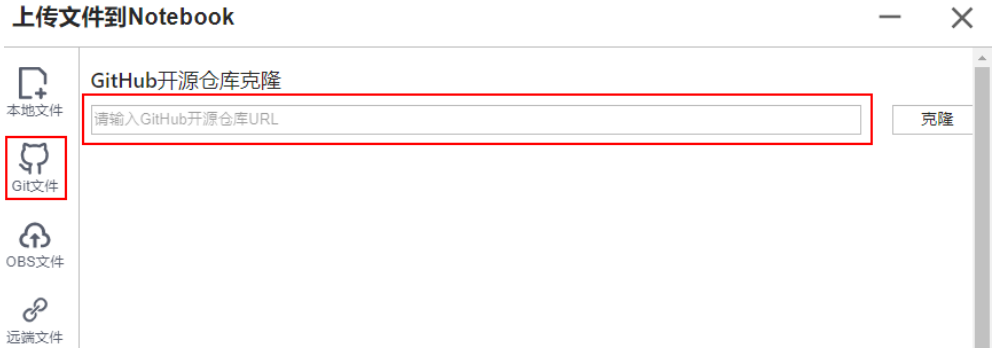
1. 通过JupyterLab打开一个运行中的Notebook。
2. 单击JupyterLab窗口上方导航栏的 ModelArts Upload Files按钮，打开文件上传窗口，选择左侧的 Git文件 进入GitHub开源仓库Clone界面。

图 5-66 上传文件图标



版权所有 © 华为云计算技术有限公司

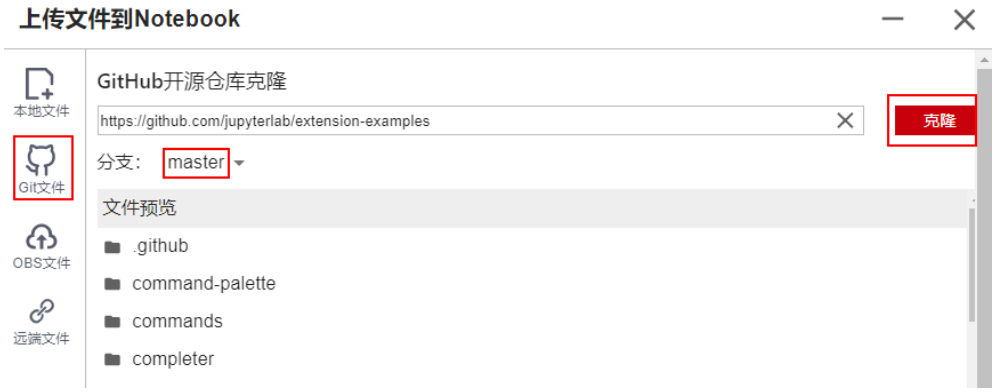
图 5-67 进入 GitHub 开源仓库 Clone 界面



3. 输入有效的GitHub开源仓库地址后会展示该仓库下的文件及文件夹，说明用户输入了有效的仓库地址，同时给出该仓库下所有的分支供选择，选择完成后单击“克隆”开始Clone仓库。

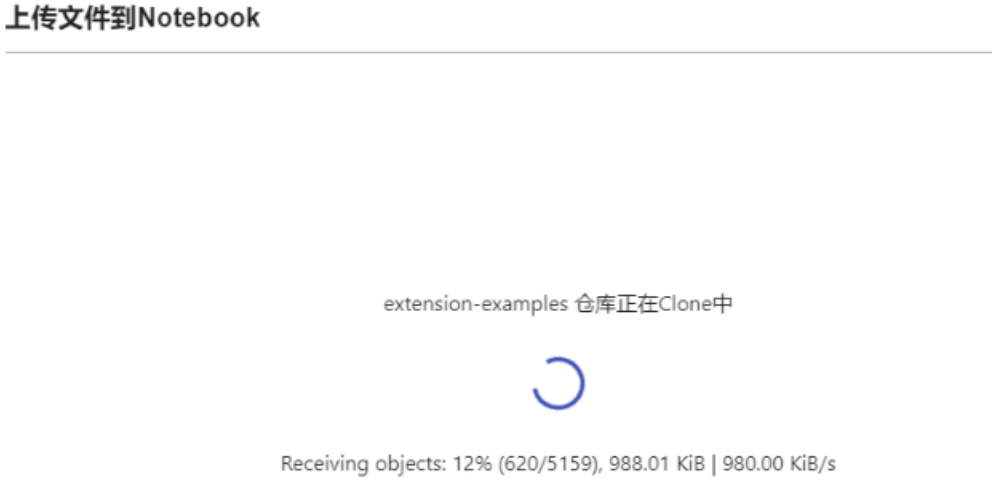
   GitHub开源仓库地址：https://github.com/jupyterlab/extension-examples

图 5-68 输入有效的 GitHub 开源仓库地址



4. Clone仓库的过程中会将进度展示出来。

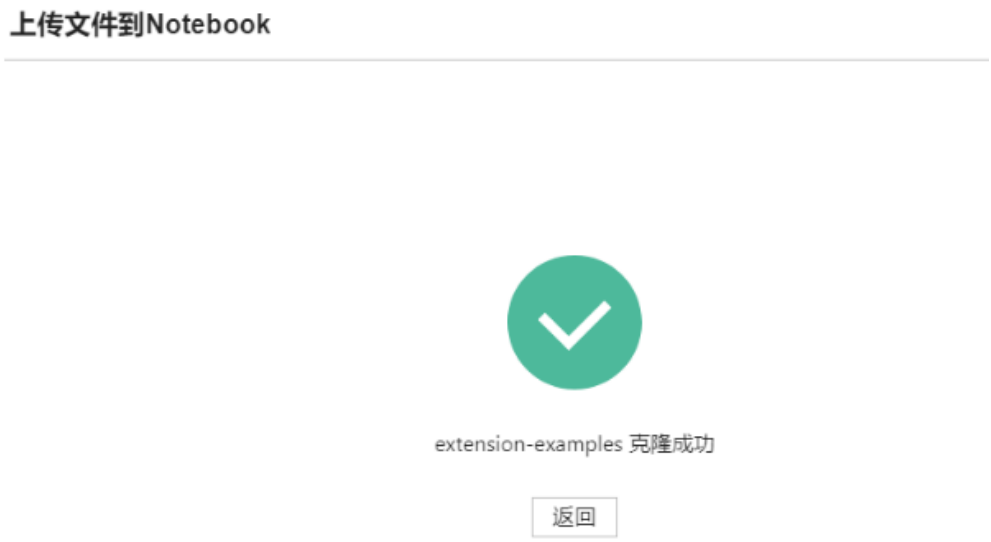
图 5-69 Clone 仓库的过程



5. Clone仓库成功。

版权所有 © 华为云计算技术有限公司

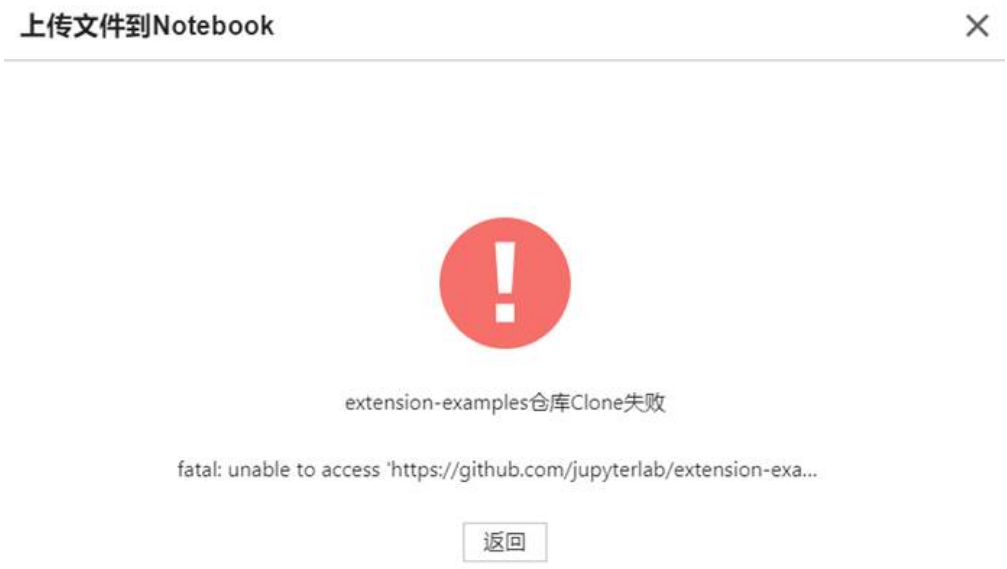
图 5-70 Clone 仓库成功



异常处理

- Clone仓库失败。可能是网络原因问题。可以在JupyterLab的Terminal中通过执行git clone https://github.com/jupyterlab/extension-examples.git测试网络连通情况。

  图 5-71 Clone 仓库失败

- 如果克隆时遇到Notebook当前目录下已有该仓库，系统给出提示仓库名称重复，此时可以单击“覆盖”继续克隆仓库，也可以单击 取消。

5.7.1.4 上传 OBS 文件到 JupyterLab

在Notebook的JupyterLab中，支持将OBS中的文件下载到Notebook。注意：文件大小不能超过10GB，否则会上传失败。

版权所有 © 华为云计算技术有限公司

1. 通过JupyterLab打开一个运行中的Notebook。
2. 单击JupyterLab窗口上方导航栏的 ModelArts Upload Files按钮，打开文件上传窗口，选择左侧的 进入OBS文件上传界面。

图 5-72 上传文件图标

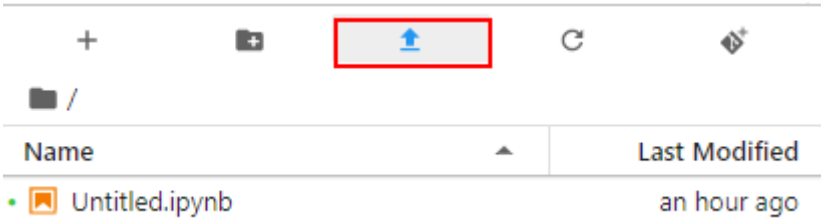
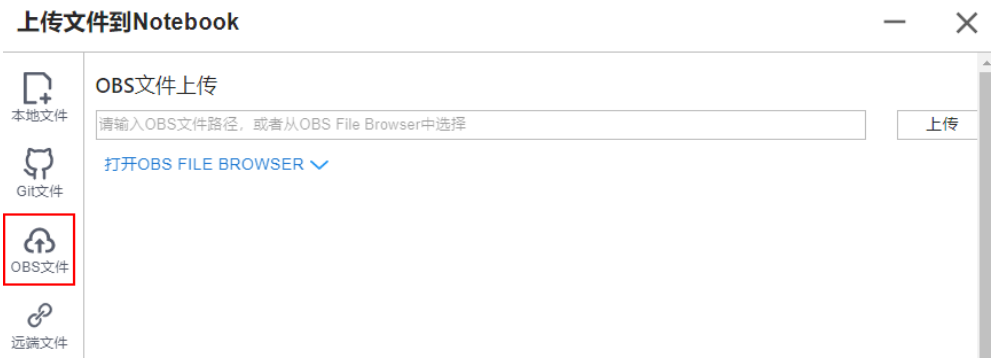
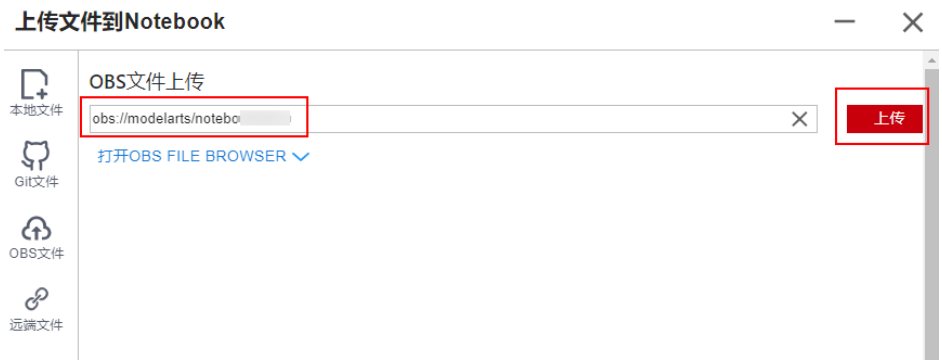


图 5-73 OBS 文件上传界面



3. 需要提供OBS文件路径，可以通过以下两种方式提供：
   - 方式一：在输入框中直接输入有效的OBS文件路径，然后单击“上传”开始传文件。

     图 5-74 输入有效的 OBS 文件路径

     说明

     此处输入的是具体的OBS文件路径，不是文件夹的路径，否则会导致上传失败。

   - 方式二：打开OBS File Browser选择OBS文件路径，然后单击“上传”，开始上传文件。

版权所有 © 华为云计算技术有限公司

图 5-75 上传 OBS 文件

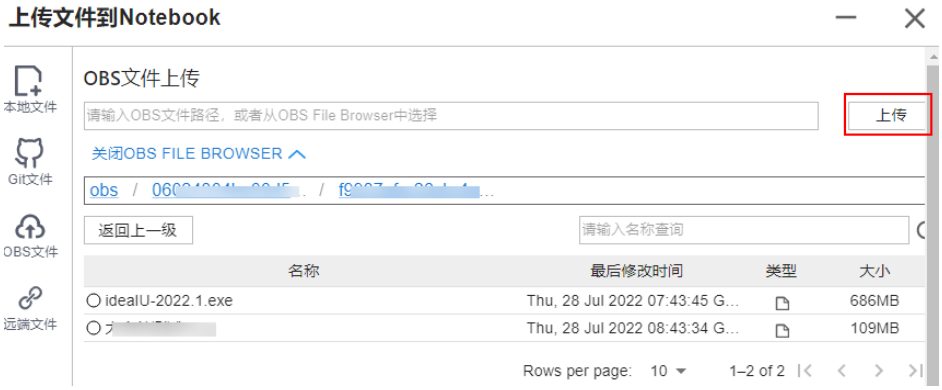
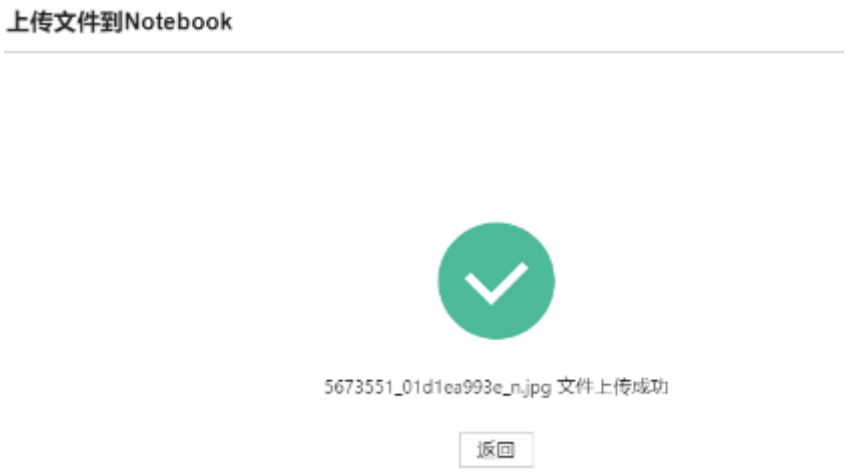


图 5-76 文件上传成功



异常处理

提示文件上传失败，有以下三种常见场景。

- **异常场景1**

版权所有 © 华为云计算技术有限公司

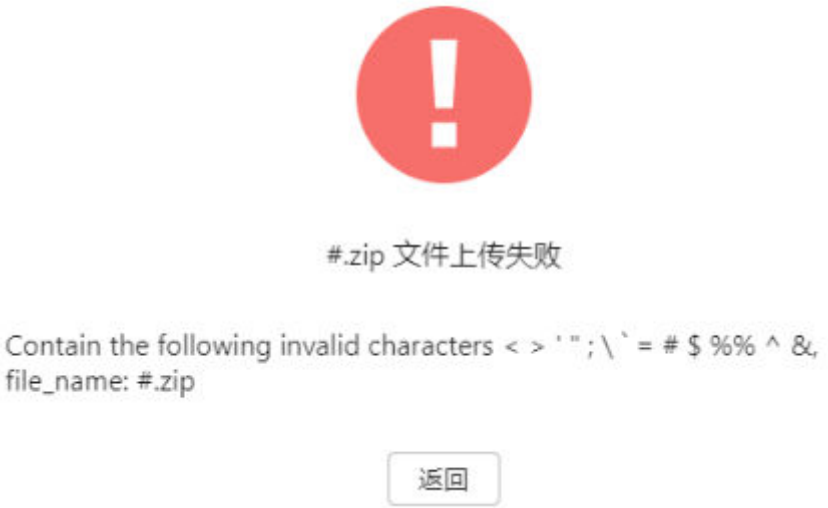
图 5-77 文件上传失败

可能原因：

- OBS路径没有设置为具体的文件路径，设置成了文件夹。
- OBS中的文件设置了加密。请前往OBS控制台查看，确保该文件未加密。
- OBS桶和Notebook不在同一个区域。请确保读取的OBS桶和Notebook处于同一站点区域，不支持跨站点访问OBS桶。例如：都在华北-北京四站点。具体操作请参见**如何查看OBS桶与ModelArts是否在同一区域**。
- 没有该OBS桶的访问权限。请确认操作Notebook的账号有权限读取OBS桶中的数据。具体操作请参见**检查您的账号是否有该OBS桶的访问权限**。
- OBS文件被删除。请确认待上传的OBS文件是否存在。

- **异常场景2**

图 5-78 文件上传失败



版权所有 © 华为云计算技术有限公司

可能原因：
文件名包含<>'";\`=#$%^&等特殊字符。

- **异常场景3**

图 5-79 文件上传失败

可能原因：
文件大小超过10GB导致上传失败。

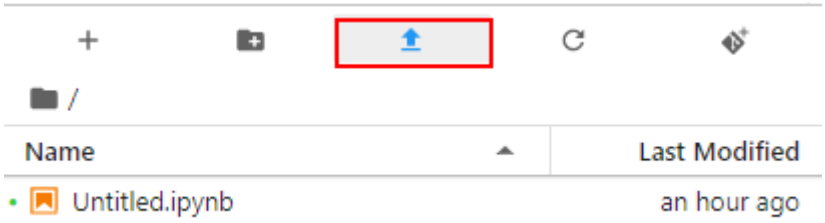
5.7.1.5 上传远端文件至 JupyterLab

在Notebook的JupyterLab中，支持通过远端文件地址下载文件。

要求：远端文件的URL粘贴在浏览器的输入框中时，可以直接下载该文件。

1. 通过JupyterLab打开一个运行中的Notebook。
2. 单击JupyterLab窗口上方导航栏的 ModelArts Upload Files按钮，打开文件上传窗口，选择左侧的 进入远端文件上传界面。

图 5-80 上传文件图标



 版权所有 © 华为云计算技术有限公司

图 5-81 进入远端文件上传界面

3. 输入有效的远端文件URL后，系统会自动识别上传文件名称，单击“上传”，开始上传文件。

图 5-82 输入有效的远端文件 URL

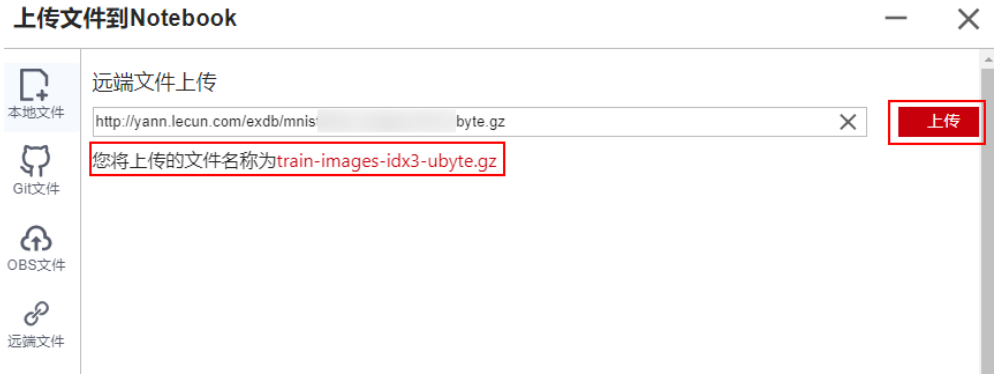
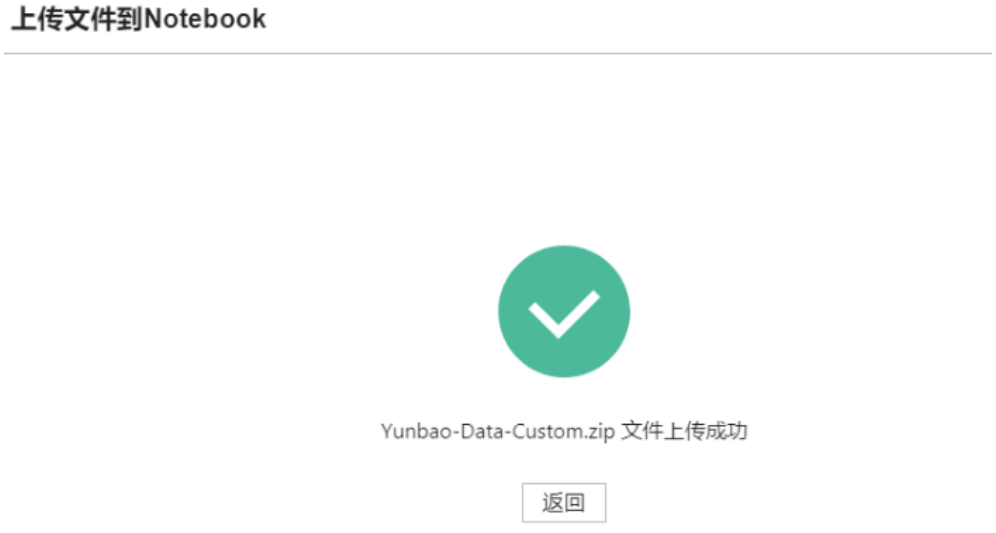


图 5-83 远端文件上传成功

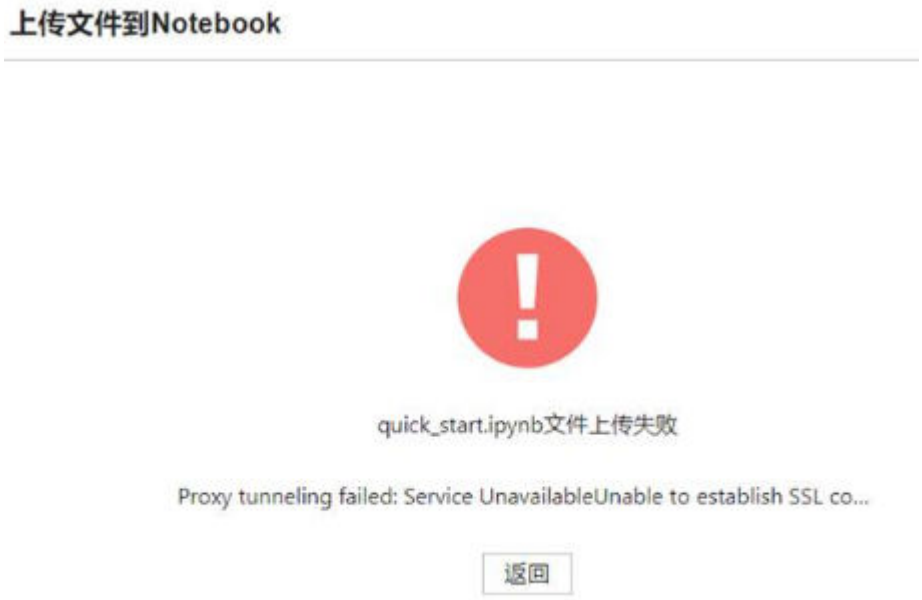


异常处理

远端文件上传失败。可能是网络原因。请先在浏览器中输入该远端文件的URL地址，测试该文件是否能下载。

版权所有 © 华为云计算技术有限公司

图 5-84 远端文件上传失败



5.7.2 从 JupyterLab 下载文件至本地

在JupyterLab中开发的文件，可以下载至本地。

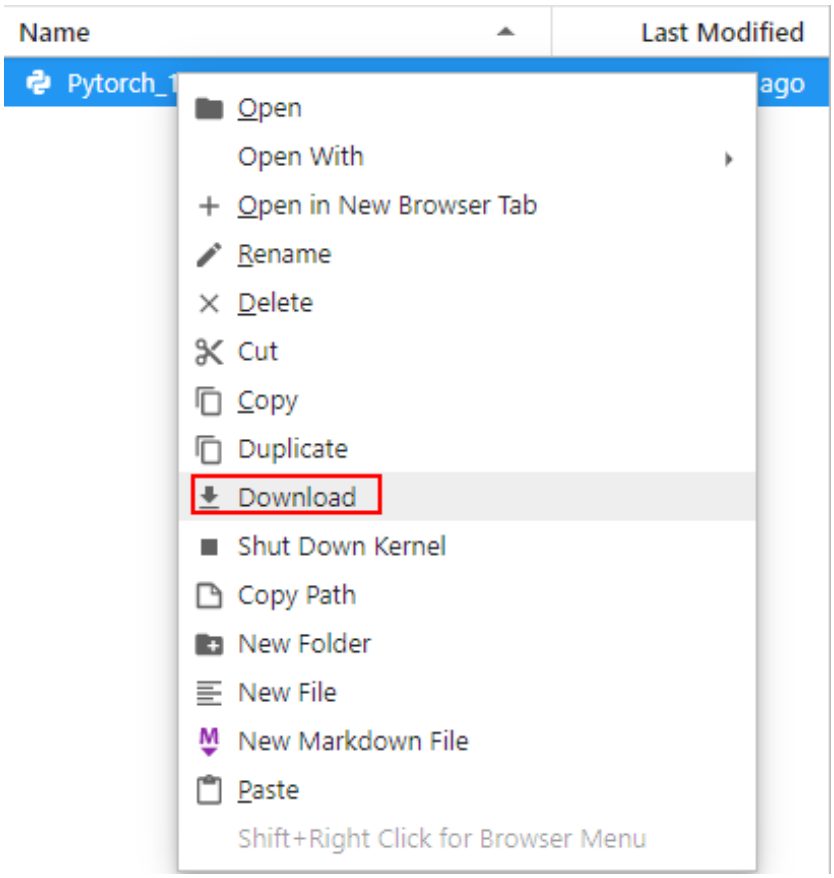
- 不大于100MB的文件，可以直接从JupyterLab中下载到本地，具体操作请参见**从JupyterLab中下载不大于100MB的文件至本地**。
- 大于100MB的文件，需要先从JupyterLab上传到OBS，再通过OBS下载到本地，具体操作请参见**从JupyterLab中下载大于100MB的文件到本地**。

从 JupyterLab 中下载不大于 100MB 的文件至本地

在JupyterLab文件列表中，选择需要下载的文件，单击右键，在操作菜单中选择“Download”下载至本地。下载的目的路径，为您本地浏览器设置的下载目录。

版权所有 © 华为云计算技术有限公司

图 5-85 下载文件



从 JupyterLab 中下载大于 100MB 的文件到本地

大于100MB的文件需要先从Notebook中上传到OBS，再从OBS下载到本地，具体操作如下：

1. 在Notebook中，新建一个大于100MB的“ipynb”文件，使用MoXing先将该文件从Notebook上传到OBS中，示例代码如下：

   ```
   import moxing as mox
   mox.file.copy('/home/ma-user/work/obs_file.txt', 'obs://bucket_name/obs_file.txt')
   ```

   其中“/home/ma-user/work/obs_file.txt”为文件在Notebook中的存储路径，“obs://bucket_name/obs_file.txt”为该文件上传到OBS的存储路径，其中“bucket_name”为OBS中创建的桶的名称，“obs_file.txt”为上传的文件。
2. 使用OBS或ModelArts SDK将OBS中的文件下载到本地。
   - 方式一：使用OBS进行下载
   - 在OBS中，可以将样例中的“obs_file.txt”下载到本地。如果您的数据较多，推荐OBS Browser+下载数据或文件夹。使用OBS下载文件的操作指导参见**下载文件**
   - 方式二：使用ModelArts SDK进行下载
     i. 在您的本地环境**下载并安装ModelArts SDK**。
     ii. 完成ModelArts SDK的**Session鉴权**。
     iii. 将OBS中的文件下载到本地，详请参见**从OBS下载数据**。示例代码如下：

     ```
     from modelarts.session import Session
     ```


版权所有 © 华为云计算技术有限公司

```
# 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
# 本示例以ak和sk保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量HUAWEICLOUD_SDK_AK和HUAWEICLOUD_SDK_SK。
__AK = os.environ["HUAWEICLOUD_SDK_AK"]
__SK = os.environ["HUAWEICLOUD_SDK_SK"]
# 如果进行了加密还需要进行解密操作
session = Session(access_key=__AK,secret_key=__SK, project_id='***', region_name='***')

session.download_data(bucket_path="/bucket_name/obs_file.txt",path="/home/user/obs_file.txt")
```

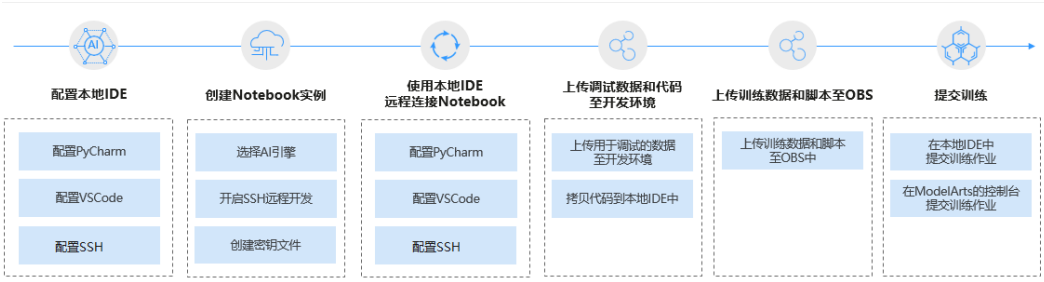

版权所有 © 华为云计算技术有限公司

6 本地 IDE

6.1 本地 IDE 操作流程

ModelArts支持通过本地IDE环境远程连接到Notebook中，开发基于PyTorch、TensorFlow和MindSpore引擎的AI模型。具体操作流程如下图所示。

图 6-1 使用本地 IDE 开发流程



1. 配置本地IDE

   在用户的PC端配置本地IDE环境。

   支持通过**PyCharm**、**VS Code**、**SSH工具**本地IDE连接云上Notebook。PyCharm和VS Code可以使用插件自动化配置，也可以手工配置。

2. **创建Notebook实例**

   在ModelArts控制台上创建一个Notebook开发环境实例，选择要使用的AI框架，并开启SSH远程开发功能。

3. 使用本地IDE远程连接到ModelArts的开发环境中。
4. **上传数据和代码至开发环境中**，进行代码调试。
   - 代码直接复制至本地IDE中即可，本地IDE中会自动同步至云上开发环境。
   - 不大于500MB数据量直接复制至本地IDE中即可。
   - **创建训练作业**大于500MB数据量请先上传到OBS中，从OBS上传到云硬盘EVS。
5. 将调试好的训练脚本和用于训练的数据集上传至OBS目录。
6. 提交训练作业。提交训练作业方式如下：

版权所有 © 华为云计算技术有限公司

- 在本地IDE中提交训练作业

  可以通过调用ModelArts提供的SDK，创建训练作业，上云训练，调用SDK创建训练作业的操作请参见**调用SDK创建训练作业**。

  - 可以基于PyCharm ToolKit直接提交训练作业，具体参考**使用PyCharm ToolKit提交训练作业**。
  - 也可以通过调用ModelArts提供的SDK，创建训练作业，上云训练，调用SDK创建训练作业的操作请参见**调用SDK创建训练作业**。

- 在ModelArts的Console控制台页面中提交训练作业，具体参考**创建训练作业**。

6.2 本地 IDE（PyCharm）

6.2.1 PyCharm Toolkit 插件连接 Notebook

6.2.1.1 PyCharm ToolKit 介绍

由于AI开发者会使用PyCharm工具开发算法或模型，为方便快速将本地代码提交到ModelArts的训练环境，ModelArts提供了一个PyCharm插件工具PyCharm ToolKit（插件下载请参见**通过Marketplace安装**），协助用户完成代码上传、提交训练作业、将训练日志获取到本地展示等，用户只需要专注于本地的代码开发即可。

使用限制

- 当前仅支持PyCharm 2019.2及以上版本，包括社区版和专业版。
- 使用PyCharm ToolKit远程连接Notebook开发环境，仅限PyCharm专业版。
- 使用PyCharm ToolKit提交训练作业，社区版和专业版都支持。
- PyCharm ToolKit工具仅支持Windows版本的PyCharm。

支持的功能

表 6-1 ToolKit（latest）功能列表

支持的功能	说明	对应操作指导
SSH远程连接	支持SSH远程连接ModelArts的Notebook开发环境。	**配置PyCharm ToolKit远程连接Notebook**
训练模型	支持将本地开发的代码，快速提交至ModelArts并自动创建训练作业，在训练作业运行期间获取训练日志并展示到本地。	• **提交训练作业** • **停止训练作业** • **查看训练日志**
OBS上传下载	上传本地文件或文件夹至OBS，从OBS下载文件或文件夹到本地。	**在PyCharm中上传数据至Notebook**

版权所有 © 华为云计算技术有限公司

6.2.1.2 下载并安装 ToolKit 工具

在使用PyCharm ToolKit之前，您需要根据如下操作指导完成在PyCharm中的安装配置。

前提条件

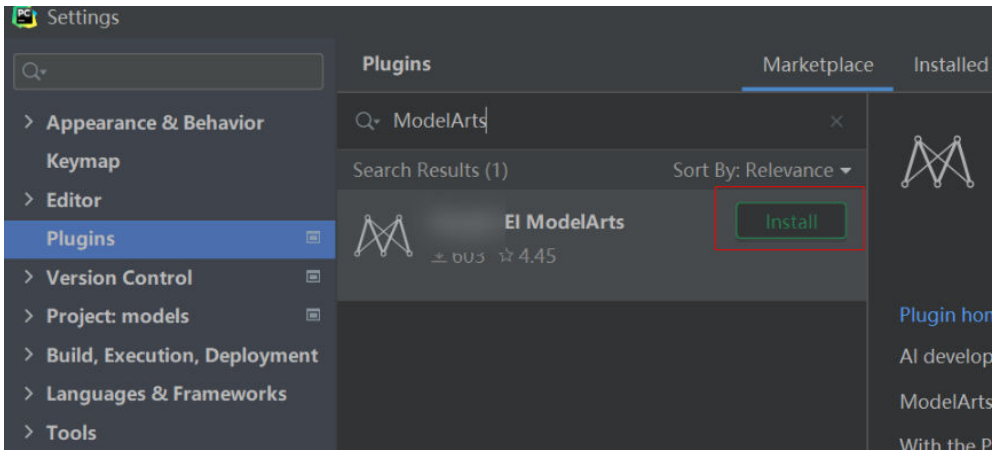
本地已安装2019.2及以上版本的PyCharm社区版或专业版。

- 使用PyCharm ToolKit远程连接Notebook开发环境，仅限PyCharm专业版。
- 使用PyCharm ToolKit提交训练作业，社区版和专业版都支持。

通过 Marketplace 安装

在PyCharm中选择“File > Settings > Plugins”，在Marketplace里搜索“ModelArts”，单击“Install”即可完成安装。

图 6-2 通过 Marketplace 安装



说明

- 通过该方式安装的PyCharm ToolKit为latest版本。
- 如果Marketplace里搜索不到ModelArts，可能是用户网络限制原因，请确保可以正常访问外网。

6.2.1.3 PyCharm ToolKit 连接 Notebook

ModelArts提供了一个PyCharm插件工具PyCharm ToolKit，协助用户完成SSH远程连接Notebook、代码上传、提交训练作业、将训练日志获取到本地展示等，用户只需要专注于本地的代码开发即可。

前提条件

本地已安装2019.2及以上版本的PyCharm专业版。SSH远程开发功能只限PyCharm专业版。单击**PyCharm工具下载地址**下载并完成安装。

说明

请下载**2023.2**或之前版本的PyCharm专业版工具。PyCharm toolkit未适配2023.2之后版本的PyCharm专业版工具。

版权所有 © 华为云计算技术有限公司

Step1 创建 Notebook 实例

创建一个Notebook实例，并开启远程SSH开发，配置远程访问IP白名单。该实例状态必须处于“运行中”，具体参见**创建Notebook实例**章节。

Step2 下载并安装 PyCharm ToolKit

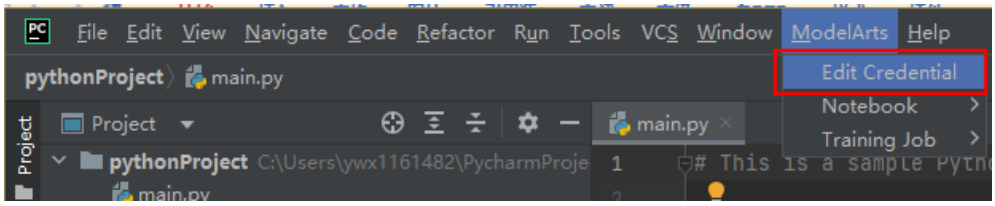
在PyCharm中选择“File > Settings > Plugins”，在Marketplace里搜索“ModelArts”，单击“Install”即可完成安装。具体下载和安装过程请参见**下载并安装ToolKit工具**。

Step3 登录插件

使用访问密钥完成登录认证操作如下：

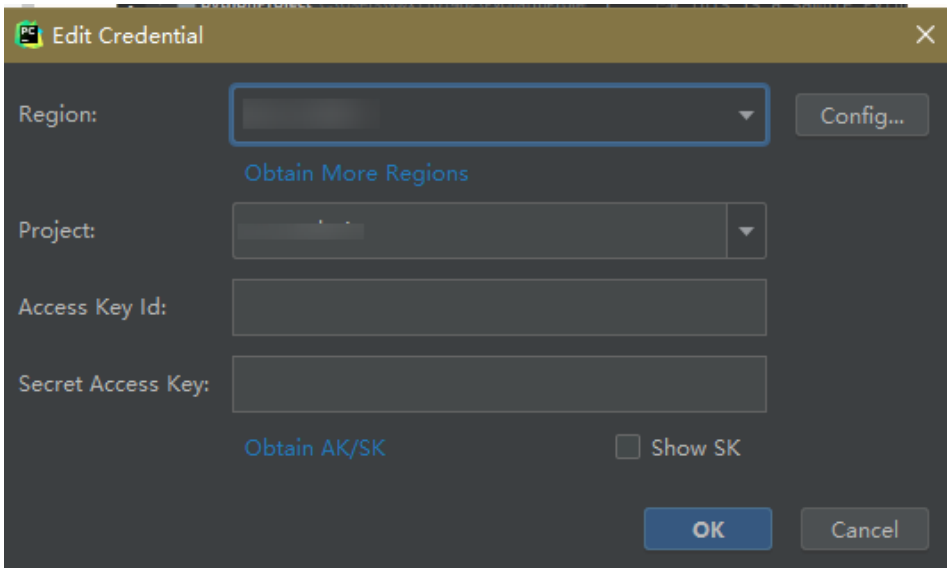
1. 打开已安装ToolKit工具的PyCharm，在菜单栏中选择“ModelArts > Edit Credential”。

图 6-3 Edit Credential



2. 在弹出的对话框中，选择您使用的ModelArts所在区域、填写AK、SK（获取方式**参考链接**），然后单击“OK”完成登录。
   - “Region”：从下拉框中选择区域。必须与ModelArts管理控制台在同一区域。
   - “Project”：Region选择后，Project自动填充为Region对应的项目。
   - “Access Key ID”：填写访问密钥的AK。
   - “Secret Access Key”：填写访问密钥的SK。

图 6-4 填写区域和访问密钥



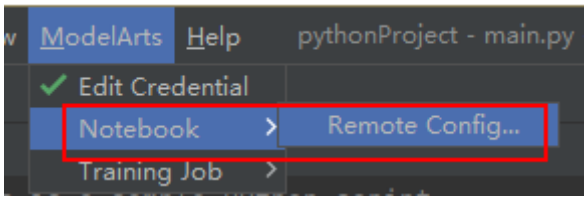
版权所有 © 华为云计算技术有限公司

3. 查看认证结果。
   在Event Log区域中，当提示如下类似信息时，表示访问密钥添加成功。
   16:01Validate Credential Success: The HUAWEI CLOUDcredential is valid.

Step4 插件自动化配置

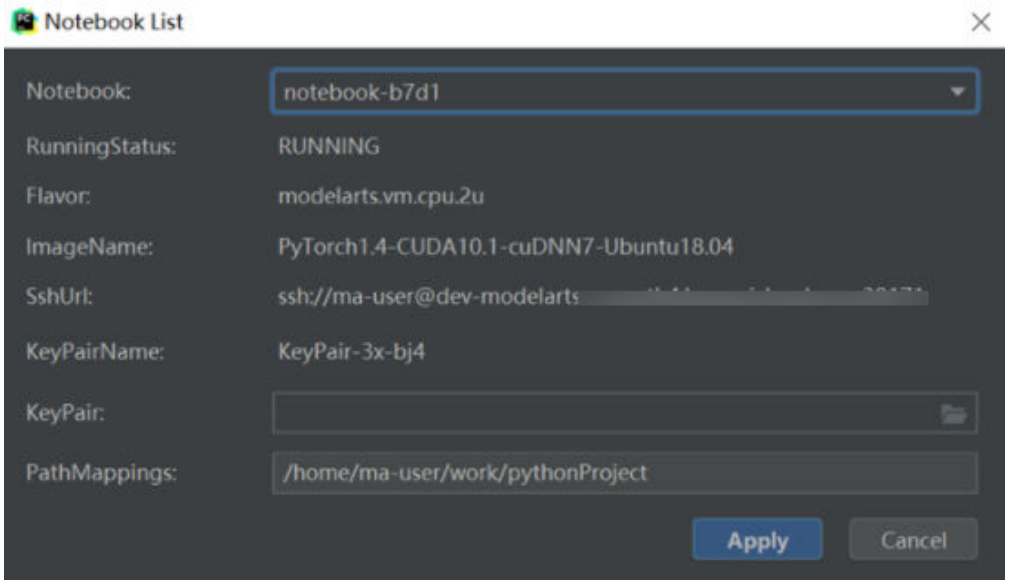
1. 在本地的PyCharm开发环境中，单击“ModelArts > Notebook > Remote Config...”，配置插件。

   图 6-5 配置插件

2. 此时，会出现该账号已创建的所有包含SSH功能的Notebook列表，下拉进行选择对应Notebook。

   图 6-6 Notebook 列表

   - KeyPair: 需要选择保存在本地的Notebook对应的keypair认证。即创建Notebook时创建的密钥对文件，创建时会直接保存到浏览器默认的下载文件夹中。
   - PathMappings: 该参数为本地IDE项目和Notebook对应的同步目录，默认为/home/ma-user/work/project名称，可根据自己实际情况更改。

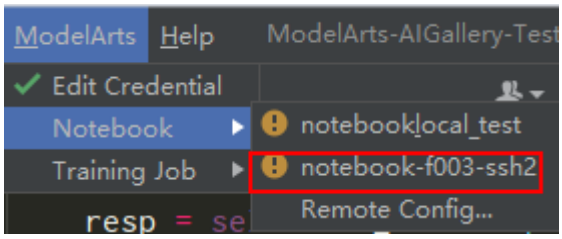
3. 单击“Apply”，配置完成后，重启IDE生效。
   重启后初次进行update python interpreter需要耗费20分钟左右。

Step5 使用插件连接云上 Notebook

与Notebook断开连接的状态下，单击Notebook名称，根据提示启动本地IDE与Notebook的连接(默认启动时间4小时)。

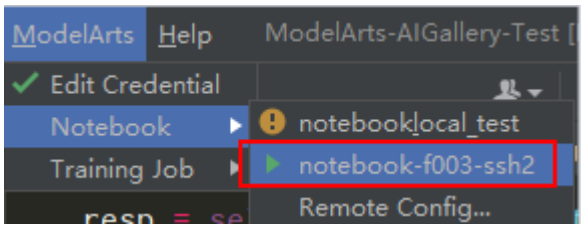
版权所有 © 华为云计算技术有限公司

图 6-7 启动连接 Notebook



连接状态下，单击Notebook名称，根据提示断开本地IDE与云上Notebook的连接。

图 6-8 停止连接 Notebook



Step6 同步上传本地文件至 Notebook

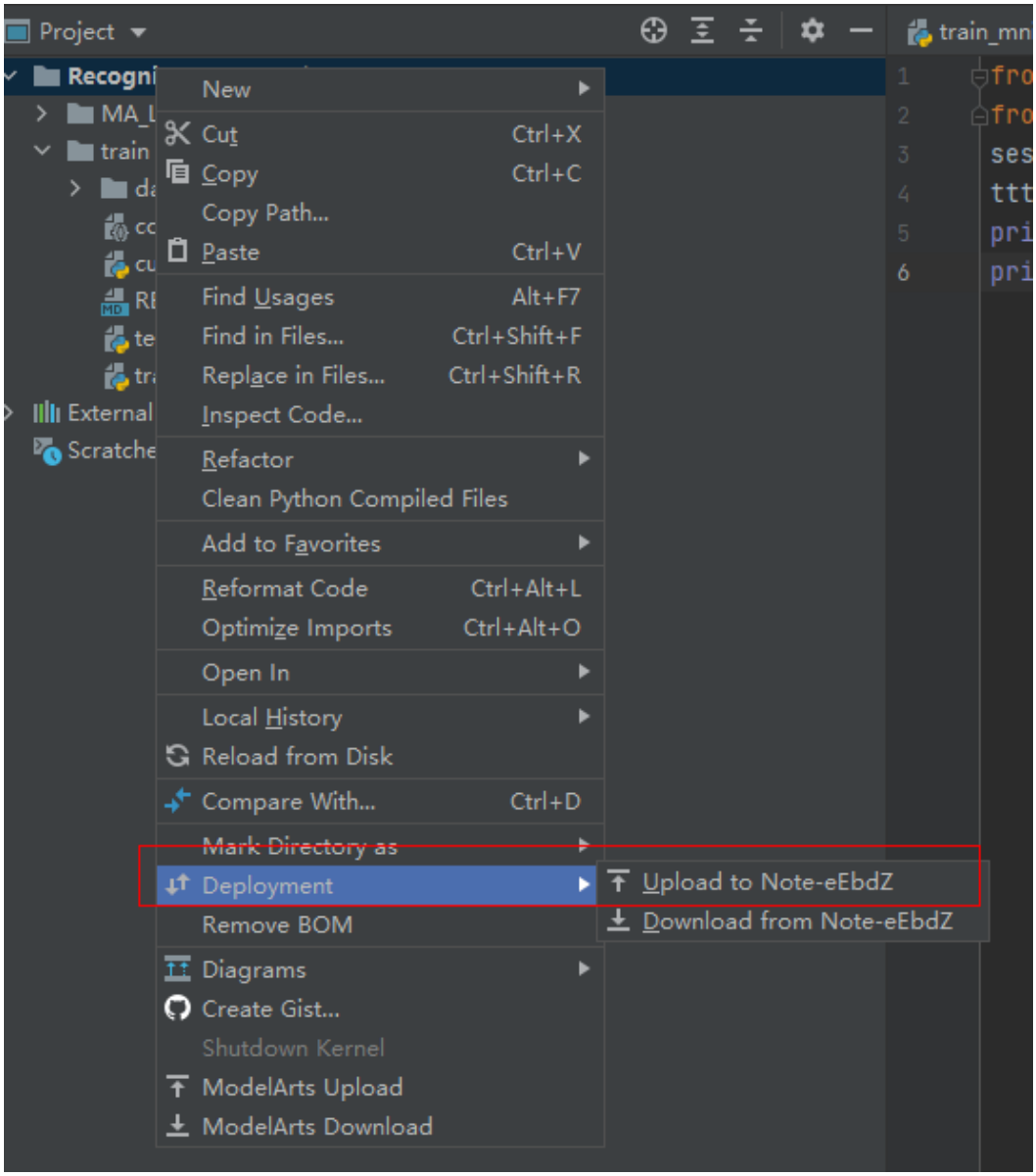
本地文件中的代码直接复制至本地IDE中即可，本地IDE中会自动同步至云上开发环境。

初始化同步：

在本地IDE的Project目录下，单击右键，选择“Deployment”，单击“Upload to xxx”（Notebook名称），将本地工程文件上传至指定的Notebook。

版权所有 © 华为云计算技术有限公司

图 6-9 同步本地文件至 Notebook



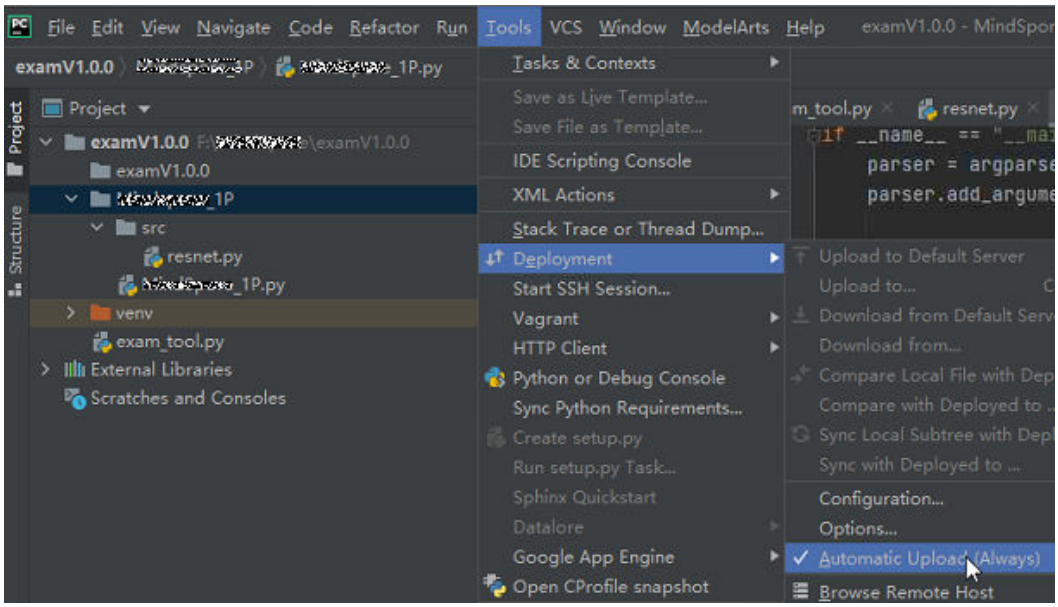
后续同步:

只需修改代码后保存（ctrl+s），即可进行自动同步。

插件安装完成后在本地IDE中开启了“Automatic Upload”，本地目录中的文件会自动上传至云端开发环境Notebook。如果未开启，请参考下图开启自动上传。

版权所有 © 华为云计算技术有限公司

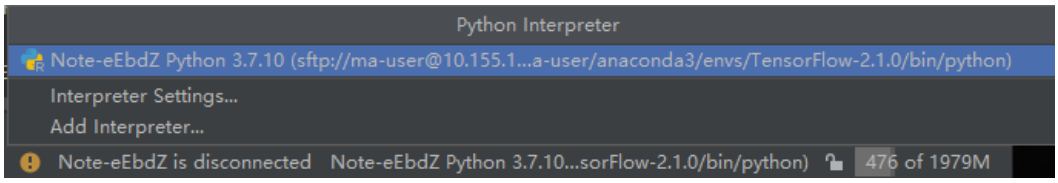
图 6-10 开启自动上传



Step7 远程调试

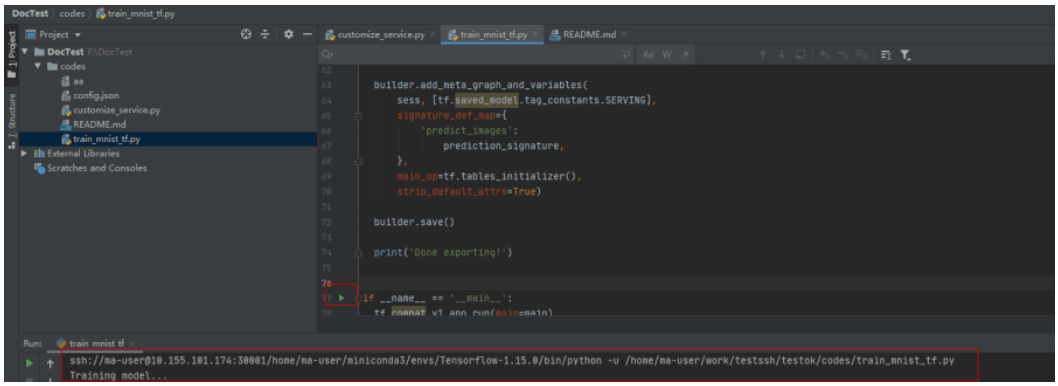
单击本地IDE右下角interpreter，选择Notebook的python解释器。

图 6-11 选择 Python 解释器



像本地运行代码一样，直接单击运行按钮运行代码即可，此时虽然是在本地IDE点的运行按钮，实际上运行的是云端Notebook里的代码，日志可以回显在本地的日志窗口。

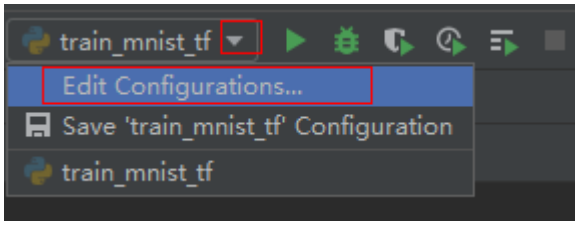
图 6-12 查看运行日志



也可以单击本地IDE右上角的Run/Debug Configuration按钮来设置运行参数。

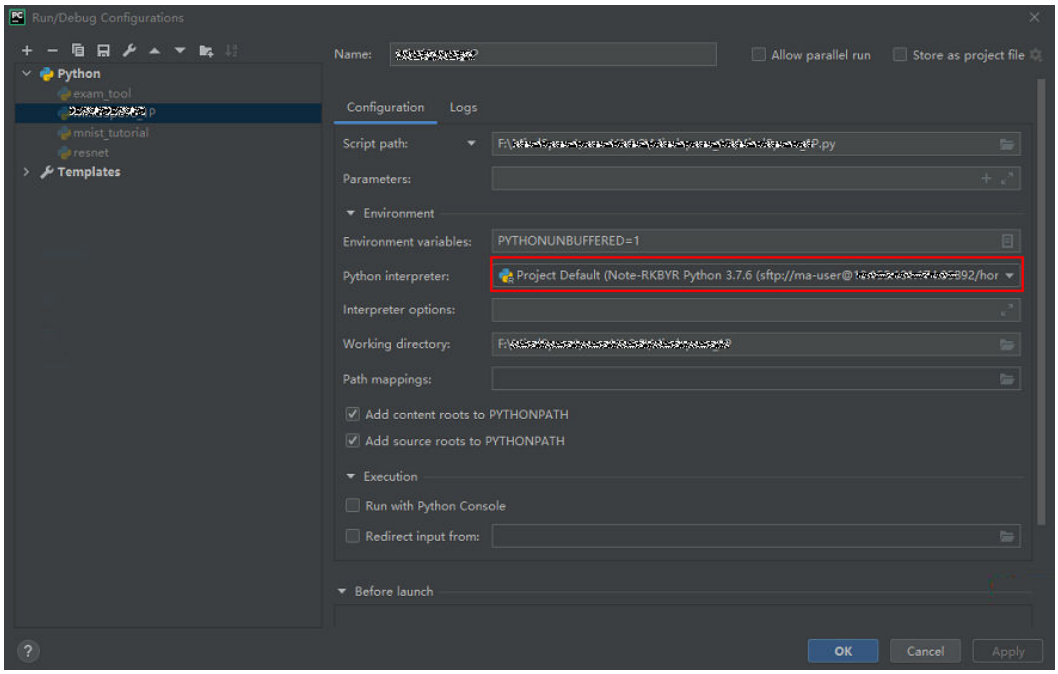
版权所有 © 华为云计算技术有限公司

图 6-13 设置运行参数（1）



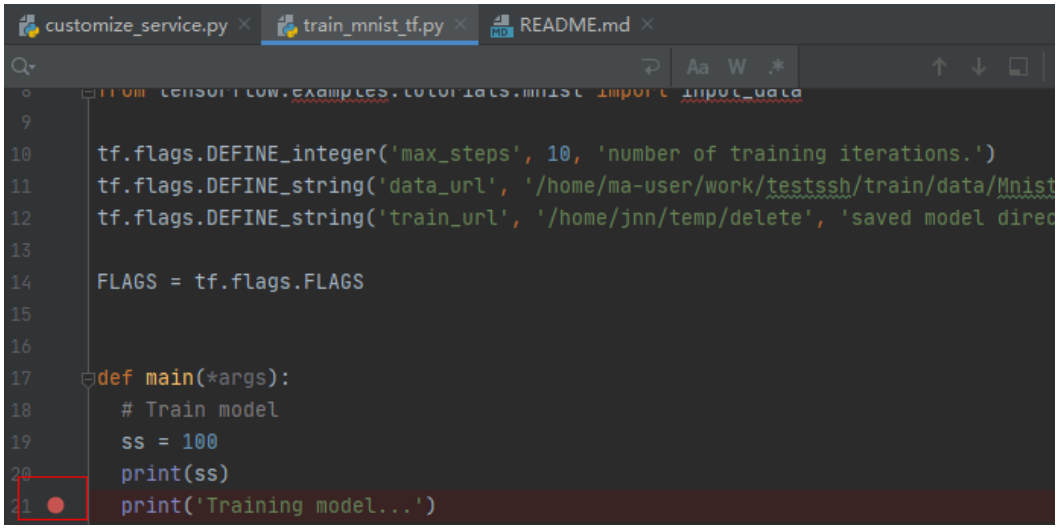
选择远程连接到云上开发环境实例对应的Python解释器。

图 6-14 设置运行参数（2）



当需要调试代码时，可以直接打断点，然后使用debug方式运行程序。

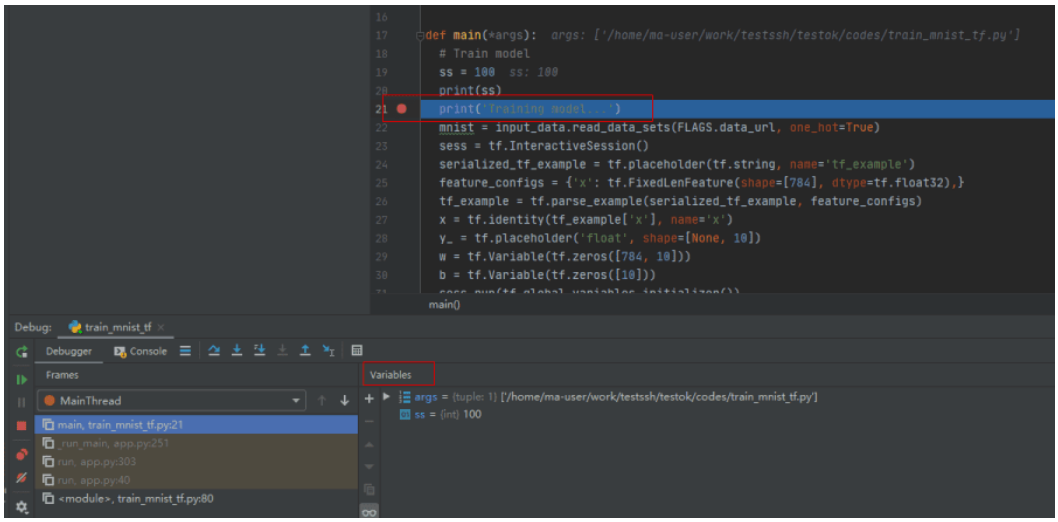
图 6-15 使用 debug 方式运行程序



版权所有 © 华为云计算技术有限公司

此时可以进入debug模式，代码运行暂停在该行，且可以查看变量的值。

图 6-16 Debug 模式下查看变量值



6.2.2 PyCharm 手动连接 Notebook

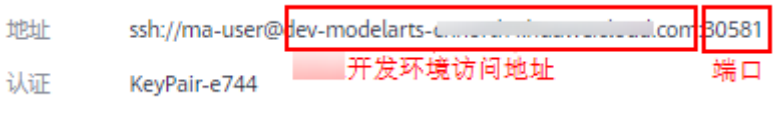
本地IDE环境支持Pycharm和VS Code。通过简单配置，即可用本地IDE远程连接到ModelArts的Notebook开发环境中，调试和运行代码。

本章节介绍基于PyCharm环境访问Notebook的方式。

前提条件

- 本地已安装2019.2及以上版本的PyCharm专业版。SSH远程调试功能只限PyCharm专业版。
- 创建一个Notebook实例，并开启远程SSH开发。该实例状态必须处于“运行中”，具体参见**创建Notebook实例**章节。
- 在Notebook实例详情页面获取开发环境IP地址（例如：dev-modelarts-cnnorth4.huaweicloud.com）和端口号。

  图 6-17 Notebook 实例详情页面

- 准备好密钥对。

  密钥对在用户第一次创建时，自动下载，之后使用相同的密钥时不会再有下载界面（用户一定要保存好），或者每次都使用新的密钥对。

Step1 配置 SSH

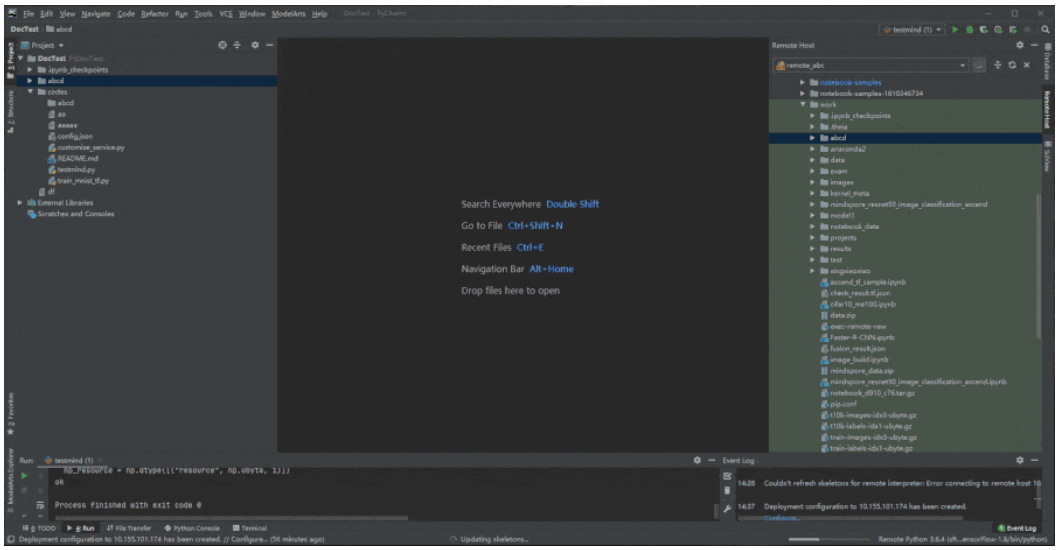
1. 在本地的PyCharm开发环境中，单击File -> Settings -> Tools -> SSH Configurations，单击+号，增加一个SSH连接配置。
   - Host: 云上开发环境的IP地址，即在开发环境实例页面远程访问模块获取的IP地址。例如：dev-modelarts-cnnorth4.huaweicloud.com

版权所有 © 华为云计算技术有限公司

- Port: 云上开发环境的端口，即在开发环境实例页面远程访问模块获取的端口号。
- User name: 固定为ma-user
- Authentication type: Key pair方式
- Private key file: 存放在本地的云上开发环境私钥文件，即在创建开发环境实例时创建并保存的密钥对文件。

2. 单击  将连接重命名，可以自定义一个便于识别的名字，单击OK。
3. 配置完成后，单击Test Connection测试连通性。
4. 选择Yes，显示Successfully connected表示网络可以连通，单击OK。
5. 在最下方再单击OK保存配置。

图 6-18 配置 SSH

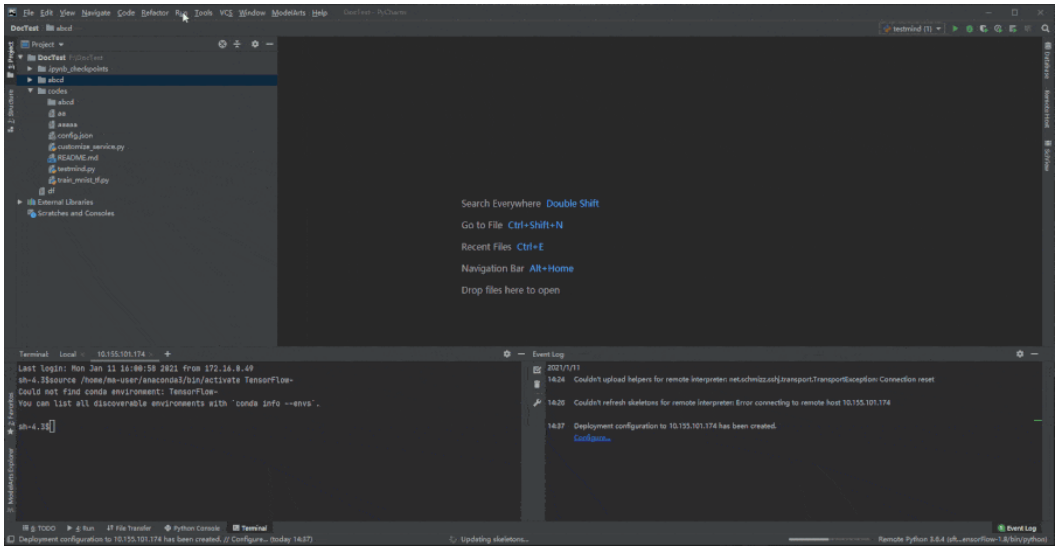


Step2 获取开发环境预置虚拟环境路径

1. 单击“Tools > Start SSH Session”，则可连接到云端开发环境内。
2. 执行如下命令可在/home/ma-user/下面的README文件查看当前环境内置的Python虚拟环境。
   ```
   cat /home/ma-user/README
   ```
3. 执行source命令可以切换到具体的Python环境中。
4. 执行which python查看python路径并复制出来，以备后续配置云上Python Interpreter使用。

版权所有 © 华为云计算技术有限公司

图 6-19 获取开发环境预置虚拟环境路径



Step3 配置云上 Python Interpreter

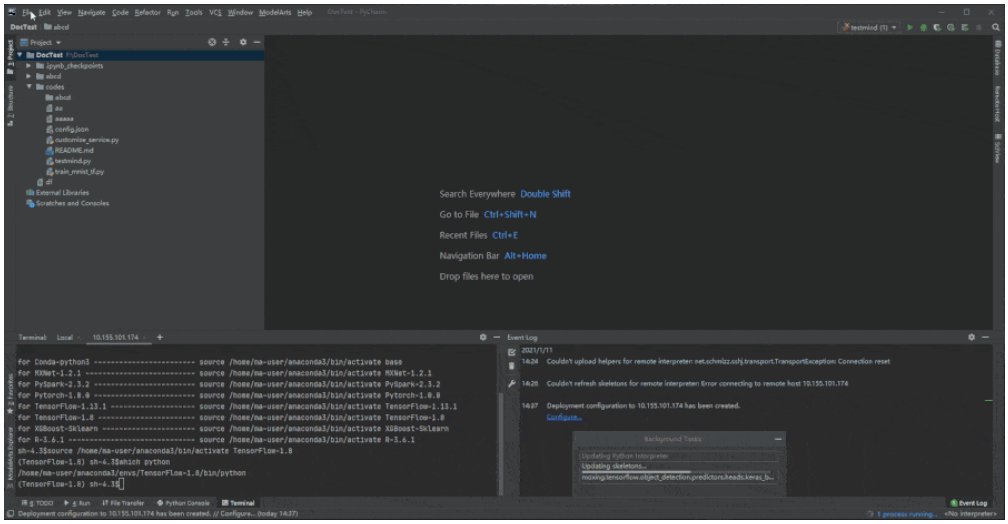
1. 单击“File > Settings > Project：PythonProject > Python Interpreter”，单击设置图标  ，再单击“Add”，添加一个新的interpreter。
2. 选择“Existing server configuration”，在下拉菜单中选择上一步配置好的SSH configuration，单击“Next”。
3. 配置Python Interpreter
   - Interpreter：填写第一步复制的python路径，例如：/home/ma-user/anaconda3/envs/Pytorch-1.0.0/bin/python
     如果路径为~/anaconda3/envs/Pytorch-1.0.0/bin/python 把~替换为/home/ma-user即可。
   - Sync folders: 需要配置本地的工程目录文件同步到云上开发环境中的某个目录，推荐配置为/home/ma-user下的某个目录中（其他目录可能没有访问权限），例如/home/ma-user/work/projects。
4. 单击右侧文件夹图标，勾选上“Automatically upload”选项，以便于本地修改的文件自动上传到容器环境中。
5. 单击“Finish”，结束配置。

   可以看到本地的工程文件已经自动往云上环境上传了。后续本地的文件每修改一次，都会自动的同步到云上的环境中。

   右下角可以看到当前的Interpreter为Remote Interpreter。

版权所有 © 华为云计算技术有限公司

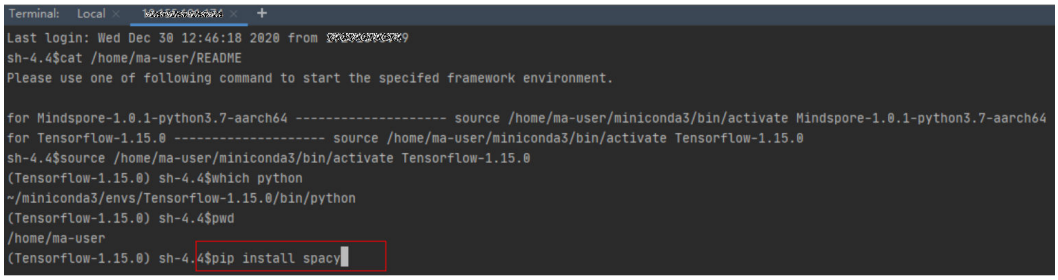
图 6-20 配置云上 Python Interpreter



Step4 云上环境依赖库安装

在进入开发环境后，可以使用不同的虚拟环境，例如TensorFlow、PyTorch等，但是实际开发中，通常还需要安装其他依赖包，此时可以通过Terminal连接到环境里操作。

单击工具栏“Tools >Start SSH session”，选择SSH Configuration中配置的开发环境。可以执行pip install安装所需要的包。



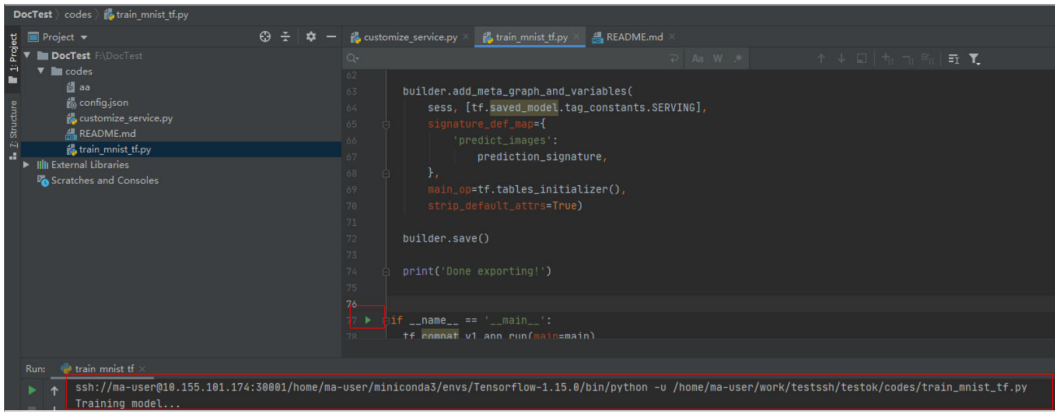
Step5 在开发环境中调试代码

由于已经连接至云端开发环境，此时可以方便的在本地PyCharm中编码、调测并运行。运行实际环境为云上开发环境，资源为云上昇腾AI处理器资源。可以做到本地编写修改代码，直接在云上环境运行。

像本地运行代码一样，直接单击运行按钮运行代码即可，此时虽然是在本地IDE单击的运行按钮，实际上运行的是云端开发环境里的代码，日志可以回显在本地的日志窗口。

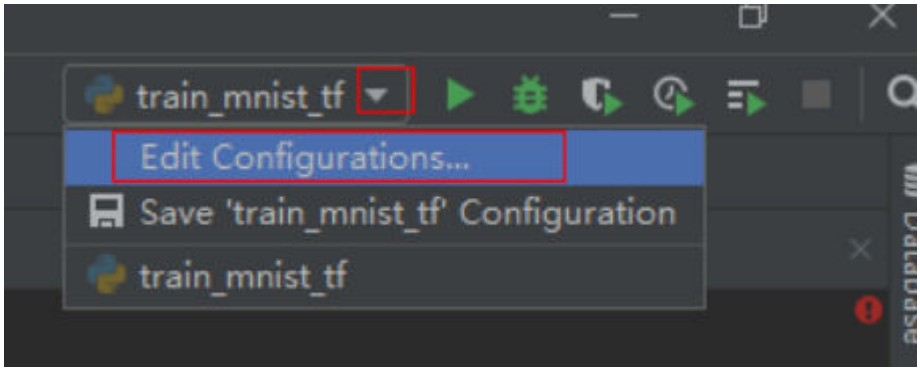
版权所有 © 华为云计算技术有限公司

图 6-21 调试代码



也可以单击右上角的Run/Debug Configuration来设置运行的参数。

图 6-22 设置运行参数



当需要调试代码时，可以直接打断点，然后使用debug方式运行程序。

版权所有 © 华为云计算技术有限公司

图 6-23 代码打断点

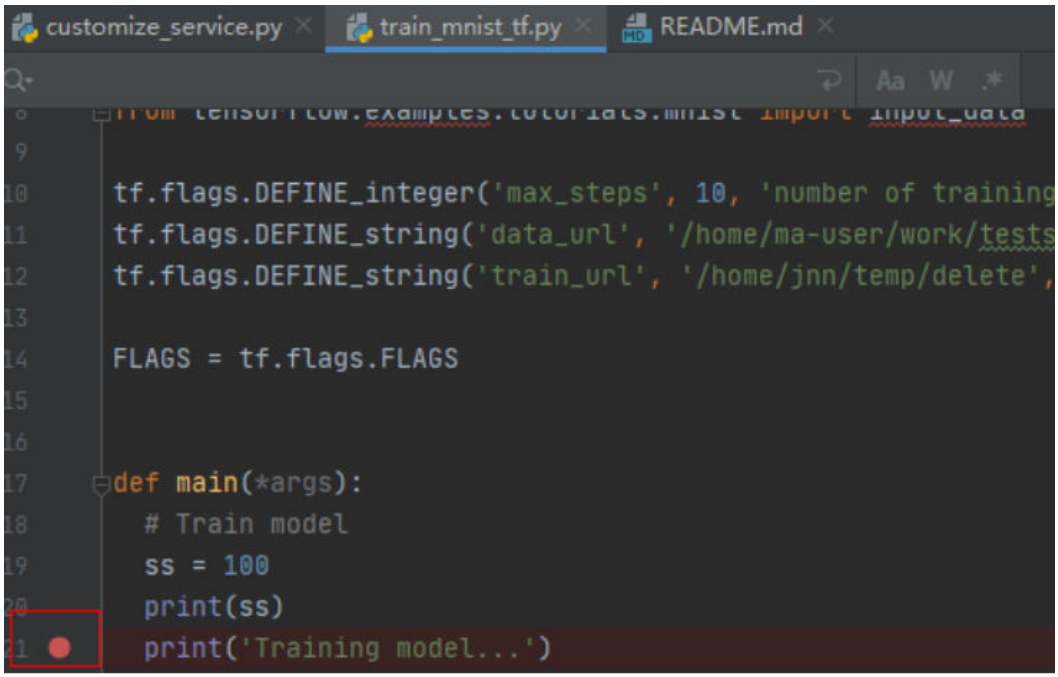
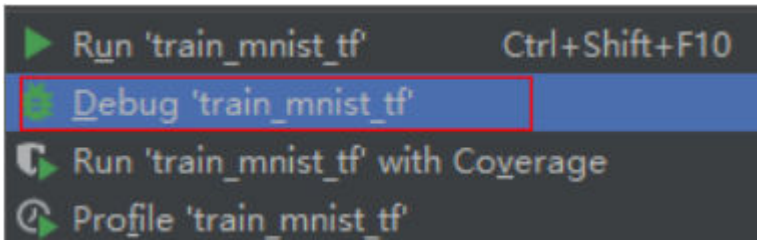
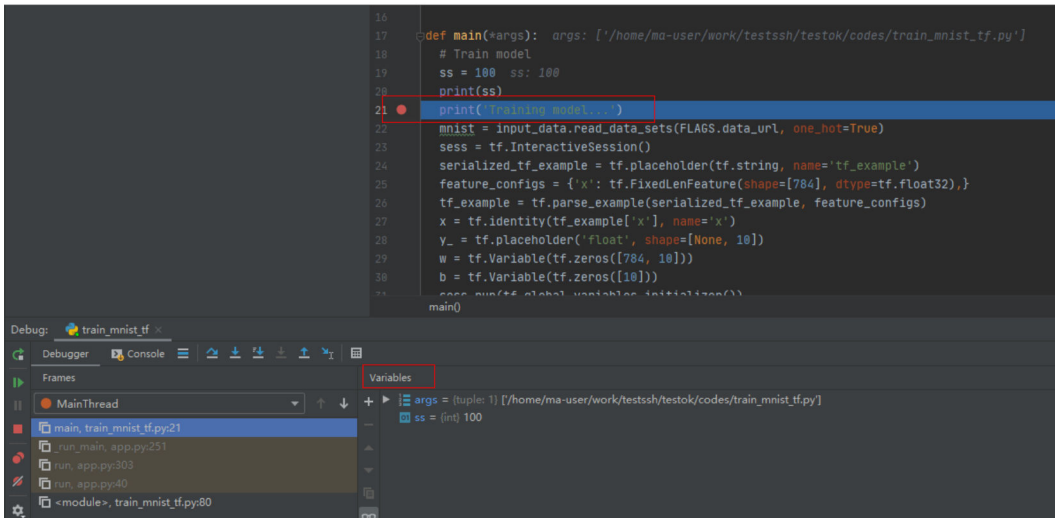


图 6-24 Debug 方式调试



此时可以进入debug模式，代码运行暂停在该行，且可以查看变量的值。

图 6-25 Debug 模式



版权所有 © 华为云计算技术有限公司

使用debug方式调试代码的前提是本地的代码和云端的代码是完全一致的，如果不一致可能会导致在本地打断点的行和实际运行时该行的代码并不一样，会出现意想不到的错误。

因此在配置云上Python Interpreter时，推荐选择Automatically upload选项，以保证本地的文件修改能自动上传到云端。如果没有选择自动上传，则本地代码修改完后，也可以参考**Step6 同步上传本地文件至Notebook**手动上传目录或代码。

6.2.3 PyCharm Toolkit 提交训练作业

6.2.3.1 提交训练作业

使用PyCharm ToolKit（latest版本）工具，可以快速将本地开发的训练代码，提交至ModelArts侧进行训练。

前提条件

- 在本地PyCharm中已有训练代码工程。
- 已在OBS中创建桶和文件夹，用于存放数据集和训练输出模型。 例如：创建命名为“test-modelarts2”的桶，创建文件夹“dataset-mnist”和“mnist-output”。训练作业使用的数据已上传至OBS，且OBS与ModelArts在同一区域。
- 已配置credential，详细请参考**使用访问密钥登录**。

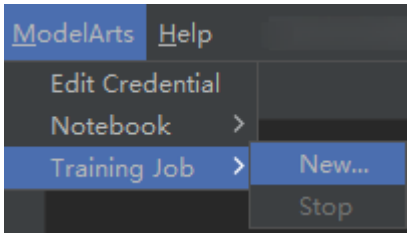
准备数据

- 训练代码工程案例请参考：在gitee的**ModelArts-Lab**工程中，单击“克隆/下载”，然后单击“下载ZIP”，下载工程。解压缩“ModelArts-Lab-master.zip”文件，然后在“\ModelArts-Lab-master\official_examples\Using_MXNet_to_Create_a_MNIST_Dataset_Recognition_Application\codes”目录中获取到训练代码文件“train_mnist.py”。
- 数据集案例请参考：从**MNIST官网**下载“Mnist-Data-Set”数据集至本地，然后解压zip包，将“Mnist-Data-Set”文件夹下的所有文件上传至“test-modelarts2/dataset-mnist”OBS路径下。

配置训练作业参数

1. 在PyCharm中，打开训练代码工程和训练启动文件，然后在菜单栏中选择“ModelArts > Training Job > New...”。

   图 6-26 选择作业配置

2. 在弹出的对话框中，设置训练作业相关参数，详细参数说明请参见**表6-2**。

版权所有 © 华为云计算技术有限公司

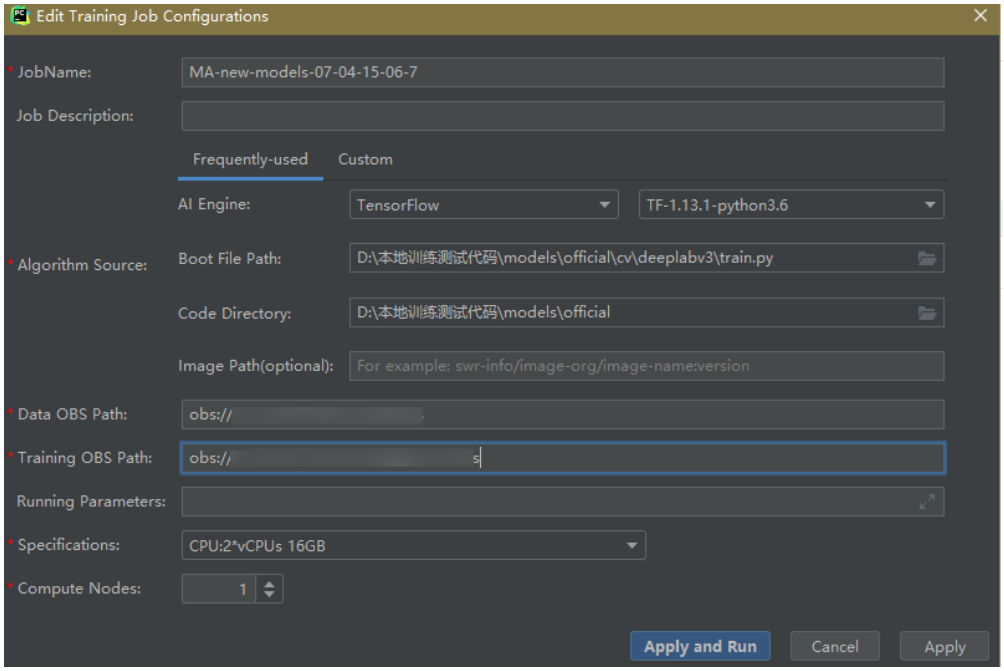
表 6-2 训练作业配置参数说明

参数	说明
Job Name	训练作业的名称。 系统会自动生成一个名称，您可以根据业务需求重新命名，命名规则如下： • 支持1~64位字符。 • 并包含大小写字母、数字、中划线（-）或下划线（_）。
Job Description	训练作业的简要描述。
Algorithm Source	训练算法来源，分为“常用框架”和“自定义镜像”两种，二者选一项即可。 常用框架指使用ModelArts训练管理中支持的常用AI引擎，当前支持的引擎列表请参见**训练管理支持的常用框架**。 如果您使用的AI引擎为支持列表之外的，建议使用自定义镜像的方式创建训练作业。
AI Engine	选择代码使用的AI引擎及其版本。支持的AI引擎与ModelArts管理控制台里**训练管理支持的常用框架**一致。
Boot File Path	训练启动文件，所选启动文件必须是当前PyCharm训练工程中的文件。当“Algorithm source”选“Frequently-used”时，显示此参数。
Code Directory	训练代码目录，系统会自动填写为训练启动文件所在的目录，用户可根据需要修改，所选目录必须是当前工程中的目录且包含启动文件。 当算法来源为自定义镜像，训练代码已预置在镜像中时，该参数可以为空。
Image Path(optional)	SWR镜像的URL地址，例如swr.cn-north-4.myhuaweicloud.com/image-org/image-name:version。关于自定义镜像的说明，请参见**自定义镜像介绍**。
Boot Command	启动本次训练作业的运行命令。例如“bash /home/work/run_train.sh python {python启动文件及参数}”。当“Algorithm source”选“Custom”时，显示此参数。 当用户输入的命令中不包含“--data_url”和“--train_url”参数时，工具在提交训练作业时会在命令后面自动添加这两个参数，分别对应存储训练数据的OBS路径和存放训练输出的OBS路径。
Data OBS Path	设置为存储训练数据的OBS路径，例如“/test-modelarts2/mnist/dataset-mnist/”，其中“test-modelarts2”为桶名称。
Training OBS Path	设置OBS路径，该路径下会自动创建用于存放训练输出模型和训练日志的目录。

版权所有 © 华为云计算技术有限公司

参数	说明
Running Parameters	运行参数。如果您的代码需要添加一些运行参数，可以在此处添加，多个运行参数使用英文分号隔开， 例如"key1=value1;key2=value2"。此参数也可以不设置，即保持为空。
Specifications	训练使用资源类型。目前支持公共资源池和专属资源池两种类型。 专属资源池规格以“Dedicated Resource Pool”标识。只有购买了专属资源池的用户才会显示专属资源池规格。
Compute Nodes	计算资源节点个数。数量设置为1时，表示单机运行；数量设置大于1时，表示后台的计算模式为分布式。
Available/Total Nodes	当“Specifications”选择专属资源池规格时，显示专属资源池的可用节点数和总节点数，用户选择“Compute Nodes”的个数不要超过可用节点数。

图 6-27 配置训练作业参数（公共资源池）



版权所有 © 华为云计算技术有限公司

图 6-28 配置训练作业参数（专属资源池）

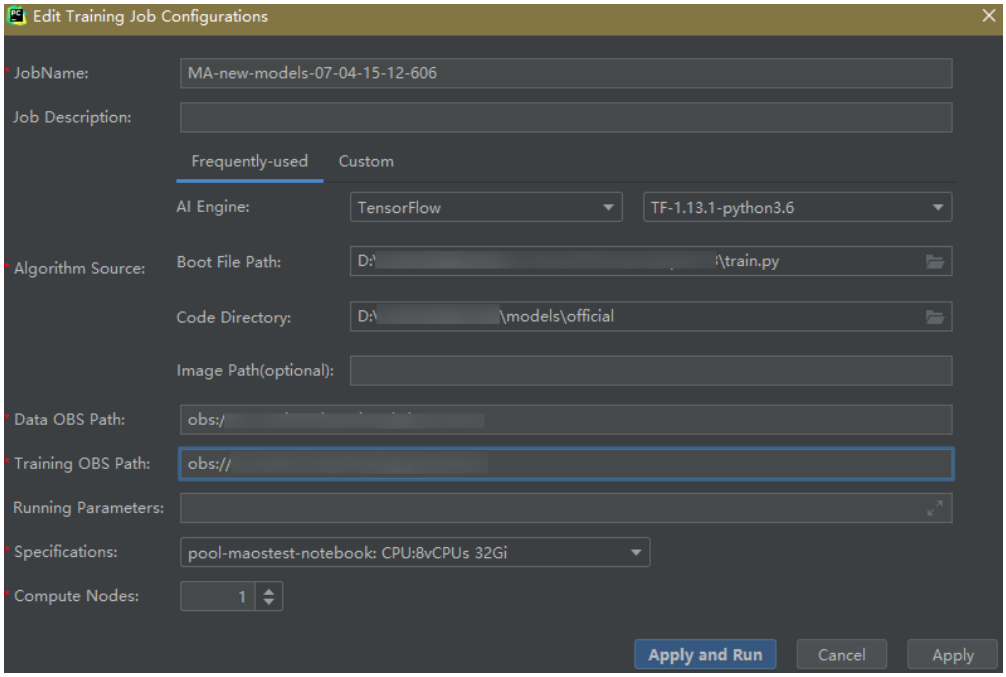
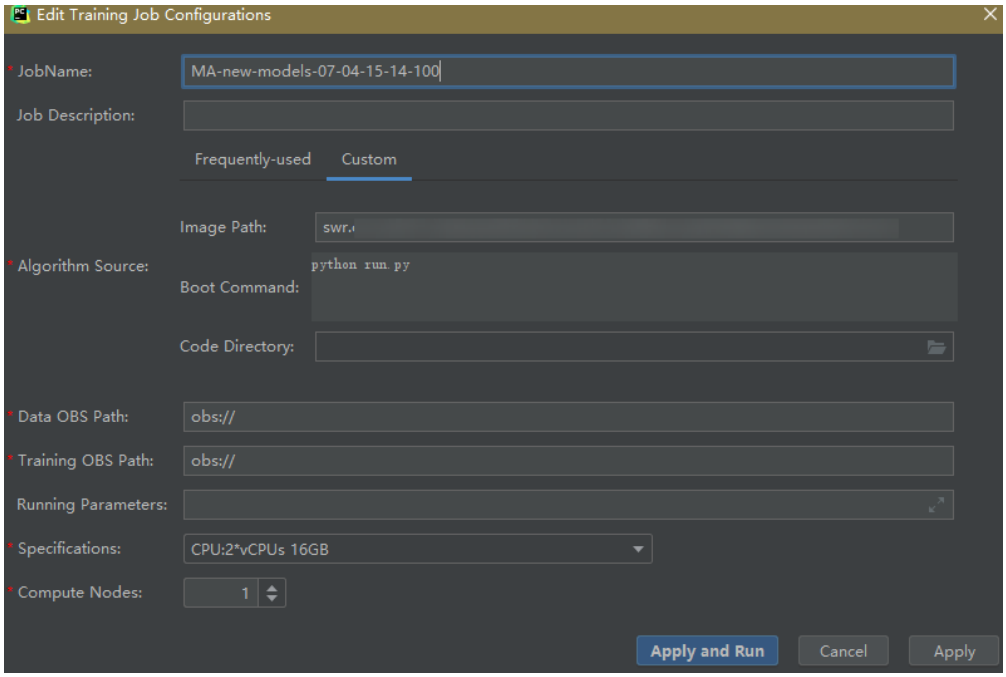


图 6-29 配置训练作业参数（自定义镜像）



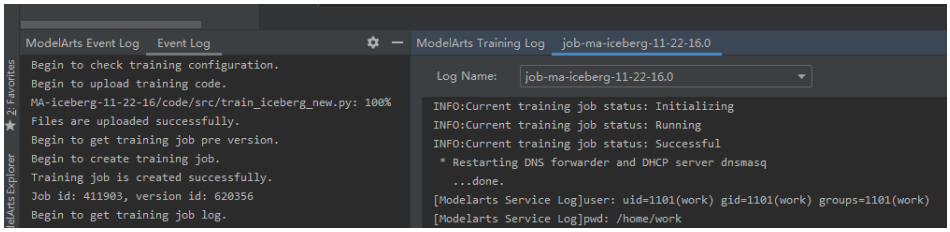
3. 参数填写完成后，单击“Apply and Run”，即自动上传本地代码至云端并启动训练，在工具下方的Training Log区域，会实时展示训练作业运行情况。当训练日志中出现“Current training job status: Successful”类似信息时，表示训练作业运行成功。

版权所有 © 华为云计算技术有限公司

说明

- 在单击“Apply and Run”按钮后，系统将自动开始执行训练作业。如果您想停止此作业，可以选择菜单栏中的“ModelArts > Training Job >Stop”停止此作业。
- 如果单击“Apply”，不会直接启动运行，只是保存训练作业的设置，如果需要启动作业，可以单击“Apply and Run”。

图 6-30 训练日志展示样例



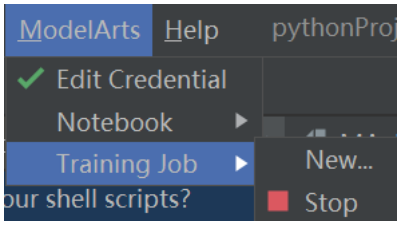
6.2.3.2 停止训练作业

当训练作业在运行过程中时，您可以执行停止作业的操作。

停止作业

当训练作业在运行过程中时，您可以在PyCharm菜单栏中，选择“ModelArts >Training Job > Stop ”停止此作业。

图 6-31 停止作业



6.2.3.3 查看训练日志

本章节介绍如何查看训练作业产生的日志。

在 OBS 中查看

提交训练作业时，系统将自动在您配置的OBS Path中，使用作业名称创建一个新的文件夹，用于存储训练输出的模型、日志和代码。

例如“train-job-01”作业，提交作业时会在“test-modelarts2”桶下创建一个命名为“train-job-01”的文件夹，且此文件夹下分别新建了三个文件夹“output”、“log”、“code”，分别用于存储输出模型、日志和训练代码。“output”文件夹还会根据您的训练作业版本再创建子文件夹，结构示例如下。

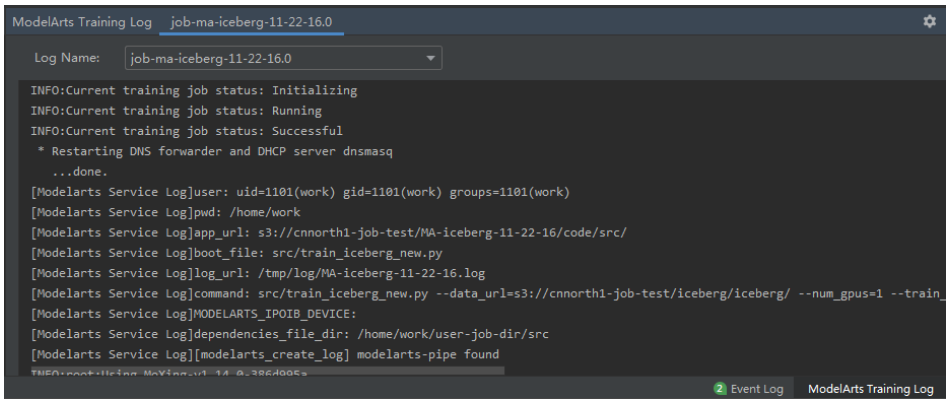
```
test-modelarts2
  |---train-job-01
      |---output
      |---log
      |---code
```


版权所有 © 华为云计算技术有限公司

在 ToolKit 工具中查看

在PyCharm工具中，单击页面右下角的ModelArts Training Log，展示训练日志。

图 6-32 查看训练日志



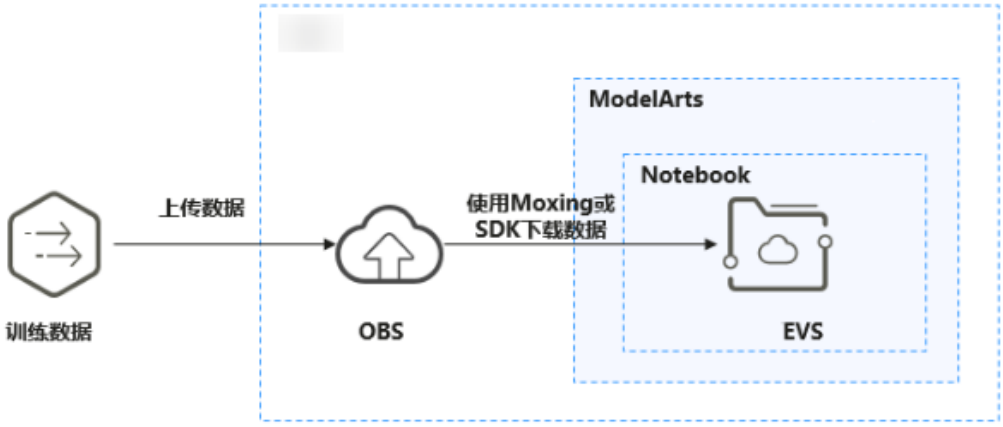
6.2.4 在 PyCharm 中上传数据至 Notebook

不大于500MB数据量，直接复制至本地IDE中即可。

大于500MB数据量，请先上传到OBS中，再从OBS上传到云上开发环境。

1. 上传数据至OBS，具体操作请参见**上传文件至OBS桶**。
2. 将OBS中的数据传至Notebook中，通过在本地IDE的Terminal中使用ModelArts提供的Moxing库的文件操作API（mox.file.copy_parallel）完成。

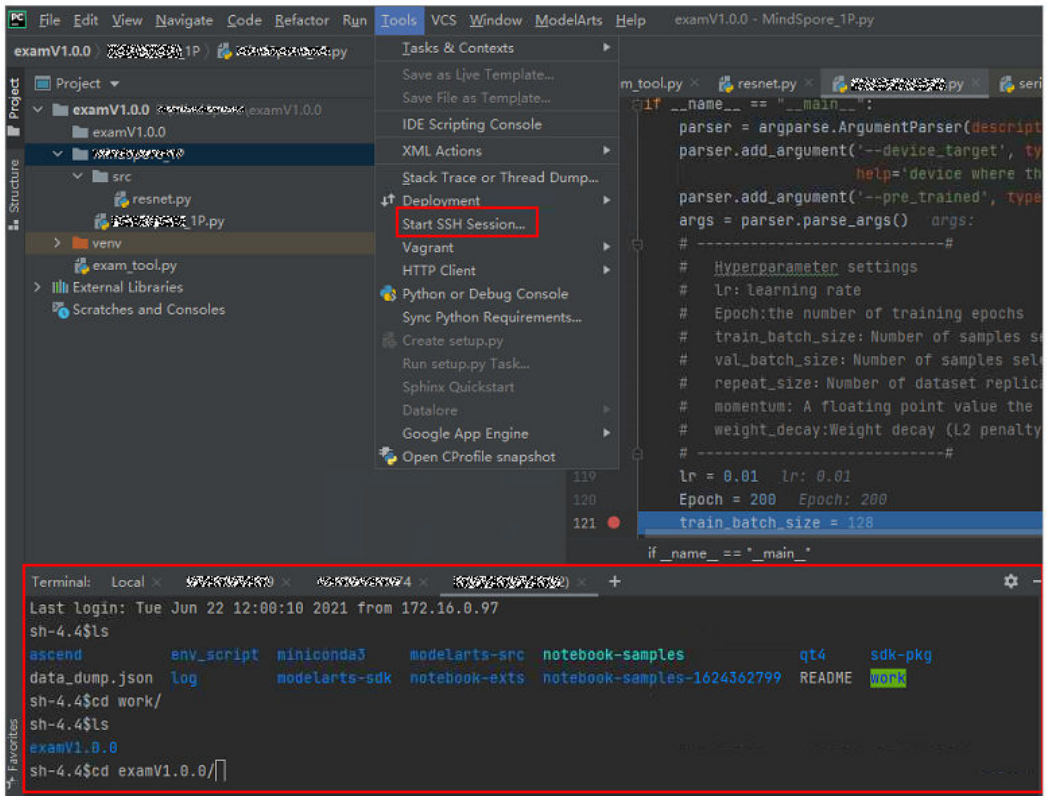
图 6-33 数据通过 OBS 中转上传到 Notebook



下图以PyCharm环境中开启Terminal为例，VS Code中操作类似。

版权所有 © 华为云计算技术有限公司

图 6-34 PyCharm 环境开启 Terminal



在本地IDE的Terminal中使用Moxing下载OBS文件到开发环境的操作示例如下：

```
#手动source进入开发环境
cat /home/ma-user/README
#然后选择要source的环境
source /home/ma-user/miniconda3/bin/activate MindSpore-python3.7-aarch64
#输入python并回车，进入python环境
python
#使用moxing
import moxing as mox
#下载一个OBS文件夹，从OBS下载至EVS（OBS -> EVS）
mox.file.copy_parallel('obs://bucket_name/sub_dir_0', '/tmp/sub_dir_0')
```

6.3 本地 IDE（VS Code）

6.3.1 VS Code 连接 Notebook 方式介绍

当用户创建完成支持SSH的Notebook实例后，使用VS Code的开发者可以通过以下三种方式连接到开发环境中：

- **VS Code一键连接Notebook**（推荐）

  该方式是指在开发环境Console控制台上提供VS Code按钮，通过该入口自动打开VS Code并连接实例。

- **VS Code ToolKit连接Notebook**（推荐）

  该方式是指用户在VS Code上使用ModelArts VS Code Toolkit插件提供的登录和连接按钮，连接云上实例。

版权所有 © 华为云计算技术有限公司

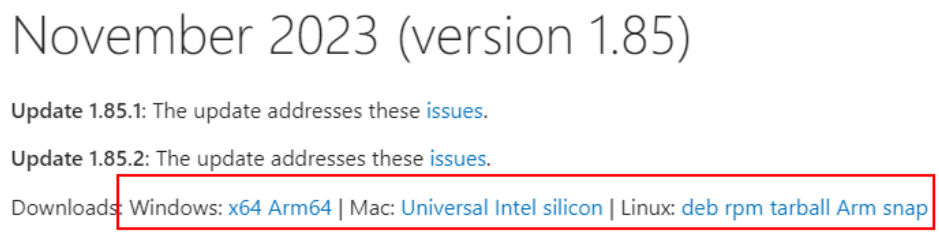
- **VS Code手动连接Notebook**
  该方式是指用户使用VS Code Remote SSH插件手工配置连接信息，连接云上实例。

6.3.2 安装 VS Code 软件

VS Code下载方式：

- 下载地址: **https://code.visualstudio.com/updates/v1_85**

图 6-35 VS Code 的下载位置

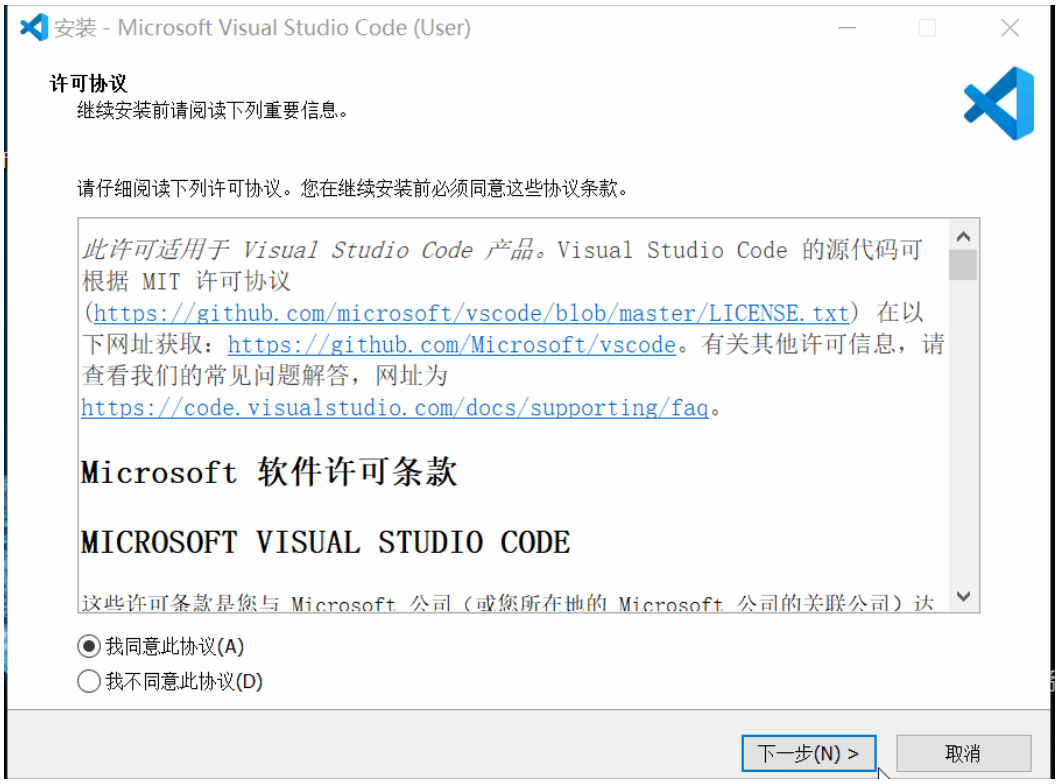


VS Code版本要求：

建议用户使用VS Code 1.85.2版本或者最新版本进行远程连接。

VS Code安装指导如下：

图 6-36 Windows 系统下 VS Code 安装指导



Linux系统下，执行命令**sudo dpkg -i code_1.85.2-1705561292_amd64.deb**安装。

版权所有 © 华为云计算技术有限公司

说明

Linux系统用户，需要在非root用户进行VS Code安装。

6.3.3 VS Code 一键连接 Notebook

前提条件

- 已经创建Notebook实例 ，实例已经开启SSH连接，实例状态为运行中。 请参考**创建Notebook实例**。
- 实例的密钥文件已经下载至本地的如下目录或其子目录中：

  Windows：C:\Users\{{user}}

  Mac/Linux: Users/{{user}}

操作步骤

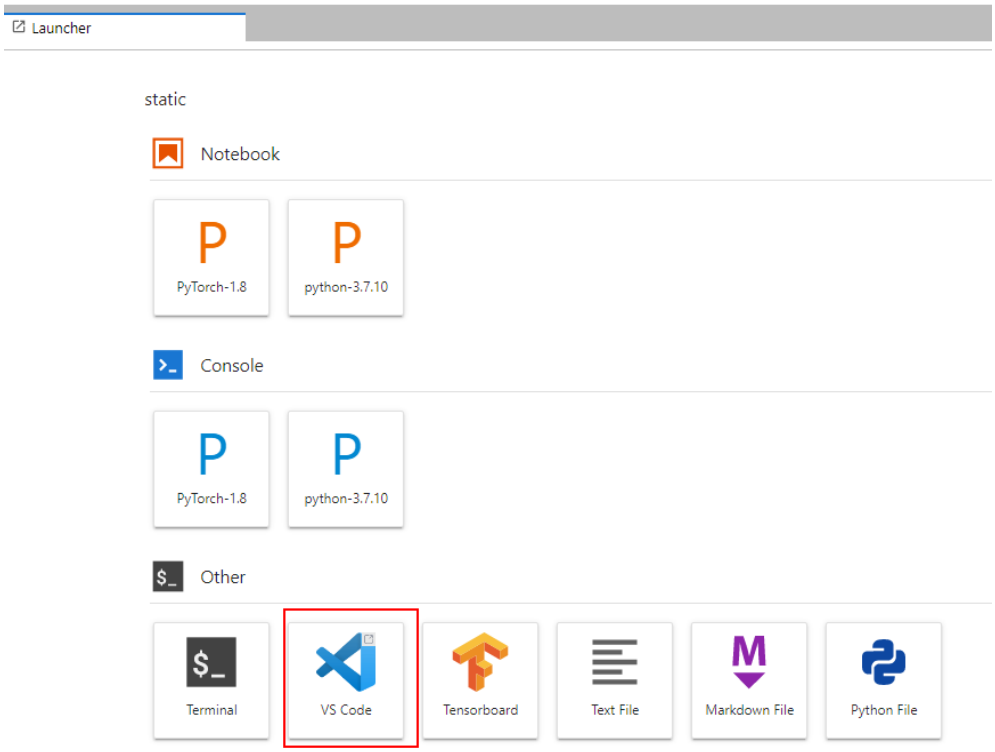
步骤1 登录ModelArts管理控制台，在左侧导航栏中选择“开发环境 > Notebook”，进入“Notebook”页面。

步骤2 该界面显示已创建实例的状态为“运行中”。当前有两种方式，可以打开VS Code连接。单击“操作”列的“更多 > VS Code接入”；或者单击“操作”列的“打开”，自动进入Launcher页面，然后单击“VS Code”。弹出“是否打开Visual Studio Code？”对话框。

图 6-37 打开 VS Code 接入

 版权所有 © 华为云计算技术有限公司

图 6-38 从 Launcher 页面打开 VS Code 接入



步骤3 如果本地已安装VS Code，请单击“打开 Visual Studio Code”，进入“Visual Studio Code”页面。

图 6-39 打开 Visual Studio Code

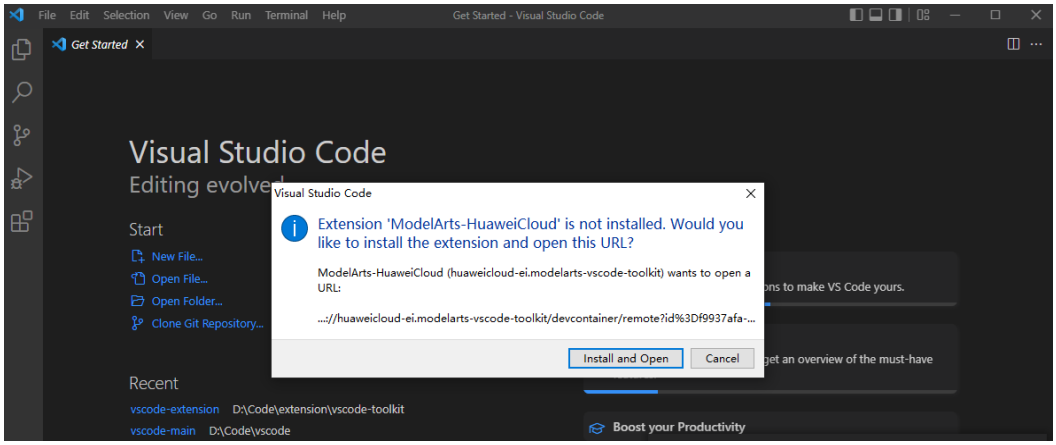
如果本地未安装VS Code，请根据实际选择“win”或“其他”下载并安装VS Code。VS Code安装请参考**安装VS Code软件**。

版权所有 © 华为云计算技术有限公司

图 6-40 下载并安装 VS Code

步骤4 如果用户之前未安装过ModelArts VS Code插件，此时会弹出安装提示，请单击“Install and Open”进行安装；如果之前已经安装过插件，则不会有该提示，请跳过此步骤，直接执行**5**。

图 6-41 安装 VS Code 插件



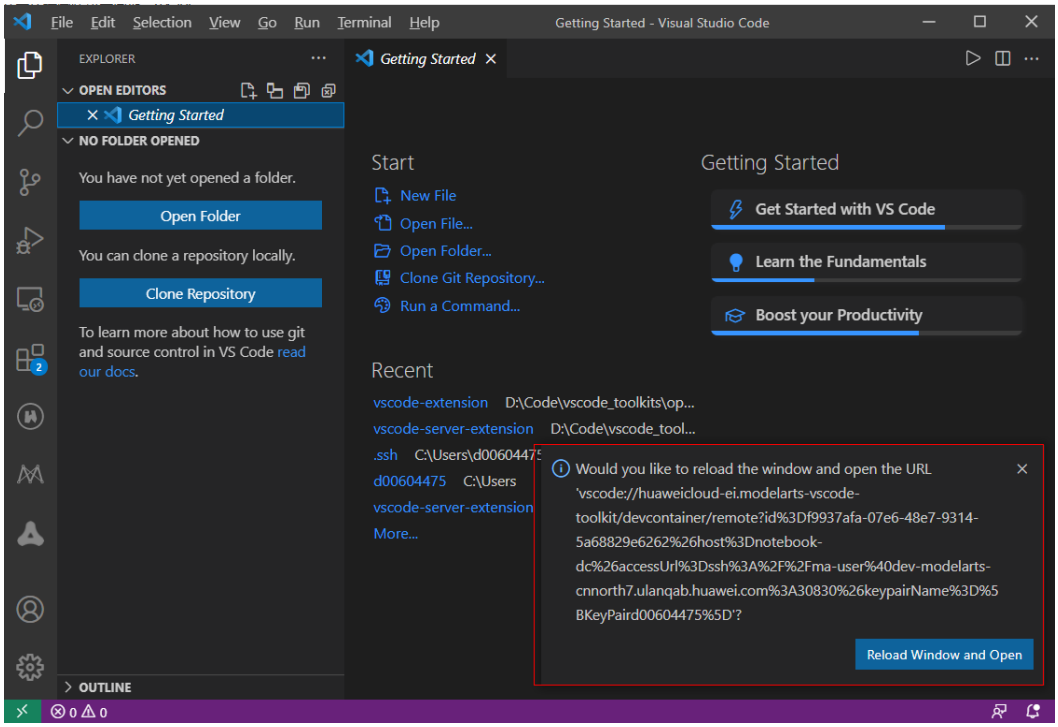
安装过程预计1~2分钟，安装完成后右下角会弹出对话框，请单击“Reload Window and Open”。

说明

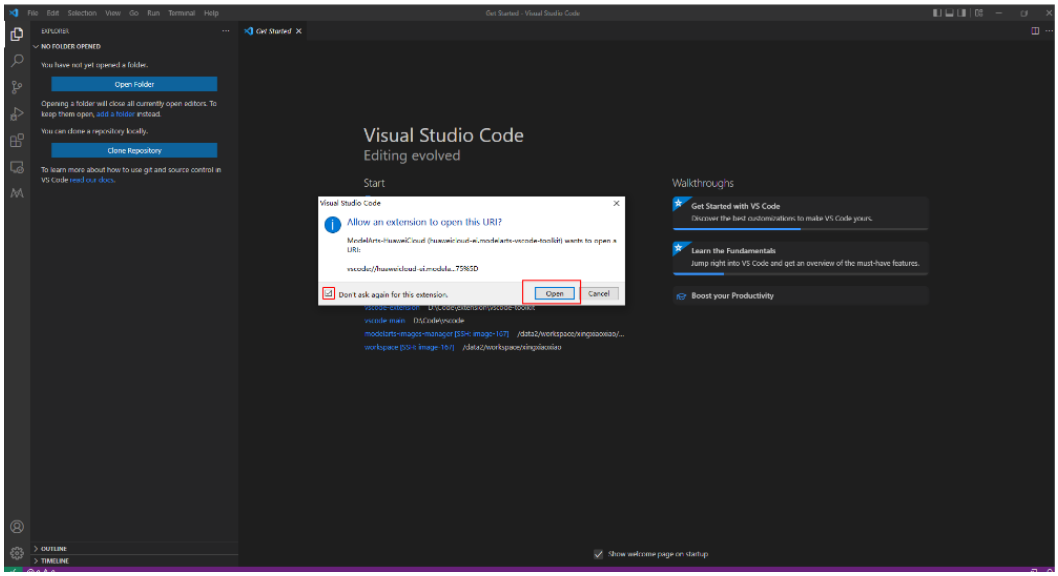
本文以VS Code 1.78.2版本的操作为例，其它版本的VS Code可能不会弹出“Reload Window and Open”，请直接执行**5**。

版权所有 © 华为云计算技术有限公司

图 6-42 Reload Window and Open



在弹出的提示中，勾选“Don't ask again for this extension”，然后单击"Open"。

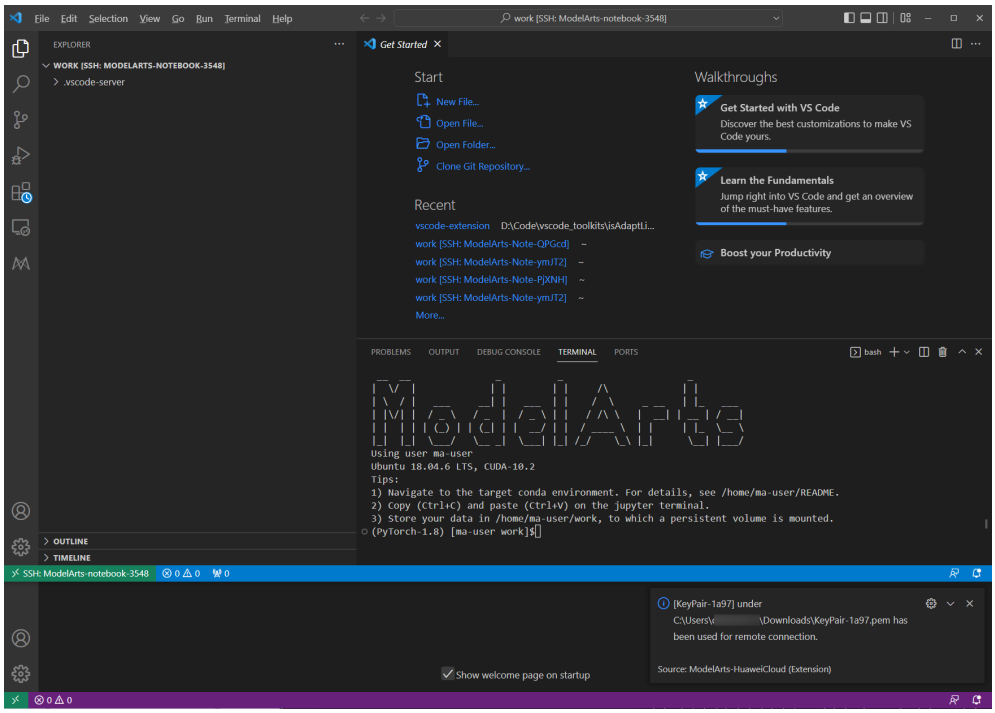


步骤5 远程连接Notebook实例。

- 远程连接执行前，会自动在（Windows：C:\Users\{{user}}\.ssh或者downloads，Mac/Linux: Users/{{user}}/.ssh或者downloads）目录下根据密钥名称查找密钥文件，如果找到则直接使用该密钥打开新窗口并尝试连接远程实例，此时无需选择密钥。

版权所有 © 华为云计算技术有限公司

图 6-43 远程连接 Notebook 实例

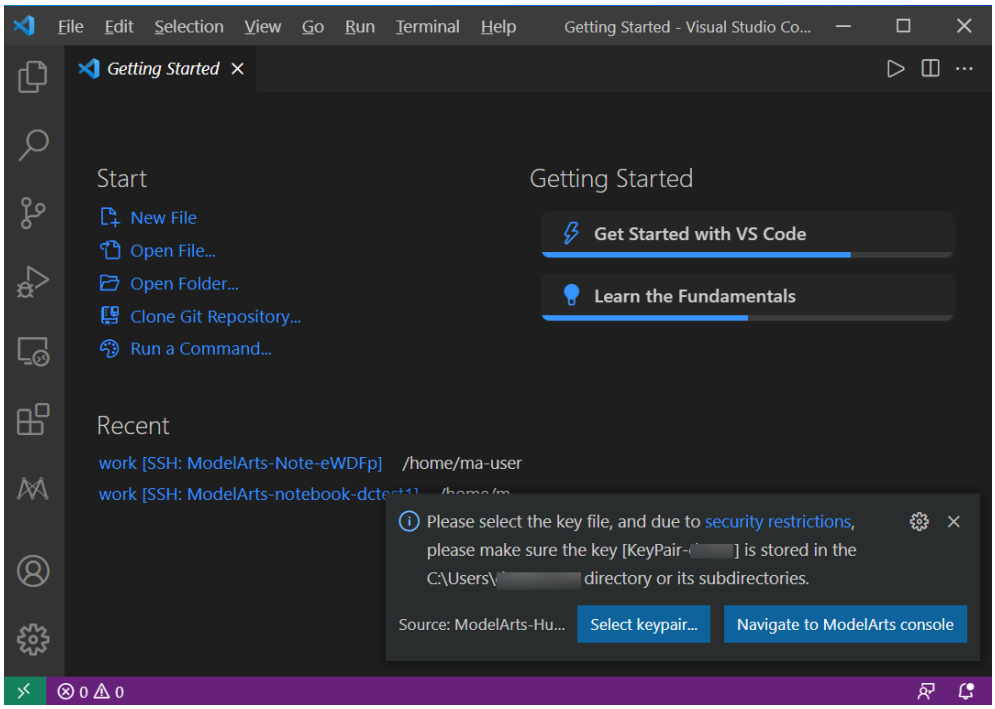


- 如果未找到会弹出选择框，请根据提示选择正确的密钥。

  说明

  密钥文件名不能包含中文字符。

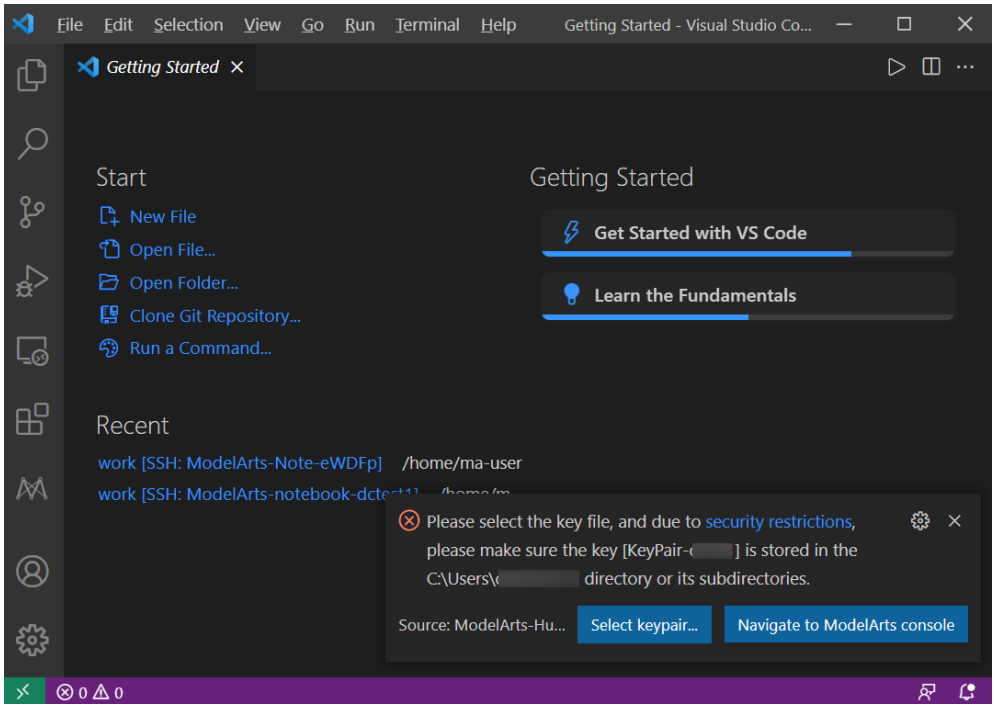
图 6-44 选择密钥文件



 版权所有 © 华为云计算技术有限公司

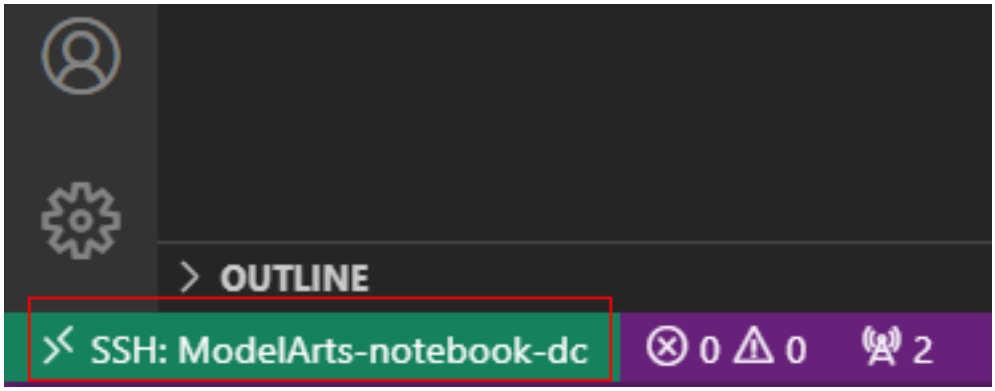
- 如果密钥选择错误，则弹出提示信息，请根据提示信息选择正确密钥。

图 6-45 选择正确的密钥文件



当左下角显示如下状态时，代表实例连接成功：

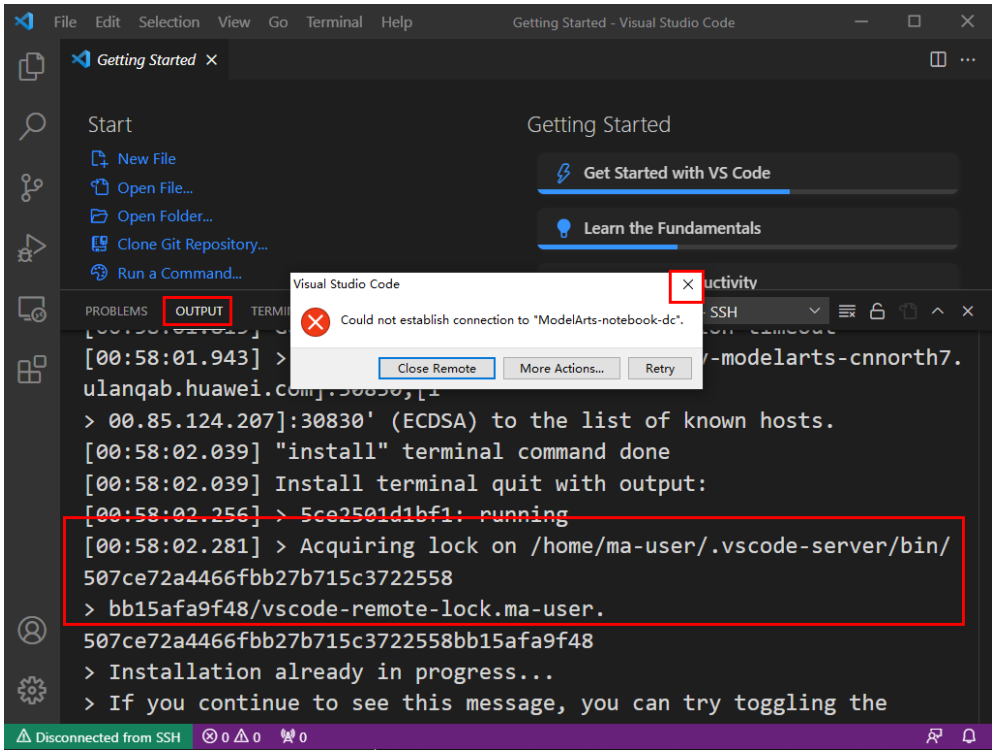
图 6-46 实例连接成功



当弹出如下错误时，代表实例连接失败，请关闭弹窗，并查看OUTPUT窗口的输出日志，请查看**FAQ**并排查失败原因。

版权所有 © 华为云计算技术有限公司

图 6-47 实例连接失败



----结束

常见问题

在ModelArts控制台界面上单击VS Code接入并在新界面单击打开，未弹出VS Code窗口

远程连接出现弹窗报错：Could not establish connection to xxx

在ModelArts控制台界面上单击VS Code接入并在新界面单击打开，VS Code打开后未进行远程连接

报错“ssh: connect to host xxx.pem port xxxxx: Connection refused”如何解决?

报错“no such identity: C:/Users/xx /test.pem: No such file or directory”如何解决?

报错“Bad owner or permissions on C:\Users\Administrator/.ssh/config”或“Connection permission denied (publickey)”如何解决?

6.3.4 VS Code ToolKit 连接 Notebook

本节介绍如何在本地使用ModelArts提供的VS Code插件工具VS Code ToolKit，协助用户完成SSH远程连接Notebook。

前提条件

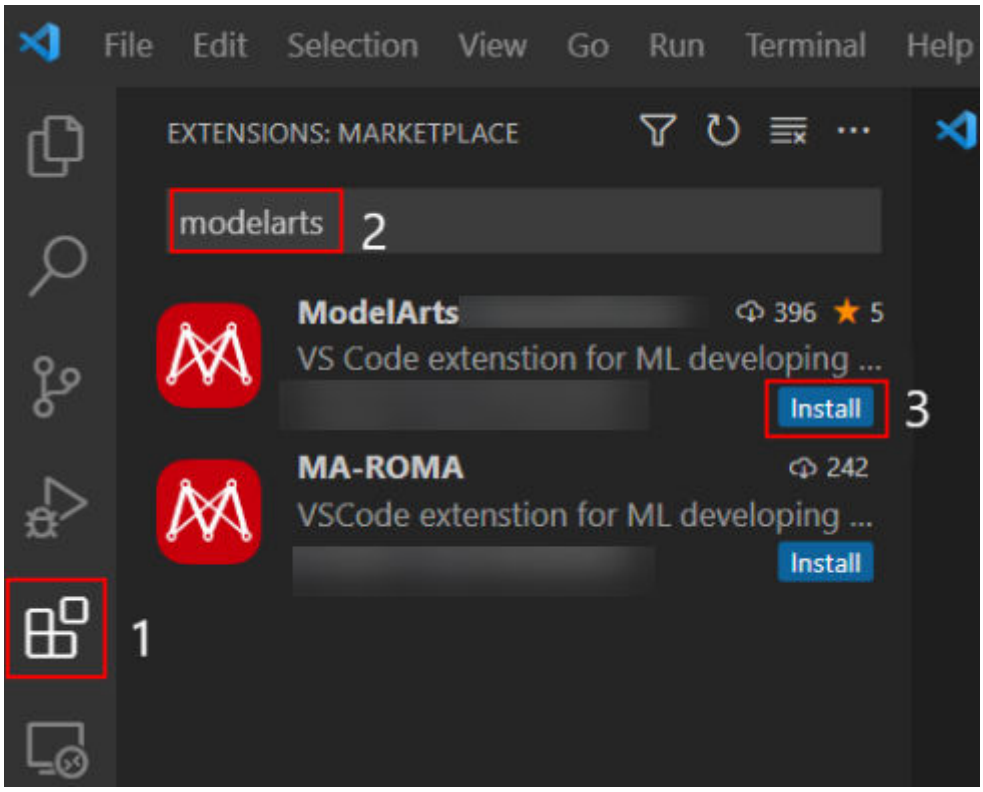
已下载并安装VS Code。详细操作请参考**安装VS Code软件**。

版权所有 © 华为云计算技术有限公司

Step1 安装 VS Code 插件

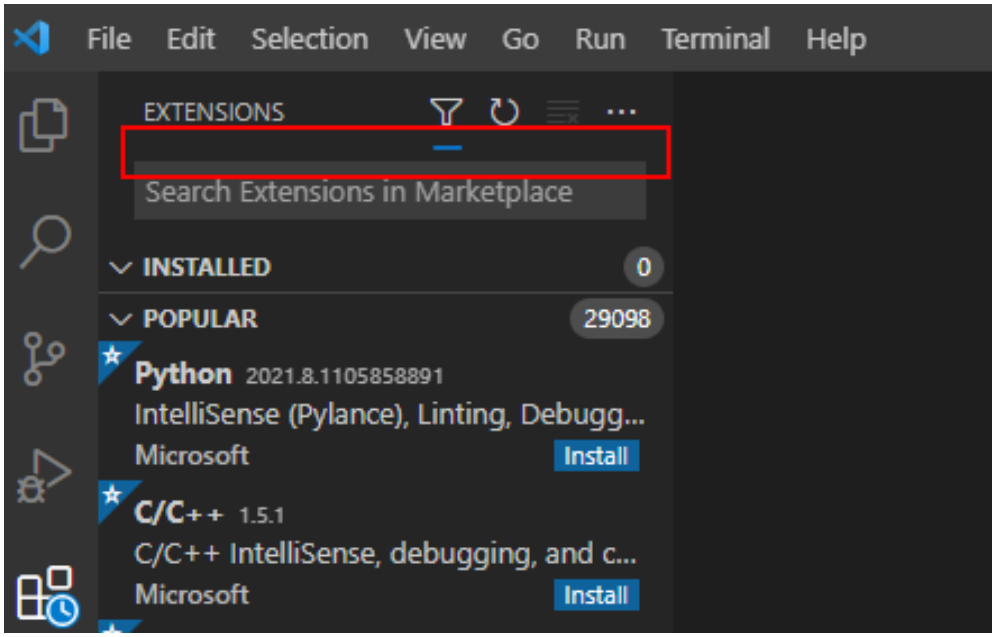
1. 在本地的VS Code开发环境中，如图6-48所示，在VS Code扩展中搜索“ModelArts-HuaweiCloud”并单击“安装”。

图 6-48 安装 VS Code 插件



2. 安装过程预计1~2分钟，如图6-49所示，请耐心等待。

图 6-49 安装过程



版权所有 © 华为云计算技术有限公司

3. 安装完成后，系统右下角提示安装完成，导航左侧出现ModelArts图标和SSH远程连接图标，表示VS Code插件安装完成。

图 6-50 安装完成提示

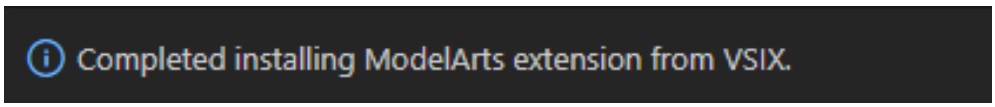
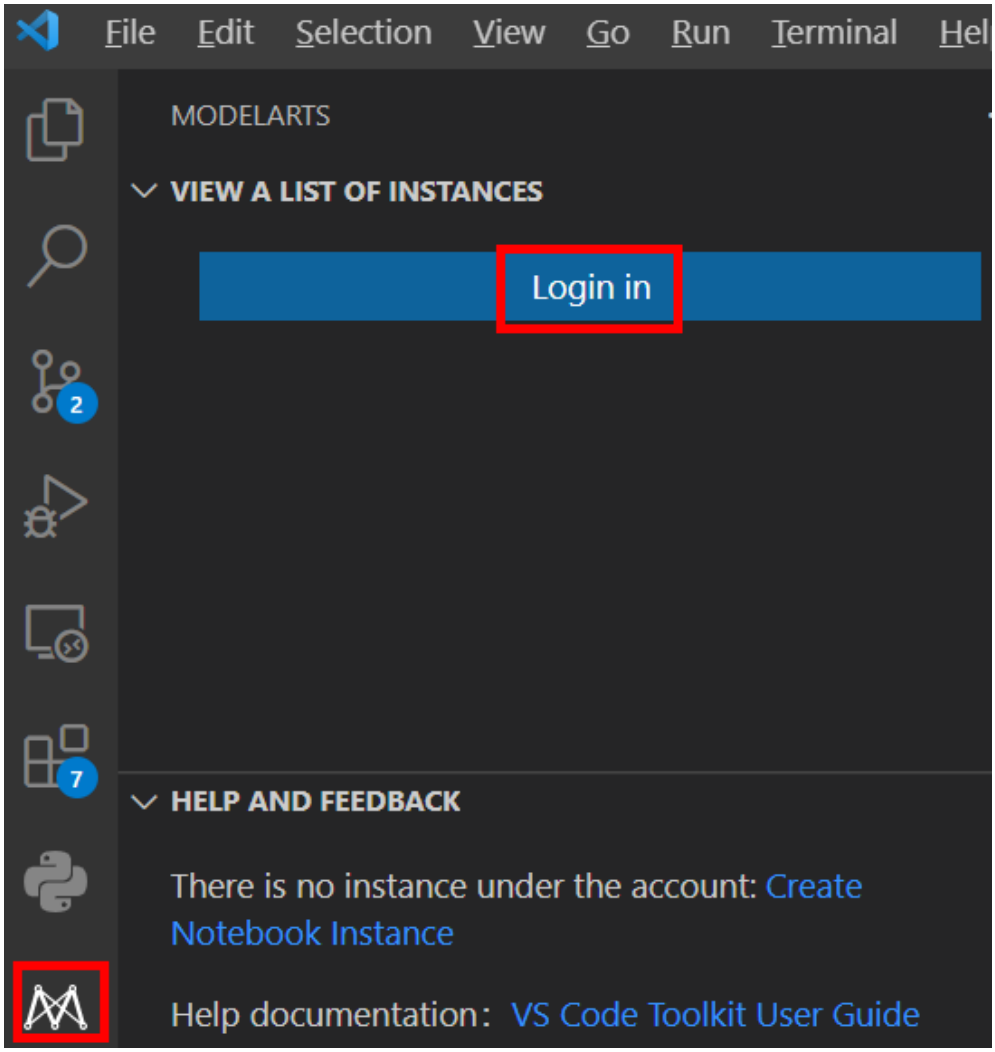


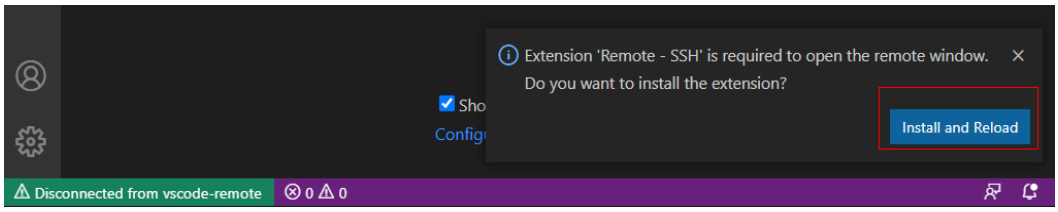
图 6-51 安装完成



当前网络不佳时SSH远程连接插件可能未安装成功，此时无需操作，在**Step4 连接Notebook实例**的**1**之后，会弹出如下图对话框，单击Install and Reload即可。

版权所有 © 华为云计算技术有限公司

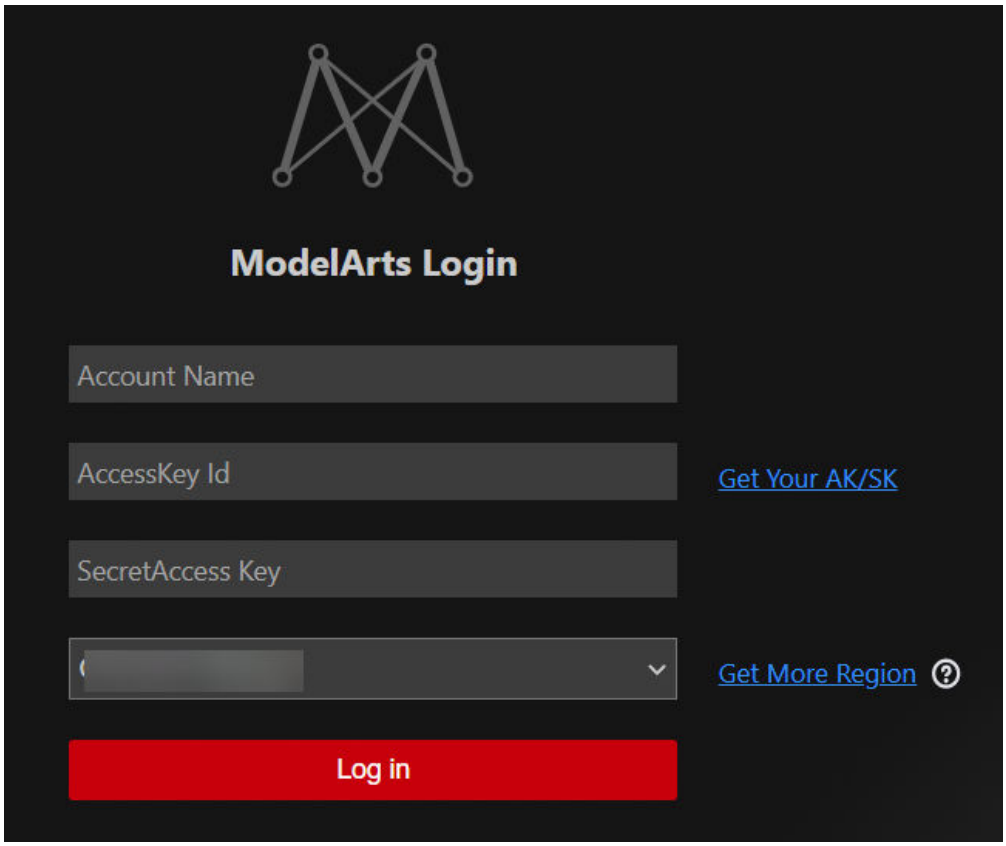
图 6-52 重新连接远程 SSH



Step2 登录 VS Code 插件

1. 在本地的VS Code开发环境中，单击ModelArts图标，单击“User Settings”，配置用户登录信息。

   图 6-53 登录插件

   输入如下用户登录信息，单击“登录”。

   - Name：自定义用户名，仅用于VS Code页面展示，不与任何华为云用户关联。
   - AK、SK：在“账号中心 > 我的凭证 > 访问密钥”中创建访问密钥，获取AK、SK（**参考链接**）。
   - 选择站点：此处的站点必须和远程连接的Notebook在同一个站点，否则会导致连接失败。

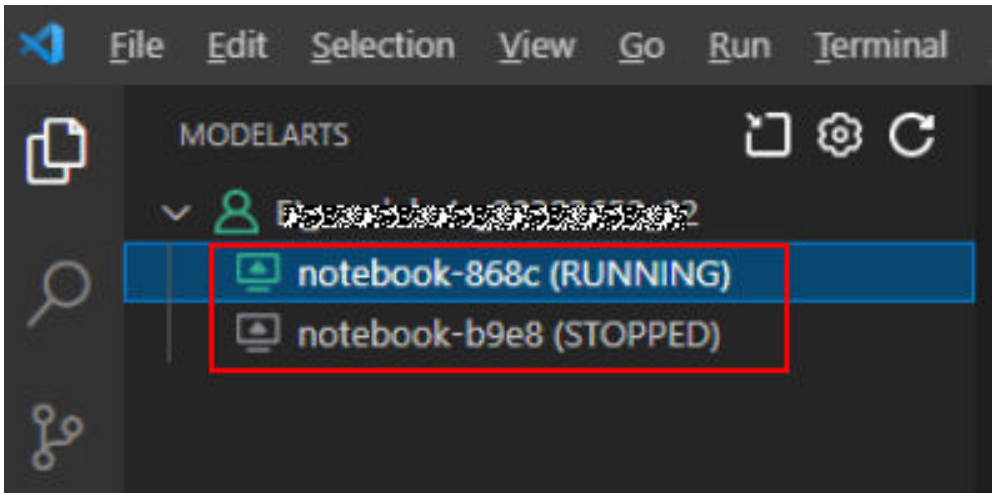
2. 登录成功后显示Notebook实例列表。

版权所有 © 华为云计算技术有限公司

说明

此处仅显示ModelArts控制台default工作空间下的Notebook实例。

图 6-54 登录成功



Step3 创建 Notebook 实例

注意

- 创建实例时，需开启“SSH远程开发”，并下载保存密钥对至本地如下目录。
  Windows：C:\Users\{{user}}
  macOS/Linux: Users/{{user}}
- 密钥对在用户第一次创建时自动下载，之后使用相同的密钥时不会再有下载界面（请妥善保管），或者每次都使用新的密钥对。

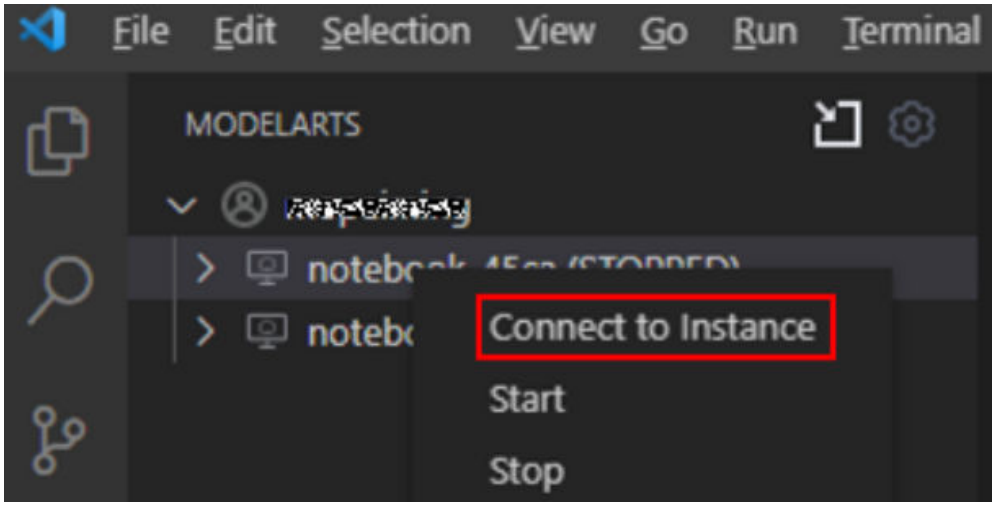
创建一个Notebook实例，并开启远程SSH开发，具体参见**创建Notebook实例**。

Step4 连接 Notebook 实例

1. 在本地的VS Code开发环境中，右键单击实例名称，单击“Connect to Instance”，启动并连接Notebook实例。
   Notebook实例状态处于“运行中”或“停止”状态都可以，如果Notebook实例是停止状态，连接Notebook时，VS Code插件会先启动实例再去连接。

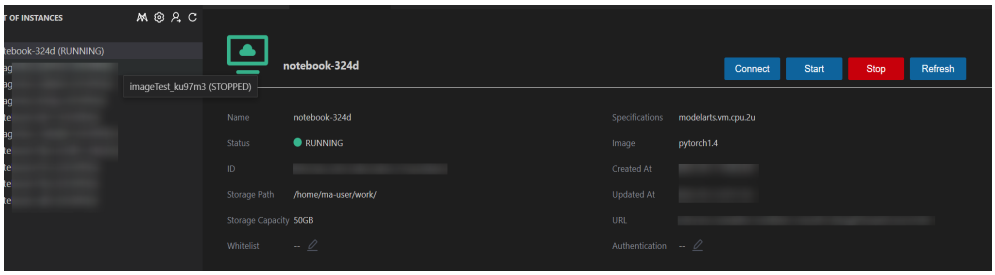
版权所有 © 华为云计算技术有限公司

图 6-55 连接 Notebook 实例



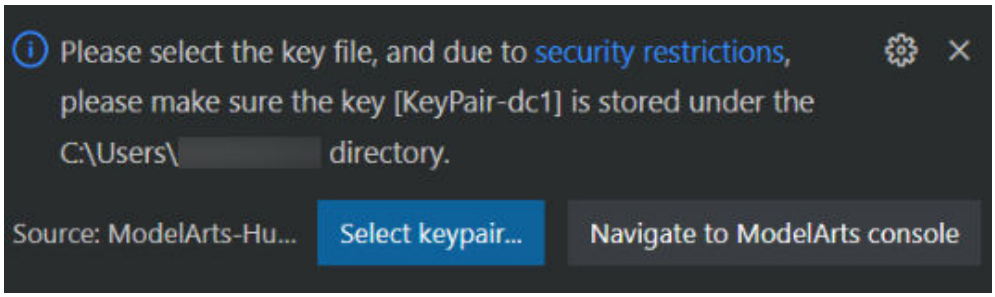
或者单击实例名称，在VS Code开发环境中显示Notebook实例详情页，单击“连接”，系统自动启动该Notebook实例并进行远程连接。

图 6-56 查看 Notebook 实例详情页



2. 第一次连接Notebook时，系统右下角会提示需要先配置密钥文件。选择本地密钥pem文件，根据系统提示单击“OK”。

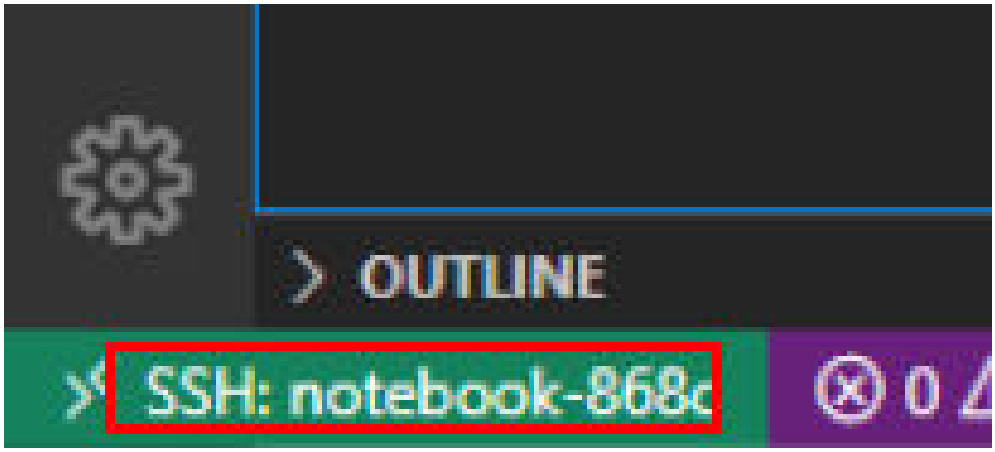
图 6-57 配置密钥文件



3. 单击“确定”后，插件自动连接远端Notebook实例。首次连接大约耗时1~2分钟，取决于本地的网络情况。VS Code环境左下角显示类似下图即为连接成功。

版权所有 © 华为云计算技术有限公司

图 6-58 连接成功

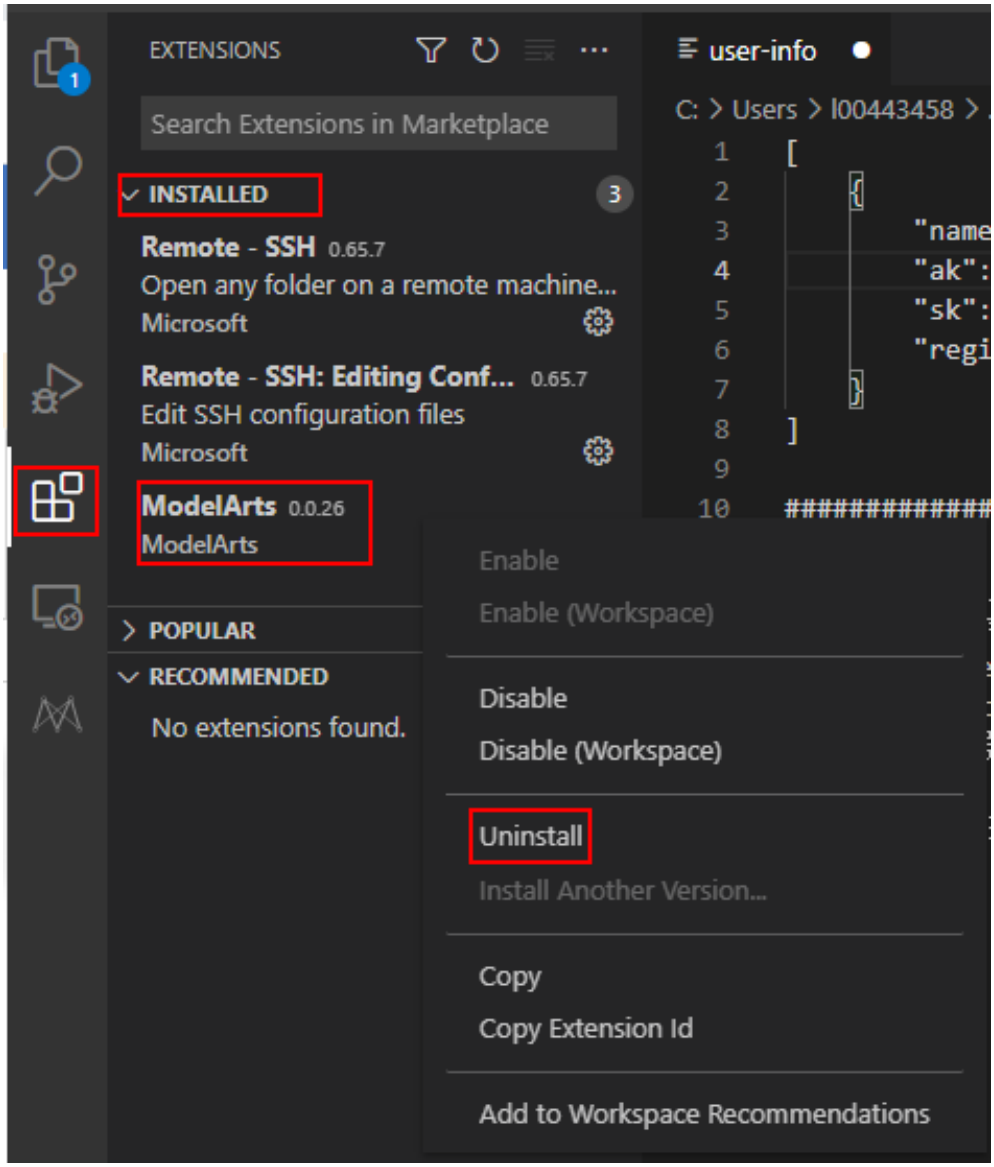


相关操作

卸载VS Code插件操作如图6-59所示。

版权所有 © 华为云计算技术有限公司

图 6-59 卸载 VS Code 插件



常见问题

报错“Permissions for 'x:/xxx.pem' are too open”如何解决?

报错“ssh: connect to host ModelArts-xxx port xxx: Connection timed out”如何解决?

报错“Host key verification failed.'或者'Port forwarding is disabled.”如何解决?

6.3.5 VS Code 手动连接 Notebook

本地IDE环境支持PyCharm和VS Code。通过简单配置，即可用本地IDE远程连接到ModelArts的Notebook开发环境中，调试和运行代码。

本章节介绍基于VS Code环境访问Notebook的方式。

版权所有 © 华为云计算技术有限公司

前提条件

- 已下载并安装VS Code。详细操作请参考**安装VS Code软件**。
- 用户本地PC或服务器的操作系统中建议先安装Python环境，详见**VSCode官方指导**。
- 创建一个Notebook实例，并开启远程SSH开发。该实例状态必须处于“运行中”，具体参见**创建Notebook实例**章节。
- 在Notebook实例详情页面获取开发环境访问地址（例如：dev-modelarts-cnnorth4.huaweicloud.com）和端口号。

  图 6-60 Notebook 实例详情页面

- 准备好密钥对。

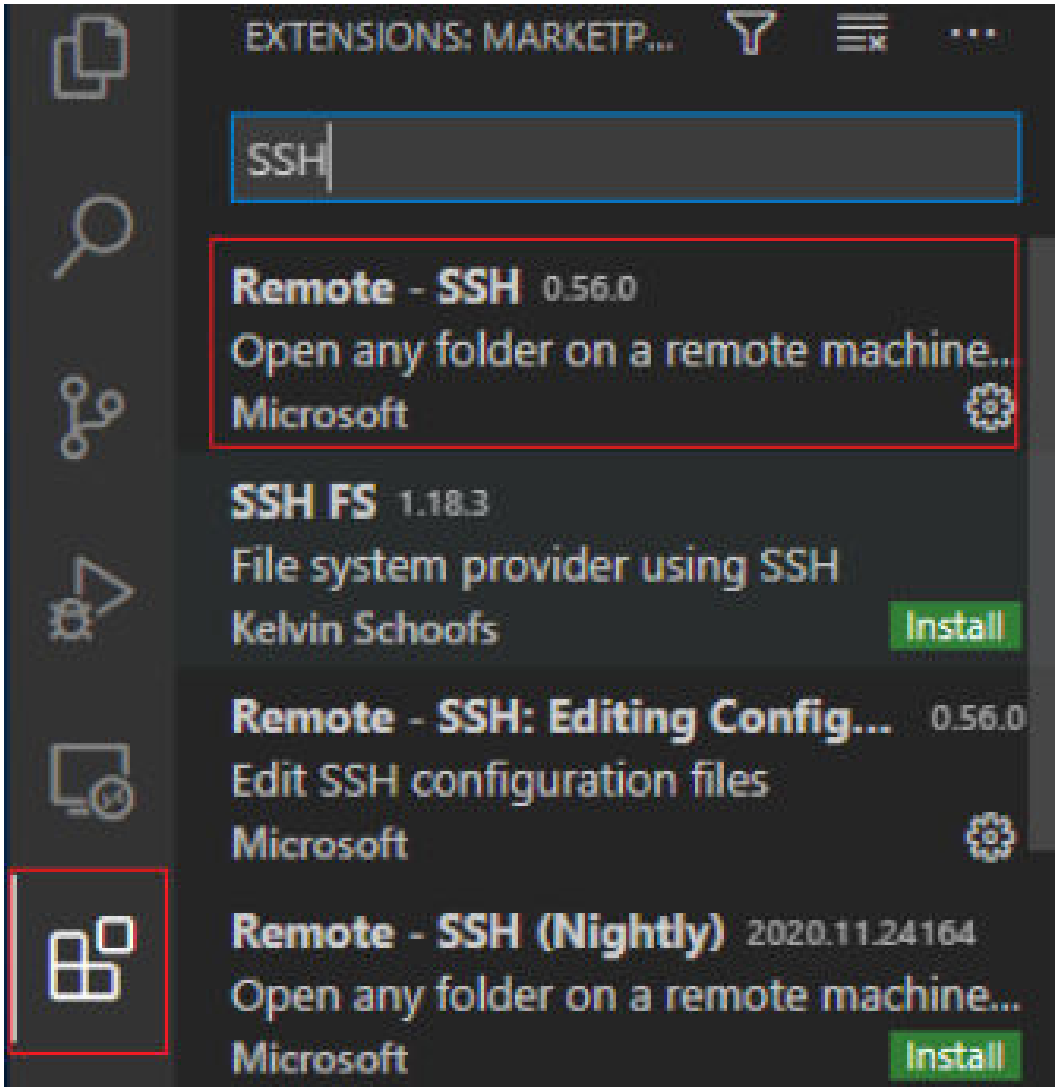
  密钥对在用户第一次创建时，自动下载，之后使用相同的密钥时不会再有下载界面（用户一定要保存好），或者每次都使用新的密钥对。

Step1 添加 Remote-SSH 插件

在本地的VS Code开发环境中，单击左侧列表的Extensions图标选项  ，在搜索框中输入SSH，单击Remote-SSH插件的install按钮，完成插件安装。

版权所有 © 华为云计算技术有限公司

图 6-61 添加 Remote-SSH 插件

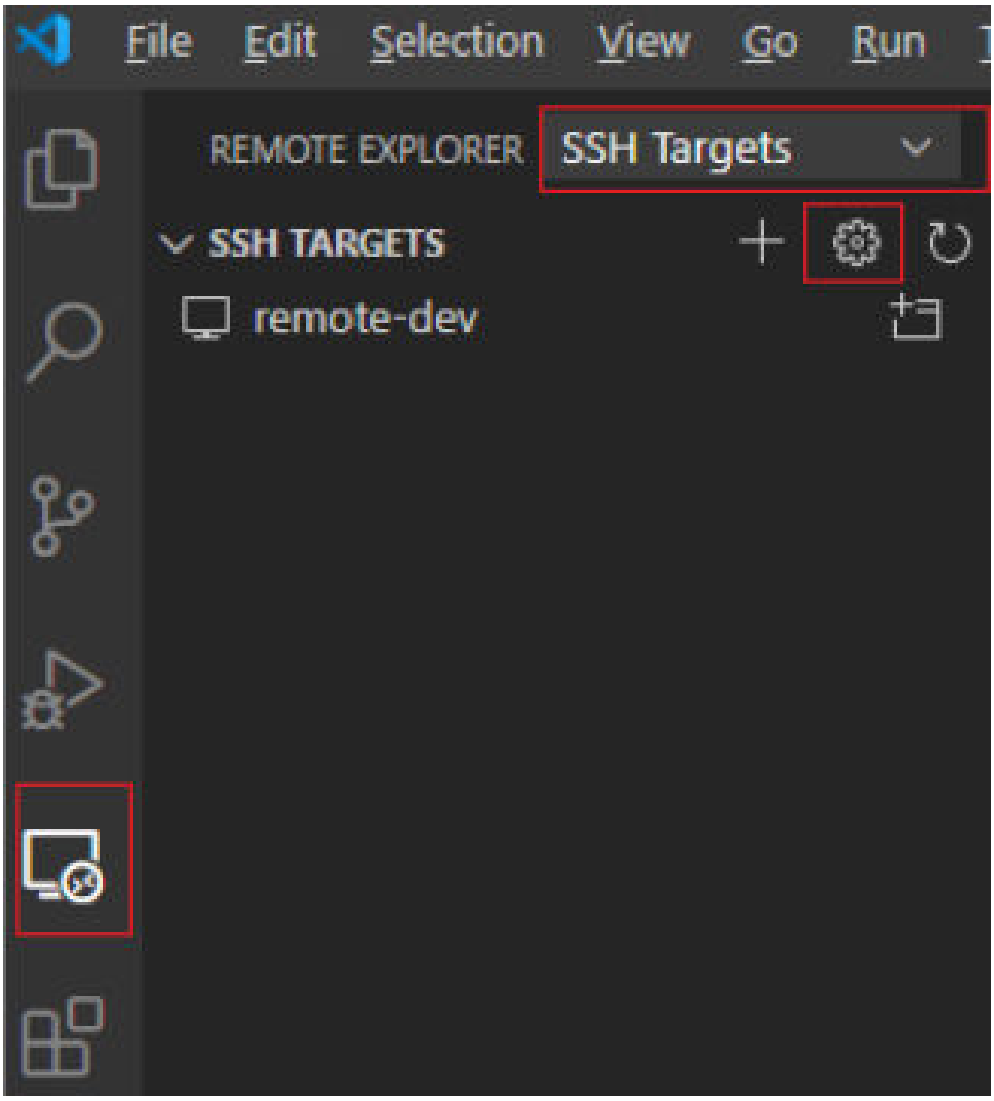


Step2 配置 SSH

1. 在本地的VS Code开发环境中，单击左侧Remote Explorer按钮，在上方的下拉列表中选择“SSH Target”，再单击页面上的设置按钮，此时会出现SSH配置文件路径。

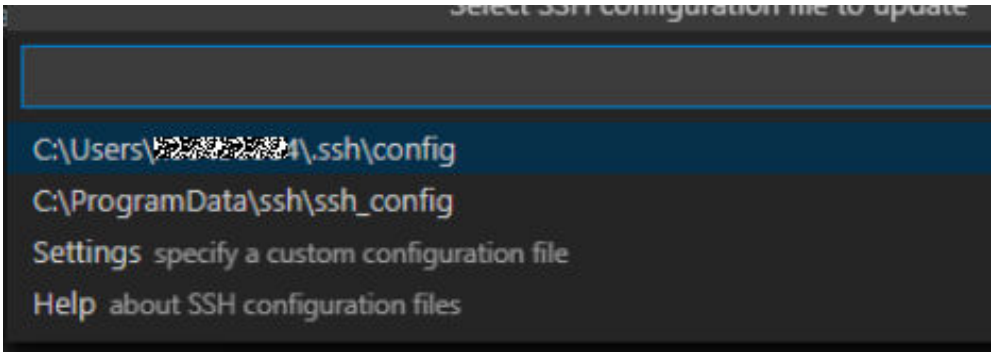
版权所有 © 华为云计算技术有限公司

图 6-62 配置 SSH Targets 页面



2. 单击列表中出现的SSH路径按钮，打开config文件，进行配置。

图 6-63 配置 SSH Config 文件




```
HOST remote-dev
    hostname <instance connection host>
    port <instance connection port>
    user ma-user
    IdentityFile ~/.ssh/test.pem
```

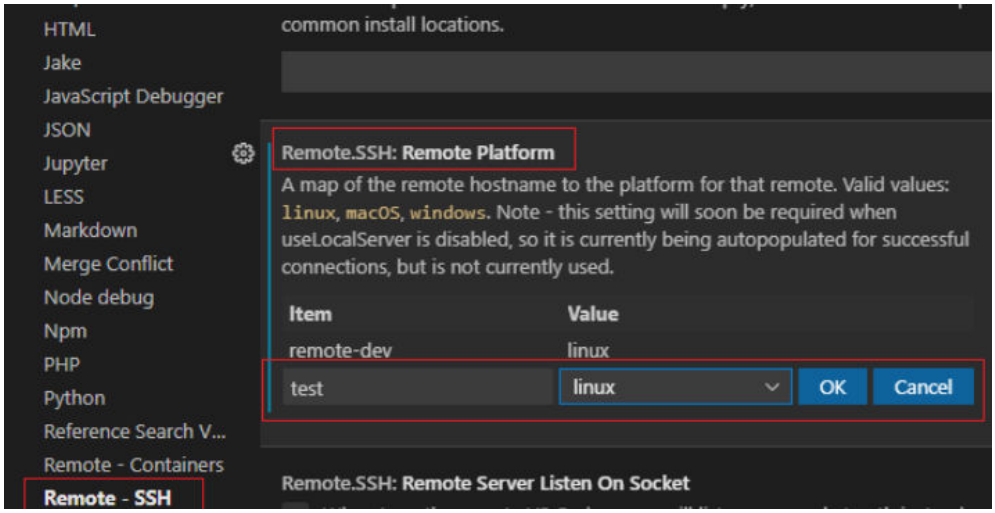

版权所有 © 华为云计算技术有限公司

```
UserKnownHostsFile=/dev/null
StrictHostKeyChecking no
```

- Host: 自定义设置的云上开发环境名称。
- HostName: 云上开发环境的访问地址，即在开发环境实例页面远程访问模块获取的访问地址。例如：dev-modelarts-cnnorth4.huaweicloud.com
- Port: 云上开发环境的端口，即在开发环境实例页面远程访问模块获取的端口号。
- User：登录用户只支持ma-user进行登录。
- IdentityFile: 存放在本地的云上开发环境私钥文件，即前提条件**准备好密钥对**中准备的密钥对。

3. 配置云上开发环境系统平台，单击“File > Preference > Settings > Extensions > Remote-SSH”，在“Remote Platform”中，单击“Add Item”选项，设置“Item”和“Value”，配置完成后，单击“OK”。

图 6-64 配置云上开发环境系统平台



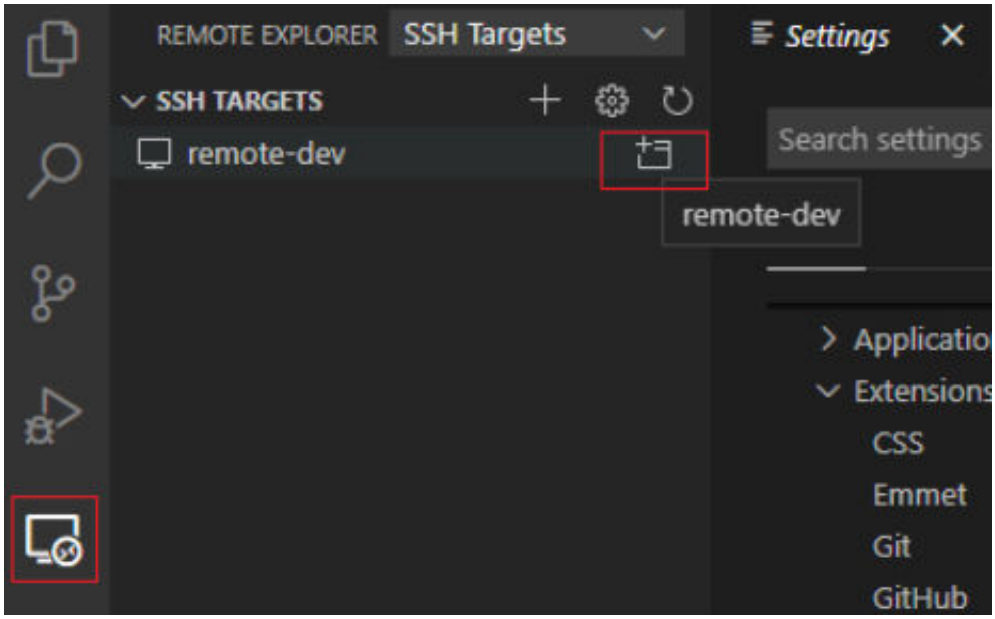
“Item”： 在SSH Config中配置的Host的名称。

“Value”： 在下拉选择框中选择远端开发环境平台。

4. 再回到SSH Targets页面，单击右侧的Connect to Host in New Window按钮  ，该按钮会显示远程开发环境名称，选中并打开。

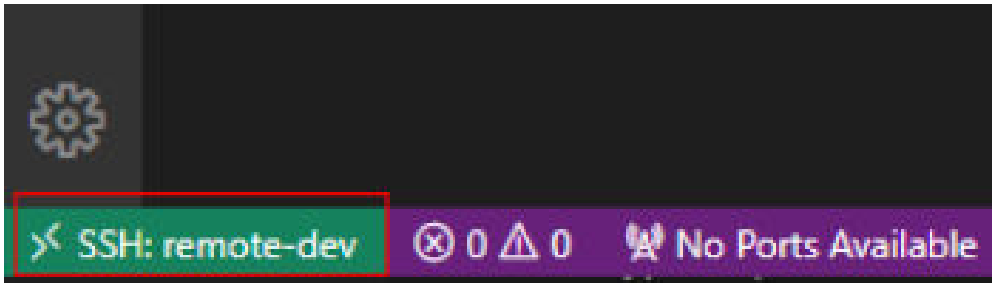
 版权所有 © 华为云计算技术有限公司

图 6-65 打开开发环境



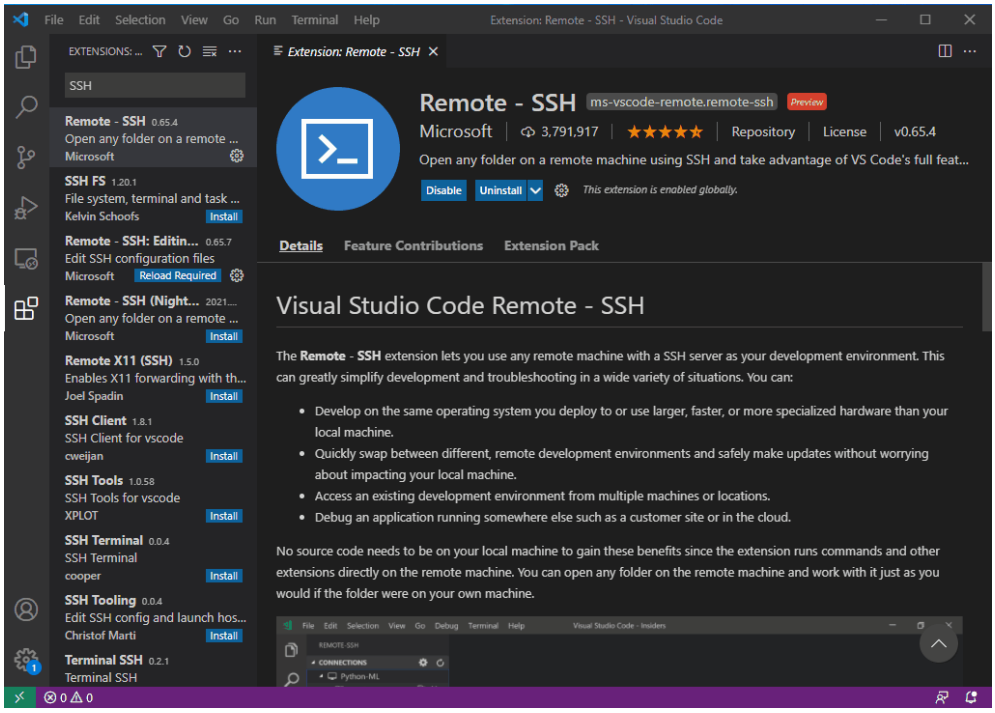
在新打开的页面中，看到下图所示界面，即表示连接成功。

图 6-66 开发环境远程连接成功



版权所有 © 华为云计算技术有限公司

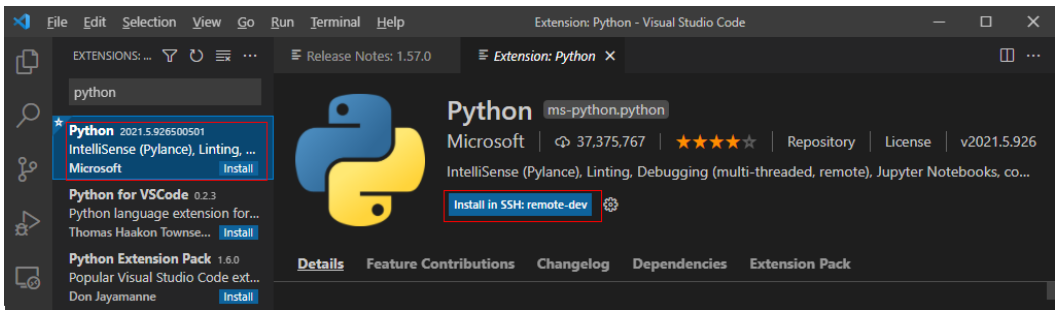
图 6-67 完整配置示例



Step3 安装云端 Python 插件

在新打开的VS Code界面，单击左侧列表的Extensions选项  ，在搜索框中输入Python，在下拉列表中单击“Install”进行安装。

图 6-68 安装云端 Python 插件



如果安装云端的Python插件不成功时，建议通过离线包的方式安装。具体操作请参见**安装远端插件时不稳定，需尝试多次**。

Step4 云上环境依赖库安装

在进入容器环境后，可以使用不同的虚拟环境，例如TensorFlow、PyTorch等，但是实际开发中，通常还需要安装其他依赖包，此时可以通过Terminal连接到环境里操作。

1. 在VS Code环境中，执行Ctrl+Shift+P。
2. 搜Python: Select Interpreter，选择对应的Python环境。

版权所有 © 华为云计算技术有限公司

3. 单击页面上方的“Terminal > New Terminal”，此时打开的命令行界面即为远端容器环境命令行。
4. 进入引擎后，通过执行如下命令安装依赖包。
```
pip install spacy
```

6.3.6 在 VS Code 中远程调试代码

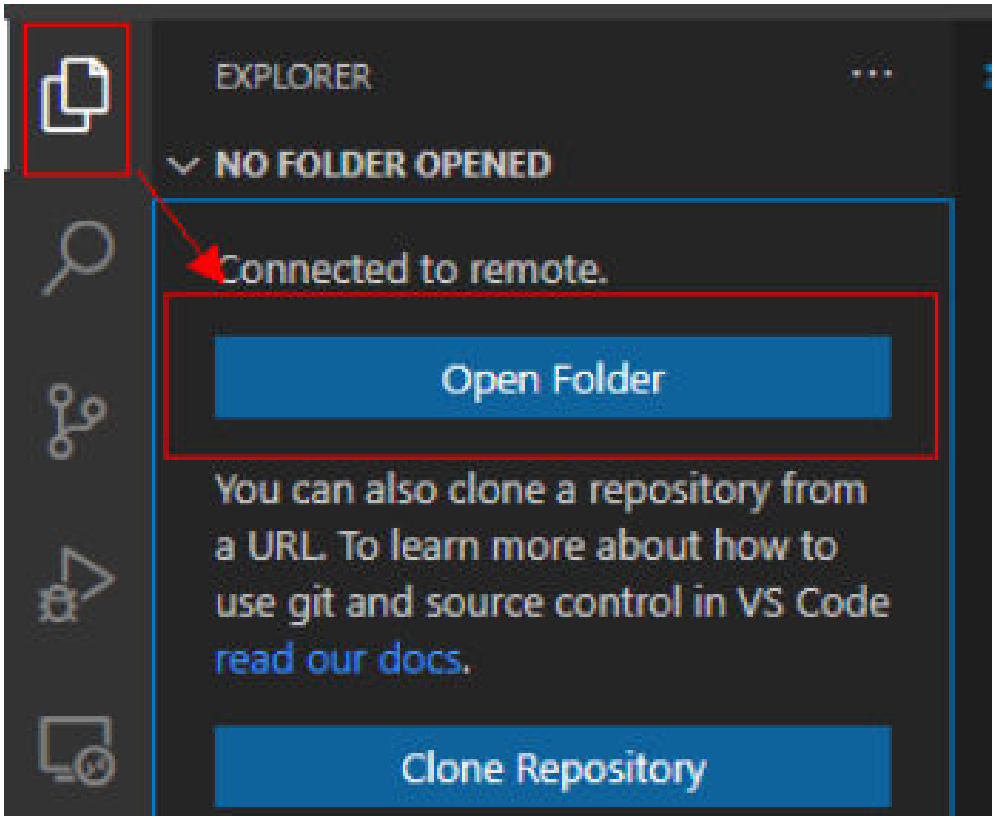
前提条件

VS Code已连接到Notebook。

Step1 上传本地代码到云端开发环境

1. 在VS Code界面，单击“File > OpenFolder”打开云端路径。

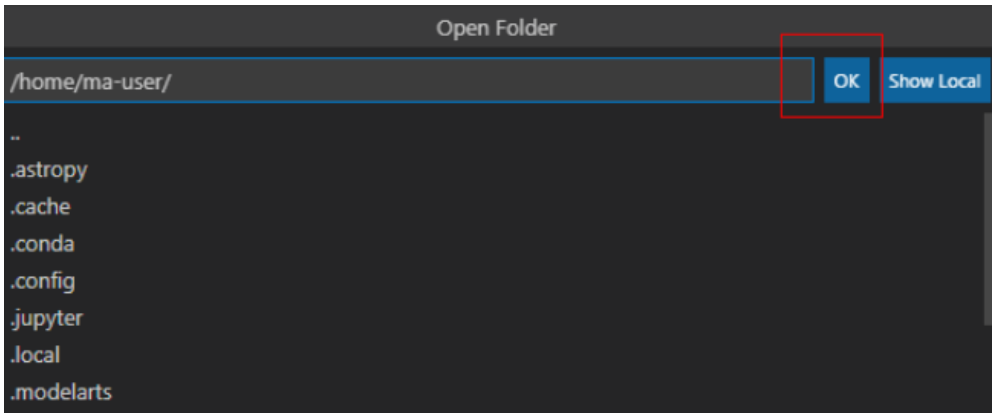
图 6-69 Open Folder



2. 选择要打开的路径，单击“OK”。

版权所有 © 华为云计算技术有限公司

图 6-70 选择文件路径

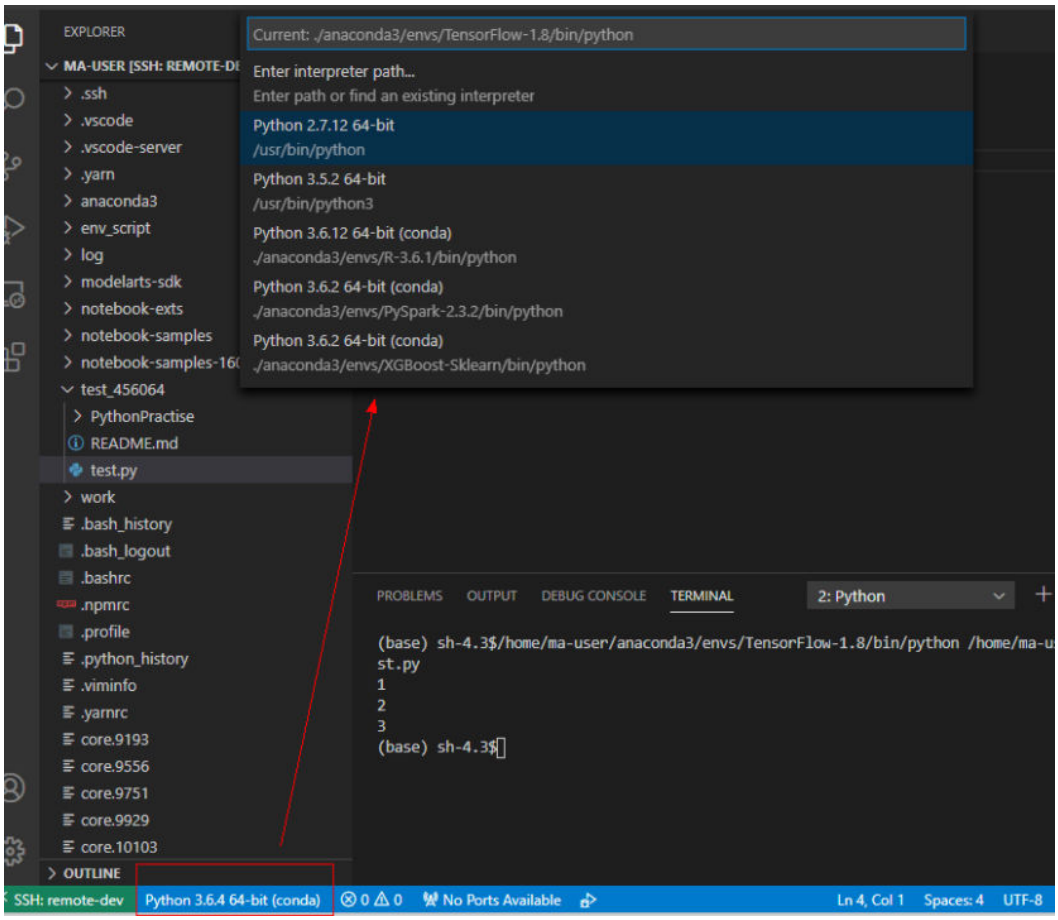


3. 此时，会在IDE左侧出现该开发环境下的目录结构，把想要上传的代码及其他文件直接拖拽至对应的文件夹内即完成本地代码上传至云端。

Step2 远程调试代码

在VS Code中打开要执行的代码文件，在执行代码之前需要选择合适的Python版本路径，单击下方默认的Python版本路径，此时在上方会出现该远程环境上所有的python版本，选择自己需要的版本即可。

图 6-71 选择 Python 版本



版权所有 © 华为云计算技术有限公司

- 对于打开的代码文件，单击run按钮，即可执行，可以在下方的Terminal中看到代码输出信息。
- 如果执行较长时间的训练任务，建议使用nohup命令后台运行，否则SSH窗口关闭或者网络断连会影响正在运行的训练任务，命令参考：
  ```
  nohup your_train_job.sh > output.log 2>&1  &  tail -f  output.log
  ```
- 如果要对代码进行debug调试，步骤如下：
  a. 单击左侧“Run > Run and Debug”。
  b. 选择当前打开的默认的python代码文件进行调试。
  c. 对当前代码进行打断点，即在代码左侧进行单击，就会出现小红点。
  d. 此时，即可按照正常的代码调试步骤对代码调试，在界面左边会显示debug信息，代码上方有相应的调试步骤。

VS Code 使用常见问题

VSCode中查看远端日志

使用VSCode调试代码时不能进入源码

使用VSCode提交代码时弹出对话框提示用户名和用户邮箱配置错误

VSCode连接远端Notebook时报错如“XHR failed”

安装远端插件时不稳定，需尝试多次

Notebook实例重新启动后，需要删除本地known_hosts才能连接

更多VS Code使用相关FAQ，请参见**FAQ**。

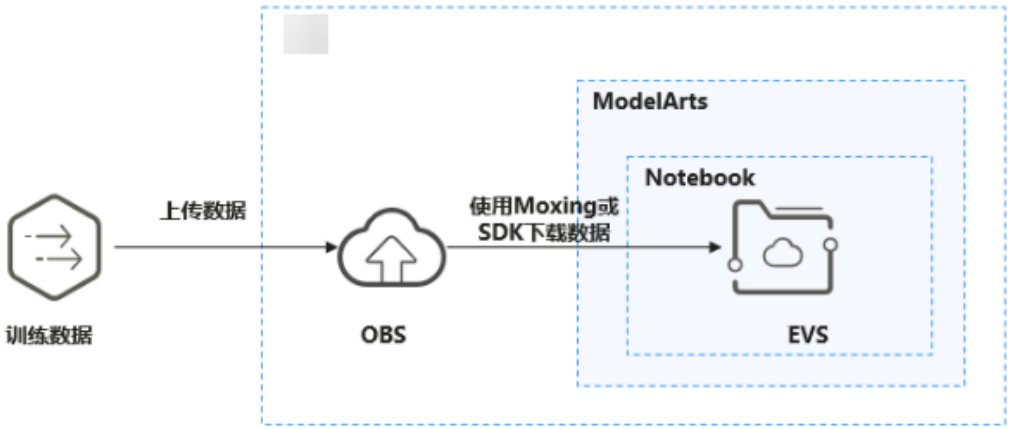
6.3.7 在 VS Code 中上传下载文件

在本地 IDE 中上传数据至 Notebook

不大于500MB数据量，直接复制至本地IDE中即可。

大于500MB数据量，请先上传到OBS中，再从OBS上传到云上开发环境。

图 6-72 数据通过 OBS 中转上传到 Notebook

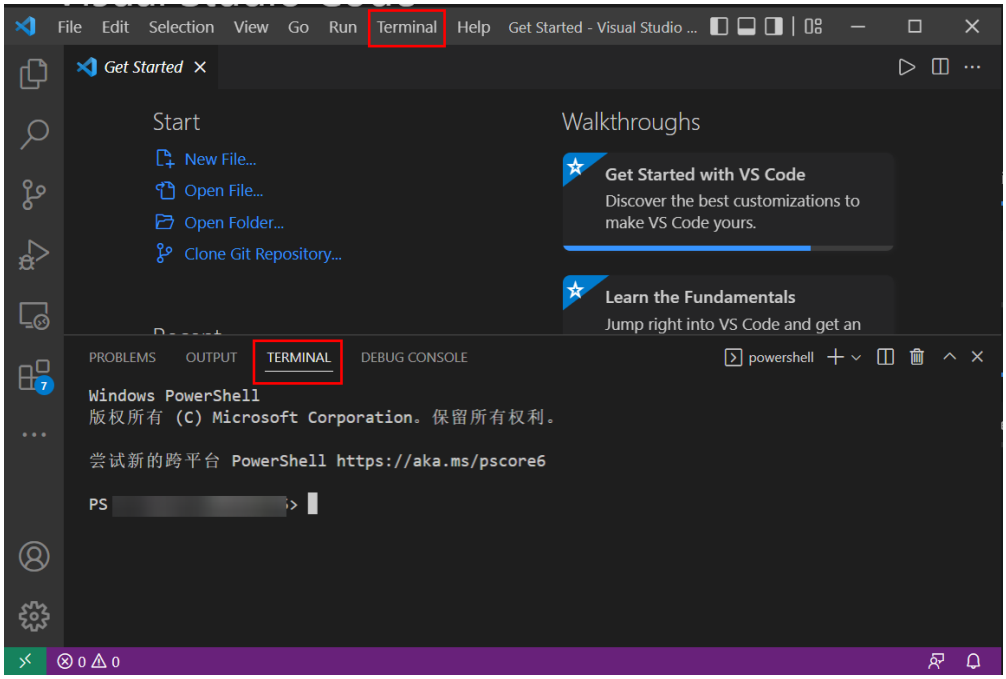


操作步骤

版权所有 © 华为云计算技术有限公司

1. 上传数据至OBS。具体操作请参见**上传文件至OBS桶**。或者在本地VS Code的Terminal中使用ModelArts SDK完成数据上传至OBS。
   首先在本地VS Code环境中开启Terminal。

   图 6-73 本地 VS Code 环境开启 Terminal

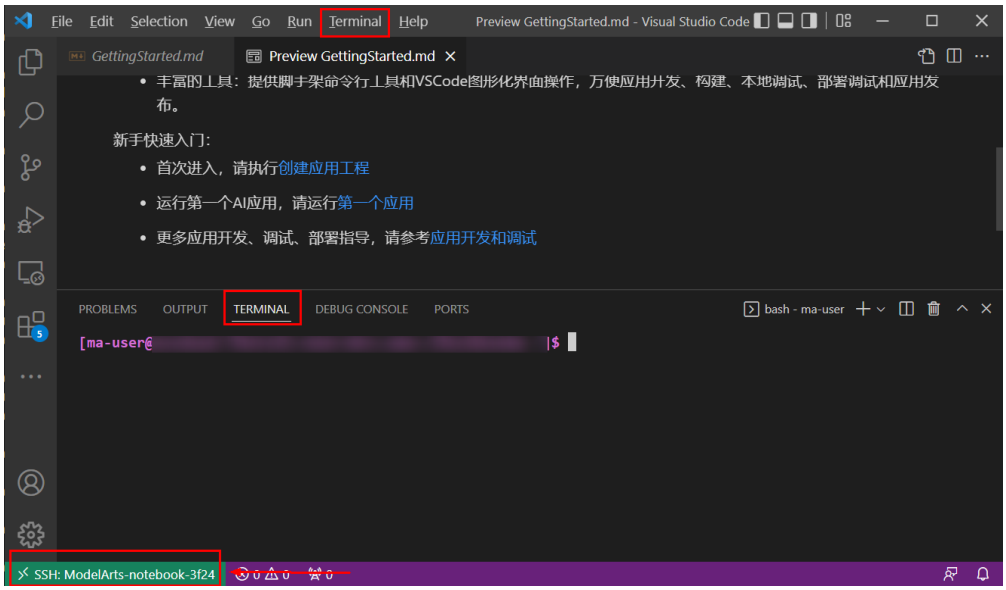
   输入python并回车，进入python环境。

   ```
   python
   ```

   在本地VS Code的Terminal中使用ModelArts SDK上传本地文件至OBS，详情请参考**文件传输**进行OBS传输操作。

2. 在远程连接VS Code的Terminal中使用ModelArts SDK下载OBS文件到开发环境的操作示例如下：

   图 6-74 远程连接 VS Code 环境开启 Terminal

版权所有 © 华为云计算技术有限公司

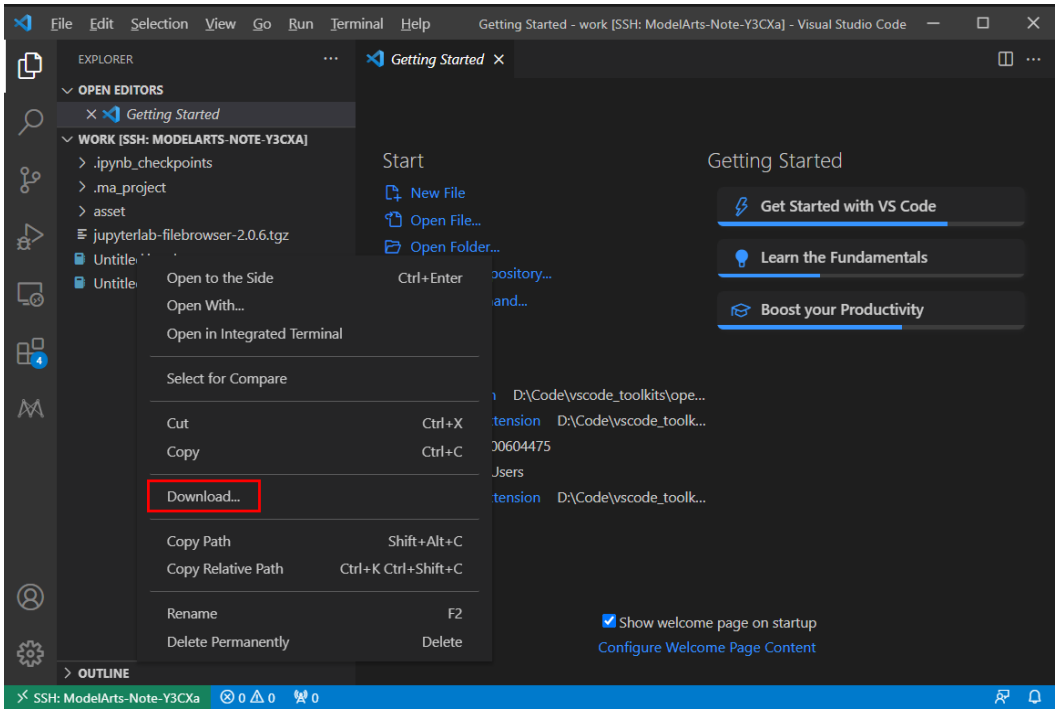
```
#手动source进入开发环境
cat /home/ma-user/README
#然后选择要source的环境
source /home/ma-user/miniconda3/bin/activate MindSpore-python3.7-aarch64
#输入python并回车，进入python环境
python
```

然后参考**文件传输**进行OBS传输操作。

下载 Notebook 中的文件至本地

在Notebook中开发的文件，可以下载至本地。在本地IDE的Project目录下的Notebook2.0工程单击右键，单击“Download...”将文件下载到本地。

图 6-75 VS Code 环境下载 Notebook 中的文件至本地



6.4 本地 IDE（SSH 工具连接）

本节操作介绍在Windows环境中使用PuTTY SSH远程登录云上Notebook实例的操作步骤。

前提条件

- 创建一个Notebook实例，并开启远程SSH开发，配置远程访问IP白名单。该实例状态必须处于“运行中”，具体参见**创建Notebook实例**章节。
- 在Notebook实例详情页面获取开发环境访问地址（例如：dev-modelarts-cnnorth4.huaweicloud.com）和端口号。

版权所有 © 华为云计算技术有限公司

图 6-76 Notebook 实例详情页面

- 准备好密钥对文件。

  密钥对在用户第一次创建时，自动下载，之后使用相同的密钥时不会再有下载界面（用户一定要保存好），或者每次都使用新的密钥对。

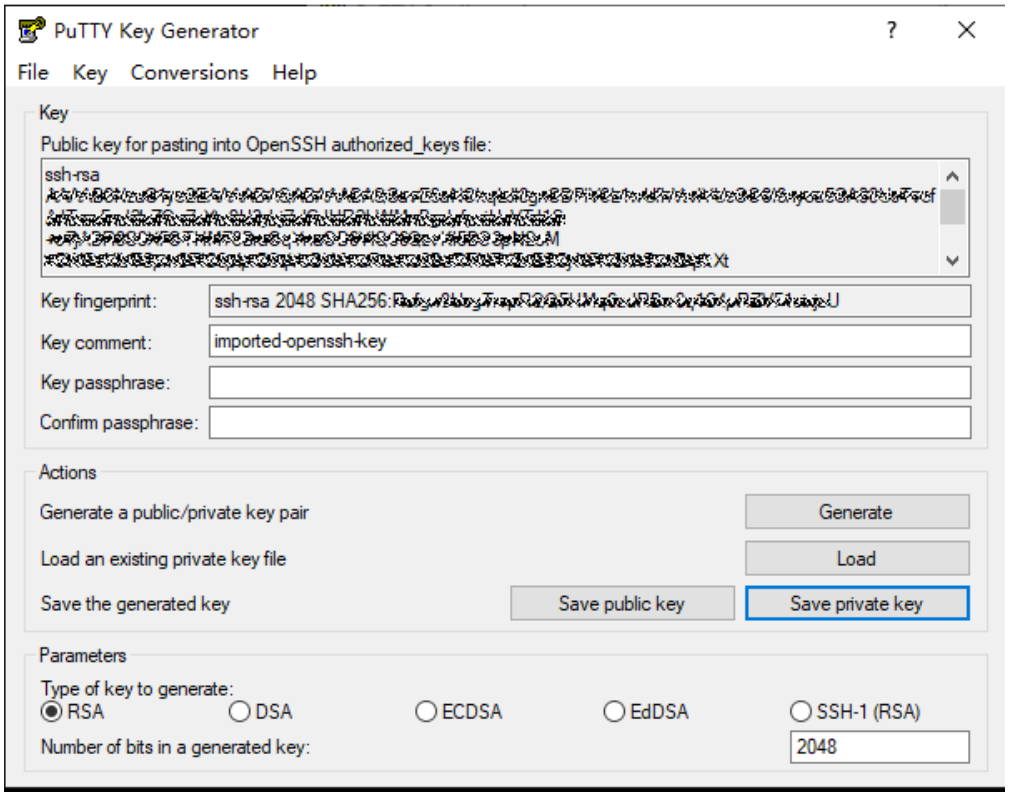
Step1 安装 SSH 工具

下载并安装SSH远程连接工具，以PuTTY为例，**下载链接**。

Step2 使用 puttygen 将密钥对.pem 文件转成.ppk 文件

1. **下载puttygen**，并双击运行puttygen。
2. 单击“Load”，上传.pem密钥（即在创建Notebook实例时创建并保存的密钥对文件）。
3. 单击“Save private key”，保存生成的.ppk文件。.ppk文件的名字可以自定义，例如key.ppk。

图 6-77 将密钥对.pem 文件转成.ppk 文件



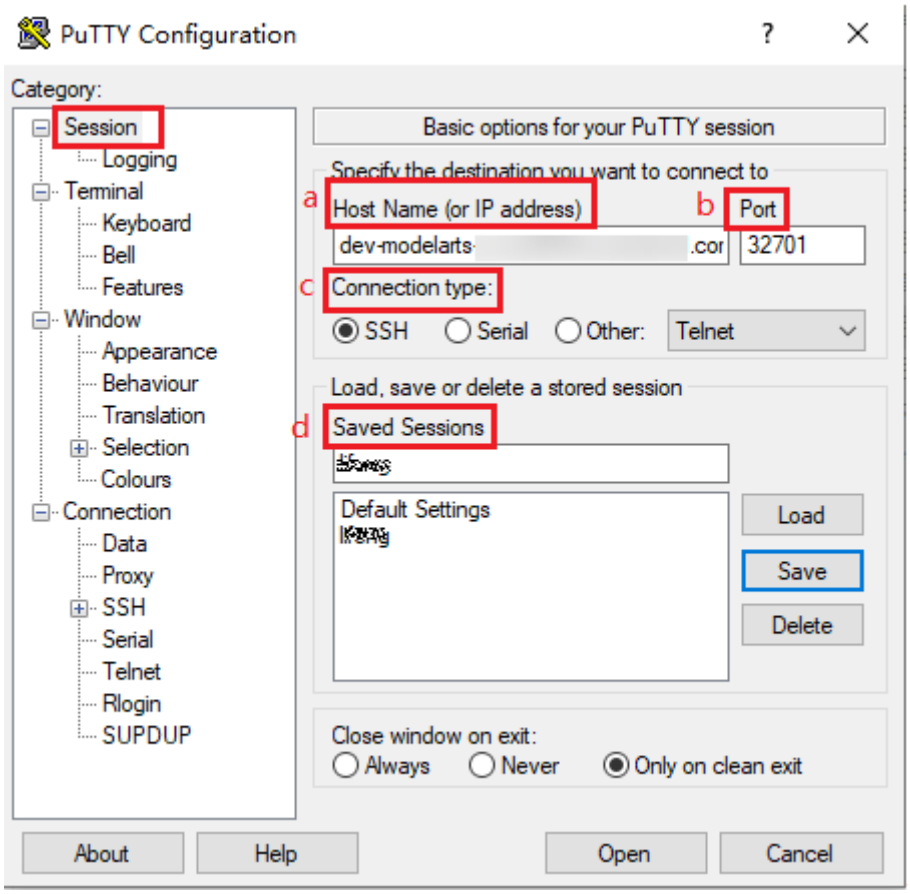
Step3 使用 SSH 工具连接云上 Notebook 实例

1. 运行PuTTY。

版权所有 © 华为云计算技术有限公司

2. 单击“Session”，填写以下参数。
   a. Host Name (or IP address)：云上开发环境Notebook实例的访问地址，即在Notebook实例详情页获取的地址。例如：dev-modelarts-cnnorth4.huaweicloud.com。
   b. Port：云上Notebook实例的端口，即在Notebook实例详情页获取的端口号。例如：32701。
   c. Connection Type：选择SSH。
   d. Saved Sessions：任务名称，在下一次使用PuTTY时就可以单击保存的任务名称，即可打开远程连接。

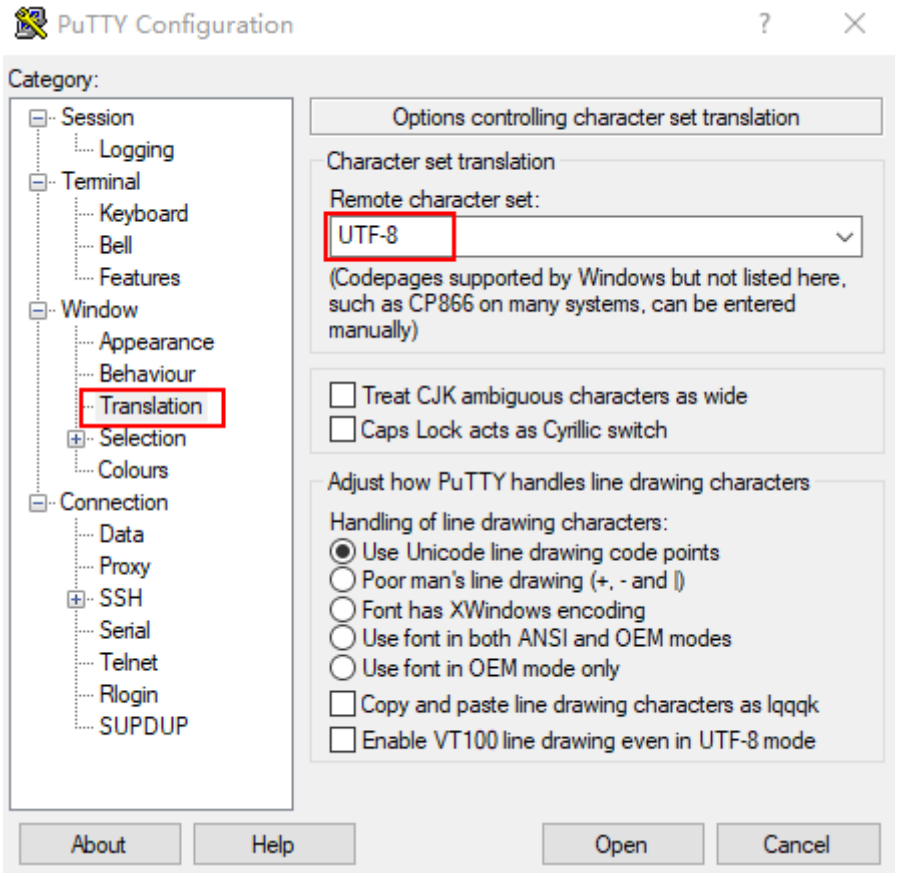
图 6-78 设置 Session



3. 选择“Window > Translation”，在“Remote character set:”中选择“UTF-8”。

版权所有 © 华为云计算技术有限公司

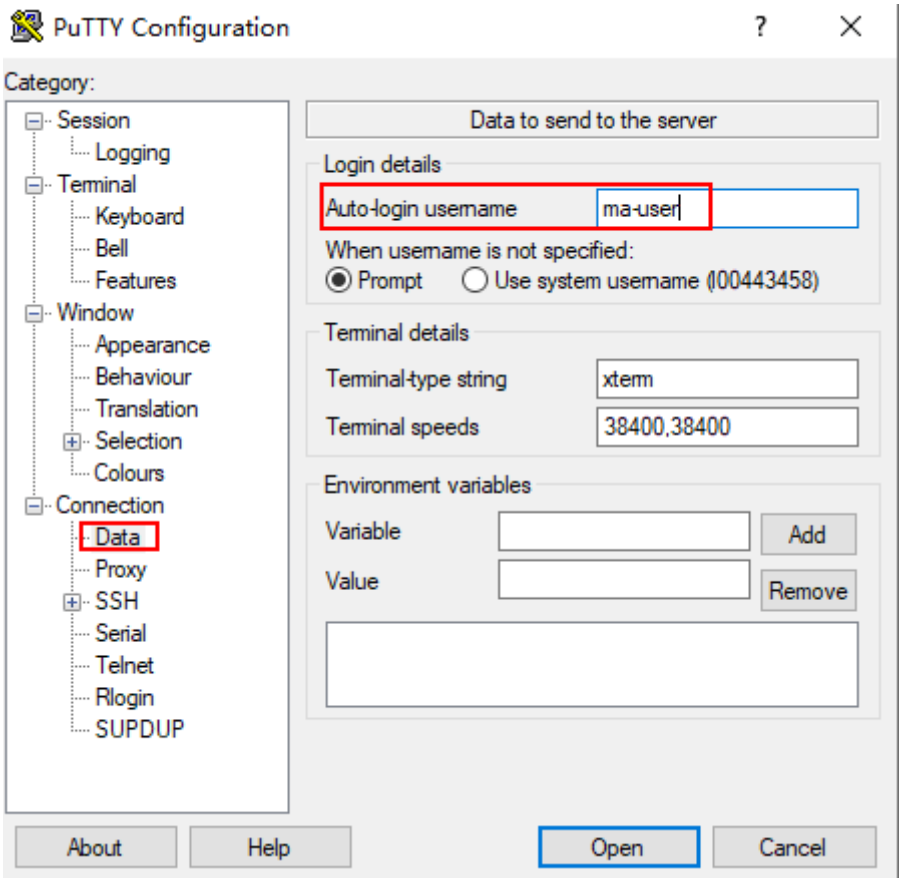
图 6-79 设置字符格式



4. 选择“Connection > Data”， 在“Auto-login username” 中填写用户名“ma-user”。

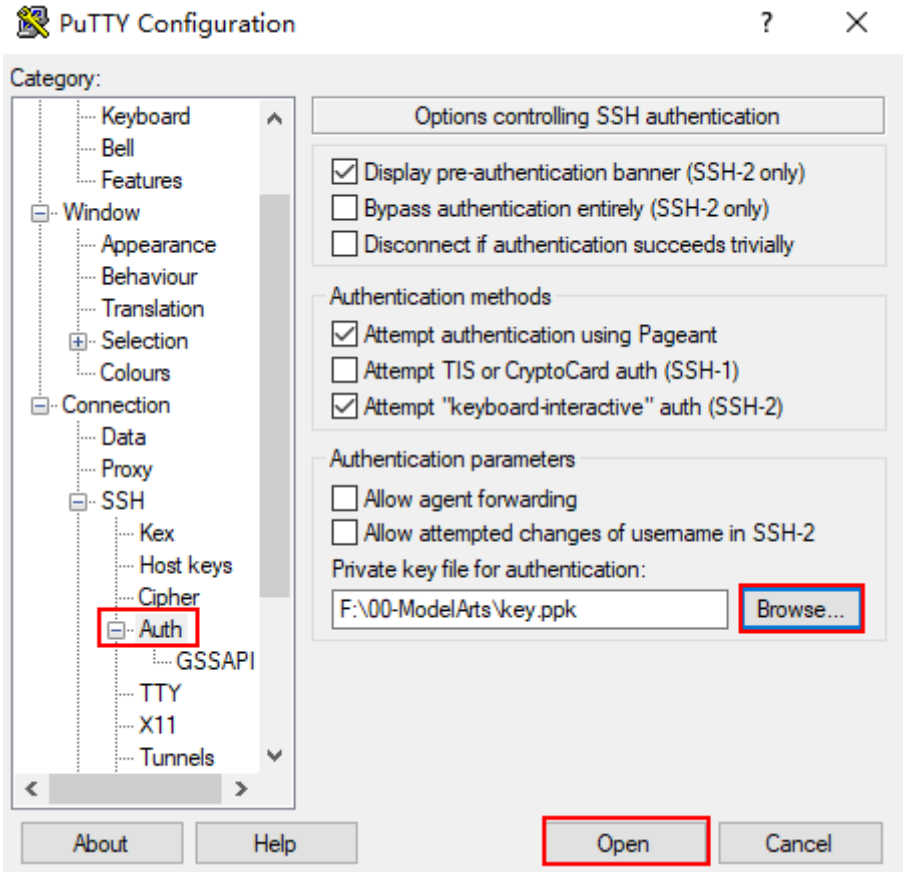
 版权所有 © 华为云计算技术有限公司

图 6-80 填写用户名



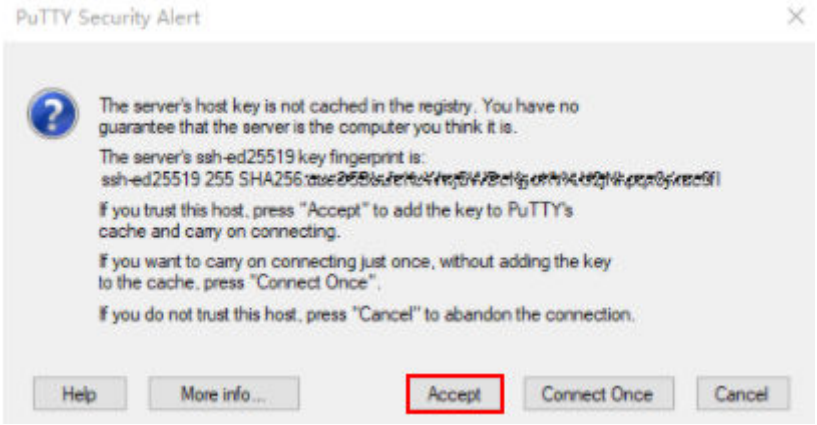
5. 选择“Connection > SSH > Auth”，单击“Browse”，选择“.ppk文件”（由**Step2**密钥对.pem文件生成）。

版权所有 © 华为云计算技术有限公司



6. 单击“Open”。如果首次登录，PuTTY会显示安全警告对话框，询问是否接受服务器的安全证书。单击“Accept”将证书保存到本地注册表中。

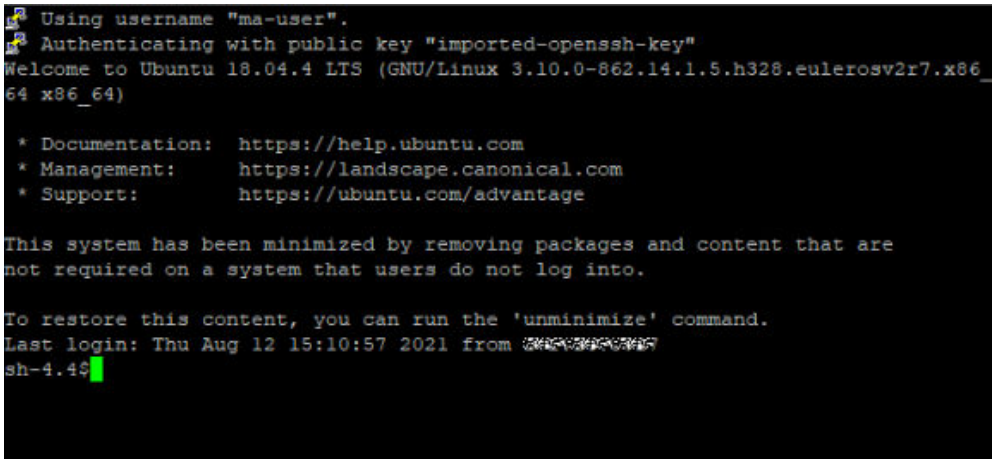
图 6-81 询问是否接受服务器的安全证书



7. 成功连接到云上Notebook实例。

 版权所有 © 华为云计算技术有限公司

图 6-82 连接到云上 Notebook 实例



版权所有 © 华为云计算技术有限公司

7 ML Studio

7.1 ML Studio 简介

ML Studio 是什么

ML Studio简称MLS，是ModelArts中的一个支持可视化机器学习建模的企业级AI开发工具，支持用户通过浏览器以全代码、少代码甚至零代码的方式开发AI模型。

MLS提供了图形化模型探索开发环境、丰富的预置算子和预置算链，并支持编写自定义算子，可帮助开发者快速构建具有实用价值的机器学习应用。

MLS为AI开发者提供可视化的操作界面来编排机器学习模型的训练、评估和预测的过程，无缝衔接数据分析和预测应用，为用户的数据挖掘分析业务提供易用、高效、高性能的工具。

了解概念

- 算子

  在MLS中，算子是一种基本功能单元，以ipynb格式保存，实质上是一段代码，对应Notebook中的一个Cell。

  在算子中，开发者可以通过代码的形式封装一系列变量、方法或类，从而实现一个独立功能。

  一个算子可以是一个机器学习算法的调用封装（比如调用Spark API封装一个随机森林算法），也可以是一个简单运算逻辑（比如完成一次简单加减运算），还可以是自行实现的任意功能脚本（比如从磁盘上读取数据）。

- 算链

  算链是由若干算子通过一定规则组合而成的复合功能单元，以JSON格式进行保存，是一个大功能或解决方案的载体。

  算链中的算子之间可以通过有向无环图（DAG）的形式组合，也可不与任何算子组合，一个算链可包含若干个DAG或零散算子。

  在运行过程中，通过DAG形式组合的算子将严格按照DAG顺序调度运行，而未按DAG形式组合的算子则按照添加至算链的先后顺序运行。

  MLS中的一个算链可转换成一个ipynb文件或一个python文件，开发者可基于转换的文件做进一步开发。

 版权所有 © 华为云计算技术有限公司

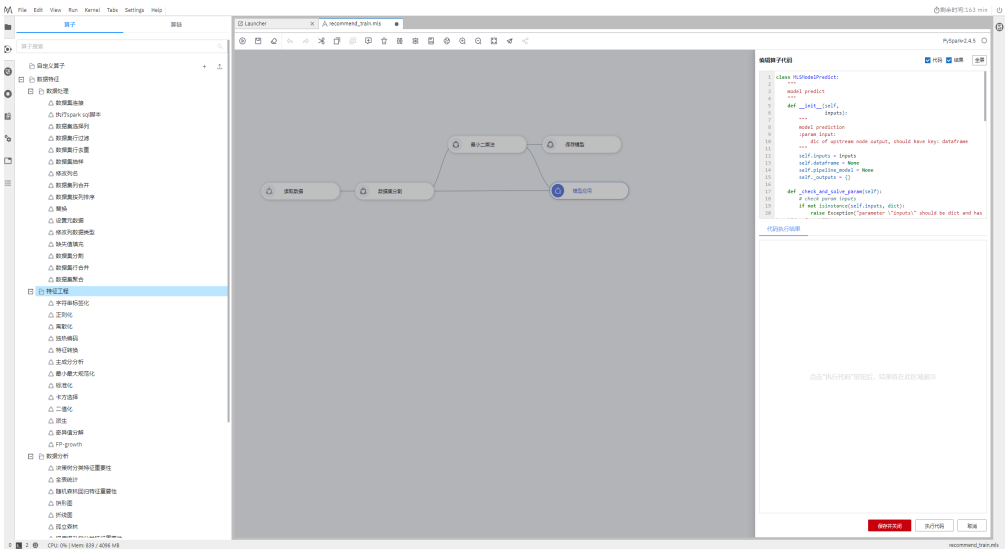
亮点特性 1：可视化建模

MLS提供了用户友好的可视化模型探索或开发环境，开发者只需要通过简单拖拉拽操作编排算子，构建算链即可完成机器学习建模。

MLS中一个算链由一组算子组成，每个算子是一个功能独立的逻辑单元，算子通过有向无环图（DAG）的形式组织。

在MLS中，开发者可在算链编排界面上直观便利地进行算子增删、算子连线、算子参数设置、算子代码修改及算子结果预览等操作，支持开发者以REPL（Read Eval Print Loop）交互式开发方式进行模型调测。

图 7-1 ML Studio



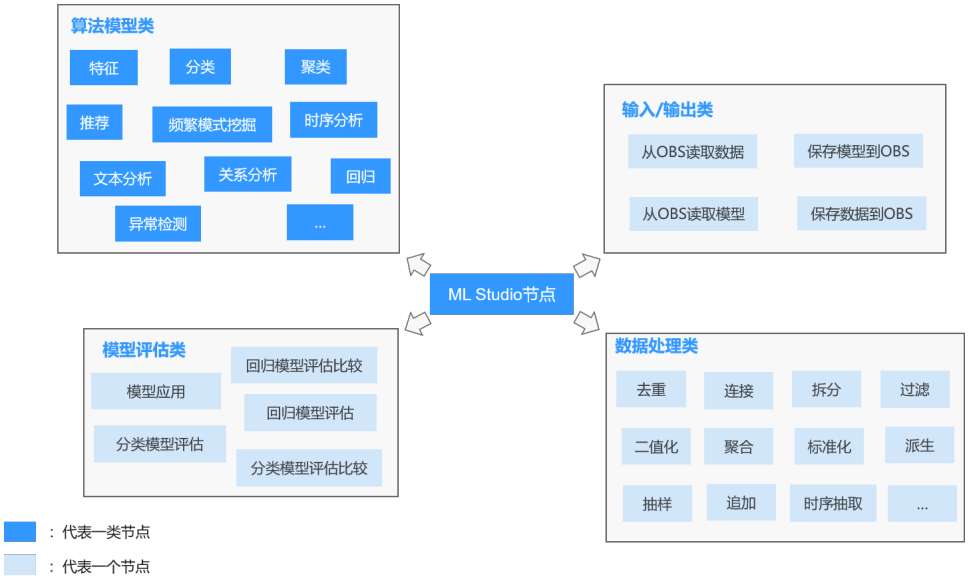
亮点特性 2：丰富的预置算子

MLS提供了丰富的预置算子，覆盖了机器学习建模全流程，包含数据分析、数据处理、特征工程、模型构建、模型评估和模型应用等多种算子类型，可极大程度地增强算法代码的可复用性，减少开发者的模型构建成本并提升开发效率。

开发者可以根据实际业务需要，方便快捷地设置预置算子参数、查看和修改预置算子源码，通过在算链中对预置算子进行参数调整和代码调整构建独特的业务场景需要的AI算法。

版权所有 © 华为云计算技术有限公司

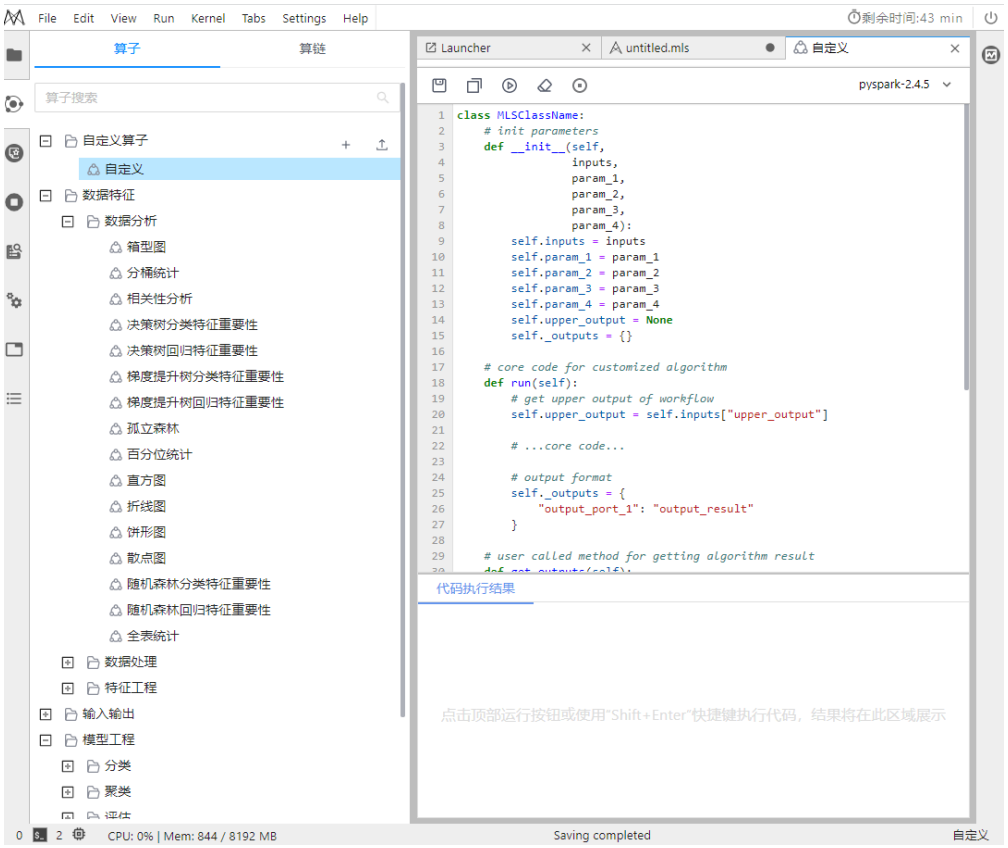
图 7-2 丰富的预置算子



亮点特性 3：提供高度开放的自定义算子开发环境

MLS提供了高度开放的自定义算子开发环境，开发者可以用自己习惯的方式编写MLS算子并拖拽至画布，构建算链完成模型构建。MLS支持全新编写自定义算子、上传自定义算子和基于预置算子开发自定义算子等方式构建贴合业务需求的个性化算子，打通业务开发全流程。

图 7-3 自定义算子开发



版权所有 © 华为云计算技术有限公司

使用方式

在Notebook开发环境中创建基于MLS引擎的Notebook实例，在JupyterLab页面打开并使用。

说明

MLStudio使用要求浏览器为Chrome，版本为81及以上。

7.2 进入 ML Studio 操作界面

在使用ML Studio开发模型前，您需要创建基于ML Studio引擎的Notebook实例，并通过在线JupyterLab页面打开MLS可视化操作界面。

背景信息

- 创建和使用Notebook需要消耗资源，需要收费。根据您选择的资源不同，收费标准不同，针对不同类型资源的价格，详情请参见**产品价格详情**。
- “运行中”的Notebook将一直收费，当您不需要使用时，建议停止Notebook，避免产生不必要的费用。在创建Notebook时，也可以选择开启自动停止功能，在指定时间内停止运行Notebook，避免产生不必要的费用。
- 只有处于“运行中”状态的Notebook，才可以执行打开、停止、删除和保存镜像操作。

Step1 创建基于 ML Studio 引擎的 Notebook 实例

1. 登录ModelArts管理控制台，在左侧全局配置中，检查是否配置了访问授权。如果未配置，请参考**使用委托授权**完成操作。
2. 登录ModelArts管理控制台，在左侧菜单栏中选择“开发环境>Notebook”，进入“Notebook”页面。
3. 单击右侧“创建”进入“创建Notebook”页面，请参见如下说明填写参数。
   a. 填写Notebook基本信息，包含名称、描述、是否自动停止，详细参数请参见**表7-1**。

      表 7-1 基本信息的参数描述

参数名称	说明
“名称”	Notebook的名称。只能包含数字、大小写字母、下划线和中划线，长度不能超过20位且不能为空。
“描述”	对Notebook的简要描述。
“自动停止”	默认开启，且默认值为“1小时”，表示该Notebook实例将在运行1小时之后自动停止，即1小时后停止计费。 开启自动停止功能后，可选择“1小时”、“2小时”、“4小时”、“6小时”或“自定义”几种模式。选择“自定义”模式时，可指定1~24小时范围内任意整数。

   b. 填写Notebook详细参数，如镜像、资源规格等，详细参数请参见**表7-2**。

版权所有 © 华为云计算技术有限公司

图 7-4 Notebook 实例参数-镜像

图 7-5 Notebook 实例参数-资源规格

表 7-2 Notebook 实例的详细参数说明

参数名称	说明
“镜像”	支持公共镜像和自定义镜像。 • 公共镜像：即预置在ModelArts内部的AI框架，“mlstudio-pyspark2.3.2-ubuntu16.04”或“mlstudio-pyspark2.4.5-ubuntu18.04”。 • 自定义镜像：可以将基于公共镜像创建的实例保存下来，作为自定义镜像使用。 一个镜像对应支持一种AI引擎，创建Notebook实例时选择好了对应AI引擎的镜像。用户可以根据需要选择镜像。 不可以在同一个Notebook实例中切换AI引擎。
“资源池”	“公共资源池”无需单独购买，即开即用，按需付费，即按您的Notebook实例运行时长进行收费。 “专属资源池”需要单独购买并创建。

版权所有 © 华为云计算技术有限公司

参数名称	说明
“类型”	芯片类型主要是CPU。
“规格”	可选资源规格。 CPU规格可选：“2核8GB”、“8核32GB”
“存储配置”	默认为“云硬盘EVS”。 ● 磁盘规格默认5GB。您可以根据实际使用量设置磁盘规格。磁盘规格的取值范围为5GB～4096GB。 从Notebook实例创建成功开始，直至实例删除成功，磁盘每GB按照规定费用收费。 ● “云硬盘EVS”的存储路径挂载在/home/ma-user/work目录下。用户在Notebook实例中的所有文件读写操作都是针对该存储目录下的内容操作，与OBS无关。停止或重启Notebook实例时，内容会被保留，不丢失。 删除Notebook实例时，内容不保留。
“SSH远程开发”	不支持。

4. 参数填写完成后，单击“立即创建”，确认规格参数。
5. 参数确认无误后，单击“提交”，完成Notebook的创建操作。

   进入Notebook列表，正在创建中的Notebook状态为“创建中”，创建过程需要几分钟，请耐心等待。当Notebook状态变为“运行中”时，表示Notebook已创建并启动完成。

Step2 打开 Notebook 实例，并进入 MLS 可视化操作界面

1. 在基于MLS引擎的Notebook实例右侧，单击操作栏的“打开”，进入Jupyterlab页面。

   图 7-6 打开 Notebook 实例

2. 在JupyterLab页面下方，单击MLS Editor图标，在弹出的“Select Kernel”对话框中选择“PySpark-2.4.5”，单击“Select”，进入一个空的算链页面。

版权所有 © 华为云计算技术有限公司

图 7-7 MLS Editor

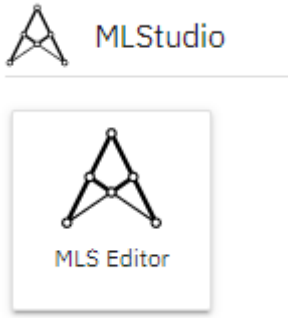


图 7-8 选择 Kernel

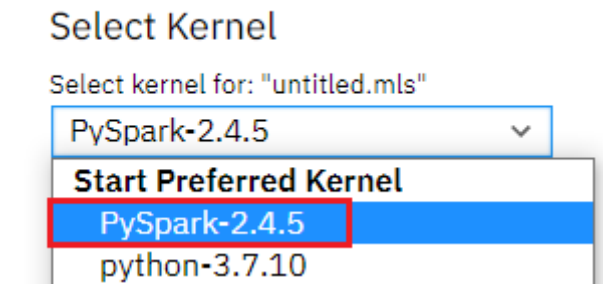
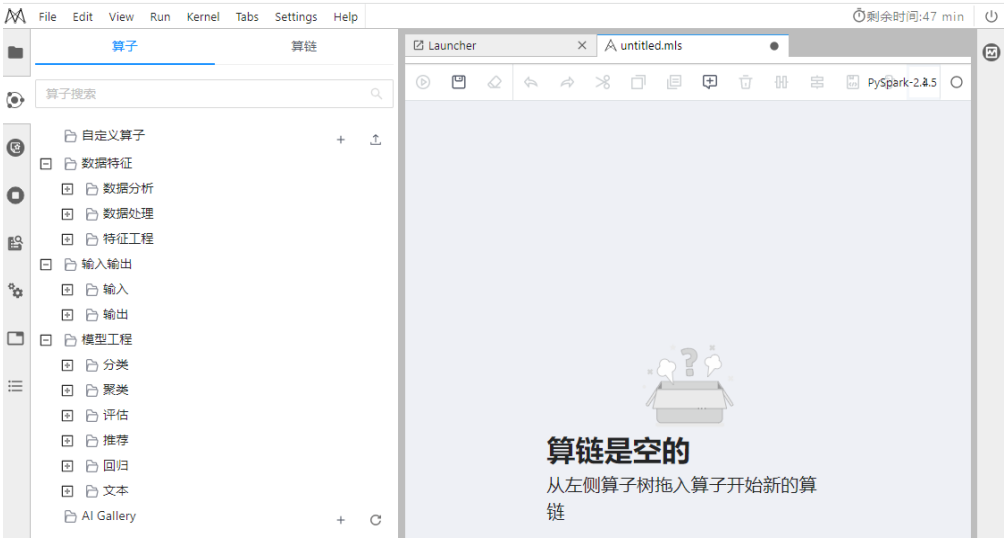


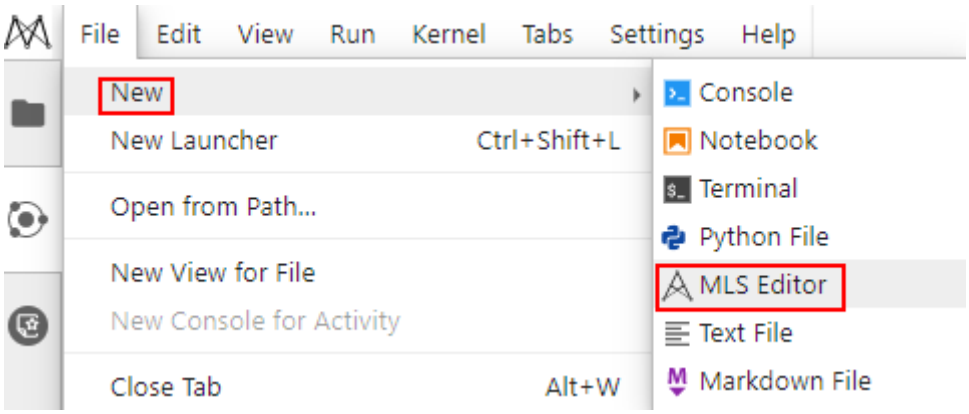
图 7-9 MLS Editor 可视化操作界面



或者单击JupyterLab导航栏的“File >New >MLS Editor”，也可以进入MLS Editor界面。

版权所有 © 华为云计算技术有限公司

图 7-10 JupyterLab 导航栏新建 MLS Editor

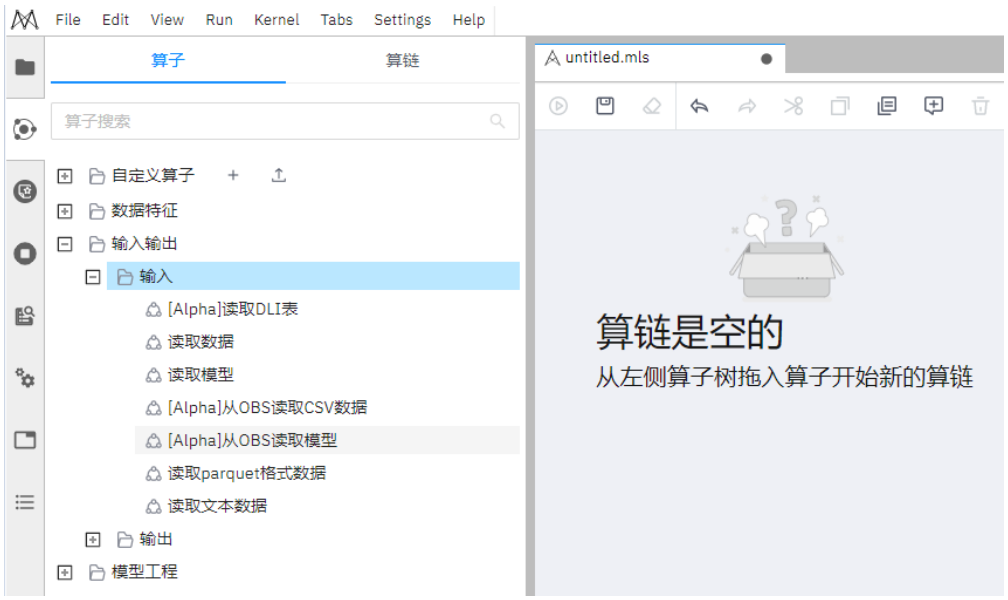


Step3 使用 MLS 操作界面

此时，已经进入了MLS的可视化编辑页面MLS Editor，可以通过拖拉拽的方式，完成模型开发。此处介绍MLS操作界面。

MLS由资产管理和资产编排两个功能模块组成，完整界面如图7-11所示，左边是资产管理界面，右边是资产编排界面。

图 7-11 MLS 资产管理和资产编排界面



其中资产管理模块主要用于管理资产，包括算子、算链两类资产。可以通过点击左侧导航条上的图标进入.

资产编排模块用于编排资产，可以通过Launcher界面的“MLS Editor”图标进入，也可通过资产管理模块中的算链子模块打开或新建算链。

资产编排过程中，开发者可以从左侧资产管理界面，拖拽预置算子或自定义算子至右侧资产编排界面，将算子连接成一个或若干个有向无环图DAG，从而构建出一个完整算链。

版权所有 © 华为云计算技术有限公司

此外，开发者也可以从左侧资产管理界面的“算链”选项卡中打开预置算链进行编排与修改，形成一个新的算链。

编排完成后，开发者可以点击算链编排界面顶部的图标一键运行，进行机器学习模型训练与预测。

- 快速上手使用MLS进行模型开发，可以参考**ML Studio快速入门**。
- MLS镜像中预置了大量的算子方便您使用，预置算子的详细说明请参考**预置算子说明**。
- 您也可以在MLS Editor中编写自己的算子，具体操作请参考**编写自定义算子**。
- MLS同时预置了部分算链，方便用户使用，用户也可以编写自己的算链，相关的详细操作可以参考**算链操作**。

7.3 ML Studio 快速入门

7.3.1 背景信息

本章提供了2个快速入门教程，通过一个餐厅经营销售量预测的算链建模示例，帮助开发者快速了解MLS的基本能力。

- 如果您想快速了解MLS的建模过程，您可以参考**使用MLS预置算链进行机器学习建模**章节，一键运行预置算链完成建模。
- 如果您了解如何从0到1在MLS上新建1条算链并完成建模，您可以参考**从0到1利用ML Studio进行机器学习建模**章节。该教程可以帮助您全面了解ML Studio的预置算子、拖拉拽建模、算链编辑、算链运行及结果查看等常用功能，快速掌握基于ML Studio的机器学习建模开发方法。

场景介绍

影响餐厅销售的因素多种多样，如果只是单纯地靠人为推算餐厅销售量，容易出现偏差。

现在，可以使用ModelArts服务中的ML Studio提供的销售预测模板，省时省力地得到餐厅未来3个月内的销售预测结果。

作为餐厅经营人员，可根据预测结果更好地判断在新地段开设哪种类型餐厅，并把预测出来的销售量较高时间段（例如每年5~7月是餐厅旺季）作为餐厅核心经营周期，从而缩短资金投入周期，提高餐厅纯利润收入。

准备工作

- 已注册华为账号并开通华为云，且在使用ModelArts前检查账号状态，账号不能处于欠费或冻结状态。
- 已经创建基于MLStudio引擎的Notebook实例开发环境，并且处于“运行中”状态。具体操作请参见**进入ML Studio操作界面**。

准备数据

本章所用示例的数据已预置在/home/ma-user/work/.ml-workspace/built-in-workflow/sales_forecast数据文件夹中，无须上传即可直接使用。如果需要在算链中使用其它数据，则需要在华为云OBS中创建桶并上传数据。

版权所有 © 华为云计算技术有限公司

本示例使用数据集基本信息可参见表7-3和表7-4。更多详细信息可见网址：https://www.kaggle.com/c/restaurant-revenue-prediction/data

表 7-3 数据源的具体字段

字段名	含义	类型	描述
Id	餐厅id	IntegerType	餐厅标识
OpenDate	开业时间	DateType	餐厅开业时间
City	城市名	StringType	餐厅所在城市
CityGroup	城市类型	StringType	餐厅所在城市类型（大型、其他）
Type	类型	StringType	餐厅类型（FC、IL、DT、MB） • FC：Food Court • IL：Inline • DT：Drive Thru • MB：Mobile
P1,P2 ~ P37	其他信息	NumericType	人口、房地产、商业数据等（1~5评分）
revenue	金额	NumericType	餐厅营业额

表 7-4 数据集的部分样本数据

Id	Open Date	City	CityGroup	Type	P1	P2	…	P36	P37	revenue
0	07/17/1999	istanbul	Big Cities	IL	4	5	…	3	4	5653753
1	02/14/2008	Ankara	Big Cities	FC	4	5	…	0	0	6923131
2	03/09/2013	Diyarbakir	Other	IL	2	4	…	0	0	2055379
3	02/02/2012	Tokat	Other	IL	6	4.5	…	12	6	2675511
4	05/09/2009	Gaziantep	Other	IL	3	4	…	3	3	4316715
5	02/12/2010	Ankara	Big Cities	FC	6	6	…	0	0	5017319

版权所有 © 华为云计算技术有限公司

7.3.2 使用 MLS 预置算链进行机器学习建模

本章节介绍如何通过一键运行预置的餐厅经营销售量预测算链，完成建模，帮助开发者快速了解MLS的建模过程。

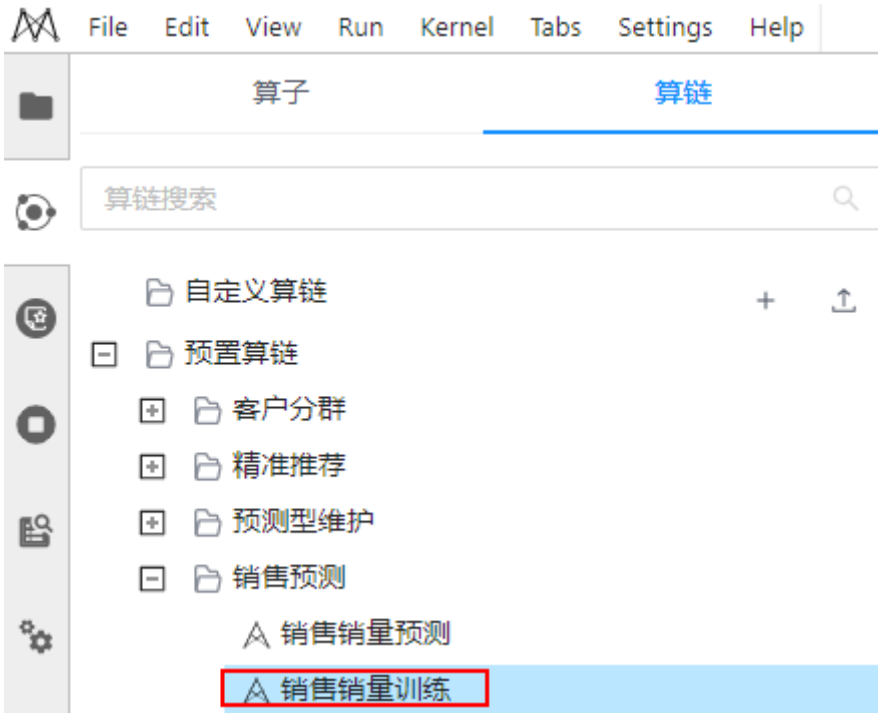
前提条件

已经创建一个基于MLStudio的Notebook镜像，并进入MLS Editor可视化编辑界面，具体参考**进入ML Studio操作界面**章节。

Step1 运行预置算链

1. 单击资产浏览图标，选择“算链”，单击展开，找到预置算链“销售销量训练”，如**图7-12**所示。

   图 7-12 预置算链

2. 双击打开销售销量训练，并选择Kernel PySpark-2.4.5。
3. 保存模型结点将训练完成的模型保存到本地默认位置，用于进行销售销量预测。您也可以右键该节点选择“参数设置”，如**图7-13**所示。在页面右侧自行设置模型路径，指定存储位置，如**图7-14**所示为默认存储路径“./output_model/sales_model”。

版权所有 © 华为云计算技术有限公司

图 7-13 右键设置参数

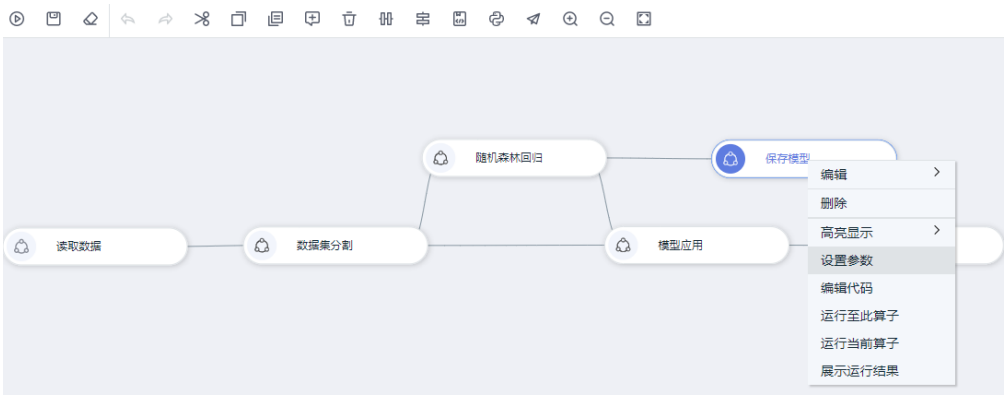
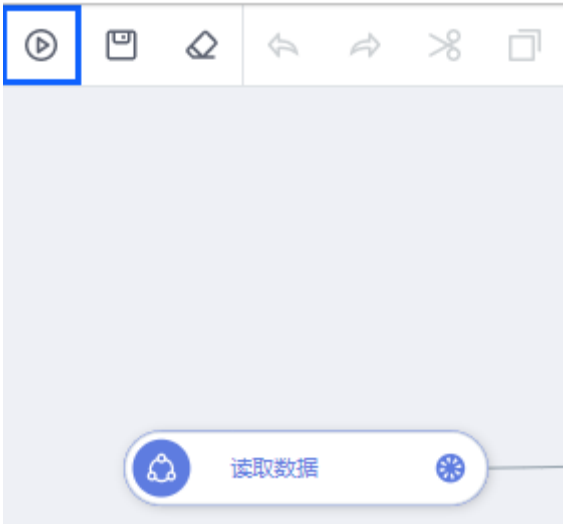


图 7-14 设置保存模型参数

4. 单击导航栏运行算链，如图7-15所示。运行过程需要几分钟，请耐心等待。当所有节点都变为绿色，表示算链运行成功，如图7-16所示。

图 7-15 单击运行



版权所有 © 华为云计算技术有限公司

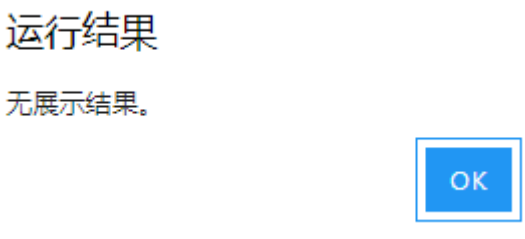
图 7-16 训练算链运行成功

5. 当算链运行完毕后，选中任意节点，右键选择“展示运行结果”，查看该节点的运行结果，如图7-17所示。

   如果无运行结果，如图7-18所示；如果有运行结果，如图7-19所示，例如模型应用节点和回归评估节点。

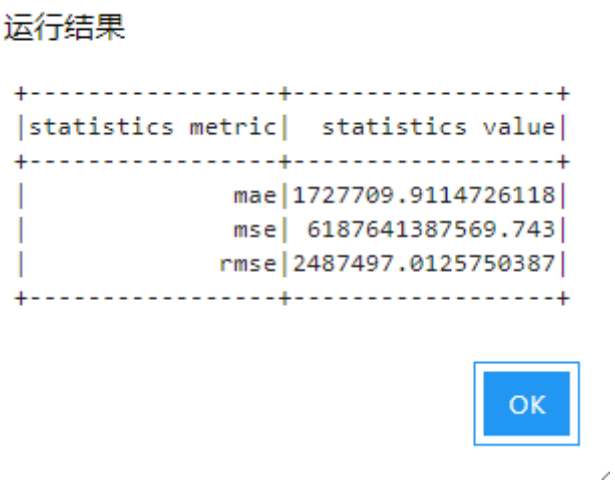
图 7-17 右键选择展示运行结果

图 7-18 无运行结果



版权所有 © 华为云计算技术有限公司

图 7-19 有运行结果

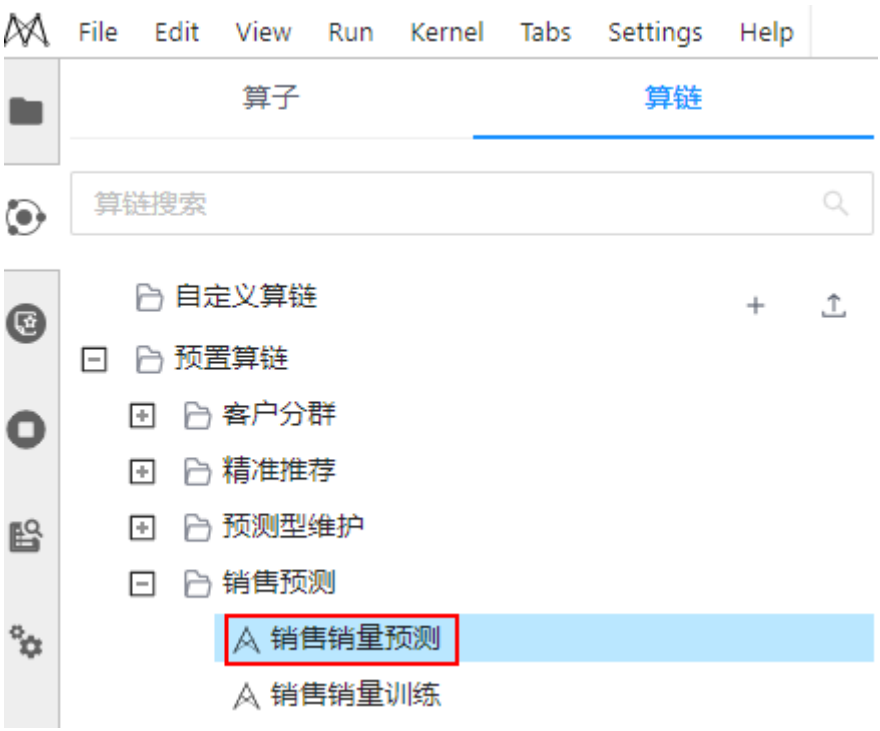


Step2 使用模型进行预测

模型建立完成后，使用已经保存好的模型和餐厅预测数据，可以预测销售额。

1. 在算链页签的预置算链目录下， 双击打开销售销量预测， 如图7-20所示。

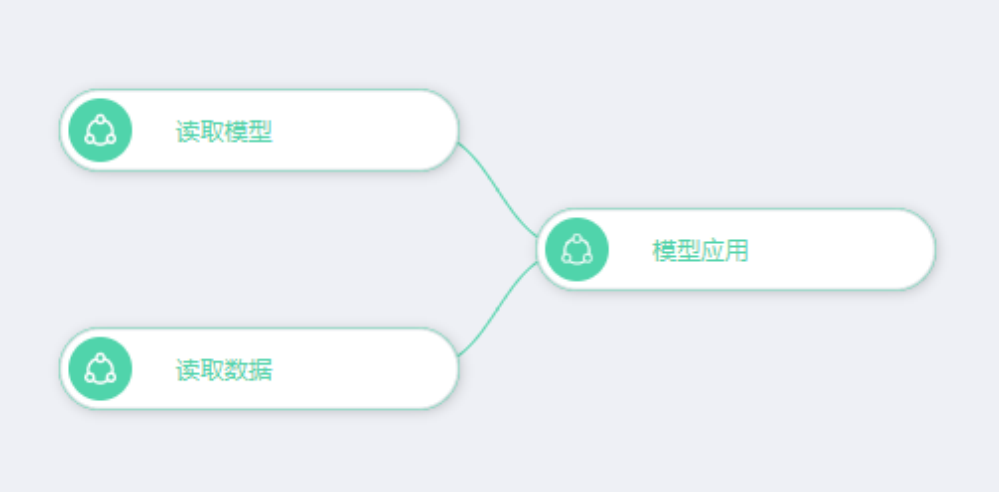
图 7-20 销售销量预测



2. 单击导航栏运行算链。运行过程需要几分钟，请耐心等待。当所有节点都变为绿色，表示算链运行成功，如图7-21所示。

版权所有 © 华为云计算技术有限公司

图 7-21 预测算链运行成功



3. 右键单击“模型应用”节点，选择“展示运行结果”，查看预测结果。

图 7-22 查看预测结果

7.3.3 从 0 到 1 利用 ML Studio 进行机器学习建模

本章节基于餐厅销量预测场景，从零开始介绍如何制作销售销量训练及销售销量预测两个算链。

前提条件

已经创建一个基于ML Studio的Notebook镜像，并进入MLS Editor可视化编辑界面，具体参考**进入ML Studio操作界面**章节。

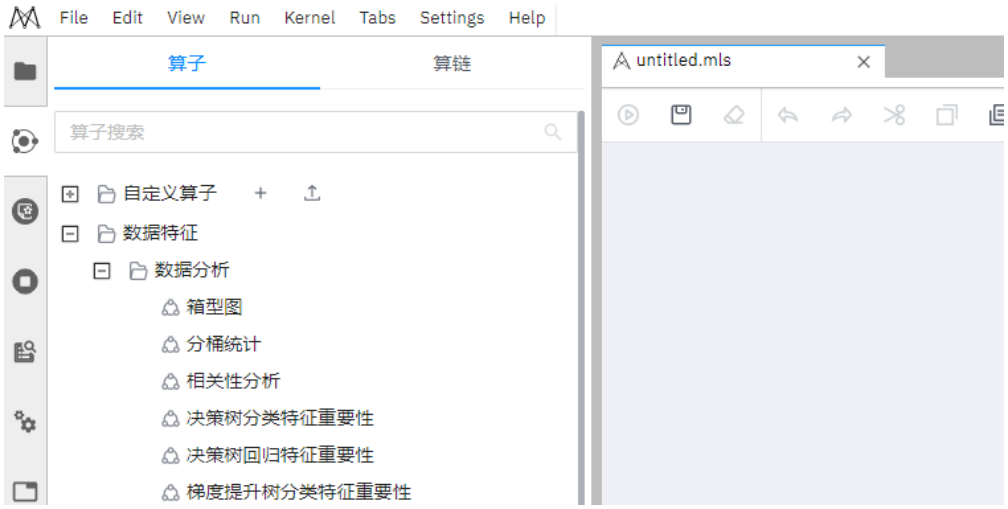
Step1 创建一个空算链

单击Launcher界面的MLS Editor，选择名为PySpark-2.4.5的Kernel，创建一个空的算链。

创建算链后，左侧界面自动跳转到资产预览界面。

版权所有 © 华为云计算技术有限公司

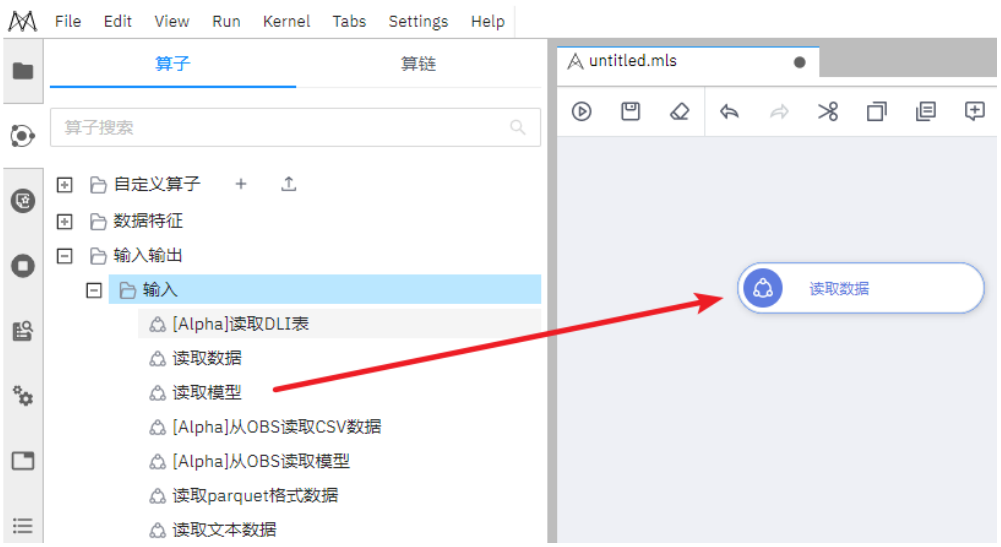
图 7-23 算链创建成功



Step2 使用 ML Studio 建模

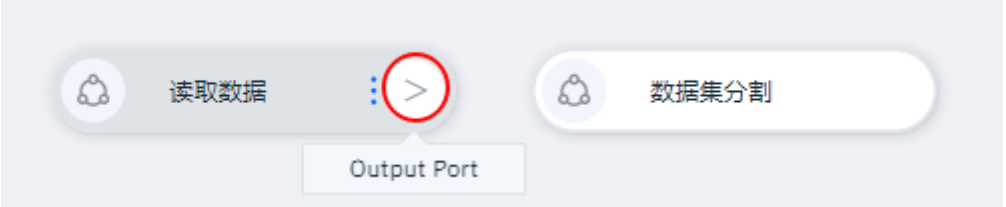
1. 从左侧资产浏览界面拖拽预置算子或自定义算子至右侧算链编辑界面，如图7-24所示，则创建算子成功。

图 7-24 拖拽创建结点



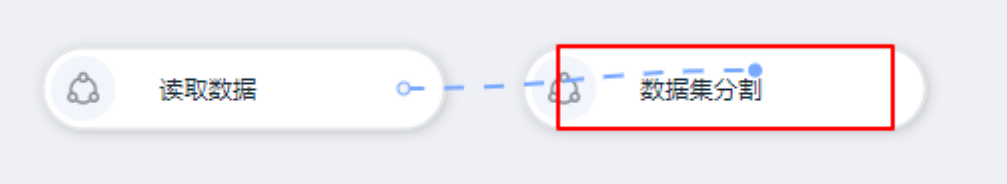
2. 在画布中，鼠标移至算子结点，从右侧输出端口，如图7-25所示，拖动连线至下一个算子结点，鼠标尽量放置至如图4 连线结束位置所示红框位置。

图 7-25 从输出端口移动至下一结点



版权所有 © 华为云计算技术有限公司

图 7-26 连线结束位置



3. 进行算子连线。

   算子之间具有数据的流入流出关系，如果源算子与目标算子的输出输入端口数量都为1，则直接连线，如图7-26所示。

4. 鼠标右键单击读取数据算子，选择“设置参数”，如图7-27所示在右侧滑出的参数设置窗口填写输入路径， 例如“/home/ma-user/work/.ml-workspace/built-in-workflow/sales_forecast/sales_train.csv”，表示读取文件为该路径下的“sales_train.csv”。

图 7-27 读取数据参数设置

5. 如果源算子和目标算子其中一个及以上具有多个输出输入端口，连线时需选择输入输出端口，如图7-28所示。

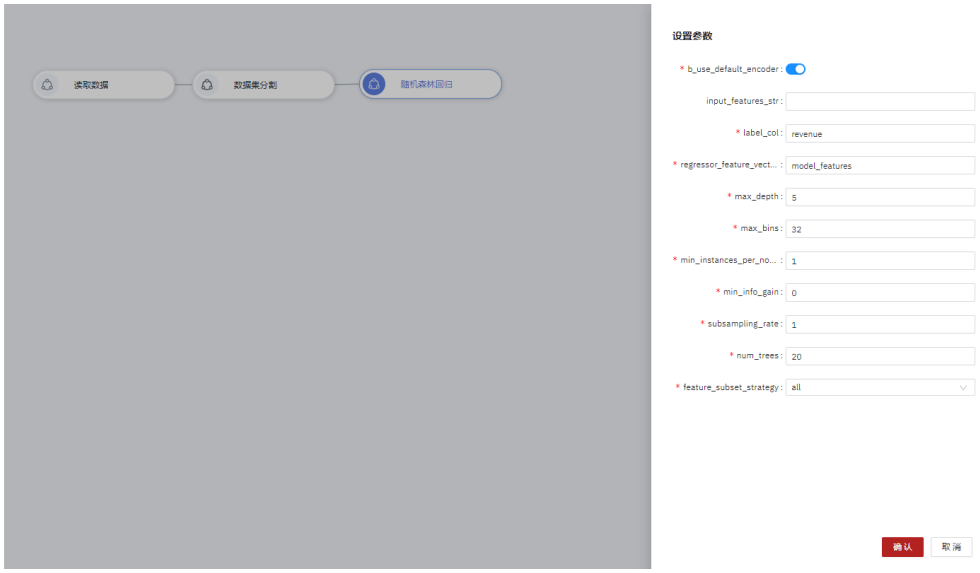
   数据集分割算子连线随机森林回归算子，数据集分割算子具有输出端口dataframe_1和dataframe_2，单击下拉框选择dataframe1为输出端口，随机森林回归算子只有输入端口dataframe，该步操作将数据dataframe1传入随机森林回归算子作为训练数据。

图 7-28 数据集分割连线随机森林回归

6. 右键单击随机森林回归算子，选择“设置参数”，在滑出的参数设置窗口填写标签列为“revenue”，如图7-29所示。

版权所有 © 华为云计算技术有限公司

图 7-29 随机森立回归参数设置



7. 如图7-30所示，随机森林回归连线模型应用，随机森林回归算子输出pipeline_model传入模型应用算子，作为模型应用算子的输入模型。

图 7-30 随机森林回归连线模型应用

8. 模型应用算子的dataframe由数据集分割算子的dataframe_2输入，如图7-31所示。

版权所有 © 华为云计算技术有限公司

图 7-31 数据集分割连线模型应用

9. 添加回归评估算子作为评估算子，将其与模型应用连线，右键选择设置参数，填写标签列为“revenue”，如图7-32所示。

图 7-32 回归评估参数设置

10. 最后添加保存模型算子，将其与随机森林回归算子连线，右键该算子选择参数设置，如图7-33所示。填写模型保存路径（文件夹级）"./output/SalesForecast"，表示输出模型保存到根目录下output/SalesForecast文件下。

图 7-33 保存模型参数设置

11. 算链创建完成，单击运行，耐心等待几分钟，运行成功，如图7-34所示。

版权所有 © 华为云计算技术有限公司

图 7-34 算链运行成功

如果运行失败，双击失败算子或者右键该算子选择编辑代码，如图7-35所示。在编辑算子代码界面可修改代码进行调试，如图7-36所示 。

图 7-35 右键选择编辑代码

图 7-36 编辑代码

 版权所有 © 华为云计算技术有限公司

Step3 使用 ML Studio 预测

1. 新建一个预测算链。
2. 拖拽读取模型算子至画布，设置“input_model_path”，为预测算链中保存模型路径，例如“./output/SalesForecast”。
3. 拉取读取数据算子，设置“input_file_path”，为测试数据的文件路径，如“/home/ma-user/work/.ml-workspace/built-in-workflow/sales_forecast/sales_predict.csv”。
4. 拉取模型应用算子，分别连接读取模型算子和读取数据算子，端口选择分别如图7-37和如图7-38所示。

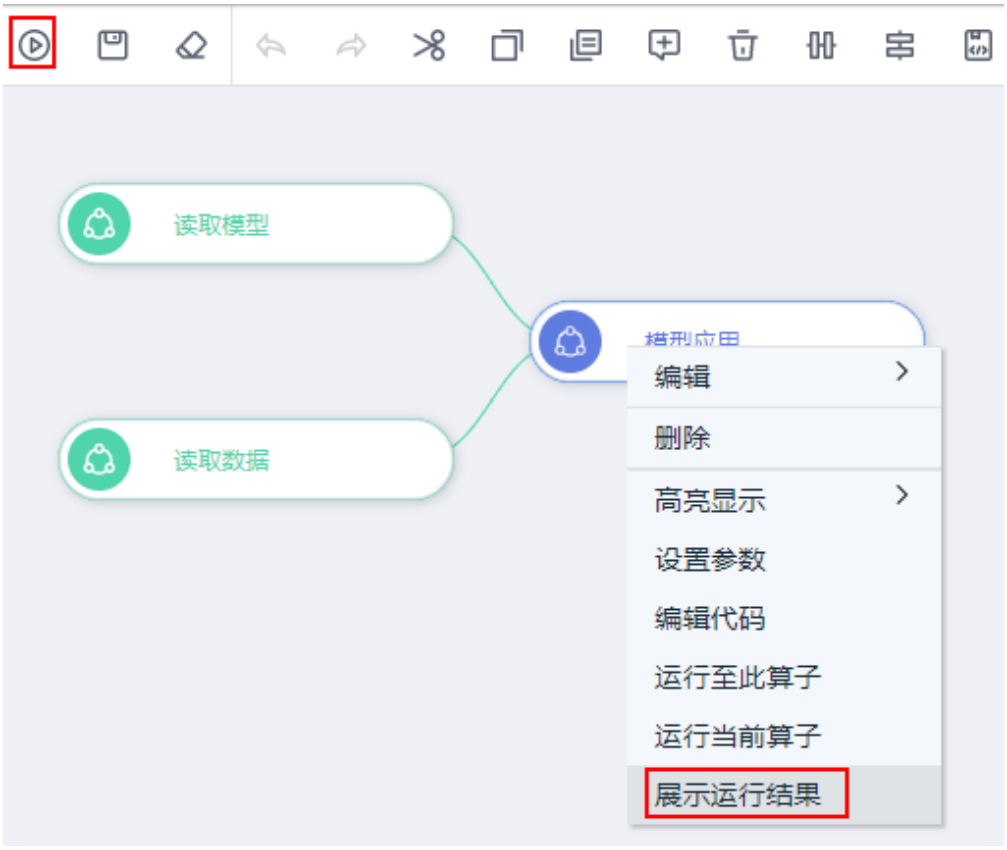
图 7-37 从读取模型算子连接模型应用算子

图 7-38 从读取数据算子连接模型应用算子

版权所有 © 华为云计算技术有限公司

5. 最终预测算链如图7-39所示。单击运行，得到并查看预测运行结果。

图 7-39 运行预测算链



7.4 算子操作

7.4.1 查看算子

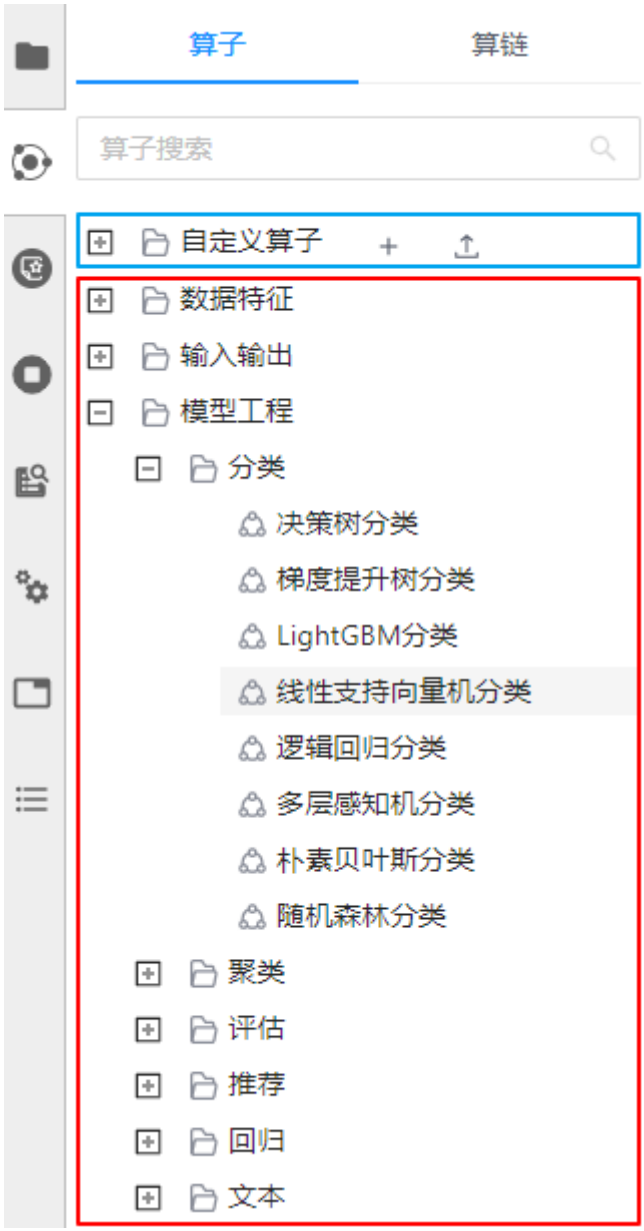
MLS提供了大量机器学习预置算子，方便用户直接使用。当预置算子不满足用户使用场景时，用户可以自行创建算子。

单击Jupyter Lab界面左侧导航条上的图标，可快速进入MLS资产管理界面，当前版本支持算子和算链两种资产。

在算子选项卡中，包含预置算子和自定义算子2类，如图7-40所示，除自定义算子外都是预置算子。

版权所有 © 华为云计算技术有限公司

图 7-40 算子列表



预置算子

如图7-40中红色框所示，预置算子列表目前分为数据特征、输入输出、模型工程三大类。带标志的是算子类别，比如数据特征类，该类包含数据特征的子类数据分析、特征工程、数据处理及其算子。详细的预置算子说明请参考**预置算子说明**章节。

- 单击算子类对象前图标，即可展开显示子类和算子。
- 双击数据特征类，展开其子类数据分析、特征工程、数据处理，双击数据分析类，展开其数据分析算子。
- 单击算子类对象前图标，即可收回其子类和算子。

版权所有 © 华为云计算技术有限公司

说明

预置算子不可修改。如果需进行修改，需拖拽至画布中，右键算子选择编辑代码进行修改，修改后会保存为新的算子，修改内容不会覆盖原有的预置算子。

自定义算子

自定义算子类别中展示开发者自行开发的算子，如图7-40中蓝色框所示，初始为空。

该界面包含新增自定义算子 ＋ 和上传自定义算子 两个按钮，开发者可以直接在线编写算子或上传本地已有的算子。

在当前版本中，一个自定义算子对应一个ipynb文件，开发者可使用Jupyter Lab直接编写算子。并对自定义算子进行删除、重命名、上传或下载操作。

说明

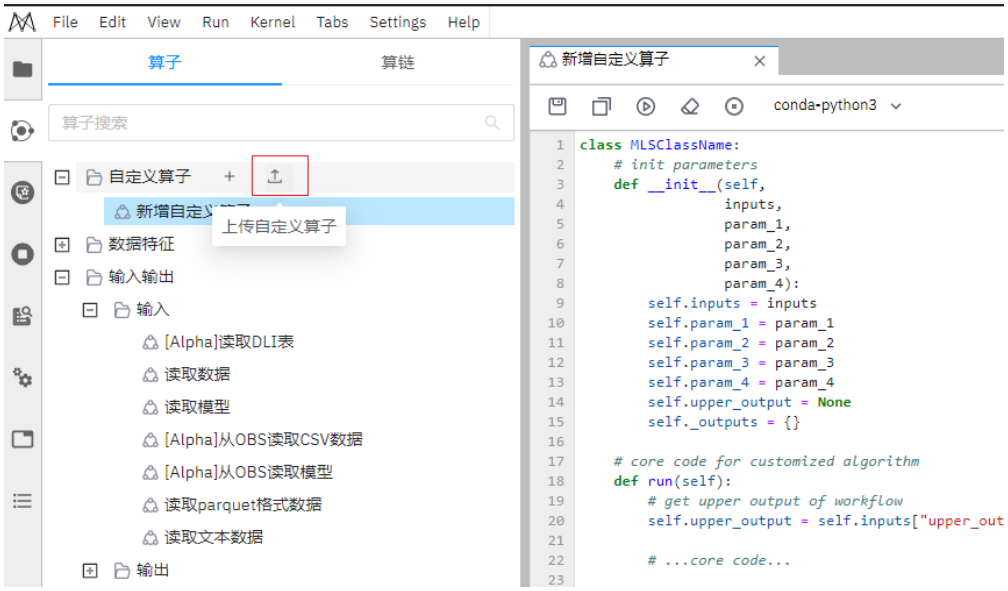
由于算子使用过程中会将ipynb文件中的所有Cell合并为一个Cell，因此建议开发者直接将算子代码写在一个Cell中。

7.4.2 上传/下载自定义算子

上传自定义算子

单击上传自定义算子图标 ，从本地上传新算子，如图7-41所示，当前版本仅支持上传使用Notebook编写的ipynb文件、python脚本。

图 7-41 上传自定义算子

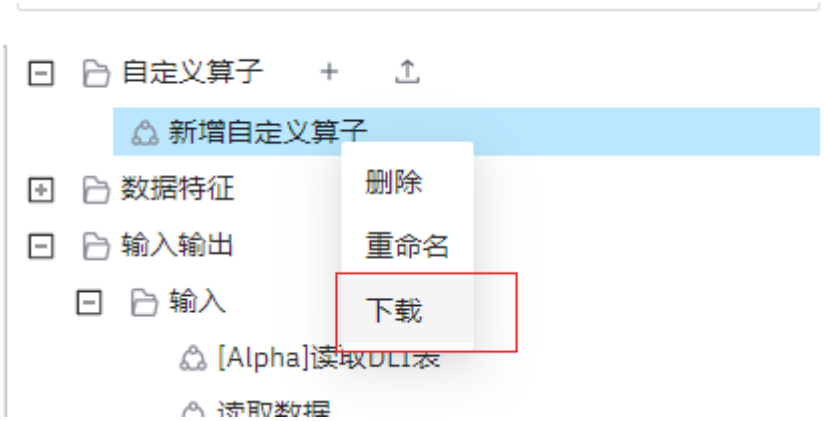


下载自定义算子

对算子单击右键，选择“下载”，即可将算子下载到本地，如图7-42所示。算子会以python文件（后缀为.py）下载到本地。

版权所有 © 华为云计算技术有限公司

图 7-42 下载自定义算子

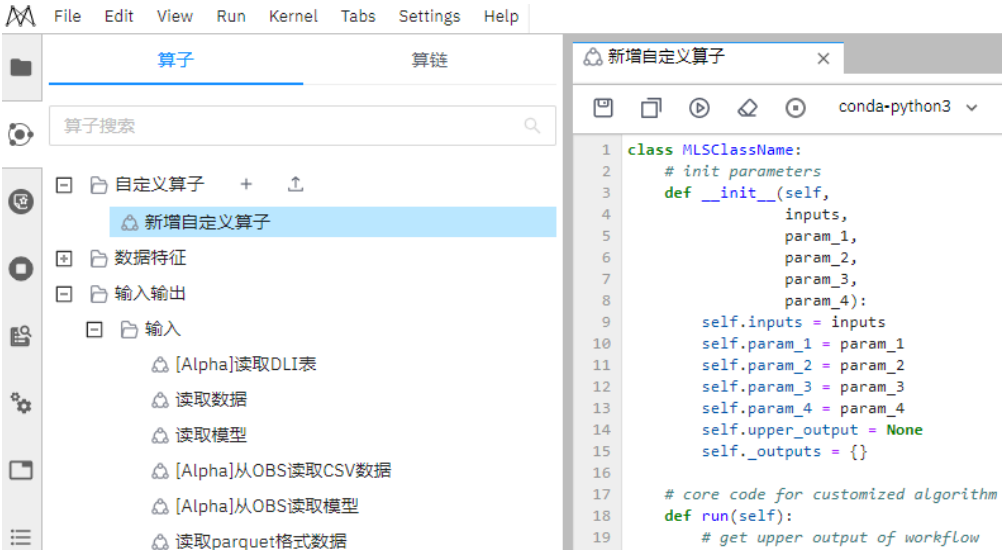


7.4.3 编写自定义算子

用户通过自定义算子功能，可以实现个性化的算子编写。

用户单击“新增自定义算子”图标 ＋ ，新建并打开一个模板算子，即一个算子编辑器（相当于Ipython Notebook的一个cell），输入自定义算子名称，即可以在新建的算子编辑器里面实现自定义算子开发，如图7-43所示。用户可进行算子编辑、调试、复制、保存等功能。

图 7-43 新增自定义算子



说明

由于算子使用过程中会将ipynb文件中的所有Cell合并为一个Cell，因此建议开发者直接将算子代码写在一个Cell中。

7.4.4 自定义算子代码模板和规范

自定义算子代码模板

新建自定义算子时，MLS Editor提供了代码模板，方便用户高效开发算子。

版权所有 © 华为云计算技术有限公司

```
class MLSClassName:
    # init parameters
    def __init__(self,
                 inputs,
                 param_1,
                 param_2):
        self.inputs = inputs
        self.param_1 = param_1
        self.param_2 = param_2
        self.upper_output = None
        self._outputs = {}

    # core code for customized algorithm
    def run(self):
        # get upper output of workflow
        self.upper_output = self.inputs["upper_output"]

        # ...core code...

        # output format
        self._outputs = {
            "output_port_1": "output_result"
        }

    # user called method for getting algorithm result
    def get_outputs(self):
        return self._outputs


# call form for algorithm
inputs = {
    "upper_output": None  #@input {"type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "param_1": "param_value_1",  #@param {"label":"param_1","type":"string","required":"false","helpTip":""}
    "param_2": "param_value_2",  #@param
{"label":"param_1","type":"enum","options":"one,two,three","required":"true","helpTip":""}
    "param_3": "param_value_3",  #@param
{"label":"param_1","type":"integer","range":"(0,none)","required":"true","helpTip":""}
    "param_4": "param_value_4"  #@param
{"label":"param_1","type":"number","range":"(0,1)","required":"true","helpTip":""}
}
mls_instance_#id# = MLSClassName(**params)
mls_instance_#id#.run()
#@output {"label":"dataframe","name":"mls_instance_#id#.get_outputs()
['output_port_1']","type":"DataFrame"}
```

输入设置编写指引

如代码模板所示，上游算子的输出作为该算子的输入，通过字典inputs的形式传给类对象，从而实现算链上游算子和当前算子的数据传递。

#@input标记，能够触发前端的界面响应，实现该算子的输入端口的定义，从而和上游算子相同类型的输出端口相连。

参数设置编写指引

如代码模板所示，算子的各种参数封装在类的__init__函数里面，通过将字典params传给类对象来实现参数输入

#@param标记，能够触发前端的界面响应，通过定义参数的名字、类型、参数提示灯选项，前端能够展示参数模板，供自定义输入。

 版权所有 © 华为云计算技术有限公司

输出设置编写指引

如代码模板所示，算子用于下游算子输入的输出数据封装在类的__output字典里，通过对外函数get_outputs暴露出去，下游算子可以通过调用上游算子类对象的get_outputs函数得到上游算子的输出数据。

#id#标记，是为了防止用户拖拽多个相同算子导致类实例的重名，前端会自动将#id#替换为唯一的id。

#@output标记，能够触发前端的界面响应，通过定义端口的名字、输出值、端口类型等，前端能够定义该算子的端口输出类型，从而和下游算子相同类型的输入端口相连。

7.5 预置算子说明

7.5.1 数据特征

7.5.1.1 数据分析

7.5.1.1.1 箱型图

概述

对数据集中选择的某些列，画出箱型图。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

无

参数说明

参数	子参数	参数说明
select_columns_str	-	列名组成的格式化字符串，例如： column_a column_a,column_b

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
```


版权所有 © 华为云计算技术有限公司

```
}
params = {
    "inputs": inputs,
    "select_columns_str": ""   # @param
{"label":"select_columns_str","type":"string","required":"true","helpTip":""}
}
box_plot____id___ = MLSBoxPlot(**params)
box_plot____id___.run()
```

7.5.1.1.2 分桶统计

概述

对数据集的某些列，进行分桶，即直方图统计。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

统计结果数据集

参数说明

参数	子参数	参数说明
select_columns_str	-	选择的列名组成的格式化字符串，例如: "column_a" "column_a,column_b"
bucket_num	-	默认桶个数为10

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "select_columns_str": "",  # @param
{"label":"select_columns_str","type":"string","required":"false","helpTip":""}
    "bucket_num": 10  # @param
{"label":"bucket_num","type":"integer","required":"true","range":"(0,2147483647)","helpTip":""}
}
bucket_statistics____id___ = MLSBucketStatistics(**params)
bucket_statistics____id___.run()
# @output {"label":"dataframe","name":"bucket_statistics____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

 版权所有 © 华为云计算技术有限公司

7.5.1.1.3 相关性分析

概述

对数据集的数值列进行相关性分析。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

参数	子参数	参数说明
output	output_port_1	dataframe类型的相关系数矩阵

参数说明

参数	是否必选	参数说明	默认值
selected_columns_str	是	选择的列组成的格式化字符串，列必须为数值类型，例如： "column_a" "column_a,column_b"	""
method	是	采用相关性分析的方法，支持"pearson"和"spearman"	"pearson"

样例

数据样本

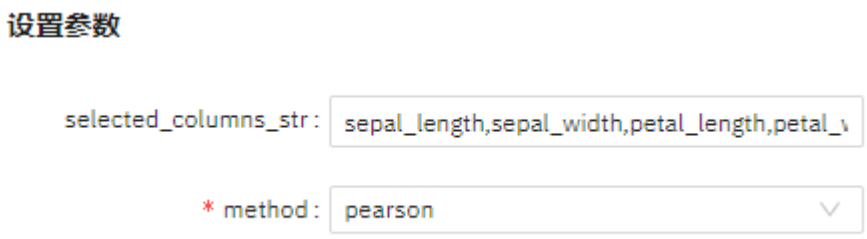
版权所有 © 华为云计算技术有限公司

	sepal_length	sepal_width	petal_length	petal_width	variety
1	5.1	3.5	1.4	.2	Setosa
2	4.9	3	1.4	.2	Setosa
3	4.7	3.2	1.3	.2	Setosa
4	4.6	3.1	1.5	.2	Setosa
5	5	3.6	1.4	.2	Setosa
6	5.4	3.9	1.7	.4	Setosa
7	4.6	3.4	1.4	.3	Setosa
8	5	3.4	1.5	.2	Setosa
9	4.4	2.9	1.4	.2	Setosa
10	4.9	3.1	1.5	.1	Setosa
11	5.4	3.7	1.5	.2	Setosa
12	4.8	3.4	1.6	.2	Setosa
13	4.8	3	1.4	.1	Setosa
14	4.3	3	1.1	.1	Setosa
15	5.8	4	1.2	.2	Setosa
16	5.7	4.4	1.5	.4	Setosa
17	5.4	3.9	1.3	.4	Setosa
18	5.1	3.5	1.4	.3	Setosa
19	5.7	3.8	1.7	.3	Setosa
20	5.1	3.8	1.5	.3	Setosa

配置流程

运行流程

参数设置



查看结果

	correlation	sepal_length	sepal_width	petal_length	petal_width
1	sepal_length	1.0	-0.11756978	0.87175375	0.8179411
2	sepal_width	-0.11756978	1.0	-0.4284401	-0.36612594
3	petal_length	0.87175375	-0.4284401	1.0	0.9628654
4	petal_width	0.8179411	-0.36612594	0.9628654	1.0

版权所有 © 华为云计算技术有限公司

7.5.1.1.4 决策树分类特征重要性

概述

采用决策树分类算法计算数据集特征的特征重要性。

输入

参数	子参数	参数说明
inputs	dataframe	参数必选，表示输入的数据集。 如果没有pipeline_model和decision_tree_classify_model参数，表示直接根据数据集训练决策树分类算法得到特征重要性
	pipeline_model	参数可选，如果含有该参数，表示根据上游的pyspark pipeline模型对象来计算特征重要性
	decision_tree_classify_model	参数可选，如果含有该参数，表示根据上游的决策树分类模型对象来计算特征重要性

输出

包含特征重要性的结果数据集

参数说明

参数	子参数	参数说明
input_columns_str	-	数据集的特征列名组成的格式化字符串，例如: "column_a" "column_a,column_b"
label_col	-	目标列名
model_input_features_col	-	特征向量的列名
classifier_label_index_col	-	将目标列按照标签编码后的列名，默认为"label_index"
prediction_index_col	-	训练模型时，预测结果对应标签的列名，默认为"prediction_index"
prediction_col	-	训练模型时，预测结果对应的列名，默认为"prediction"
max_depth	-	树的最大深度

版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
max_bins	-	分割特征时的最大分箱个数
min_instances_per_node	-	决策树分裂时要求每个节点必须包含的实例数目
min_info_gain	-	最小信息增益
impurity	-	计算信息增益的标准，支持"gini"和"entropy"

样例

```
inputs = {
    "dataframe": None,  # @input {"label":"dataframe","type":"DataFrame"}
    "pipeline_model": None,  # @input {"label":"pipeline_model","type":"PipelineModel"}
    "decision_tree_classify_model": None
}
params = {
    "inputs": inputs,
    "input_columns_str": "",  # @param
{"label":"input_columns_str","type":"string","required":"false","helpTip":""}
    "label_col": "",  # @param {"label":"label_col","type":"string","required":"true","helpTip":""}
    "model_input_features_col": "model_features",  # @param
{"label":"model_input_features_col","type":"string","required":"false","helpTip":""}
    "classifier_label_index_col": "label_index",  # @param
{"label":"classifier_label_index_col","type":"string","required":"false","helpTip":""}
    "prediction_index_col": "prediction_index",  # @param
{"label":"prediction_index_col","type":"string","required":"false","helpTip":""}
    "prediction_col": "prediction",  # @param
{"label":"prediction_col","type":"string","required":"false","helpTip":""}
    "max_depth": 5,  # @param
{"label":"max_depth","type":"integer","required":"false","range":"(0,2147483647]","helpTip":""}
    "max_bins": 32,  # @param
{"label":"max_bins","type":"integer","required":"false","range":"(0,2147483647]","helpTip":""}
    "min_instances_per_node": 1,  # @param
{"label":"min_instances_per_node","type":"integer","required":"false","range":"(0,2147483647]","helpTip":""}
    "min_info_gain": 0.0,  # @param {"label":"min_info_gain","type":"number","required":"false","helpTip":""}
    "impurity": "gini"  # @param
{"label":"impurity","type":"enum","required":"false","options":"entropy,gini","helpTip":""}
}
dt_classify_feature_importance____id___ = MLSDecisionTreeClassifierFeatureImportance(**params)
dt_classify_feature_importance____id___.run()
# @output {"label":"dataframe","name":"dt_classify_feature_importance____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.1.5 决策树回归特征重要性

概述

采用决策树回归算法计算数据集特征的特征重要性。

版权所有 © 华为云计算技术有限公司

输入

参数	子参数	参数说明
inputs	dataframe	参数必选，表示输入的数据集；如果没有pipeline_model和decision_tree_regressor_model参数，表示直接根据数据集训练决策树回归算法得到特征重要性
	pipeline_model	参数可选，如果含有该参数，表示根据上游的pyspark pipeline模型对象来计算特征重要性
	decision_tree_regressor_model	参数可选，如果含有该参数，表示根据上游的决策树回归模型对象来计算特征重要性

输出

包含特征重要性的结果数据集

参数说明

参数	子参数	参数说明
input_columns_str	-	数据集的特征列名组成的格式化字符串，例如: "column_a" "column_a,column_b"
label_col	-	目标列名
model_input_features_col	-	特征向量的列名
prediction_col	-	训练模型时，预测结果对应的列名，默认为"prediction"
max_depth	-	树的最大深度，默认为5
max_bins	-	分割特征时的最大分箱个数，默认为32
min_instances_per_node	-	决策树分裂时要求每个节点必须包含的实例数目，默认为1
min_info_gain	-	最小信息增益，默认为0.0

样例

```
inputs = {
    "dataframe": None,  # @input {"label":"dataframe","type":"DataFrame"}
    "pipeline_model": None,  # @input {"label":"pipeline_model","type":"PipelineModel"}
    "decision_tree_regressor_model": None
}
params = {
```


版权所有 © 华为云计算技术有限公司

```
    "inputs": inputs,
    "input_columns_str": "",  # @param
{"label":"input_columns_str","type":"string","required":"false","helpTip":""}
    "label_col": "",  # @param {"label":"label_col","type":"string","required":"true","helpTip":""}
    "model_input_features_col": "model_features",  # @param
{"label":"model_input_features_col","type":"string","required":"false","helpTip":""}
    "prediction_col": "prediction",  # @param
{"label":"prediction_col","type":"string","required":"false","helpTip":""}
    "max_depth": 5,  # @param
{"label":"max_depth","type":"integer","required":"false","range":"(0,2147483647]","helpTip":""}
    "max_bins": 32,  # @param
{"label":"max_bins","type":"integer","required":"false","range":"(0,2147483647]","helpTip":""}
    "min_instances_per_node": 1,  # @param
{"label":"min_instances_per_node","type":"integer","required":"false","range":"(0,2147483647]","helpTip":""}
    "min_info_gain": 0.0,  # @param {"label":"min_info_gain","type":"number","required":"false","helpTip":""}
    "impurity": "variance"
}
dt_regression_feature_importance____id___ = MLSDecisionTreeRegressorFeatureImportance(**params)
dt_regression_feature_importance____id___.run()
# @output {"label":"dataframe","name":"dt_regression_feature_importance____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.1.6 梯度提升树分类特征重要性

概述

采用梯度提升树分类算法计算数据集特征的特征重要性。

输入

参数	子参数	参数说明
inputs	dataframe	参数必选，表示输入的数据集；如果没有pipeline_model和gbt_classify_model参数，表示直接根据数据集训练gbdt分类模型得到特征重要性
	pipeline_model	参数可选，如果含有该参数，表示根据上游的pyspark pipeline模型对象pipeline_model来计算特征重要性
	gbt_classify_model	参数可选，如果含有该参数，表示根据上游的gbt_classify_model对象来计算特征重要性

输出

特征重要性结果数据集

参数说明

参数	子参数	参数说明
input_columns_str	-	数据集的特征列名组成的格式化字符串，例如: "column_a" "column_a,column_b"

 版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
label_col	-	目标列名
model_input_features_col	-	特征向量的列名
classifier_label_index_col	-	将目标列按照标签编码后的列名，默认为"label_index"
prediction_index_col	-	训练模型时，预测结果对应标签的列名，默认为"prediction_index"
prediction_col	-	训练模型时，预测结果对应的列名，默认为"prediction"
max_depth	-	树的最大深度
max_bins	-	特征分裂时的最大分箱个数
min_instances_per_node	-	树分裂时要求每个节点必须包含的实例数目，默认为1
min_info_gain	-	最小信息增益
max_iter	-	最大迭代次数
step_size	-	步长
subsampling_rate	-	训练每棵树时，对训练集的抽样率

样例

```
inputs = {
    "dataframe": None,  # @input {"label":"dataframe","type":"DataFrame"}
    "pipeline_model": None,  # @input {"label":"pipeline_model","type":"PipelineModel"}
    "gbt_classify_model": None
}
params = {
    "inputs": inputs,
    "input_columns_str": "",  # @param {"label": "input_columns_str", "type": "string", "required": "false", "helpTip": ""}
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": ""}
    "model_input_features_col": "model_features",  # @param {"label": "model_input_features_col", "type": "string", "required": "false", "helpTip": ""}
    "classifier_label_index_col": "label_index",  # @param {"label": "classifier_label_index_col", "type": "string", "required": "false", "helpTip": ""}
    "prediction_index_col": "prediction_index",  # @param {"label": "prediction_index_col", "type": "string", "required": "false", "helpTip": ""}
    "prediction_col": "prediction",  # @param {"label": "prediction_col", "type": "string", "required": "false", "helpTip": ""}
    "max_depth": 5,  # @param {"label": "max_depth", "type": "integer", "required": "false","range":"(0,2147483647]", "helpTip": ""}
    "max_bins": 32,  # @param {"label": "max_bins", "type": "integer", "required": "false","range":"(0,2147483647]", "helpTip": ""}
    "min_instances_per_node": 1,  # @param {"label": "min_instances_per_node", "type": "integer",
```


版权所有 © 华为云计算技术有限公司

```
"required": "false","range":"(0,2147483647]", "helpTip": ""}
    "min_info_gain": 0.0,  # @param {"label": "min_info_gain", "type": "number", "required": "false",
"helpTip": ""}
    "loss_type": "logistic",
    "max_iter": 20,  # @param {"label": "max_iter", "type": "integer", "required":
"false","range":"(0,2147483647]", "helpTip": ""}
    "step_size": 0.1,  # @param {"label": "step_size", "type": "number", "required": "false", "helpTip": ""}
    "subsampling_rate": 1.0  # @param {"label": "subsampling_rate", "type": "number", "required": "false",
"helpTip": ""}
}
gbt_classifier_feature_importance____id___ = MLSGBTClassifierFeatureImportance(**params)
gbt_classifier_feature_importance____id___.run()
# @output {"label":"dataframe","name":"gbt_classifier_feature_importance____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.1.7 梯度提升树回归特征重要性

概述

采用梯度提升树回归算法计算数据集特征的特征重要性。

输入

参数	子参数	参数说明
inputs	dataframe	参数必选，表示输入的数据集；如果没有pipeline_model和gbt_regressor_model参数，表示直接根据数据集训练梯度提升树回归模型得到特征重要性
	pipeline_model	参数可选，如果含有该参数，表示根据上游的pyspark pipeline模型对象pipeline_model来计算特征重要性
	gbt_regressor_model	参数可选，如果含有该参数，表示根据上游的gbt_regressor_model对象来计算特征重要性

输出

特征重要性结果数据集

参数说明

参数	子参数	参数说明
input_columns_str	-	数据集的特征列名组成的格式化字符串，例如: "column_a" "column_a,column_b"
label_col	-	目标列名
model_input_features_col	-	特征向量的列名

 版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
prediction_col	-	训练模型时，预测结果对应的列名，默认为"prediction"
max_depth	-	树的最大深度，默认为5
max_bins	-	特征分裂时的最大分箱个数，默认为32
min_instances_per_node	-	决策树分裂时要求每个节点必须包含的实例数目，默认为1
min_info_gain	-	最小信息增益，默认为0
subsampling_rate	-	训练每棵树时，对训练集的抽样率，默认为1
max_iter	-	最大迭代次数，默认为20
step_size	-	步长，默认为0.1

样例

```
inputs = {
    "dataframe": None,  # @input {"label":"dataframe","type":"DataFrame"}
    "pipeline_model": None,  # @input {"label":"pipeline_model","type":"PipelineModel"}
    "gbt_regressor_model": None
}
params = {
    "inputs": inputs,
    "input_columns_str": "",  # @param {"label": "input_columns_str", "type": "string", "required": "false", "helpTip": ""}
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": ""}
    "model_input_features_col": "model_features",  # @param {"label": "model_input_features_col", "type": "string", "required": "false", "helpTip": ""}
    "prediction_col": "prediction",  # @param {"label": "prediction_col", "type": "string", "required": "false", "helpTip": ""}
    "max_depth": 5,  # @param {"label": "max_depth", "type": "integer", "required": "false","range":"(0,2147483647]", "helpTip": ""}
    "max_bins": 32,  # @param {"label": "max_bins", "type": "integer", "required": "false","range":"(0,2147483647]", "helpTip": ""}
    "min_instances_per_node": 1,  # @param {"label": "min_instances_per_node", "type": "integer", "required": "false","range":"(0,2147483647]", "helpTip": ""}
    "min_info_gain": 0.0,  # @param {"label": "min_info_gain", "type": "number", "required": "false", "helpTip": ""}
    "subsampling_rate": 1.0,  # @param {"label": "subsampling_rate", "type": "number", "required": "false", "helpTip": ""}
    "loss_type": "squared",  # @param {"label": "loss_type", "type": "enum", "required": "false", "options": "squared, absolute", "helpTip": ""}
    "max_iter": 20,  # @param {"label": "max_iter", "type": "integer", "required": "false","range":"(0,2147483647]", "helpTip": ""}
    "step_size": 0.1,  # @param {"label": "step_size", "type": "number", "required": "false", "helpTip": ""}
    "impurity": "variance"
}
gbt_regression_feature_importance____id___ = MLSGBTRegressorFeatureImportance(**params)
gbt_regression_feature_importance____id___.run()
# @output {"label":"dataframe","name":"gbt_regression_feature_importance____id___.get_outputs()['output_port_1']","type":"DataFrame"}
```


版权所有 © 华为云计算技术有限公司

7.5.1.1.8 孤立森林

概述

对sklearn孤立森林算法的封装。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

参数说明

参数	子参数	参数说明
select_columns_str	-	列名组成的格式化字符串，例如： "column_a" "column_a,column_b"
n_estimators	-	基学习器的数量，默认为100
max_samples	-	从数据集中抽取多少个样本来训练，支持"auto"、int类型、float类型
contamination	-	-
max_features	-	从数据集中抽取多少数量的特征来训练每个基训练器
bootstrap	-	构建树时，下次是否替换采样，True表示替换，False表示不替换

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "select_columns_str": "",  # @param
{"label":"select_columns_str","type":"string","required":"false","helpTip":""}
    "n_estimators": 100,  # @param {"label":"n_estimators","type":"integer","required":"false","helpTip":""}
    "max_samples": "auto",  # @param {"label":"max_samples","type":"string","required":"false","helpTip":""}
    "contamination": "auto",  # @param
{"label":"contamination","type":"string","required":"false","helpTip":""}
    "max_features": 1.0,  # @param {"label":"max_features","type":"number","required":"false","helpTip":""}
    "bootstrap": False  # @param {"label":"bootstrap","type":"boolean","required":"false","helpTip":""}
```


版权所有 © 华为云计算技术有限公司

```
}
isolation_forest____id___ = MLSIsolationForest(**params)
isolation_forest____id___.run()
# @output {"label":"dataframe","name":"isolation_forest____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.1.9 百分位

概述

百分位是统计学术语，用于计算数据表列数据的百分位。可以将一组数据从小到大排序，并计算相应数据的百分位，则某百分位所对应数据的值称为该百分位的百分位数。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

参数	子参数	参数说明
output	output_port_1	dataframe类型的百分位结果

参数说明

参数	是否必选	参数说明	默认值
input_cols	是	进行百分位处理的列名，列名间用空格分隔	""
quantile_col	否	quantile列的列名	"quantile"

样例

数据样本

版权所有 © 华为云计算技术有限公司

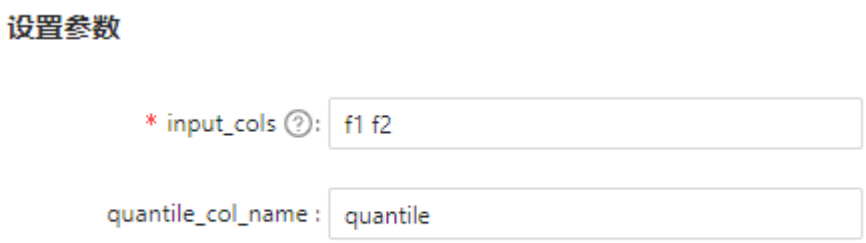
	f1	f2
1	1	11
2	2	22
3	3	33
4	4	44
5	5	55
6	6	66
7	7	77
8	8	88
9	9	99
10	10	110
11	11	10
12	12	20
13	13	30
14	14	40
15	15	50
16	16	60
17	17	70
18	18	80
19	19	90
20	20	100
21	21	110
22	22	120
23	23	130
24	24	140
25	25	150
26	26	160
27	27	170
28	28	180
29	29	190
30	30	200
31	31	210

配置流程

运行流程

版权所有 © 华为云计算技术有限公司

算法参数设置



查看结果

版权所有 © 华为云计算技术有限公司

图 7-44 部分结果展示

	quantile	f1	f2
1	0	1.0	6.0
2	1	2.0	10.0
3	2	4.0	16.0
4	3	6.0	18.0
5	4	8.0	20.0
6	5	10.0	21.0
7	6	12.0	22.0
8	7	14.0	24.0
9	8	16.0	26.0
10	9	18.0	27.0
11	10	20.0	29.0
12	11	22.0	30.0
13	12	24.0	32.0
14	13	26.0	33.0
15	14	28.0	35.0
16	15	30.0	36.0
17	16	32.0	38.0
18	17	34.0	40.0
19	18	36.0	41.0
20	19	38.0	43.0
21	20	40.0	44.0

7.5.1.1.10 百分位统计

概述

对用户选择的数值列进行百分位统计。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

统计结果数据集

版权所有 © 华为云计算技术有限公司

参数说明

参数	子参数	参数说明
select_columns_str	-	选择的列名组成的格式化字符串，例如: "column_a" "column_a,column_b"

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "select_columns_str": ""  # @param
{"label":"select_columns_str","type":"string","required":"false","helpTip":""}
}
percentile_statistics____id___ = MLSPercentileStatistics(**params)
percentile_statistics____id___.run()
# @output {"label":"dataframe","name":"percentile_statistics____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.1.11 直方图

概述

对数据集选择出来的某列，画出其条形图，字符串列显示每个特征出现的数目，数值列显示每个数值区间对应的样本数目。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

无

参数说明

参数	子参数	参数说明
select_column_name	-	选择列的列名
string_bucket_show_num	-	如果选择列为字符串列，该参数表示条形图显示的条的数量
numerical_bucket_show_num	-	如果选择列为数值列，该参数表示条形图显示的条的数量

版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
numerical_interval	-	如果选择列为数值列，该参数表示特征值的区间长度

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "select_column_name": "",  # @param
{"label":"select_column_name","type":"string","required":"true","helpTip":""}
    "string_bucket_show_num": 10,  # @param
{"label":"string_bucket_show_num","type":"integer","required":"true","helpTip":""}
    "numerical_bucket_show_num": 10,  # @param
{"label":"numerical_bucket_show_num","type":"integer","required":"true","helpTip":""}
    "numerical_interval": 0.05  # @param
{"label":"numerical_interval","type":"float","required":"true","helpTip":""}
}
plot_bar_chart____id___ = MLSPlotBarChart(**params)
plot_bar_chart____id___.run()
```

7.5.1.1.12 折线图

概述

对数据集中选择的某些列，画出对应的折线图。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象。

输出

无

参数说明

参数	子参数	参数说明
select_columns_str	-	列名组成的格式化字符串，例如： "column_a" "column_a,column_b"
start_index	-	画折线图时，数据集转成的数组的起始索引
end_index	-	画折线图时，数据集转成的数组的终点索引

版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
figure_length	-	图的长度
figure_width	-	图的宽度

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "select_columns_str": "",  # @param
{"label":"select_columns_str","type":"string","required":"true","helpTip":""}
    "start_index": 0,  # @param {"label":"start_index","type":"integer","required":"true","helpTip":""}
    "end_index": 0,  # @param {"label":"end_index","type":"integer","required":"true","helpTip":""}
    "figure_length": 30,  # @param {"label":"figure_length","type":"integer","required":"false","helpTip":""}
    "figure_width": 10  # @param {"label":"figure_width","type":"integer","required":"false","helpTip":""}
}
plot_line____id___ = MLSPlotLine(**params)
plot_line____id___.run()
```

7.5.1.1.13 饼形图

概述

对数据集的某一列，画出对应的饼形图。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

无

参数说明

参数	子参数	参数说明
select_column_name	-	选择列的列名
numeric_intervals_str	-	画饼形图时，每个区间的长度组成的字符串，逗号隔开

版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
numeric_interval_length	-	如果numeric_intervals_str没有设置，默认饼形图的每个区间的长度一样，numeric_interval_length表示此时的区间长度
show_share_number	-	饼形图的份额数目，默认为5
figure_length	-	图的长度
figure_width	-	图的宽度

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "select_column_name": "",  # @param
{"label":"select_column_name","type":"string","required":"true","helpTip":""}
    "numeric_intervals_str": "",  # @param
{"label":"numeric_intervals_str","type":"string","required":"false","helpTip":""}
    "numeric_interval_length": "",  # @param
{"label":"numeric_interval_length","type":"string","required":"false","helpTip":""}
    "show_share_number": 5,  # @param
{"label":"show_share_number","type":"integer","required":"false","range":"(0,2147483647]","helpTip":""}
    "figure_length": "",  # @param
{"label":"figure_length","type":"integer","required":"false","range":"(0,2147483647]","helpTip":""}
    "figure_width": ""   # @param
{"label":"figure_width","type":"integer","required":"false","range":"(0,2147483647]","helpTip":""}
}
plot_pie____id___ = MLSPlotPie(**params)
plot_pie____id___.run()
```

7.5.1.1.14 散点图

概述

对数据集画出对应的散点图。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

无

版权所有 © 华为云计算技术有限公司

参数说明

参数	子参数	参数说明
start_index	-	只对数据集转成的数组的某个区间内元素化散点图，start_index表示开始位置
end_index	-	只对数据集转成的数组的某个区间内元素化散点图，end_index表示结束位置
x_axis_column_name	-	散点图x轴的列名
y_axis_columns_str	-	散点图y轴的某些列，y_axis_columns_str表示用列名逗号隔开的字符串
figure_length	-	图的长度
figure_width	-	图的宽度

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "start_index": "",  # @param
{"label":"start_index","type":"integer","required":"true","range":"[0,2147483647]","helpTip":""}
    "end_index": "",  # @param
{"label":"end_index","type":"integer","required":"true","range":"[0,2147483647]","helpTip":""}
    "x_axis_column_name": "",  # @param
{"label":"x_axis_column_name","type":"string","required":"false","helpTip":""}
    "y_axis_columns_str": "",  # @param
{"label":"y_axis_columns_str","type":"string","required":"false","helpTip":""}
    "figure_length": "",  # @param
{"label":"figure_length","type":"integer","required":"false","range":"[0,2147483647]","helpTip":""}
    "figure_width": ""   # @param
{"label":"figure_width","type":"integer","required":"false","range":"[0,2147483647]","helpTip":""}
}
plot_scatter____id___ = MLSPlotScatter(**params)
plot_scatter____id___.run()
```

7.5.1.1.15 随机森林分类特征重要性

概述

采用随机森林分类算法计算数据集特征的特征重要性

输入

参数	子参数	参数说明
inputs	dataframe	参数必选，表示输入的数据集；如果没有pipeline_model和random_forest_classify_model参数，表示直接根据数据集训练随机森林分类模型得到特征重要性

版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
	pipeline_model	参数可选，如果含有该参数，表示根据上游的pyspark pipeline模型对象pipeline_model来计算特征重要性
	random_forest_classify_model	参数可选，如果含有该参数，表示根据上游的random_forest_classify_model对象来计算特征重要性

输出

特征重要性结果数据集

参数说明

参数	子参数	参数说明
input_columns_str	-	数据集的特征列名组成的格式化字符串，例如: "column_a" "column_a,column_b"
label_col	-	目标列名
model_input_features_col	-	特征向量的列名
classifier_label_index_col	-	将目标列按照标签编码后的列名，默认为"label_index"
prediction_index_col	-	训练模型时，预测结果对应标签的列名，默认为"prediction_index"
prediction_col	-	训练模型时，预测结果对应的列名，默认为"prediction"
max_depth	-	树的最大深度，默认为5
max_bins	-	特征分裂时的最大分箱个数，默认为32
min_instances_per_node	-	树分裂时要求每个节点必须包含的实例数目，默认为1
min_info_gain	-	最小信息增益，默认为0.0
impurity	-	纯度，支持"gini"和"entropy"，默认为"gini"
num_trees	-	树的个数，默认为20
feature_subset_strategy	-	每个树节点分裂时使用的特征个数,默认为"all"

版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
subsampling_rate	-	训练每棵树时，对训练集的抽样率，默认为1.0
seed	-	随机数种子，默认为0

样例

```
inputs = {
    "dataframe": None,  # @input {"label":"dataframe","type":"DataFrame"}
    "pipeline_model": None,  # @input {"label":"pipeline_model","type":"PipelineModel"}
    "random_forest_classify_model": None
}
params = {
    "inputs": inputs,
    "input_columns_str": "",  # @param {"label": "input_columns_str", "type": "string", "required": "false",
"helpTip": ""}
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": ""}
    "model_input_features_col": "model_features",  # @param {"label": "model_input_features_col", "type":
"string", "required": "false", "helpTip": ""}
    "classifier_label_index_col": "label_index",  # @param {"label": "classifier_label_index_col", "type":
"string", "required": "false", "helpTip": ""}
    "prediction_index_col": "prediction_index",  # @param {"label": "prediction_index_col", "type": "string",
"required": "false", "helpTip": ""}
    "prediction_col": "prediction",  # @param {"label": "prediction_col", "type": "string", "required": "false",
"helpTip": ""}
    "max_depth": 5,  # @param {"label": "max_depth", "type": "integer", "required":
"false","range":"(0,2147483647]", "helpTip": ""}
    "max_bins": 32,  # @param {"label": "max_bins", "type": "integer", "required":
"false","range":"(0,2147483647]", "helpTip": ""}
    "min_instances_per_node": 1,  # @param {"label": "min_instances_per_node", "type": "integer",
"required": "false","range":"(0,2147483647]", "helpTip": ""}
    "min_info_gain": 0.0,  # @param {"label": "min_info_gain", "type": "number", "required": "false",
"helpTip": ""}
    "impurity": "gini",  # @param {"label": "impurity", "type": "enum", "required": "false", "options":
"entropy,gini", "helpTip": ""}
    "num_trees": 20,  # @param {"label": "num_trees", "type": "integer", "required":
"false","range":"(0,2147483647]", "helpTip": ""}
    "feature_subset_strategy": "all",  # @param {"label": "feature_subset_strategy", "type": "enum",
"options":"auto,all,onethird,sqrt,log2", "required": "false", "helpTip": ""}
    "subsampling_rate": 1.0,  # @param {"label": "subsampling_rate", "type": "number", "required": "false",
"helpTip": ""}
    "seed": 0  # @param {"label": "seed", "type": "integer", "required": "false","range":"[0,2147483647]",
"helpTip": ""}
}
rf_classify_feature_importance____id___ = MLSRandomForestClassifierFeatureImportance(**params)
rf_classify_feature_importance____id___.run()
# @output {"label":"dataframe","name":"rf_classify_feature_importance____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.1.16 随机森林回归特征重要性

概述

采用随机森林回归算法计算数据集特征的特征重要性

版权所有 © 华为云计算技术有限公司

输入

参数	子参数	参数说明
inputs	dataframe	参数必选，表示输入的数据集；如果没有pipeline_model和random_forest_regressor_model参数，表示直接根据数据集训练随机森林分类模型得到特征重要性
	pipeline_model	参数可选，如果含有该参数，表示根据上游的pyspark pipeline模型对象pipeline_model来计算特征重要性
	random_forest_regressor_model	参数可选，如果含有该参数，表示根据上游的random_forest_regressor_model对象来计算特征重要性

输出

特征重要性结果数据集

参数说明

参数	子参数	参数说明
input_columns_str	-	数据集的特征列名组成的格式化字符串，例如: "column_a" "column_a,column_b"
label_col	-	目标列名
model_input_features_col	-	特征向量的列名
prediction_col	-	训练模型时，预测结果对应的列名，默认为"prediction"
max_depth	-	树的最大深度，默认为5
max_bins	-	特征分裂时的最大分箱个数，默认为32
min_instances_per_node	-	树分裂时要求每个节点必须包含的实例数目，默认为1
min_info_gain	-	最小信息增益，默认为0.0
subsampling_rate	-	训练每棵树时，对训练集的抽样率，默认为1.0
num_trees	-	树的个数，默认为20

版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
feature_subset_strategy	-	每个树节点分裂时使用的特征个数,默认为"auto"

样例

```
inputs = {
    "dataframe": None,  # @input {"label":"dataframe","type":"DataFrame"}
    "pipeline_model": None,  # @input {"label":"pipeline_model","type":"PipelineModel"}
    "random_forest_regressor_model": None
}
params = {
    "inputs": inputs,
    "input_columns_str": "",  # @param {"label": "input_columns_str", "type": "string", "required": "false",
"helpTip": ""}
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": ""}
    "model_input_features_col": "model_features",  # @param {"label": "model_input_features_col", "type":
"string", "required": "false", "helpTip": ""}
    "prediction_col": "prediction",  # @param {"label": "prediction_col", "type": "string", "required": "false",
"helpTip": ""}
    "max_depth": 5,  # @param {"label": "max_depth", "type": "integer", "required":
"false","range":"(0,2147483647]", "helpTip": ""}
    "max_bins": 32,  # @param {"label": "max_bins", "type": "integer", "required":
"false","range":"(0,2147483647]", "helpTip": ""}
    "min_instances_per_node": 1,  # @param {"label": "min_instances_per_node", "type": "integer",
"required": "false","range":"(0,2147483647]", "helpTip": ""}
    "min_info_gain": 0.0,  # @param {"label": "min_info_gain", "type": "number", "required": "false",
"helpTip": ""}
    "impurity": "variance",
    "subsampling_rate": 1.0,  # @param {"label": "subsampling_rate", "type": "number", "required": "false",
"helpTip": ""}
    "num_trees": 20,  # @param {"label": "num_trees", "type": "integer", "required":
"false","range":"(0,2147483647]", "helpTip": ""}
    "feature_subset_strategy": "auto"  # @param {"label": "feature_subset_strategy", "type": "enum",
"options":"auto,all,onethird,sqrt,log2", "required": "false", "helpTip": ""}
}
rf_regression_feature_importance____id___ = MLSRandomForestRegressorFeatureImportance(**params)
rf_regression_feature_importance____id___.run()
# @output {"label":"dataframe","name":"rf_regression_feature_importance____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.1.17 全表统计

概述

对数据集指定的某些列做全表统计，包括元素总数、null值个数、nan值个数、最小值、最大值、方差、标准差等。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

版权所有 © 华为云计算技术有限公司

输出

表 7-5

参数	子参数	参数说明
output	output_port_1	dataframe类型的统计结果

参数说明

参数	是否必选	参数说明	默认值
select_columns_str	是	列名组成的格式化字符串，例如： "column_a" "column_a,column_b" ""则表示全选	""

样例

数据样本

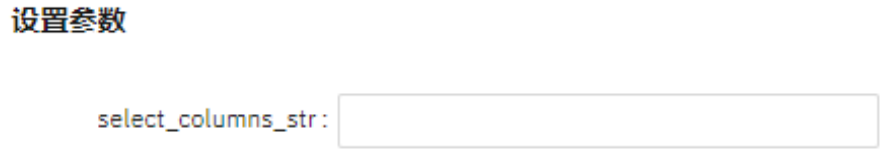
	sepal_length	sepal_width	petal_length	petal_width	variety
1	5.1	3.5	1.4	.2	Setosa
2	4.9	3	1.4	.2	Setosa
3	4.7	3.2	1.3	.2	Setosa
4	4.6	3.1	1.5	.2	Setosa
5	5	3.6	1.4	.2	Setosa
6	5.4	3.9	1.7	.4	Setosa
7	4.6	3.4	1.4	.3	Setosa
8	5	3.4	1.5	.2	Setosa
9	4.4	2.9	1.4	.2	Setosa
10	4.9	3.1	1.5	.1	Setosa
11	5.4	3.7	1.5	.2	Setosa
12	4.8	3.4	1.6	.2	Setosa
13	4.8	3	1.4	.1	Setosa
14	4.3	3	1.1	.1	Setosa
15	5.8	4	1.2	.2	Setosa
16	5.7	4.4	1.5	.4	Setosa
17	5.4	3.9	1.3	.4	Setosa
18	5.1	3.5	1.4	.3	Setosa
19	5.7	3.8	1.7	.3	Setosa
20	5.1	3.8	1.5	.3	Setosa

配置流程

运行流程：

版权所有 © 华为云计算技术有限公司

算法参数设置：



查看结果：

	statistics metric	sepal_length	sepal_width	petal_length	petal_width	variety
1	column_type	number	number	number	number	string
2	elem_count	150	150	150	150	150
3	null_count	0	0	0	0	0
4	nan_count	0	0	0	0	
5	valid_count	150	150	150	150	150
6	unique_count	35	23	43	22	3
7	positive_infinity_count	0	0	0	0	
8	negative_infinity_count	0	0	0	0	
9	min	4.3	2.0	1.0	0.1	
10	max	7.9	4.4	6.9	2.5	
11	mean	5.843333333333335	3.057333333333334	3.7580000000000027	1.199333333333334	
12	variance	0.6811222222221929	0.18871288888887605	3.0955026666666488	0.5771328888888878	
13	standard_deviation	0.8280661279778452	0.43586628493668345	1.7652982332594613	0.7622376689603458	
14	standard_error	0.06761131622759714	0.035588333139247204	0.14413599717741055	0.062236445056044226	
15	kurtosis	-0.573567948935144	0.18097631750897358	-1.3955358863989842	-1.3360674052315624	
16	skewness	0.30864072959002775	0.3126147039239179	-0.2694109303052909	-0.10091656529579222	
17	2nd_moment	34.82566666666665	9.535999999999992	17.21806666666667	2.015533333333334	
18	3rd_moment	211.63327333333334	30.33453333333333	86.48922	3.7569533333333336	
19	4th_moment	1310.611336666667	98.38535199999994	454.8411006666669	7.389707333333331	
20	2nd_central_moment	0.6811222222221929	0.18871288888887605	3.0955026666666488	0.5771328888888878	
21	3rd_central_moment	0.17524607407449366	0.025886255407492342	-1.482072575999922	-0.04469240059259283	
22	4th_central_moment	1.1256885107359267	0.11328269225668919	15.374194561978697	0.5542266145771801	
23	sum	876.5000000000002	458.60000000000014	563.7000000000004	179.90000000000012	
24	square_sum	5223.849999999998	1430.399999999999	2582.7100000000005	302.3300000000001	
25	cube_sum	31744.990999999998	4550.179999999999	12973.383	563.543	
26	4th_power_sum	196591.70050000006	14757.80279999999	68226.16510000003	1108.4560999999997	
27	l1_norm	876.5000000000002	458.60000000000014	563.7000000000004	179.90000000000012	
28	l2_norm	72.27620631992245	37.82062929143299	50.8203699317508	17.38763928772391	

7.5.1.1.18 单样本 t 检验

说明

单样本t检验目前仅支持在ML Studio镜像内运行，不支持发布到dli。

概述

t检验也称为Student t检验，它是一种使用假设检验来评估一个或两个总体均值的工具。单样本t检验可用于检验一个正态分布的总体的均值是否在满足零假设的值之内。

版权所有 © 华为云计算技术有限公司

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

参数	子参数	参数说明
output	T_Test_Result	dataframe类型的检验结果

参数说明

参数	是否必选	参数说明	默认值
feature_column	是	进行T检验的特征列列名	""
alternative_hypothesis	是	备择假设，取值如下： ● two_sided，双侧检验 ● greater，单侧检验 ● less，单侧检验	""
confidence_level	是	置信水平，取值包括：0.8,0.9,0.95,0.99,0.995,0.999	""
except_value	是	零假设值，即t检验中的μ值	0

样例

数据样本

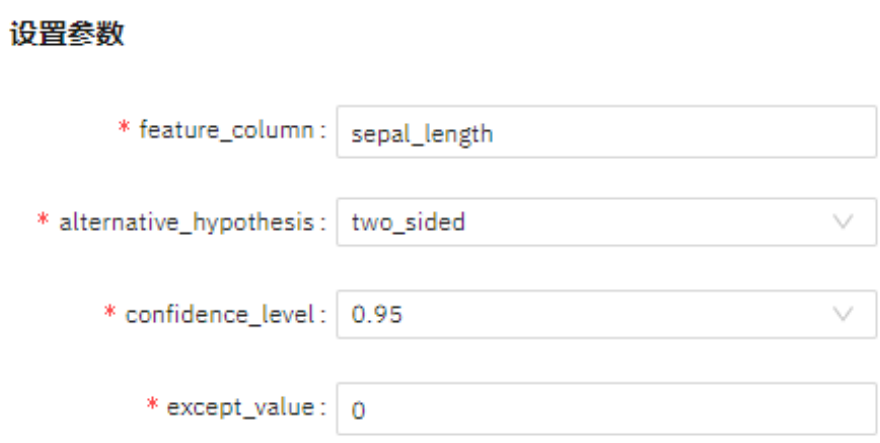
版权所有 © 华为云计算技术有限公司

	sepal_length	sepal_width	petal_length	petal_width	variety
1	5.1	3.5	1.4	.2	Setosa
2	4.9	3	1.4	.2	Setosa
3	4.7	3.2	1.3	.2	Setosa
4	4.6	3.1	1.5	.2	Setosa
5	5	3.6	1.4	.2	Setosa
6	5.4	3.9	1.7	.4	Setosa
7	4.6	3.4	1.4	.3	Setosa
8	5	3.4	1.5	.2	Setosa
9	4.4	2.9	1.4	.2	Setosa
10	4.9	3.1	1.5	.1	Setosa
11	5.4	3.7	1.5	.2	Setosa
12	4.8	3.4	1.6	.2	Setosa
13	4.8	3	1.4	.1	Setosa
14	4.3	3	1.1	.1	Setosa
15	5.8	4	1.2	.2	Setosa
16	5.7	4.4	1.5	.4	Setosa
17	5.4	3.9	1.3	.4	Setosa
18	5.1	3.5	1.4	.3	Setosa
19	5.7	3.8	1.7	.3	Setosa
20	5.1	3.8	1.5	.3	Setosa

配置流程

运行流程

算法参数设置



查看结果

版权所有 © 华为云计算技术有限公司

	One Sample T Test	value
1	AlternativeHypthesis	mean not equals to 0
2	ConfidenceInterval	67, 5.976934185159302]
3	ConfidenceLevel	0.95
4	alpha	0.050000000000000044
5	df	149
6	mean	5.843333333333335
7	p	0.0
8	stdDeviation	0.8280661279778637
9	t	86.42537461721697

7.5.1.1.19 直方图（多字段）

概述

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，以长条图（bar）的形式展现。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

参数	子参数	参数说明
output	dataframe	dataframe类型的统计结果

参数说明

参数	是否必选	参数说明	默认值
bins	是	直方图中条形数量	2
selected_columns_str	是	需要统计的特征列列名，支持多列以“，”分隔	""

样例

数据样本

版权所有 © 华为云计算技术有限公司

	sepal_length	sepal_width	petal_length	petal_width	variety
1	5.1	3.5	1.4	.2	Setosa
2	4.9	3	1.4	.2	Setosa
3	4.7	3.2	1.3	.2	Setosa
4	4.6	3.1	1.5	.2	Setosa
5	5	3.6	1.4	.2	Setosa
6	5.4	3.9	1.7	.4	Setosa
7	4.6	3.4	1.4	.3	Setosa
8	5	3.4	1.5	.2	Setosa
9	4.4	2.9	1.4	.2	Setosa
10	4.9	3.1	1.5	.1	Setosa
11	5.4	3.7	1.5	.2	Setosa
12	4.8	3.4	1.6	.2	Setosa
13	4.8	3	1.4	.1	Setosa
14	4.3	3	1.1	.1	Setosa
15	5.8	4	1.2	.2	Setosa
16	5.7	4.4	1.5	.4	Setosa
17	5.4	3.9	1.3	.4	Setosa
18	5.1	3.5	1.4	.3	Setosa
19	5.7	3.8	1.7	.3	Setosa
20	5.1	3.8	1.5	.3	Setosa

配置流程

运行流程

算法参数设置

查看结果

	colname	histogram
1	sepal_length	{[4.30, 5.50): 52, [5.50, 6.70): 70, [6.70, 7.90]: 28}
2	sepal_width	{[2.00, 2.80): 47, [2.80, 3.60): 88, [3.60, 4.40]: 15}
3	petal_length	{[1.00, 2.97): 50, [2.97, 4.93): 54, [4.93, 6.90]: 46}
4	petal_width	{[0.10, 0.90): 50, [0.90, 1.70): 52, [1.70, 2.50]: 48}

版权所有 © 华为云计算技术有限公司

说明

目前直方图（多字段）暂不支持绘图，仅对数据进行统计。

7.5.1.1.20 卡方拟合性检验

说明

卡方拟合检验目前仅支持在ML Studio镜像内运行，不支持发布到dli。

概述

卡方拟合检验，即卡方拟合优度检验。对每个类别中的实测频率和期望频率进行比较，以检验是否所有类别包含相同比例的值，或检验是否每个类别包含用户指定比例的值。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

参数	子参数	参数说明
output	output_port_1	dataframe类型的计算过程数据
output	statistic_summary	dataframe类型的统计信息

参数说明

参数	是否必选	参数说明	默认值
selected_columns_str	是	需要进行卡方拟合检验列列名	""
labels_rate	是	检验列类别比例，格式为：label1:0.5,label2:0.5，比例和为1	""

样例

数据样本

版权所有 © 华为云计算技术有限公司

	sepal_length	sepal_width	petal_length	petal_width	variety
1	5.1	3.5	1.4	.2	Setosa
2	4.9	3	1.4	.2	Setosa
3	4.7	3.2	1.3	.2	Setosa
4	4.6	3.1	1.5	.2	Setosa
5	5	3.6	1.4	.2	Setosa
6	5.4	3.9	1.7	.4	Setosa
7	4.6	3.4	1.4	.3	Setosa
8	5	3.4	1.5	.2	Setosa
9	4.4	2.9	1.4	.2	Setosa
10	4.9	3.1	1.5	.1	Setosa
11	5.4	3.7	1.5	.2	Setosa
12	4.8	3.4	1.6	.2	Setosa
13	4.8	3	1.4	.1	Setosa
14	4.3	3	1.1	.1	Setosa
15	5.8	4	1.2	.2	Setosa
16	5.7	4.4	1.5	.4	Setosa
17	5.4	3.9	1.3	.4	Setosa
18	5.1	3.5	1.4	.3	Setosa
19	5.7	3.8	1.7	.3	Setosa
20	5.1	3.8	1.5	.3	Setosa

配置流程

运行流程

算法参数设置

查看结果

版权所有 © 华为云计算技术有限公司

	variety	observed	excepted	resident
1	Virginica	50	45.0	0.7453559924999299
2	Setosa	50	60.0	-1.2909944487358056
3	Versicolor	50	45.0	0.7453559924999299

	Chi-Square Statistics	value
1	comment	皮尔逊卡方
2	df	2
3	p-value	0.24935220877729625
4	value	2.7777777777777777

7.5.1.1.21 卡方独立性检验

说明

卡方独立性检验目前仅支持在MLStudio镜像内运行，不支持发布到dli。

概述

卡方独立性检验是检验两个变量之间是否存在相关性。一般认为这两个变量是分类变量，我们认为两者之间是不相关的，可以通过检验来确定该看法。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

参数	子参数	参数说明
output	output_port_1	dataframe类型的计算过程数据
output	statistic_summary	dataframe类型的统计检验结果

参数说明

参数	是否必选	参数说明	默认值
selected_column1	是	卡方独立性检验特征列列名	""

版权所有 © 华为云计算技术有限公司

参数	是否必选	参数说明	默认值
selected_column2	是	卡方独立性检验特征列列名	""

样例

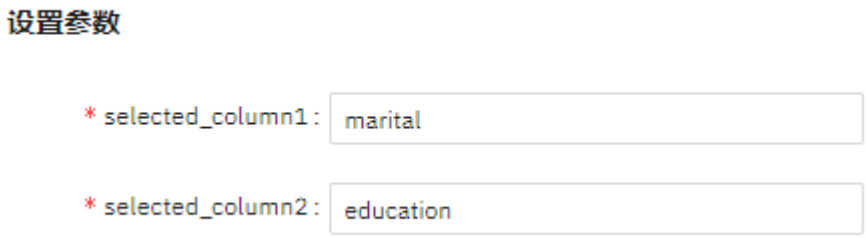
数据样本

	id	age	job	marital	education	housing	loan	label
1	0	59	admin.	married	secondary	yes	no	yes
2	1	56	admin.	married	secondary	no	no	yes
3	2	41	technician	married	secondary	yes	no	yes
4	3	55	services	married	secondary	yes	no	yes
5	4	54	admin.	married	tertiary	no	no	yes
6	5	42	management	single	tertiary	yes	yes	yes
7	6	56	management	married	tertiary	yes	yes	yes
8	7	60	retired	divorced	secondary	yes	no	yes
9	8	39	technician	single	unknown	yes	no	yes
10	9	37	technician	married	secondary	yes	no	yes
11	10	34	admin.	married	secondary	no	no	yes
12	11	55	unemployed	divorced	secondary	yes	no	yes
13	12	28	services	single	secondary	yes	no	yes
14	13	30	technician	married	secondary	yes	no	yes
15	14	36	technician	married	secondary	yes	yes	yes
16	15	37	admin.	single	secondary	yes	yes	yes
17	16	45	blue-collar	married	secondary	yes	no	yes
18	17	53	services	divorced	primary	yes	yes	yes

配置流程

运行流程

算法参数设置



查看结果

版权所有 © 华为云计算技术有限公司

	marital	education	observed	excepted	resident
1	divorced	unknown	0	0.16666666666666666	-0.16666666666666666
2	married	unknown	0	0.6111111111111112	-0.6111111111111112
3	single	unknown	1	0.2222222222222222	0.7777777777777778
4	divorced	tertiary	0	0.5	-0.5
5	married	tertiary	2	1.8333333333333333	0.16666666666666674
6	single	tertiary	1	0.6666666666666666	0.33333333333333337
7	divorced	secondary	2	2.1666666666666665	-0.16666666666666652
8	married	secondary	9	7.944444444444445	1.0555555555555554
9	single	secondary	2	2.888888888888889	-0.8888888888888888
10	divorced	primary	1	0.16666666666666666	0.8333333333333334
11	married	primary	0	0.6111111111111112	-0.6111111111111112
12	single	primary	0	0.2222222222222222	-0.2222222222222222

	Chi-Square Statistics	value
1	comment	皮尔逊卡方
2	df	6
3	p-value	0.14214191510856633
4	value	9.60839160839161

7.5.1.1.22 协方差矩阵

概述

协方差，在概率论与统计学中用于衡量随机变量的联合变化程度。正态形式的协方差大小可以显示变量之间线性关系的强弱，如：皮尔逊相关系数。但是协方差的数值大小也取决于变量的大小。协方差矩阵是多个变量之间的协方差所构成的矩阵表示形式。方差是协方差的一种特殊形式。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

参数	子参数	参数说明
output	output_port_1	dataframe类型的协方差矩阵

版权所有 © 华为云计算技术有限公司

参数说明

参数	是否必选	参数说明	默认值
selected_columns_str	是	计算协方差矩阵的特征列列名，支持多列用','分隔	""

样例

数据样本

	sepal_length	sepal_width	petal_length	petal_width	variety
1	5.1	3.5	1.4	.2	Setosa
2	4.9	3	1.4	.2	Setosa
3	4.7	3.2	1.3	.2	Setosa
4	4.6	3.1	1.5	.2	Setosa
5	5	3.6	1.4	.2	Setosa
6	5.4	3.9	1.7	.4	Setosa
7	4.6	3.4	1.4	.3	Setosa
8	5	3.4	1.5	.2	Setosa
9	4.4	2.9	1.4	.2	Setosa
10	4.9	3.1	1.5	.1	Setosa
11	5.4	3.7	1.5	.2	Setosa
12	4.8	3.4	1.6	.2	Setosa
13	4.8	3	1.4	.1	Setosa
14	4.3	3	1.1	.1	Setosa
15	5.8	4	1.2	.2	Setosa
16	5.7	4.4	1.5	.4	Setosa
17	5.4	3.9	1.3	.4	Setosa
18	5.1	3.5	1.4	.3	Setosa
19	5.7	3.8	1.7	.3	Setosa
20	5.1	3.8	1.5	.3	Setosa

配置流程

运行流程

算法参数设置

查看结果

版权所有 © 华为云计算技术有限公司

	covariance	sepal_length	sepal_width	petal_length	petal_width
1	sepal_length	0.6856935	-0.042434003	1.2743155	0.5162707
2	sepal_width	-0.042434003	0.18997942	-0.32965636	-0.12163937
3	petal_length	1.2743155	-0.32965636	3.116278	1.2956094
4	petal_width	0.5162707	-0.12163937	1.2956094	0.5810063

7.5.1.1.23 孤立森林[PySpark 版]

概述

孤立森林（Isolation Forest），简称为iForest，用于挖掘异常（Anomaly）数据，从数据中找出与其它数据的规律不符合的数据。通常用于网络安全中的攻击检测和流量异常等分析，金融机构则用于挖掘出欺诈行为。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象。

输出

参数	子参数	参数说明
output	output_port_1	output为字典类型，output_port_1为pyspark中的PipelineModel类型对象，为训练出的孤立森林模型。
output	output_port_2	output为字典类型，output_port_1为pyspark中的DataFrame类型对象，孤立森林算法的检测结果。

参数说明

参数	是否必选	参数说明	默认值
b_use_default_encoder	是	是否对数据中的类别型特征列进行编码处理。	True
input_features_str	否	选择特征列，逗号分隔。	无
num_trees	是	孤立森林中树的个数。	100

版权所有 © 华为云计算技术有限公司

参数	是否必选	参数说明	默认值
bootstrap	是	采样数据构建孤立树时是否为有放回采样。	False
max_samples	是	训练单棵孤立树的最大样本个数，改值小于1.0时该值乘以总样本数取整得到单棵孤立树的训练样本数，大于1.0时取整得到单棵数的训练样本数。	256.0
max_features	是	参与训练的特征数，小于等于1.0时特征为该值乘以总特征个数。	1.0
feature_vector_col	是	input_features_str中的特征列处理为向量列后的列名。	"assembled_features"
prediction_col	是	预测结果列名。	"prediction"
score_col	是	异常分数列，该列数值为孤立森林算法中每个样本的分数值，值越大异常可能越大。	"outlier_score"
contamination	是	异常值比例，取值0到1浮点数，score_col列中数值大于contamination * 100%分位数值的样本视为异常值， 如果为0.0则prediction_col列输出均为0.0非异常。	0.0
contamination_error	是	计算分位数时允许的误差，如果为0.0则实际计算时为contamination * 0.01。	0.0
seed	是	随机种子。	0

样例

数据样本为信用卡欺诈检测数据，包含Time，V1，V2，V3，V4，V5，V6，V7，V8，V9，V10，V11，V12，V13，V14，V15，V16，V17，V18，V19，V20，V21，V22，V23，V24，V25，V26，V27，V28，Amount等特征。

版权所有 © 华为云计算技术有限公司

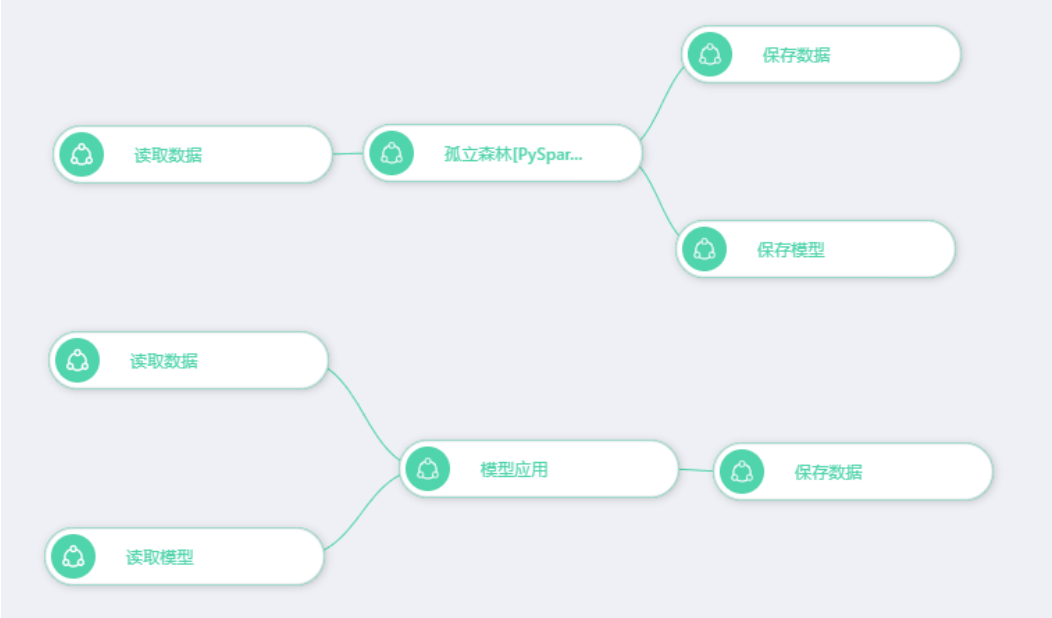
图 7-45 数据样本

	Time	V1	V2	V3	V4	V5	V6	V7	V8
1	0.0	-1.3598071336738	-0.0727811733098497	2.53634673796914	1.37815522427443	-0.33832076994251803	0.462387777762292	0.239598554061257	0.0986979012610507
2	0.0	1.1918571113148602	0.26615071205963	0.16648011335321	0.448154078460911	0.0600176492822243	-0.0823608088155687	-0.0788029833323113	0.0851016549148104
3	1.0	-1.35835406159823	-1.3401630747360902	1.77320934263119	0.3797795930343279	-0.503198133318193	1.80049938079263	0.7914609564504219	0.247675786588991 03
4	1.0	-0.9662717115720871	-0.185226008082898	1.79299333957872	-0.863291275036453	-0.0103088796030823	1.24720316752486	0.23760893977178	0.377435874652262
5	2.0	-1.1582330934952298	0.8777367548484508	1.548717846511	0.40303393395512105	-0.40719337731165295	0.0959214624684256	0.5929407453855451	-0.270532677192281 97
6	2.0	-0.4259658844124 54	0.960523044882985	1.1411093423221899	-0.16825207976030201	0.42098688077219004	-0.0297275516639742	0.476200948720027 05	0.260314333074874
7	4.0	1.22965763450793	0.141003507049326	0.0453707735899449	1.20261273673594	0.19188098859764496	0.272708122899098	-0.005159002882509829	0.0812129398830894
8	7.0	-0.6442694423481 46	1.4179635454738502	1.0743803763556	-0.49219901849501496	0.9489340947641569	0.428118462833089	1.1206313583835301	-3.807864238735890 5
9	7.0	-0.8942860822028199	0.286157196276544	-0.113192212729871	-0.271526130088604	2.6695986595986	3.7218180611275096	0.370145127676916 04	0.851084443200905
10	9.0	-0.3382617524257 4996	1.11959337641566	1.04436655157316	-0.222187276738296	0.49936080649727	-0.24676110061991	0.651583206489972	0.0695385865186387
11	10.0	1.4490437811471497	-1.1763388253596598	0.913859832832795	-1.3756666549994299	-1.97138316545323	-0.629152138897 34	-1.4232356010358997	0.0484558879088564
12	10.0	0.38497821518095	0.6161094591764721	-0.874299702595052	-0.0940186259679115	2.92458437838817	3.31702716826156	0.470454671805878 97	0.538247228376955
13	10.0	1.249998742053	-1.221636809218 16	0.383930151282291	-1.2348986876689199	-1.48541947377961	-0.753230164566149 1	-0.689404975426345	-0.227487227519552
14	11.0	1.0693735878819002	0.287722129331455	0.8286127266342809	2.71252042961718	-0.178398016248009	0.337543730282967 97	-0.096716861739596 2	0.115981735546596 98
15	12.0	-2.7918547659338997	-0.327770756658658	1.64175016056605	1.76747274389883	-0.136588446465306	0.807596468265319 9	-0.422911389711497	-1.907107476240 96

配置流程

下图上边部分运行孤立森林算子，得到异常检测结果和孤立森林模型，下边部分加载保存的模型和新的数据进行预测。

图 7-46 配置流程



参数设置

版权所有 © 华为云计算技术有限公司

图 7-47 参数设置界面

查看结果

图 7-48 查看运行结果

	V26	V27	V28	Amount	Class	isolation_forest_assembled_features	assembled_features	outlier_score	prediction
1	-0.189114843888824	0.13355837674038698	-0.02105305345382 15	149.62	0	4038698,-0.02105305345382 15,149.62]	76,0.5654080810508573]	0.3733114094406259	0.0
2	0.125894532368176	-0.00898309914322813	0.0147241691924927	2.69	0	9914322813,0.0147241691924927,2.69]	0.010165403943502247]	0.3500874726797695	0.0
3	-0.139096571514147	-0.05535279403842 61	-0.05975184059292 04	378.66	0	0384261,-0.05975184059292 04,378.66]	34,1.4309412108723274]	0.4714709531599 1144	0.0
4	-0.221928444458406 97	0.06272284872930 33	0.0614576285006353	123.5	0	487293033,0.0614576285006353,123.5]	32,0.4667016308633931]	0.3961689139277004	0.0
5	0.502292224181569	0.21942222951334 8	0.21515314749920603	69.99	0	29513348,0.21515314749920603,69.99]	3,0.26448945055974804]	0.3805561333731212	0.0
6	0.105914779097956 98	0.25384422473933 7	0.0810802569229443	3.67	0	4224739337,0.0810802569229443,3.67]	5,0.013868785305819 05]	0.353860024882022	0.0
7	-0.257236845917138 97	0.03450742974384 13	0.00516776890624 91605	4.99	0	7438413,0.00516776890624 91605,4.99]	0.018857013263225357]	0.3724813038632 5795	0.0
8	-0.05163429692624 94	-1.20692108094258	-1.08533918832377	40.8	0	92108094258,-1.08533918832377,40.8]	42,0.15418159141074 04]	0.4866391168784549	0.0
9	-0.38415730770229 4	0.01174735645819 96	0.142404329921 47	93.2	0	473564581996,0.142404329921 47,93.2]	96,0.3521991254774756]	0.4025265158751509	0.0
10	0.09419883395149 61	0.24621930461992 603	0.0830756493473326	3.68	0	0461992603,0.0830756493473326,3.68]	0.013906574911556977]	0.3590252792519574	0.0
11	-0.12947795372661802	0.04284987093814 61	0.0162532619375515	7.8	0	98709381461,0.0162532619375515,7.8]	0.029475892475582723]	0.3850524235384043	0.0
12	-0.49220829534001703	0.04247244191902 7	-0.05433738837321 22	9.99	0	441919027,-0.05433738837321 22,9.99]	7,0.037751816132188 64]	0.4209990951949953	0.0
13	-0.35499003967396 21	0.02641554907761 07	0.0424220887282304	121.5	0	490776107,0.0424220887282304,121.5]	5,0.45914370971580776]	0.3871549734124868	0.0
14	0.10409415316278098	0.02149105836431 89	0.021293311477486	27.5	0	10583643189,0.021293311477486,27.5]	3,0.10392141577929806]	0.3615917189853906 6	0.0
15	-0.235557217541 17	-0.16477751177654	-0.03015363655922 53	58.8	0	751177654,-0.03015363655922 53,58.8]	74,0.22220288173900 82]	0.4326369233363324	0.0
16	-0.08708647321469 62	-0.18099750009272 103	0.129394059390202	15.99	0	0009272103,0.129394059390202,15.99]	1,0.06042557957494458]	0.4022230856051522	0.0
17	-0.38226057411321707	0.09280918746048 7	0.0375055169810008	12.99	0	187460487,0.0375055169810008,12.99]	0.049088697853566614]	0.3509998870064638	0.0
18	0.04902672863395099 4	0.07969239915515 05	0.1310237894523 11	0.89	0	23991551505,0.1310237894523 11,0.89]	.0033632749106754646]	0.3755388831300 2254	0.0
19	-0.621272013713977	0.39205328955774 4	0.9495942455048 46	46.8	0	532895577 44,0.9495942455048 46,46.8]	3,0.17685535485349632]	0.4933634697762262 5	0.0

7.5.1.1.24 皮尔森系数

概述

皮尔森系数是一种线性相关系数，用于反映两个变量线性相关程度的统计量。选择输入的dataframe中的两列数值列，计算其皮尔森系数。

版权所有 © 华为云计算技术有限公司

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

参数	子参数	参数说明
output	output_port_1	dataframe类型的皮尔森系数结果

参数说明

参数	是否必选	参数说明	默认值
input_col1	是	皮尔森系数计算第一列列名	""
input_col2	是	皮尔森系数计算第二列列名	""

样例

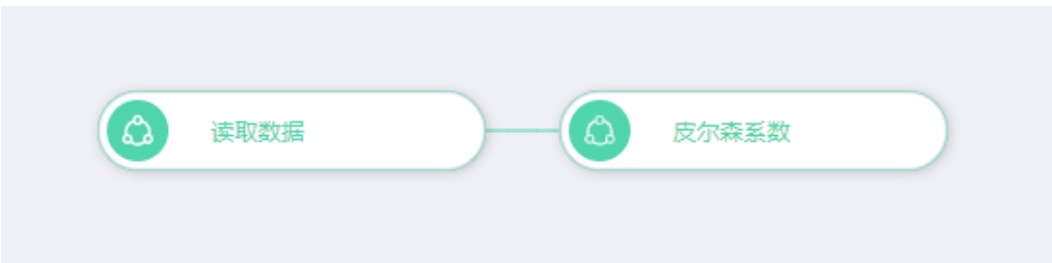
数据样本

图 7-49 数据样本

f1	f2
1	2.2
2	3.9
3	5.8
4	8.3
5	10.7
6	12.3
7	13.3
8	15.6
9	17.9
10	20.1

配置流程

运行流程



版权所有 © 华为云计算技术有限公司

算法参数设置

查看结果

表 7-6 查看运行结果

col1_name	col2_name	total	valid	pearson
f1	f2	10	10	0.9842079934534723

7.5.1.1.25 离散特征分析

概述

离散值特征分析通过每个离散特征的gini，entropy，gini gain，information gain，information gain ratio等和每个离散值对应的gini，entropy指标，方便对离散特征进行理解。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

参数	子参数	参数说明
output	output_cnt_table	指向一个pyspark的DataFrame类型对象，该对象中包含各个特征及其取值的统计信息
output	output_value_table	指向一个pyspark的DataFrame类型对象，该对象中包含各个特征的gini，entropy，gini gain，information gain，information gain ratio指标

版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
output	output_enum_value_table	指向一个pyspark的DataFrame类型对象，该对象中包含各个特征取值的gini，entropy指标

参数说明

参数	是否必选	参数说明	默认值
feature_cols	是	待分析的特征名称	""
label_col	是	标签列的名称	""
enable_sparse	是	输入数据是否为稀疏格式，取值为**{true,false}**	false
kv_delimiter	是	当输入数据为稀疏格式时，kv对之间的分隔符	":"
item_delimiter	是	当输入数据为稀疏格式时，key和value之间的分隔符	","
sparse_feature_list	否	稀疏格式的特征名称	""

样例

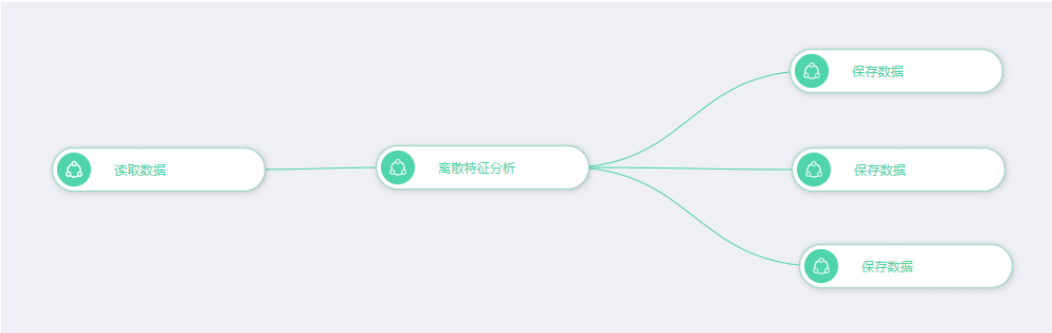
数据样本

```
f1,f2,label
1,1,0
1,1,1
1,1,1
1,0,1
1,0,1
2,0,0
2,0,1
```

配置流程

运行流程

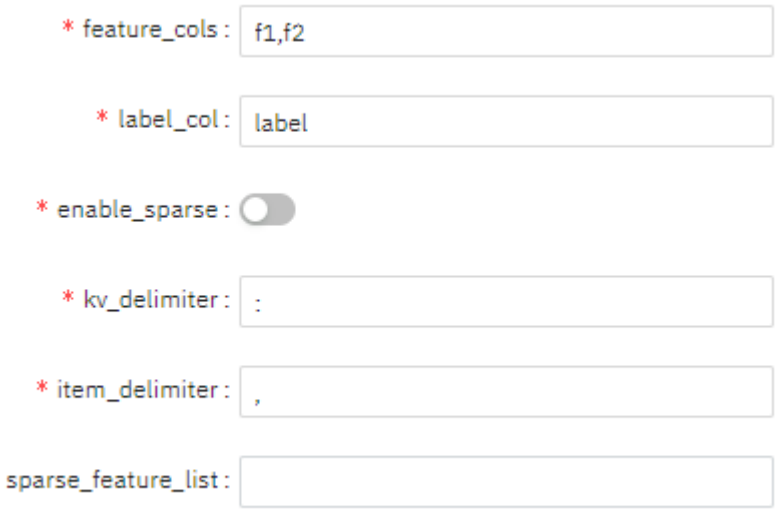
图 7-50 运行流程



版权所有 © 华为云计算技术有限公司

参数设置

图 7-51 参数设置



查看结果

output_cnt_table:

```
col_name,col_value,label_value,cnt
f2,1,1,2
f2,1,0,1
f2,0,1,3
f2,0,0,1
f1,1,1,4
f1,1,0,1
f1,2,0,1
f1,2,1,1
```

output_value_table:

```
col_name,feature_gini,feature_entropy,feature_gini_gain,feature_entropy_gain,feature_entropy_ratio
f2,0.40476190476190477,0.8571428571428571,0.003401360544217691,0.00597771142377390 15,0.006925696874193348
f1,0.37142857142857133,0.8013772106338303,0.03673469387755113,0.06174335793280072 4,0.07153503251039055
```

output_enum_value_table：

```
col_name,col_value,feature_value_gini,feature_value_entropy
f2,1,0.19047619047619047,0.39355535745192405
f1,1,0.22857142857142845,0.5156629249195446
f2,0,0.21428571428571427,0.46358749969093305
f1,2,0.14285714285714285,0.2857142857142857
```

7.5.1.2 数据处理

7.5.1.2.1 修改列名

概述

修改数据集表头的列名。

版权所有 © 华为云计算技术有限公司

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

参数说明

参数	子参数	参数说明
new_column_name_map_str	-	旧列名和新列名组成的格式化字符串，例如： "column_a:column_a_new" "column_a:column_a_new,column_b:column_b_new"

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "new_column_name_map_str": ""  # @param
{"label":"new_column_name_map_str","type":"string","required":"true","helpTip":""}
}
change_column_name____id___ = MLSChangeColumnName(**params)
change_column_name____id___.run()
# @output {"label":"dataframe","name":"change_column_name____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.2.2 数据集列合并

概述

将两个包含相同行数的数据集，按照列拼接，形成一个新的数据集。

输入

两个行数相同的数据集

参数	子参数	参数说明
inputs	left_dataframe	数据集一，列合并后将在左边
	right_dataframe	数据集二，列合并后将在右边

版权所有 © 华为云计算技术有限公司

输出

数据集

参数说明

无

样例

```
inputs = {
    "left_dataframe": None,  # @input {"label":"left_dataframe","type":"DataFrame"}
    "right_dataframe": None  # @input {"label":"right_dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs
}
column_append____id___ = MLSColumnAppend(**params)
column_append____id___.run()
# @output {"label":"dataframe","name":"column_append____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.2.3 数据集聚合

概述

对数据集进行各种聚合运算，包括求平均值、最大值、最小值、方差，对某些列执行分组操作。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

 版权所有 © 华为云计算技术有限公司

参数说明

参数	子参数	参数说明
agg_operators_str	-	代表各种聚合操作的格式化字符串，例如： "sum,old_column_a,new_column_a" "sum,old_column_a,new_column_a;covar,old_column_b,new_column_b,additional_column_b" 聚合操作有： sum：求和 sum_distinct：去重后求和 avg：均值 avg_distinct：去重后求均值 min：最小值 max：最大值 count：计数 count_distinct：去重后计数 stddev_pop：标准差 stddev_samp：样本标准差 var_pop：方差 var_samp：样本方差 covar_pop：协方差 covar_samp：样本协方差 corr：相关系数 percentile_approx：近似百分比
group_by_columns_str	-	代表对某些列进行分组操作的格式化字符串，例如： "column_a" "column_a,column_b,column_c"

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "agg_operators_str": "",  # @param
{"label":"agg_operators_str","type":"string","required":"true","helpTip":""}
    "group_by_columns_str": ""  # @param
{"label":"group_by_columns_str","type":"string","required":"true","helpTip":""}
}
dataset_aggregate____id___ = MLSDatasetAggerate(**params)
dataset_aggregate____id___.run()
# @output {"label":"dataframe","name":"dataset_aggregate____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```


版权所有 © 华为云计算技术有限公司

7.5.1.2.4 数据集行合并

概述

多个数据集按照行合并为一个数据集。

输入

参数	子参数	参数说明
inputs	dataframe_1	被合并的数据集一
	dataframe_2	被合并的数据集二
	dataframe_3	被合并的数据集三（可缺省）
	dataframe_4	被合并的数据集四（可缺省）

输出

合并后的新数据集。

参数说明

无

样例

配置流程

版权所有 © 华为云计算技术有限公司

图 7-52 配置流程

7.5.1.2.5 数据集行过滤

概述

根据过滤条件，对数据集按照行进行过滤。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

版权所有 © 华为云计算技术有限公司

输出

参数	子参数	参数说明
output	output_port_1	output为字典类型，output_port_1为pyspark中的DataFrame类型对象，为数据集过滤结果

参数说明

参数	是否必选	参数说明	默认值
column_name	是	列名，对该列按照过滤条件进行数据集的行过滤，不同列之间用分号分隔	无

condition_map_str	是	过滤条件组装后的格式化字符串，格式参考："!=:filter_value;IS NULL;BETWEEN:left_value,right_value;REGEXP:expr"
		该字符串将被分号分割为多个条件，每个条件对应的字符串将被冒号分割为过滤条件和过滤值，最终将被转换成如下字典：
		{
		"=": "filter_value",
		"!=": "filter_value",
		">": "filter_value",
		">=": "filter_value",
		"<": "filter_value",
		"<=": "filter_value",
		"IS NULL": "",
		"IS NOT NULL": "",
		"BETWEEN": "filter_value_left,filter_value_right",
		"NOT BETWEEN": "filter_value_left,filter_value_right",
		"LIKE": "filter_value_expr",
		"NOT LIKE": "filter_value_expr",
		"REGEXP": "filter_value_expr"
		}
			无

版权所有 © 华为云计算技术有限公司

样例

数据样本

鸢尾花数据集，species列代表鸢尾花种类，共有Iris-setosa、Iris-versicolor和Iris-virginica三种类别，每种类别样本数量为50。

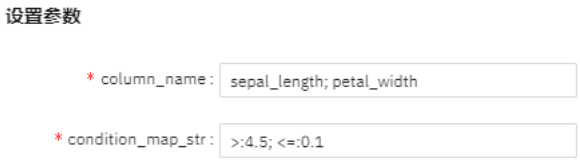
sepal_length	sepal_width	petal_length	petal_width	species
6.3	3.3	6.0	2.5	Iris-virginica
7.1	3.0	5.9	2.1	Iris-virginica
6.5	3.0	5.8	2.2	Iris-virginica
7.6	3.0	6.6	2.1	Iris-virginica
7.2	3.6	6.1	2.5	Iris-virginica
6.8	3.0	5.5	2.1	Iris-virginica
5.8	2.8	5.1	2.4	Iris-virginica
6.4	3.2	5.3	2.3	Iris-virginica
7.7	3.8	6.7	2.2	Iris-virginica
7.7	2.6	6.9	2.3	Iris-virginica
6.9	3.2	5.7	2.3	Iris-virginica
6.7	3.3	5.7	2.1	Iris-virginica
6.4	2.8	5.6	2.1	Iris-virginica
6.4	2.8	5.6	2.2	Iris-virginica
7.7	3.0	6.1	2.3	Iris-virginica
6.3	3.4	5.6	2.4	Iris-virginica
6.9	3.1	5.4	2.1	Iris-virginica
6.7	3.1	5.6	2.4	Iris-virginica
6.9	3.1	5.1	2.3	Iris-virginica
6.8	3.2	5.9	2.3	Iris-virginica
6.7	3.3	5.7	2.5	Iris-virginica
6.7	3.0	5.2	2.3	Iris-virginica
6.2	3.4	5.4	2.3	Iris-virginica

配置流程

运行流程

参数设置

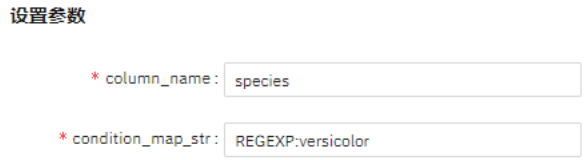
图 7-53 参数设置（过滤出 sepal_length>4.5 且 petal_length<=0.1 的数据）



版权所有 © 华为云计算技术有限公司

图 7-54 参数设置（选择该列中后缀为"setosa"的样本）

图 7-55 参数设置（选择该列中包含"versicolor"的样本）



查看结果

图 7-56 参数设置 1 结果

sepal_length	sepal_width	petal_length	petal_width	species
4.9	3.1	1.5	0.1	Iris-setosa
4.8	3.0	1.4	0.1	Iris-setosa
5.2	4.1	1.5	0.1	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa

图 7-57 参数设置 2 结果

sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5.0	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5.0	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa

版权所有 © 华为云计算技术有限公司

图 7-58 参数设置 3 结果

sepal_length	sepal_width	petal_length	petal_width	species
7.0	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
5.5	2.3	4.0	1.3	Iris-versicolor
6.5	2.8	4.6	1.5	Iris-versicolor
5.7	2.8	4.5	1.3	Iris-versicolor
6.3	3.3	4.7	1.6	Iris-versicolor
4.9	2.4	3.3	1.0	Iris-versicolor
6.6	2.9	4.6	1.3	Iris-versicolor
5.2	2.7	3.9	1.4	Iris-versicolor

7.5.1.2.6 数据集连接

概述

“连接”节点是关系数据库中常用的方法之一，用于以特定的方式将两个数据集联接在一起。

输入

参数	子参数	参数说明
inputs	left_dataframe	inputs为字典类型，left_dataframe为执行连接操作的左数据集
	right_dataframe	inputs为字典类型，right_dataframe为执行连接操作的右数据集

输出

参数	子参数	参数说明
outputs	output_port_1	dataframe类型的数据集合并结果

版权所有 © 华为云计算技术有限公司

参数说明

参数名称	是否必选	参数描述	默认值
join_column_pairs_str	是	关联条件，等式对之间以”；“分割，等式之间以”，“分割； eg. left_col1,right_col1;left_col2,right_col2;	无
join_type	是	支持左连接、右连接、内连接和全连接(left_join、right_join、inner_join和full join)。	left_join
mapjoin	否	是否进行mapjoin优化(将小表进行broadcoast广播)。	true
output_left_cols	否	左表输出字段列	如果不选择，则默认为所有字段
output_right_cols	否	右表输出字段列	如果不选择，则默认为所有字段

样例

数据样本

"Persons" 表：

Id_P	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing

"Orders" 表：

Id_O	OrderNo	Id_P
1	77895	3

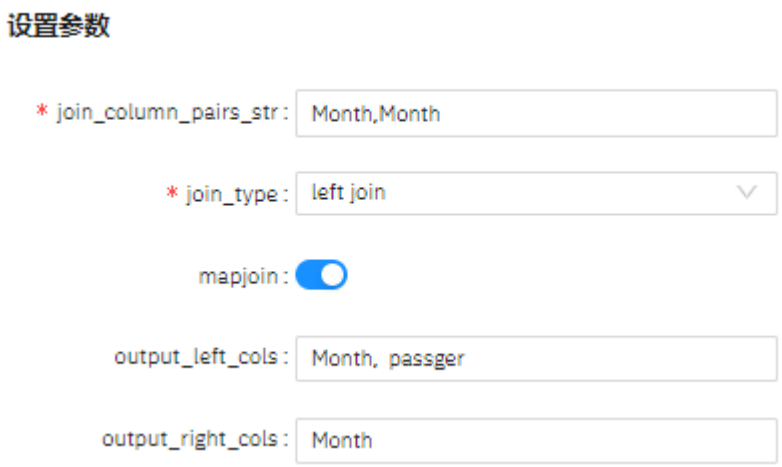
版权所有 © 华为云计算技术有限公司

Id_O	OrderNo	Id_P
2	44678	3
3	22456	1
4	24562	1
5	34764	65

配置流程

运行流程

算法参数设置



查看结果

	LastName	FirstName	OrderNo
1	Adams	John	22456
2	Adams	John	24562
3	Carter	Thomas	77895
4	Carter	Thomas	44678
5	Bush	George	

版权所有 © 华为云计算技术有限公司

7.5.1.2.7 数据集抽样

概述

按照配置的比例参数，对数据集进行随机抽样。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

参数说明

参数	子参数	参数说明
fraction	-	抽样比例
seed	-	随机抽样的随机数种子

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "fraction": 0.7,  # @param
{"label":"fraction","type":"number","required":"true","range":"(0.0,1.0)","helpTip": ""}
    "seed": 0  # @param
{"label":"seed","type":"integer","required":"true","range":"(0,2147483647]","helpTip":"seed"}
}
dataset_sample____id___ = MLSDatasetSample(**params)
dataset_sample____id___.run()
# @output {"label":"dataframe","name":"dataset_sample____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.2.8 数据集拆分

概述

将数据集按照比例或阈值拆分为两个子数据集。

版权所有 © 华为云计算技术有限公司

输入

参数	子参数	参数说明
inputs	dataframe	pyspark中DataFrame类型的对象

输出

参数	子参数	参数说明
output	output_port_1	output为字典类型，output_port_1为pyspark中的DataFrame类型对象，为拆分的子数据集1。
output	output_port_2	output_port_2为pyspark中的DataFrame类型对象，为拆分的子数据集2。

参数说明

参数	是否必选	参数说明	默认值
fraction	否	分割比例，比例值对应第一个数据集的行数。	0.7
id_col	否	id列，按比例拆分的情况下，如果设置该列，则该列相同的样本不会被拆分，按阈值拆分则设置该列无效。	无
threshold_col	否	用于阈值拆分的列。	无
thrshold	否	拆分阈值。	0.0
seed	否	随机数种子。	1234

说明

如果设置fraction，则按照比例拆分，阈值列和阈值设置无效。

样例

数据样本

鸢尾花数据集，species列代表鸢尾花种类，共有Iris-setosa、Iris-versicolor和Iris-virginica三种类别，每种类别样本数量为50。

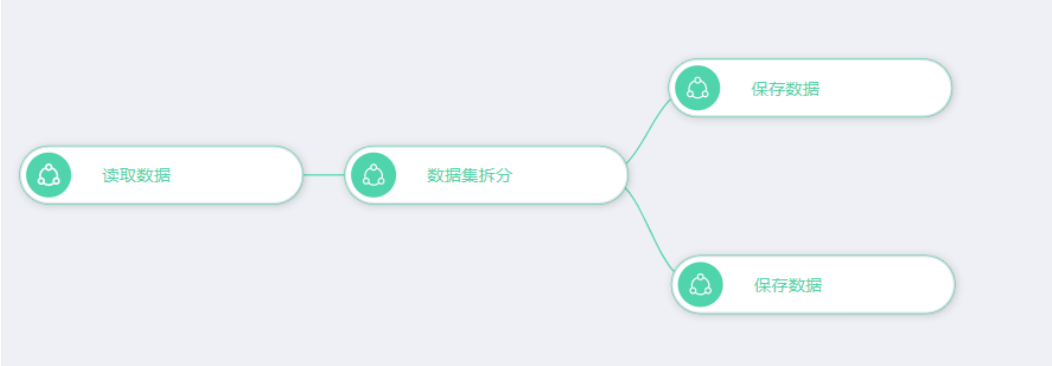
版权所有 © 华为云计算技术有限公司

图 7-59 数据样本

	sepal_length	sepal_width	petal_length	petal_width	species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3	1.4	0.1	Iris-setosa
14	4.3	3	1.1	0.1	Iris-setosa
15	5.8	4	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
18	5.1	3.5	1.4	0.3	Iris-setosa
19	5.7	3.8	1.7	0.3	Iris-setosa
20	5.1	3.8	1.5	0.3	Iris-setosa

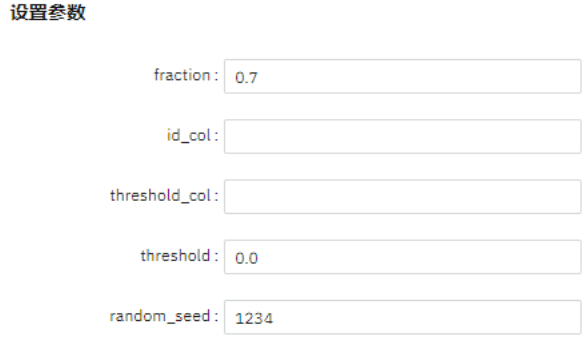
配置流程

运行流程



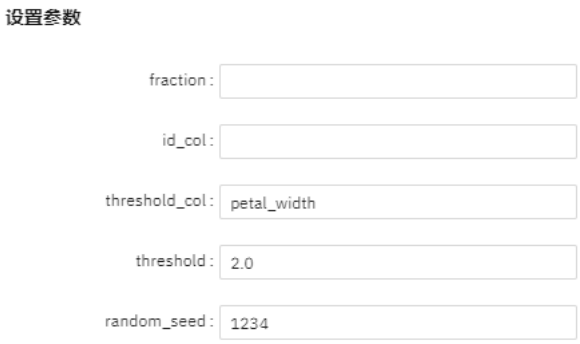
参数设置

图 7-60 参数设置（按比例拆分）



版权所有 © 华为云计算技术有限公司

图 7-61 参数设置（按阈值拆分）



按petal_width列划分，小于等于2.0的数据划分至子数据集1，大于2.0的数据划分至子数据集2。

查看结果

- 按比例拆分

图 7-62 子数据集 1

sepal_length	sepal_width	petal_length	petal_width	species
4.4	3.2	1.3	0.2	Iris-setosa
4.5	2.3	1.3	0.3	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
4.6	3.2	1.4	0.2	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
4.6	3.6	1.0	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.7	3.2	1.6	0.2	Iris-setosa
4.8	3.0	1.4	0.1	Iris-setosa
4.8	3.0	1.4	0.3	Iris-setosa
4.8	3.4	1.6	0.2	Iris-setosa
4.8	3.4	1.9	0.2	Iris-setosa
4.9	2.4	3.3	1.0	Iris-versicolor
4.9	2.5	4.5	1.7	Iris-virginica
4.9	3.0	1.4	0.2	Iris-setosa

版权所有 © 华为云计算技术有限公司

图 7-63 子数据集 2

sepal_length	sepal_width	petal_length	petal_width	species
4.3	3.0	1.1	0.1	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.4	3.0	1.3	0.2	Iris-setosa
4.8	3.1	1.6	0.2	Iris-setosa
5.0	3.3	1.4	0.2	Iris-setosa
5.0	3.4	1.5	0.2	Iris-setosa
5.0	3.6	1.4	0.2	Iris-setosa
5.1	3.4	1.5	0.2	Iris-setosa
5.1	3.8	1.5	0.3	Iris-setosa
5.2	2.7	3.9	1.4	Iris-versicolor
5.2	4.1	1.5	0.1	Iris-setosa
5.3	3.7	1.5	0.2	Iris-setosa
5.4	3.4	1.5	0.4	Iris-setosa
5.5	2.3	4.0	1.3	Iris-versicolor
5.6	2.9	3.6	1.3	Iris-versicolor

- 按阈值拆分

图 7-64 子数据集 1

sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5.0	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5.0	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
5.4	3.7	1.5	0.2	Iris-setosa
4.8	3.4	1.6	0.2	Iris-setosa
4.8	3.0	1.4	0.1	Iris-setosa
4.3	3.0	1.1	0.1	Iris-setosa
5.8	4.0	1.2	0.2	Iris-setosa
5.7	4.4	1.5	0.4	Iris-setosa
5.4	3.9	1.3	0.4	Iris-setosa

版权所有 © 华为云计算技术有限公司

图 7-65 子数据集 2

sepal_length	sepal_width	petal_length	petal_width	species
6.3	3.3	6.0	2.5	Iris-virginica
7.1	3.0	5.9	2.1	Iris-virginica
6.5	3.0	5.8	2.2	Iris-virginica
7.6	3.0	6.6	2.1	Iris-virginica
7.2	3.6	6.1	2.5	Iris-virginica
6.8	3.0	5.5	2.1	Iris-virginica
5.8	2.8	5.1	2.4	Iris-virginica
6.4	3.2	5.3	2.3	Iris-virginica
7.7	3.8	6.7	2.2	Iris-virginica
7.7	2.6	6.9	2.3	Iris-virginica
6.9	3.2	5.7	2.3	Iris-virginica
6.7	3.3	5.7	2.1	Iris-virginica
6.4	2.8	5.6	2.1	Iris-virginica
6.4	2.8	5.6	2.2	Iris-virginica
7.7	3.0	6.1	2.3	Iris-virginica
6.3	3.4	5.6	2.4	Iris-virginica
6.9	3.1	5.4	2.1	Iris-virginica
6.7	3.1	5.6	2.4	Iris-virginica
6.9	3.1	5.1	2.3	Iris-virginica
6.8	3.2	5.9	2.3	Iris-virginica
6.7	3.3	5.7	2.5	Iris-virginica
6.7	3.0	5.2	2.3	Iris-virginica
6.2	3.4	5.4	2.3	Iris-virginica

7.5.1.2.9 数据集行去重

概述

“去重”节点用于删除数据集中的重复行（假如有两行相同，保留其中一行）。

对于那些不允许有重复记录输入的节点算法，可以先使用该算法做预处理。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

去重后的数据集

参数说明

无

版权所有 © 华为云计算技术有限公司

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs
}
drop_duplicates____id___ = MLSDropDuplicates(**params)
drop_duplicates____id___.run()
# @output {"label":"dataframe","name":"drop_duplicates____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.2.10 执行 spark sql 脚本

概述

对数据集执行spark sql脚本操作。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

参数说明

参数	子参数	参数说明
sql_string	-	sql脚本的格式化字符串，例如： "SELECT age FROM adult_table WHERE age > 50"

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "sql_string": ""  # @param {"label":"sql_string","type":"string","required":"true","helpTip":""}
}
execute_sql____id___ = MLSExecuteSql(**params)
execute_sql____id___.run()
# @output {"label":"dataframe","name":"execute_sql____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```


版权所有 © 华为云计算技术有限公司

7.5.1.2.11 替换

概述

“替换”节点用于对数据中指定属性名满足条件的内容进行替换。

用户可以根据需要，从输入数据集中依据条件替换某一个或多个属性，可选的条件包括“=”、“!=”、“Like”、“Between”等。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

参数说明

参数	子参数	参数说明
conditions_str	-	替换条件组成的格式化字符串，例如: "column_a,>,50,1" 表示将column_a列大于50的值替换为1 "column_b,like,HS%,IS HS;column_c,between,50,100,1;column_d,is null,0.0"

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "conditions_str": ""  # @param {"label":"conditions_str","type":"string","required":"true","helpTip":""}
}
field_replace____id___ = MLSFieldReplace(**params)
field_replace____id___.run()
# @output {"label":"dataframe","name":"field_replace____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.2.12 缺失值填充

概述

“缺失值填充”节点用来将某些列出现的缺失值（如空值、指定的值）替换为均值或者中位数。

版权所有 © 华为云计算技术有限公司

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

参数说明

参数	子参数	参数说明
input_features_str	-	列名组成的字符串，例如: "column_a" "column_a,column_b"
missing_value	-	类型为数值，表示该值为缺失值，将要被填充
strategy	-	填充策略，支持mean和median
output_col_postfix	-	输出特征列的后缀

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "outer_pipeline_stages": None,
    "input_features_str": "",  # @param
{"label":"input_features_str","type":"string","required":"true","helpTip":""}
    "missing_value": "",  # @param
{"label":"missing_value","type":"number","required":"false","range":"(none,none)","helpTip":""}
    "strategy": "mean",  # @param
{"label":"strategy","type":"enum","options":"mean,median","required":"true","helpTip":""}
    "output_col_postfix": "_impute"  # @param
{"label":"output_col_postfix","type":"string","required":"true","helpTip":""}
}
missing_value_impute____id___ = MLSMissingValueImpute(**params)
missing_value_impute____id___.run()
# @output {"label":"dataframe","name":"missing_value_impute____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.2.13 缺省值填充

概述

通过给定一个缺省值的配置表，来实现将输入表的缺省值或固定值填充为定义的值。

版权所有 © 华为云计算技术有限公司

- 将数值型的空值替换为最大值，最小值，均值或者一个自定义的值。
- 将字符串类型、日期类型的空值、或者固定值，替换为一个自定义的值。
- 数值型替换可以自定义，也可以直接选择替换成数值最大值，最小值或者均值。

组件配置方式

方式一：以配置表方式确定填充策略

输入

参数	子参数	参数说明
inputs	dataDF	inputs为字典类型，dataDF为pyspark中的DataFrame类型对象，输入数据
inputs	paramDF	inputs为字典类型，paramDF为pyspark中的DataFrame类型对象，待修改字段的配置

输出

参数	参数说明
dataDF	inputs为字典类型，dataDF为pyspark中的DataFrame类型对象，填充后的数据输出
paramDF	inputs为字典类型，paramDF为pyspark中的DataFrame类型对象，已被修改字段的配置

样例

```
dataDF:
+---+----+-----------+--------+---------+-------+----+
|id |age |job        |marital |education|housing|loan|
+---+----+-----------+--------+---------+-------+----+
|0  |59  |admin.     |married |secondary|yes    |no  |
|1  |56  |admin.     |married |secondary|no     |no  |
|2  |41  |technician |married |secondary|yes    |no  |
|3  |55  |services   |married |secondary|yes    |no  |
|4  |54  |admin.     |married |tertiary |no     |no  |
|5  |null|management |single  |tertiary |yes    |yes |
|6  |56  |management |married |tertiary |yes    |yes |
|7  |60  |retired    |divorced|secondary|yes    |no  |
|8  |39  |technician |single  |unknown  |yes    |no  |
|9  |37  |technician |married |secondary|yes    |no  |
|10 |34  |admin.     |married |secondary|no     |no  |
|11 |null|null       |divorced|secondary|yes    |no  |
|12 |28  |services   |single  |secondary|yes    |no  |
|13 |30  |technician |married |secondary|yes    |no  |
|14 |36  |technician |married |secondary|yes    |yes |
|15 |37  |admin.     |single  |secondary|yes    |yes |
|16 |null|blue-collar|married |secondary|yes    |no  |
|17 |53  |services   |divorced|primary  |yes    |yes |
+---+----+-----------+--------+---------+-------+----+
paramDF
+---------
+-----------------------------------------------------------------------------------------------------------------------+
```


版权所有 © 华为云计算技术有限公司

```
|feature  |json                                                                                                   |
+---------
+-------------------------------------------------------------------------------------------------------------+
|age      |{"name":"fillMissingValues","type":"IntegerType","paras":
{"missing_value_type":"null","replaced_value":"45.0"}}      |
|job      |{"name":"fillMissingValues","type":"StringType","paras":
{"missing_value_type":"null","replaced_value":"blue-collar"}}|
|education|{"name":"fillMissingValues","type":"StringType","paras":
{"missing_value_type":"unknown","replaced_value":"primary"}} |
+---------
+-------------------------------------------------------------------------------------------------------------+
结果:
数据输出结果:
+---+---+-----------+--------+---------+-------+----+
|id |age|job        |marital |education|housing|loan|
+---+---+-----------+--------+---------+-------+----+
|0  |59 |admin.     |married |secondary|yes    |no  |
|1  |56 |admin.     |married |secondary|no     |no  |
|2  |41 |technician |married |secondary|yes    |no  |
|3  |55 |services   |married |secondary|yes    |no  |
|4  |54 |admin.     |married |tertiary |no     |no  |
|5  |45 |management |single  |tertiary |yes    |yes |
|6  |56 |management |married |tertiary |yes    |yes |
|7  |60 |retired    |divorced|secondary|yes    |no  |
|8  |39 |technician |single  |primary  |yes    |no  |
|9  |37 |technician |married |secondary|yes    |no  |
|10 |34 |admin.     |married |secondary|no     |no  |
|11 |45 |blue-collar|divorced|secondary|yes    |no  |
|12 |28 |services   |single  |secondary|yes    |no  |
|13 |30 |technician |married |secondary|yes    |no  |
|14 |36 |technician |married |secondary|yes    |yes |
|15 |37 |admin.     |single  |secondary|yes    |yes |
|16 |45 |blue-collar|married |secondary|yes    |no  |
|17 |53 |services   |divorced|primary  |yes    |yes |
+---+---+-----------+--------+---------+-------+----+
配置输出结果:
+---------
+-------------------------------------------------------------------------------------------------------------+
|feature  |json                                                                                                   |
+---------
+-------------------------------------------------------------------------------------------------------------+
|age      |{"name":"fillMissingValues","type":"IntegerType","paras":
{"missing_value_type":"null","replaced_value":"45.0"}}      |
|job      |{"name":"fillMissingValues","type":"StringType","paras":
{"missing_value_type":"null","replaced_value":"blue-collar"}}|
|education|{"name":"fillMissingValues","type":"StringType","paras":
{"missing_value_type":"unknown","replaced_value":"primary"}} |
+---------
+-------------------------------------------------------------------------------------------------------------+
```

方式二：以配置表方式确定填充策略

输入

参数	子参数	参数说明
inputs	dataDF	inputs为字典类型，dataDF为pyspark中的DataFrame类型对象，输入数据

输出

参数	参数说明
dataDF	inputs为字典类型，dataDF为pyspark中的DataFrame类型对象，填充后的数据输出
paramDF	inputs为字典类型，paramDF为pyspark中的DataFrame类型对象，已被修改字段的配置

参数说明

参数	参数说明
configs	第一个是列名1，被填充值1，填充值1；列名2，被填充值2，填充值2 例：col_double,null,mean;col_string,null-empty,str_type_empty

样例

```
dataDF:
+---+----+-----------+--------+---------+-------+----+
|id |age |job        |marital |education|housing|loan|
+---+----+-----------+--------+---------+-------+----+
|0  |59  |admin.     |married |secondary|yes    |no  |
|1  |56  |admin.     |married |secondary|no     |no  |
|2  |41  |technician |married |secondary|yes    |no  |
|3  |55  |services   |married |secondary|yes    |no  |
|4  |54  |admin.     |married |tertiary |no     |no  |
|5  |null|management |single  |tertiary |yes    |yes |
|6  |56  |management |married |tertiary |yes    |yes |
|7  |60  |retired    |divorced|secondary|yes    |no  |
|8  |39  |technician |single  |unknown  |yes    |no  |
|9  |37  |technician |married |secondary|yes    |no  |
|10 |34  |admin.     |married |secondary|no     |no  |
|11 |null|null       |divorced|secondary|yes    |no  |
|12 |28  |services   |single  |secondary|yes    |no  |
|13 |30  |technician |married |secondary|yes    |no  |
|14 |36  |technician |married |secondary|yes    |yes |
|15 |37  |admin.     |single  |secondary|yes    |yes |
|16 |null|blue-collar|married |secondary|yes    |no  |
|17 |53  |services   |divorced|primary  |yes    |yes |
+---+----+-----------+--------+---------+-------+----+
configs:
age,null,mean;job,null,blue-collar;education,unknown,primary

结果:
数据输出结果:
+---+---+-----------+--------+---------+-------+----+
|id |age|job        |marital |education|housing|loan|
+---+---+-----------+--------+---------+-------+----+
|0  |59 |admin.     |married |secondary|yes    |no  |
|1  |56 |admin.     |married |secondary|no     |no  |
|2  |41 |technician |married |secondary|yes    |no  |
|3  |55 |services   |married |secondary|yes    |no  |
|4  |54 |admin.     |married |tertiary |no     |no  |
|5  |45 |management |single  |tertiary |yes    |yes |
|6  |56 |management |married |tertiary |yes    |yes |
|7  |60 |retired    |divorced|secondary|yes    |no  |
|8  |39 |technician |single  |primary  |yes    |no  |
|9  |37 |technician |married |secondary|yes    |no  |
```


版权所有 © 华为云计算技术有限公司

```
|10 |34 |admin.      |married |secondary|no      |no  |
|11 |45 |blue-collar|divorced|secondary|yes     |no  |
|12 |28 |services   |single  |secondary|yes     |no  |
|13 |30 |technician |married |secondary|yes     |no  |
|14 |36 |technician |married |secondary|yes     |yes |
|15 |37 |admin.      |single  |secondary|yes     |yes |
|16 |45 |blue-collar|married |secondary|yes     |no  |
|17 |53 |services   |divorced|primary  |yes     |yes |
+---+---+-----------+--------+---------+-------+----+
配置输出结果:
+---------
+-------------------------------------------------------------------------------------------------------------------------------+
|feature  |json                                                                                                                 |
+---------
+-------------------------------------------------------------------------------------------------------------------------------+
|age      |{"name":"fillMissingValues","type":"IntegerType","paras":
{"missing_value_type":"null","replaced_value":"45.0"}}      |
|job      |{"name":"fillMissingValues","type":"StringType","paras":
{"missing_value_type":"null","replaced_value":"blue-collar"}}|
|education|{"name":"fillMissingValues","type":"StringType","paras":
{"missing_value_type":"unknown","replaced_value":"primary"}} |
+---------
+-------------------------------------------------------------------------------------------------------------------------------+
```

7.5.1.2.14 修改列数据类型

概述

修改数据集相应列的数据类型。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

参数说明

参数	子参数	参数说明
column_type_map_str	-	指定相应列的数据类型的规范化字符串，例如："column_a:string,column_b:integer"，列类型可以是：string,integer,long,float,double,bool,date,timestamp
timestamp_format	-	取值yyyy-MM-dd HH:mm:ss
date_format	-	取值yyyy-MM-dd

 版权所有 © 华为云计算技术有限公司

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "column_type_map_str": "",  # @param
{"label":"column_type_map_str","type":"string","required":"true","helpTip":""}
    "timestamp_format": "yyyy-MM-dd HH:mm:ss",  # @param
{"label":"timestamp_format","type":"string","required":"false","helpTip":""}
    "date_format": "yyyy-MM-dd"  # @param
{"label":"date_format","type":"string","required":"false","helpTip":""}
}
modify_data_type____id___ = MLSModifyDataType(**params)
modify_data_type____id___.run()
# @output {"label":"dataframe","name":"modify_data_type____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.2.15 数据集选择列

概述

根据输入数据集，选择一些列生成新的数据集。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

参数说明

参数	子参数	参数说明
select_columns_str	-	将选择的列名按照逗号分隔形成的字符串，例如："column_a" 或者 "column_a,column_b,column_c"

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "select_columns_str": ""  # @param
{"label":"select_columns_str","type":"string","required":"true","helpTip":""}
}
select_columns____id___ = MLSSelectColumns(**params)
select_columns____id___.run()
```

 版权所有 © 华为云计算技术有限公司

```
# @output {"label":"dataframe","name":"select_columns____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.2.16 设置元数据

概述

设置数据集的元数据信息。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

参数说明

参数	子参数	参数说明
column_type_map_str	-	指定相应列的数据类型的规范化字符串，例如："column_a:string,column_b:integer"，列类型可以是：string,integer,long,float,double,bool,date,timestamp

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "column_type_map_str": ""  # @param
{"label":"column_type_map_str","type":"string","required":"true","helpTip":""}
}
set_metadata____id___ = MLSSetMetadata(**params)
set_metadata____id___.run()
# @output {"label":"dataframe","name":"set_metadata____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.2.17 数据集按列排序

概述

对输入数据集，按照选择的一些列，进行排序，生成新的数据集。

 版权所有 © 华为云计算技术有限公司

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

参数说明

参数	子参数	参数说明
columns_str	-	选择的一些列，列名按照逗号分隔，例如："column_a"或者"column_a,column_b"
ascend_tags_str	-	对选择的列是升序还是降序，升序标记为1，降序标记为0，按照逗号分隔，例如: "1"或者"1,0"

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "columns_str": "",  # @param {"label":"columns_str","type":"string","required":"true","helpTip":""}
    "ascend_tags_str": ""  # @param {"label":"ascend_tags_str","type":"string","required":"true","helpTip":""}
}
sort_with_columns____id___ = MLSSortWithColumns(**params)
sort_with_columns____id___.run()
# @output {"label":"dataframe","name":"sort_with_columns____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.2.18 增加序列号

概述

提供的增加序号列组件。您可以在数据表的第一列追加ID列。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

版权所有 © 华为云计算技术有限公司

输出

数据集

参数说明

参数名	参数类型	是否必选	参数含义	默认值
selected_col_names	String	否	输入表中，参与训练的列，列名以英文逗号（,）分隔不限定数据类型。	-

样例

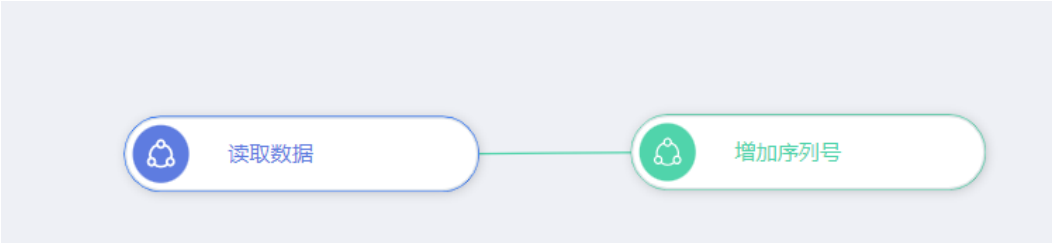
输入数据集

```
+---+-----------+--------+---------+-------+----+-----+
|age|        job| marital|education|housing|loan|label|
+---+-----------+--------+---------+-------+----+-----+
| 59|     admin.| married|secondary|    yes|  no|  yes|
| 56|     admin.| married|secondary|     no|  no|  yes|
| 41| technician| married|secondary|    yes|  no|  yes|
| 55|   services| married|secondary|    yes|  no|  yes|
| 54|     admin.| married| tertiary|     no|  no|  yes|
| 42| management|  single| tertiary|    yes| yes|  yes|
| 56| management| married| tertiary|    yes| yes|  yes|
| 60|    retired|divorced|secondary|    yes|  no|  yes|
| 39| technician|  single|  unknown|    yes|  no|  yes|
| 37| technician| married|secondary|    yes|  no|  yes|
| 34|     admin.| married|secondary|     no|  no|  yes|
| 55| unemployed|divorced|secondary|    yes|  no|  yes|
| 28|   services|  single|secondary|    yes|  no|  yes|
| 30| technician| married|secondary|    yes|  no|  yes|
| 36| technician| married|secondary|    yes| yes|  yes|
| 37|     admin.|  single|secondary|    yes| yes|  yes|
| 45|blue-collar| married|secondary|    yes|  no|  yes|
| 53|   services|divorced|  primary|    yes| yes|  yes|
| 38|     admin.|  single|secondary|    yes|  no|  yes|
| 30|blue-collar| married|secondary|    yes|  no|  yes|
+---+-----------+--------+---------+-------+----+-----+
only showing top 20 rows
```


版权所有 © 华为云计算技术有限公司

配置流程

运行流程



输出结果

```
+---+------------+--------+---------+-------+----+-----+---------+
|age|         job| marital|education|housing|loan|label|append_id|
+---+------------+--------+---------+-------+----+-----+---------+
| 53|  technician| married|secondary|    yes|  no|  yes|        0|
| 50|entrepreneur| married| tertiary|    yes|  no|  yes|        1|
| 57|     retired|divorced|  primary|     no| yes|  yes|        2|
| 47|    services| married|secondary|     no| yes|  yes|        3|
| 33|  technician| married|secondary|     no|  no|  yes|        4|
| 41|  technician| married|secondary|     no|  no|  yes|        5|
| 54|  unemployed|  single|secondary|     no|  no|  yes|        6|
| 64|     retired| married|  primary|     no|  no|  yes|        7|
| 39| blue-collar| married|secondary|    yes| yes|  yes|        8|
| 34|     student|  single|  unknown|     no|  no|  yes|        9|
| 68|     retired| married| tertiary|     no|  no|  yes|       10|
| 33|    services|  single|  unknown|     no|  no|  yes|       11|
| 42| blue-collar| married|  primary|    yes|  no|   no|       12|
| 58|      admin.| married|  primary|     no|  no|   no|       13|
| 47|  management| married|  primary|    yes|  no|   no|       14|
| 48|entrepreneur| married|secondary|     no|  no|   no|       15|
| 59|     retired|divorced|  primary|     no| yes|   no|       16|
| 31| blue-collar|divorced|secondary|    yes|  no|   no|       17|
| 33|  technician|  single|secondary|     no| yes|   no|       18|
| 29|  management|divorced| tertiary|    yes| yes|   no|       19|
+---+------------+--------+---------+-------+----+-----+---------+
only showing top 20 rows
```

7.5.1.2.19 普通表转 KV 表

概述

将普通的table表转为KV（Key:Value）格式的表。

KV表格式定义：Key是列名的index，Value支持BIGINT，DOUBLE和STRING类型。在该组件中可以输入用户定义的key_map表，是列名和Key的映射，但无论是否输入key_map表，该组件都会输出key_map表记录转化后的列名和Key的映射。例如1:10，2:20和3:30。

key_map表格式定义：包含列名和index的映射以及类型信息的col_name，col_index和col_datatype，这三列类型要求是STRING。

版权所有 © 华为云计算技术有限公司

说明与约束

- 转换后的结果表不会显示原表中的空值。您可以在结果表中指定需要保留的列，并且输出的列与原表的列一致。
- 如果存在输入Key_map表，则转化的列为Key_map表与KV表中Key的交集。（ 在col和keymap表中都存在 ）
- 如果存在的输入Key_map表与输入表类型冲突，则输出的Key_map表使用您指定的类型。(和读入的keymap中col_datatype和判断得到的datatype不同 以keymap的类型为主)
- 输入表中需要转换为KV的列只能为BIGINT或DOUBLE类型。
- 当selected_col_name 为空时，默认选择整张表。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象
	keymap_dataframe	keymap_dataframe为输入的keymap表，类型：pyspark中的DataFrame类型对象

输出

参数	子参数	参数说明
outputs	output_dataframe	转化后的dataframe表
	output_keymap_dataframe	转化后的keymap_dataframe表

参数说明

参数名称	是否必选	参数描述	默认值
selected_col_names	否	选择的列名称，只能为BIGINT或DOUBLE类型。	默认选择整张表。
append_col_names	否	需要保留的列名称，该列会被原样写入至输出表中。	无
kv_delimiter	否	Key和Value的分割符。	半角冒号（:）

版权所有 © 华为云计算技术有限公司

参数名称	是否必选	参数描述	默认值
item_delimiter	否	KV间的分割符。默认为半角逗号（,）。	半角逗号（,）
convert_col2indexId	否	指定是否将列转换为编号。取值如下： 1为转换。 0不转换	0
keymap_dataframe	否	输入的索引表。 该参数仅当convertColToIndexId=1时有效。如果未指定该参数，则程序自动计算一套编号。	None

样例

数据样本

table表

rowid	col0	col1	col2
0	1	2	3
1	4	5	6
2	7	8	
3	10	11	12

input_keymap表

col_name	col_index	col_datatype
col	mycol1	bigint
col2	mycol2	bigint

convert=0

```
inputs = {
    "dataframe": input_df,
    "keymap_dataframe": None
}
```


版权所有 © 华为云计算技术有限公司

```
params = {
    "inputs": inputs,
    "selected_col_names": "col1,col2",
    "append_col_names": "rowid",
    "kv_delimiter": ":",
    "item_delimiter": ",",
    "convert_col2indexId": 0,
}
```

```
+-----+---------------+
|rowid|             kv|
+-----+---------------+
|    0|  col1:2,col2:3|
|    1|  col1:5,col2:6|
|    2|         col1:8|
|    3|col1:11,col2:12|
+-----+---------------+
```

convert=1，无输入的keymap表

```
inputs = {
    "dataframe": input_df,
    "keymap_dataframe": None
}
params = {
    "inputs": inputs,
    "selected_col_names": "col1,col2",
    "append_col_names": "rowid",
    "kv_delimiter": ":",
    "item_delimiter": ",",
    "convert_col2indexId": 1,
}
```

```
+-----+---------+
|rowid|       kv|
+-----+---------+
|    0|  0:2,1:3|
|    1|  0:5,1:6|
|    2|      0:8|
|    3|0:11,1:12|
+-----+---------+

None
+--------+---------+------------+
|col_name|col_index|col_datatype|
+--------+---------+------------+
|    col1|        0|         int|
|    col2|        1|         int|
+--------+---------+------------+
```

convert=1，有输入的keymap表

```
inputs = {
    "dataframe": input_df,
```


版权所有 © 华为云计算技术有限公司

```
    "keymap_dataframe": input_key_map_df
}
params = {
    "inputs": inputs,
    "selected_col_names": "col1,col2",
    "append_col_names": "rowid",
    "kv_delimiter": ":",
    "item_delimiter": ",",
    "convert_col2indexId": 1,
}
```

```
+-----+-------------------+
|rowid|                 kv|
+-----+-------------------+
|    0|  mycol1:2,mycol2:3|
|    1|  mycol1:5,mycol2:6|
|    2|           mycol1:8|
|    3|mycol1:11,mycol2:12|
+-----+-------------------+

None
+--------+---------+------------+
|col_name|col_index|col_datatype|
+--------+---------+------------+
|    col2|   mycol2|      bigint|
|    col1|   mycol1|      bigint|
+--------+---------+------------+
```

配置流程

运行流程

7.5.1.2.20 KV 表转普通表

概述

用于将KV（Key:Value）格式的表为普通表格式。其中Key转换成表的某列名，Value转换成该列在对应行的值。

表格式定义

KV表格式定义：Key是列名的index，Value支持BIGINT，DOUBLE和STRING类型。在该组件中可以输入用户定义的key_map表，是列名和Key的映射，但无论是否输入key_map表，该组件都会输出key_map表记录转化后的列名和Key的映射。例如1:10，2:20和3:30。

版权所有 © 华为云计算技术有限公司

key_map表格式定义：包含列名和index的映射以及类型信息的col_name，col_index和col_datatype，这三列类型要求是STRING，当col_datatype缺失时，默认值为double类型。

col_name	col_index	col_datatype
col1	1	bigint
col2	2	double

说明与约束

- key_map表中数据类型暂时只支持bigint、string、double3种类型。
- 如果有输入key_map表，则：
  a. 转化的列是： key_map表中的Key和KV表中的Key的交集。
  b. 转化后key列类型和key_map表中一致。如果key_map表无类型，则转化后key列类型为DOUBLE。
- 如果没有key_map表，
  a. 转化的列是： key_map表kv_col_name中的key。
  b. 转化的列只支持数值类型，默认为double。
- 转化后key列名称的命名规则是：
  - 无key_map表，使用kv_col_name+"_"+key 命名。
  - 有key_map表，使用key_map对应的col_name命名。
  - 不支持以下字符：%&()*+-./;<>=?
  - 指定的Append列，不能与Append列名和转化后Key列名相同；会造成列名冲突原因，会报错。
  - 如果输入表中列名长度超过128个字符时，列名会被截断成128个字符。
  - 同一行中不能存在重复的key，否则会报错。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象
	keymap_dataframe	keymap_dataframe为输入的keymap表，类型：pyspark中的DataFrame类型对象

输出

参数	子参数	参数说明
outputs	output_dataframe	输出结果表

版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
	output_keymap_dataframe	输出索引表

参数说明

参数名称	是否必选	参数描述	默认值
input_dataframe	是	输入df	无
input_keymap_dataframe	否	输入索引表对应的df，非必须	无
kv_col_name	是	KV列名	无
append_col_names	否	附加列名，支持多列	无
kv_delimiter	否	Key和Value之间分隔符	默认”:”
item_delimiter	否	KV对之间分隔符	默认”,”
top1200	否	是否只截取前1200列 true false	true

样例

没有kepmap表

```
inputs = {      "dataframe": input_df,      "keymap_dataframe":None} params = {      "inputs": inputs,
"kv_col_name": "kv",      "append_col_names": "rowid",      "kv_delimiter": ":",      "item_delimiter": ",",
"top1200": True, }
```

输入

rowid	kv
0	col1:2,col2:3
1	col1:5,col2:6
2	col1:8
3	col1:11,col2:12

输出

版权所有 © 华为云计算技术有限公司

col_name	col_index	col_type
kv_col1	col1	double
kv_col2	col2	double

kv_col1	kv_col2	rowid
2.0	3.0	0
5.0	6.0	1
8.0		2
11.0	12.0	3

有kepmap表

```
inputs = {      "dataframe": input_df,      "keymap_dataframe": input_key_map_df } params = {      "inputs": inputs,      "kv_col_name": "kv",      "append_col_names": "rowid,kv_1",      "kv_delimiter": ":",      "item_delimiter": ",",      "top1200": True, }
```

输入

rowid	kv	kv_1
0	col1:2,col2:3	col3:2,col4:3
1	col1:5,col2:6	
2	col1:8	
3	col1:11,col2:12	

col_name	col_index	col_type
mycol1	col1	bigint
mycol2	col2	bigint

输出

col_name	col_index	col_type
mycol1	col1	bigint
mycol2	col2	bigint

版权所有 © 华为云计算技术有限公司

mycol1	mycol2	rowid	kv_1
2	3	0	col3:2,col4:3
5	6	1	
8		2	
11	12	3	

运行示例

7.5.1.2.21 分层采样

概述

分层采样是一种数据采样算法，依据数据集中某一代表数据类别的列，按照数量或比例对不同类别的数据进行采样。

说明

算法实现采用spark自带的sample函数，采样数量会存在一定误差（按比例采样和按数量采样均会存在）。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型。

输出

参数	子参数	参数说明
output	output_port_1	output为字典类型，output_port_1为pyspark中的DataFrame类型对象，为分层采样结果。

版权所有 © 华为云计算技术有限公司

参数说明

参数	是否必选	参数说明	默认值
strata_col	是	分层列, 按此列进行分层采样。	无
sample_size	否	采样个数。为整数时：表示每个层的采样个数；为字符串时：格式为strata0:n0,strata1:n1,…表示每个层分别设置的采样个数。	无
sample_ratio	否	采样比例。为数字时：范围 (0,1) 表示每个层的采样比例；字符串时：格式为 strata0:r0,strata1:r1,…表示每个层分别设置的采样比例。	0.2
random_seed	是	随机种子。	123

样例

数据样本

鸢尾花数据集，species列代表鸢尾花种类，共有Iris-setosa、Iris-versicolor和Iris-virginica三种类别，每种类别样本数量为50。

	sepal_length	sepal_width	petal_length	petal_width	species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3	1.4	0.1	Iris-setosa
14	4.3	3	1.1	0.1	Iris-setosa
15	5.8	4	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
18	5.1	3.5	1.4	0.3	Iris-setosa
19	5.7	3.8	1.7	0.3	Iris-setosa
20	5.1	3.8	1.5	0.3	Iris-setosa

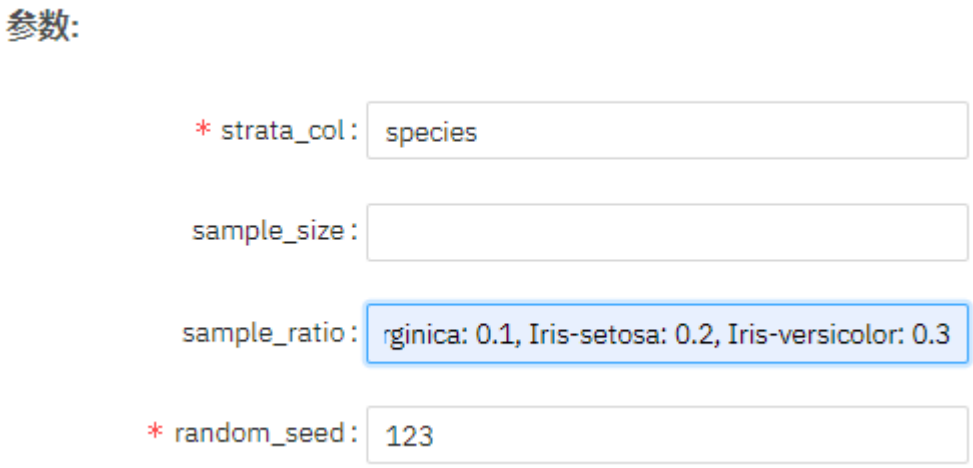
配置流程

运行流程

版权所有 © 华为云计算技术有限公司

参数设置

按比例采样，并分别指定每个种类的采样比例，如果sample_ratio为数字例如0.3，则所有类别数据均采样30%



7.5.1.2.22 加权采样

概述

加权采样是一种数据采样算法，依据数据集中权重列进行数据采样，权重越大的样本被采样的概率越大。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型。

输出

参数	子参数	参数说明
output	output_port_1	output为字典类型，output_port_1为pyspark中的DataFrame类型对象，为加权采样结果。

版权所有 © 华为云计算技术有限公司

参数说明

参数	是否必选	参数说明	默认值
weight_col	是	权重列。	"weight"
sample_size	否	采样数量。	100
sample_ratio	否	采样比例，范围(0,1)，如果sample_size和sample_ratio同时设置，以sample_size为准。	0.2
with_replacement	是	是否有放回采样。	True
partitions	否	分区数，有放回采样需设置该参数，数据量较大时可将该参数调大。	10
random_seed	否	随机种子。	1234

样例

截取部分样例数据如下，weight为权重列，class列中包括A，B，C，D四种类别，权重分别为1，2，4，10。

版权所有 © 华为云计算技术有限公司

	class	weight
1	A	1
2	B	2
3	C	4
4	D	10
5	A	1
6	B	2
7	C	4
8	D	10
9	A	1
10	B	2
11	C	4
12	D	10
13	A	1
14	B	2
15	C	4
16	D	10
17	A	1
18	B	2
19	C	4
20	D	10

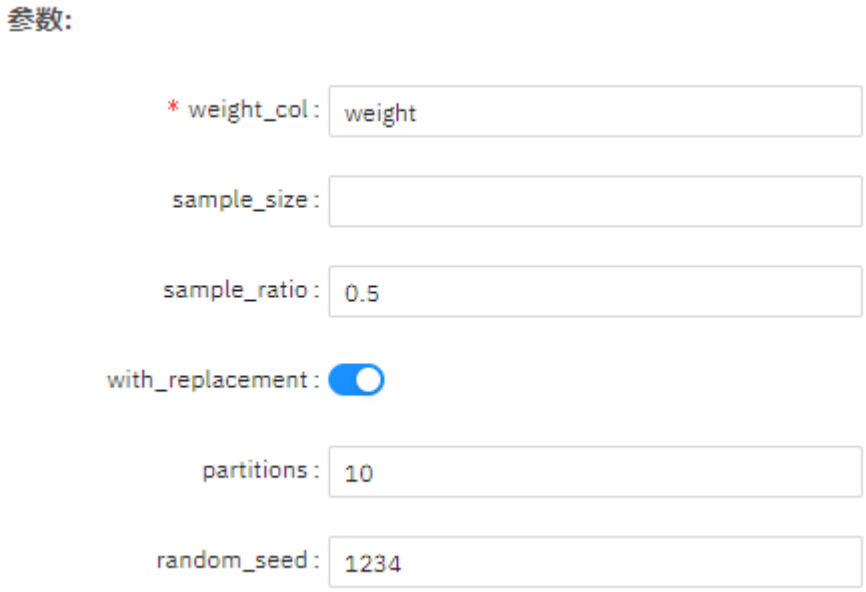
配置流程

运行流程

参数设置

以按比例采样为例，设置有放回采样

版权所有 © 华为云计算技术有限公司



查看结果

将最终采样结果分组统计，查看A，B，C，D四种类别数据的采样数据量，结果如下，符合权重越大采样概率越大

```
+-----+-----+
|class|count|
+-----+-----+
|    B|  260|
|    D| 1227|
|    C|  446|
|    A|  110|
+-----+-----+
```

7.5.1.3 特征工程

7.5.1.3.1 二值化

概述

“二值化”节点用于将数值型的字段转换成二值化形式。

例如：数据集中有一列整型数据属性为“Age”，取值为：“20-40”，设置阈值为30。二值化后当“Age”小于等于“30”时，“Age”这一列的取值就为“0”；当“Age”大于“30”时，“Age”这一列的取值就为“1”。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

版权所有 © 华为云计算技术有限公司

输出

数据集

参数说明

参数	子参数	参数说明
input_col	-	输入列名
output_col	-	对应的输出列名
threshold	-	阈值，列中大于该值的设置为1，小于该值的设置为0

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "outer_pipeline_stages": None,
    "input_col": "",  # @param {"label":"input_col","type":"string","required":"true","helpTip":""}
    "output_col": "binarizered_feature",  # @param
{"label":"output_col","type":"string","required":"true","helpTip":""}
    "threshold": 0.0  # @param
{"label":"threshold","type":"number","required":"true","range":"(none,none)","helpTip":""}
}
binarizer____id___ = MLSBinarizer(**params)
binarizer____id___.run()
# @output {"label":"dataframe","name":"binarizer____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.3.2 卡方选择

概述

采用卡方检验来进行特征选择。

卡方检验（Chi-Squared Test或χ^2 Test）的基本思想是通过特征变量与目标变量之间的偏差大小来选择相关性较大的特征变量。首先假设两个变量是独立的，然后观察实际值与理论值的偏差程度，该偏差程度代表两个变量之间的相关性。如果某个特征变量与目标变量之间偏差程度越大，则它们的相关性越高，最后根据相关性对特征变量进行排序，并选择与目标变量相关性较大的特征变量。卡方检验中假设理论值为E，第i个样本的实际值为x_i，则偏差程度的计算公式如下：

$$\sum_{i=1}^{n} \frac{(x_i - E)^2}{E}$$

版权所有 © 华为云计算技术有限公司

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

参数说明

参数	子参数	参数说明
input_features_str	-	输入列名组成的格式化字符串，例如: "column_a" "column_a,column_b"
label_col	-	目标列，基于该列进行卡方检验
chi_label_index_col	-	目标列通过标签编码得到的新列名，默认为label_index
chi_features_col	-	调用spark卡方选择需要的输入特征向量列名，默认为input_features
chi_output_col	-	调用spark卡方选择需要的输入特征向量列名，默认为output_features
selector_type	-	卡方选择的选择方法，支持numTopFeatures,percentile,fpr,fdr,fwe
num_top_features	-	选择的特征个数，默认为50
percentile	-	选择的特征个数占原始特征数量的比例，默认为0.1
fpr	-	最高的p-value，默认为0.05
fdr	-	期望的错误观察率的最大值，默认为0.05
fwe	-	默认为0.05
max_categories	-	特征的最大类别数，默认为1000

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
```


版权所有 © 华为云计算技术有限公司

```
    "b_output_action": True,
    "b_use_default_encoder": True,
    "input_features_str": "",  # @param
{"label":"input_features_str","type":"string","required":"false","helpTip":""}
    "outer_pipeline_stages": None,
    "label_col": "",  # @param {"label":"label_col","type":"string","required":"true","helpTip":""}
    "chi_label_index_col": "label_index",  # @param
{"label":"chi_label_index_col","type":"string","required":"true","helpTip":""}
    "chi_features_col": "input_features",  # @param
{"label":"chi_features_col","type":"string","required":"true","helpTip":""}
    "chi_output_col": "output_features",  # @param
{"label":"chi_output_col","type":"string","required":"true","helpTip":""}
    "selector_type": "numTopFeatures",  # @param
{"label":"selector_type","type":"enum","required":"true","options":"numTopFeatures,percentile,fpr,fdr,fwe","h
elpTip":""}
    "num_top_features": 50,  # @param
{"label":"num_top_features","type":"integer","required":"true","range":"(0,2147483647]","helpTip":""}
    "percentile": 0.1,  # @param
{"label":"percentile","type":"number","required":"true","range":"[0,1]","helpTip":""}
    "fpr": 0.05,  # @param {"label":"fpr","type":"number","required":"true","range":"[0,1]","helpTip":""}
    "fdr": 0.05,  # @param {"label":"fdr","type":"number","required":"true","range":"[0,1]","helpTip":""}
    "fwe": 0.05,  # @param {"label":"fwe","type":"number","required":"true","range":"[0,1]","helpTip":""}
    "max_categories": 1000  # @param
{"label":"max_categories","type":"number","required":"true","range":"(0,2147483647]","helpTip":""}
}
chi_square_selector____id___ = MLSChiSquareSelector(**params)
chi_square_selector____id___.run()
# @output {"label":"dataframe","name":"chi_square_selector____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.3.3 派生

概述

“派生”节点用于在数据集中生成任意可行的新属性字段，对现有数据的某个属性操作，例如2*某个属性、两个属性乘积等，允许用户自定义生成属性名称，并将生成的新属性字段添加到原数据集中。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

 版权所有 © 华为云计算技术有限公司

参数说明

参数	子参数	参数说明
derive_operators_str	-	派生操作组成的格式化字符串，例如: "2*column_a as new_column_a" "abs(column_a) as new_column_a;case when column_b > 50 then 1 else 0 end as new_column_b;column_c+column_d as new_column_cd"

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "derive_operators_str": ""  # @param
{"label":"derive_operators_str","type":"string","required":"true","helpTip": ""}
}
feature_derive____id___ = MLSFeatureDerive(**params)
feature_derive____id___.run()
# @output {"label":"dataframe","name":"feature_derive____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.3.4 特征转换

概述

将对应列的数据乘以相应的权重得到新的列，只支持数字列。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

参数说明

参数	子参数	参数说明
input_columns_str	-	输入的列名组成的格式化字符串，以逗号分隔，例如: "column_a" "column_a,column_b"

版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
input_weights_str	-	输入的权重组成的格式化字符串，以逗号分隔，例如: "0.5" "0.4,0.8"

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "input_columns_str": "",  # @param
{"label":"input_columns_str","type":"string","required":"true","helpTip": ""}
    "input_weights_str": ""  # @param {"label":"input_weights_str","type":"string","required":"true","helpTip":
""}
}
feature_transform____id___ = MLSFeatureTransform(**params)
feature_transform____id___.run()
# @output {"label":"dataframe","name":"feature_transform____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.3.5 FP-growth

概述

“FP-Growth”节点用于挖掘频繁模式，该算法使用了一种称为频繁模式树（Frequent Pattern Tree）的数据结构。FP-tree是一种特殊的前缀树，由频繁项头表和项前缀树构成。FP-Growth算法基于以上的结构加快整个挖掘过程。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集和spark pipeline类型的模型

参数说明

参数	子参数	参数说明
input_features_str	-	数据集的特征列名组成的格式化字符串，例如: "column_a" "column_a,column_b"

版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
fp_items_col	-	fp-growth模型训练所需要的项目数组的列名
prediction_col	-	预测列名
min_support	-	最小支持度
min_confidence	-	生成关联规则的最小置信度

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "input_features_str": "",  # @param
{"label":"input_features_str","type":"string","required":"false","helpTip":""}
    "fp_items_col": "values",  # @param {"label":"fp_items_col","type":"string","required":"false","helpTip":""}
    "prediction_col": "prediction",  # @param
{"label":"prediction_col","type":"string","required":"false","helpTip":""}
    "min_support": 0.3,  # @param
{"label":"min_support","type":"number","required":"true","range":"(none,none)","helpTip":""}
    "min_confidence": 0.8  # @param
{"label":"min_confidence","type":"number","required":"true","range":"(none,none)","helpTip":""}
}
fp_growth____id___ = MLSFPGrowth(**params)
fp_growth____id___.run()
# @output {"label":"pipeline_model","name":"fp_growth____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
# @output {"label":"dataframe","name":"fp_growth____id___.get_outputs()
['output_port_2']","type":"DataFrame"}
```

7.5.1.3.6 最小最大规范化

概述

将数据集指定的某些数字列，转换到一定的数值范围（例如0和1之间）。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

版权所有 © 华为云计算技术有限公司

参数说明

参数	子参数	参数说明
input_features_str	-	输入的列名以逗号分隔组成的字符串，例如： "column_a" "column_a,column_b"
min	-	转换后的最小值，默认为0.0
max	-	转换后的最大值，默认为1.0
input_vector_column	-	输入的向量列的列名，默认为"input_features"
output_vector_column	-	结果输出的向量列的列名，默认为"output_scaler_features"

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "outer_pipeline_stages": None,
    "input_features_str": "",  # @param
{"label":"input_features_str","type":"string","required":"false","helpTip":""}
    "min": 0.0,  # @param {"label":
"min","type":"number","required":"true","range":"(none,none)","helpTip":""}
    "max": 1.0,  # @param
{"label":"max","type":"number","required":"true","range":"(none,none)","helpTip":""}
    "input_vector_column": "input_features",  # @param
{"label":"input_vector_column","type":"string","required":"true","helpTip":""}
    "output_vector_column": "minmax_scaler_features"  # @param
{"label":"output_vector_column","type":"string","required":"true","helpTip":""}
}
min_max_scaler____id___ = MLSMinMaxScaler(**params)
min_max_scaler____id___.run()
# @output {"label":"pipeline_model","name":"min_max_scaler____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
# @output {"label":"dataframe","name":"min_max_scaler____id___.get_outputs()
['output_port_2']","type":"DataFrame"}
```

7.5.1.3.7 正则化

概述

使用p范式对向量进行正则化。

版权所有 © 华为云计算技术有限公司

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

参数说明

参数	子参数	参数说明
input_features_str	-	数据集的特征列名组成的格式化字符串，例如: "column_a" "column_a,column_b"
output_col	-	输出的结果列名
p	-	用第几范式进行正则化

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "outer_pipeline_stages": None,
    "input_features_str": "",  # @param
{"label":"input_features_str","type":"string","required":"false","helpTip":""}
    "output_col": "norm_output_features",  # @param
{"label":"output_col","type":"string","required":"false","helpTip":""}
    "p": 2  # @param {"label":"p","type":"integer","required":"false","helpTip":""}
}
normalizer____id___ = MLSNormalizer(**params)
normalizer____id___.run()
# @output {"label":"pipeline_model","name":"normalizer____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
```

7.5.1.3.8 独热编码

概述

将用户指定的一些列进行one-hot编码。

版权所有 © 华为云计算技术有限公司

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

参数说明

参数	子参数	参数说明
input_features_str	-	输入的列名以逗号分隔组成的字符串，例如： "column_a" "column_a,column_b"
output_column_postfix	-	结果输出的列名，在输入列名后面增加的后缀，默认为"_one_hot"
handle_invalid	-	是否处理无效值，支持"keep"、"error"，默认为"keep"。
drop_last	-	在编码向量中，是否去除最后一个类别的编码，默认为true

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "outer_pipeline_stages": None,
    "input_features_str": "",  # @param
{"label":"input_features_str","type":"string","required":"false","helpTip":""}
    "output_column_postfix": "_one_hot",  # @param
{"label":"output_column_postfix","type":"string","required":"true","helpTip":""}
    "handle_invalid": "keep",  # @param
{"label":"handle_invalid","type":"enum","options":"keep,error","required":"true","helpTip":""}
    "drop_last": True  # @param {"label":"drop_last","type":"boolean","required":"true","helpTip":""}
}
one_hot_encoder____id___ = MLSOneHotEncoder(**params)
one_hot_encoder____id___.run()
# @output {"label":"pipeline_model","name":"one_hot_encoder____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
# @output {"label":"dataframe","name":"one_hot_encoder____id___.get_outputs()
['output_port_2']","type":"DataFrame"}
```


版权所有 © 华为云计算技术有限公司

7.5.1.3.9 主成分分析

概述

主成分分析（Principal Components Analysis，PCA）是统计分析中简化数据集的一种算法，常用于减少数据集的维数，同时保持数据集中对方差贡献最大的特征。该算法主要通过对原始数据矩阵进行奇异值分解（Singular Value Decomposition，SVD），以得出数据的主成分（即特征向量），应用场景主要为高维数据降维等。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

参数说明

参数	子参数	参数说明
input_features_str	-	输入的特征列名以逗号分隔组成的格式化字符串，例如: "column_a" "column_a,column_b"
pca_input_features_col	-	输入的特征向量列名，默认为"pca_input_features"
pca_output_vector_col	-	输出的特征向量列名，默认为"pca_output_features"
k	-	主成分的维度数

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "b_use_default_encoder": True,
    "input_features_str": "",  # @param
{"label":"input_features_str","type":"string","required":"false","helpTip":""}
    "outer_pipeline_stages": None,
    "pca_input_features_col": "pca_input_features",  # @param
{"label":"pca_input_features_col","type":"string","required":"true","helpTip":""}
    "pca_output_vector_col": "pca_output_features",  # @param
{"label":"pca_output_vector_col","type":"string","required":"true","helpTip":""}
    "k": 5  # @param {"label":"k","type":"integer","required":"true","range":"(0,2147483647]","helpTip":"k"}
```


版权所有 © 华为云计算技术有限公司

```
}
pca____id___ = MLSPCA(**params)
pca____id___.run()
# @output {"label":"dataframe","name":"pca____id___.get_outputs()['output_port_1']","type":"DataFrame"}
```

7.5.1.3.10 离散化

概述

根据用户输入的桶的个数，按照分位数分桶，将用户指定的某个数值列离散化。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

参数说明

参数	子参数	参数说明
input_col	-	输入的列名
output_col	-	离散化后输出的列名，默认为"quantile_discretizer_result"
num_buckets	-	桶的个数，默认为2
handle_invalid	-	处理无效值的策略，支持skip、keep、error，默认为skip
relative_error	-	相对错误值，取值范围是[0, 1]，默认为0.001

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "outer_pipeline_stages": None,
    "input_col": "",  # @param {"label":"input_col","type":"string","required":"true","helpTip":""}
    "output_col": "quantile_discretizer_result",  # @param
{"label":"output_col","type":"string","required":"true","helpTip":""}
    "num_buckets": 2,  # @param
{"label":"num_buckets","type":"integer","required":"true","range":"(0,2147483647]","helpTip": ""}
    "handle_invalid": "skip",  # @param
{"label":"handle_invalid","type":"enum","options":"skip,keep,error","required":"true","helpTip":""}
    "relative_error": 0.001  # @param
{"label":"relative_error","type":"number","required":"true","range":"[0,1]","helpTip":""}
```

 版权所有 © 华为云计算技术有限公司

```
}
quantile_discretizer____id___ = MLSQuantileDiscretizer(**params)
quantile_discretizer____id___.run()
# @output {"label":"dataframe","name":"quantile_discretizer____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.3.11 标准化

概述

对数据集的某些数值列，根据均值和方差进行标准化。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

参数说明

参数	子参数	参数说明
input_features_str	-	输入的特征列名以逗号分隔组成的格式化字符串，例如: "column_a" "column_a,column_b"
input_vector_column	-	算子输入的向量列的列名，默认为"input_features"
output_vector_column	-	算子输出的向量列的列名，默认为"standard_features"
with_std	-	是否按照方差进行标准化，默认为True
with_mean	-	是否按照均值进行标准化，默认为False

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "outer_pipeline_stages": None,
    "input_features_str": "",  # @param {"label":"input_features_str","type":
"string","required":"false","helpTip": ""}
    "input_vector_column": "input_features",  # @param
```

 版权所有 © 华为云计算技术有限公司

```
{"label":"input_vector_column","type":"string","required":"true","helpTip":""}
    "output_vector_column": "standard_features",  # @param
{"label":"output_vector_column","type":"string","required":"true","helpTip":""}
    "with_std": True,  # @param {"label":"with_std","type":"boolean","required":"true","helpTip":""}
    "with_mean": False  # @param {"label":"with_mean","type":"boolean","required":"true","helpTip":""}
}
standard_scaler____id___ = MLSStandardScaler(**params)
standard_scaler____id___.run()
# @output {"label":"pipeline_model","name":"standard_scaler____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
# @output {"label":"dataframe","name":"standard_scaler____id___.get_outputs()
['output_port_2']","type":"DataFrame"}
```

7.5.1.3.12 字符串标签化

概述

对数据集的指定列进行StringIndexer编码，即标签编码。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

数据集

参数说明

参数	子参数	参数说明
input_column_str	-	输入的列名
output_column_str	-	算子输出的列名
handle_invalid	-	处理无效值的策略，支持skip、error、keep，默认为keep

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "outer_pipeline_stages": None,
    "input_column_str": "",  # @param
{"label":"input_column_str","type":"string","required":"true","helpTip":""}
    "output_column_str": "",  # @param
{"label":"output_column_str","type":"string","required":"true","helpTip":""}
    "handle_invalid": "keep"  # @param
{"label":"handle_invalid","type":"enum","options":"skip,error,keep","required":"true","helpTip":""}
}
```

 版权所有 © 华为云计算技术有限公司

```
string_indexer____id___ = MLSStringIndexer(**params)
string_indexer____id___.run()
# @output {"label":"dataframe","name":"string_indexer____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.1.3.13 奇异值分解

概述

奇异值分解（Singular Value Decomposition，SVD）一般用于数据挖掘、建模等领域的特征工程过程，是线性代数中一种重要的矩阵分解方法，奇异值分解算子可将1个矩阵分解为3个矩阵。

比如对于$m \times n$ 的矩阵**A**，可根据以下SVD计算公式得到左奇异向量组成的$m \times k$ 矩阵**U**、奇异值组成的$k \times k$ 矩阵**Σ**（对角线上元素被称为奇异值）和右奇异向量组成的$n \times k$ 矩阵**V**：$\mathbf{A}=\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{T}$。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

3个数据集，分别对应矩阵**U**、矩阵**Σ**的对角线元素（k个奇异值）和矩阵**V**。

参数说明

参数	子参数	参数说明
k	-	奇异值的个数
computeU	-	是否计算U，默认为False
rCond	-	条件值，默认为1e-9

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "k": "",  # @param {"label":"k","type":"integer","required":"true","range":"(0,2147483647]","helpTip":""}
    "computeU": False,  # @param {"label":"computeU","type":"boolean","required":"false","helpTip":""}
    "rCond": 1e-9  # @param
{"label":"rCond","type":"integer","required":"false","range":"(0,2147483647]","helpTip":""}
}
svd____id___ = MLSSVD(**params)
svd____id___.run()
# @output {"label":"dataframe","name":"svd____id___.get_outputs()['output_port_1']","type":"DataFrame"}
# @output {"label":"dataframe","name":"svd____id___.get_outputs()['output_port_2']","type":"DataFrame"}
# @output {"label":"dataframe","name":"svd____id___.get_outputs()['output_port_3']","type":"DataFrame"}
```

 版权所有 © 华为云计算技术有限公司

7.5.1.3.14 过滤式特征选择

概述

过滤式特征选择根据特征对标签的重要性对特征进行筛选，特征重要性较高的特征，提升训练的精度和效率。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

参数	子参数	参数说明
output	output_feature_importance	dataframe类型的特征重要性结果
output	output_selected_dataframe	dataframe类型的特征筛选结果

参数说明

参数	是否必选	参数说明	默认值
feature_columns_str	是	特征列列名，支持多列用','分隔	"”
label_column	是	标签列列名	""
discretization_columns_str	否	需要进行离散化特征列列名，支持多列用','分隔	""
method	是	过滤选择的方法，取值如下： • IV：根据IV值计算特征的重要性，注：IV法仅支持2分类； • Gini增益：根据Gini增益计算特征重要性； • 信息增益：根据信息增益计算特征重要性； • Lasso：采用Lasso回归计算特征重要性；	""

版权所有 © 华为云计算技术有限公司

参数	是否必选	参数说明	默认值
select_feature_num	是	选择的TopN个特征，如果大于输入特征数，则输出所以特征	None
discretization_method	否	离散化连续特征方法，取值如下： ● equidistant division：根据特征的最小、最大值等距离分隔	""
discretization_bin_num	否	离散化连续特征区间数量	None
is_sparse	是	是否是K:V的稀疏特征	False
kv_col	否	稀疏特征列名	""
item_spliter	否	K:V特征中每个item之间的分隔符	","
kv_spliter	否	K:V特征中每个key与value之间的分隔符	":"

样例一（常规数据）

数据样本

sepal_length	sepal_width	petal_length	petal_width	variety
5.1	3.5	1.4	.2	Setosa
4.9	3	1.4	.2	Setosa
4.7	3.2	1.3	.2	Setosa
4.6	3.1	1.5	.2	Setosa
5	3.6	1.4	.2	Setosa
5.4	3.9	1.7	.4	Setosa
4.6	3.4	1.4	.3	Setosa
5	3.4	1.5	.2	Setosa
4.4	2.9	1.4	.2	Setosa
4.9	3.1	1.5	.1	Setosa
5.4	3.7	1.5	.2	Setosa
4.8	3.4	1.6	.2	Setosa
4.8	3	1.4	.1	Setosa
4.3	3	1.1	.1	Setosa
5.8	4	1.2	.2	Setosa
5.7	4.4	1.5	.4	Setosa
5.4	3.9	1.3	.4	Setosa
5.1	3.5	1.4	.3	Setosa
5.7	3.8	1.7	.3	Setosa
5.1	3.8	1.5	.3	Setosa
5.4	3.4	1.7	.2	Setosa
5.1	3.7	1.5	.4	Setosa
4.6	3.6	1	.2	Setosa

版权所有 © 华为云计算技术有限公司

配置流程

运行流程

算法参数设置

设置参数

* feature_columns_str :	sepal_length,sepal_width,petal_length,petal_
* label_column :	variety
discretization_columns_str :	sepal_length,sepal_width
* method :	lasso
* select_feature_num :	3
discretization_method :	equidistant division
discretization_bin_num :	3
* is_sparse :	False
kv_col :	
item_spliter :	,
kv_spliter :	:

查看结果

版权所有 © 华为云计算技术有限公司

	sepal_width	petal_length	petal_width
1	1.0	1.4	0.2
2	1.0	1.4	0.2
3	1.0	1.3	0.2
4	1.0	1.5	0.2
5	1.0	1.4	0.2
5	2.0	1.7	0.4
7	1.0	1.4	0.3
3	1.0	1.5	0.2
9	1.0	1.4	0.2
10	1.0	1.5	0.1
11	2.0	1.5	0.2
12	1.0	1.6	0.2
13	1.0	1.4	0.1
14	1.0	1.1	0.1
15	2.0	1.2	0.2
16	2.0	1.5	0.4
17	2.0	1.3	0.4
18	1.0	1.4	0.3
19	2.0	1.7	0.3
20	2.0	1.5	0.3

	feature	importances
1	petal_width	0.48845932
2	petal_length	0.29287928
3	sepal_width	0.037839856
4	sepal_length	0.011933952

样例二（KV 稀疏数据）

数据样本

版权所有 © 华为云计算技术有限公司

	iris	variety
1	petal_length:1.4,petal_width:0.2	Setosa
2	sepal_length:4.9,petal_width:0.2	Setosa
3	petal_length:1.3,petal_width:0.2	Setosa
4	sepal_length:4.6,petal_length:1.5	Setosa
5	sepal_width:3.6,petal_length:1.4	Setosa
6	sepal_length:5.4,sepal_width:3.9,petal_length:1.7,petal_width:0.4	Setosa
7	sepal_width:3.4,petal_length:1.4,petal_width:0.3	Setosa
8	sepal_length:5.0,sepal_width:3.4,petal_length:1.5,petal_width:0.2	Setosa
9	sepal_length:4.4	Setosa
10	sepal_length:4.9,sepal_width:3.1	Setosa
11	sepal_length:5.4,petal_length:1.5,petal_width:0.2	Versicolor
12	sepal_width:3.4,petal_width:0.2	Versicolor
13	sepal_length:4.8,petal_length:1.4,petal_width:0.1	Versicolor
14	sepal_length:4.3,sepal_width:3.0,petal_length:1.1,petal_width:0.1	Versicolor
15	sepal_length:5.8,petal_length:1.2	Versicolor
16	sepal_length:5.7,petal_length:1.5	Versicolor
17	sepal_length:5.4	Versicolor
18	petal_length:1.4	Versicolor
19	sepal_length:5.7	Versicolor
20	sepal_length:5.1,sepal_width:3.8,petal_length:1.5,petal_width:0.3	Versicolor

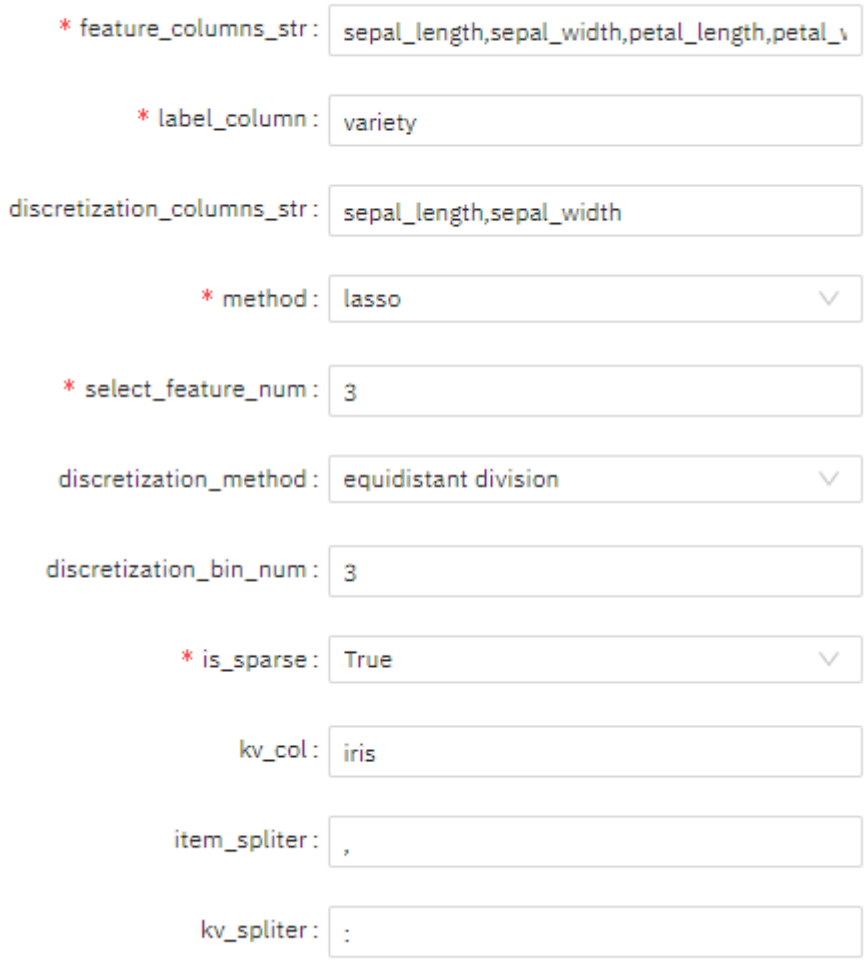
配置流程

运行流程

算法参数设置

版权所有 © 华为云计算技术有限公司

设置参数



查看结果

版权所有 © 华为云计算技术有限公司

	sepal_length	sepal_width	petal_width
1			0.2
2	1.0		0.2
3			0.2
4	0.0		
5		2.0	
6	2.0	2.0	0.4
7		1.0	0.3
8	1.0	1.0	0.2
9	0.0		
10	1.0	0.0	
11	2.0		0.2
12		1.0	0.2
13	1.0		0.1
14	0.0	0.0	0.1
15	2.0		
16	2.0		
17	2.0		
18			
19	2.0		
20	1.0	2.0	0.3

	feature	importances
1	sepal_length	0.61519873
2	petal_width	0.44991094
3	sepal_width	0.23486607
4	petal_length	0.187027

7.5.1.3.15 线性特征重要性

概述

用线性模型计算训练数据的特征重要性。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

版权所有 © 华为云计算技术有限公司

输出

特征的重要性和特征在线性模型中的weights，格式是dataFrame。

列名	公式
weight	abs(w_)
importance	abs(w_j) * STD(f_i)

参数说明

参数	参数说明
feature_cols	特征列
label_col	label列
item_spliter	稀疏特征的iterm之间的分割符
kv_spliter	稀疏特征中每个iterm的KV分割符
model_path	线性模型的输入路径

样例

输入数据

版权所有 © 华为云计算技术有限公司

	sepal_length	sepal_width	petal_length	petal_width	species
1	5.1	3.5	1.4	0.2	0
2	4.9	3	1.4	0.2	0
3	4.7	3.2	1.3	0.2	0
4	4.6	3.1	1.5	0.2	0
5	5	3.6	1.4	0.2	0
6	5.4	3.9	1.7	0.4	0
7	4.6	3.4	1.4	0.3	0
8	5	3.4	1.5	0.2	0
9	4.4	2.9	1.4	0.2	0
10	4.9	3.1	1.5	0.1	0
11	5.4	3.7	1.5	0.2	0
12	4.8	3.4	1.6	0.2	0
13	4.8	3	1.4	0.1	0
14	4.3	3	1.1	0.1	0
15	5.8	4	1.2	0.2	0
16	5.7	4.4	1.5	0.4	0
17	5.4	3.9	1.3	0.4	0
18	5.1	3.5	1.4	0.3	0
19	5.7	3.8	1.7	0.3	0
20	5.1	3.8	1.5	0.3	0
21	5.4	3.4	1.7	0.2	0
22	5.1	3.7	1.5	0.4	0
23	4.6	3.6	1	0.2	0
24	5.1	3.3	1.7	0.5	0
25	4.8	3.4	1.9	0.2	0
26	5	3	1.6	0.2	0
27	5	3.4	1.6	0.4	0
28	5.2	3.5	1.5	0.2	0
29	5.2	3.4	1.4	0.2	0
30	4.7	3.2	1.6	0.2	0
31	4.8	3.1	1.6	0.2	0

配置流程

运行流程

输出结果

版权所有 © 华为云计算技术有限公司

```
+------------+------------------+------------------+
|    features|           weights|        importance|
+------------+------------------+------------------+
|sepal_length|0.5171578116981289|0.2129535720798462|
| sepal_width| 5.759773101425531|1.3053391233618572|
|petal_length|13.135690942155964|27.563137770538713|
| petal_width|12.849304439714638| 4.120538310099398|
+------------+------------------+------------------+
```

7.5.1.3.16 特征尺度变换

概述

支持对稠密或稀疏的数值类特征进行常见的尺度变换，支持常见的log2、log10、ln、abs及sqrt等尺度变化函数。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

dataFrame

参数说明

参数	参数说明	默认值
scale_cols	需要被进行尺度变换的特征名	-
scale_method	尺度变换的方法	"ln"
item_spliter	离散型特征的，iterm之间的分割符	","
kv_spliter	离散型特征KV的分割符	":"

样例

输入数据

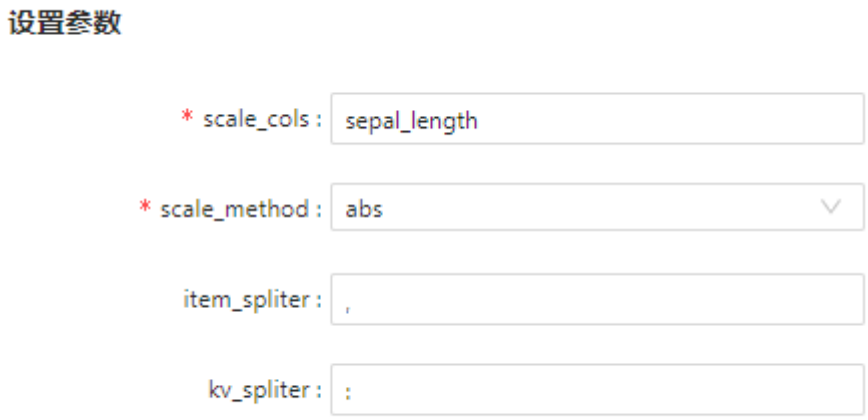
版权所有 © 华为云计算技术有限公司

	sepal_length	sepal_width	petal_length	petal_width	species
1	-5.1	3.5	1.4	0.2	0
2	4.9	3	1.4	0.2	0
3	4.7	3.2	1.3	0.2	0
4	-4.6	3.1	1.5	0.2	0
5	5	3.6	1.4	0.2	0
6	5.4	3.9	1.7	0.4	0
7	4.6	3.4	1.4	0.3	0
8	5	3.4	1.5	0.2	0
9	4.4	2.9	1.4	0.2	0
10	4.9	3.1	1.5	0.1	0
11	5.4	3.7	1.5	0.2	0
12	4.8	3.4	1.6	0.2	0
13	4.8	3	1.4	0.1	0
14	4.3	3	1.1	0.1	0
15	5.8	4	1.2	0.2	0
16	5.7	4.4	1.5	0.4	0
17	5.4	3.9	1.3	0.4	0

配置流程

运行流程

参数设置



输出结果

版权所有 © 华为云计算技术有限公司

运行结果

sepal_length	sepal_width	petal_length	petal_width	species	sepal_length_scale
-5.1	3.5	1.4	0.2	0	5.1
4.9	3.0	1.4	0.2	0	4.9
4.7	3.2	1.3	0.2	0	4.7
-4.6	3.1	1.5	0.2	0	4.6
5.0	3.6	1.4	0.2	0	5.0
5.4	3.9	1.7	0.4	0	5.4
4.6	3.4	1.4	0.3	0	4.6
5.0	3.4	1.5	0.2	0	5.0
4.4	2.9	1.4	0.2	0	4.4
4.9	3.1	1.5	0.1	0	4.9
5.4	3.7	1.5	0.2	0	5.4
4.8	3.4	1.6	0.2	0	4.8
4.8	3.0	1.4	0.1	0	4.8
4.3	3.0	1.1	0.1	0	4.3
5.8	4.0	1.2	0.2	0	5.8
5.7	4.4	1.5	0.4	0	5.7
5.4	3.9	1.3	0.4	0	5.4
5.1	3.5	1.4	0.3	0	5.1
5.7	3.8	1.7	0.3	0	5.7
5.1	3.8	1.5	0.3	0	5.1

only showing top 20 rows

7.5.1.3.17 特征异常检测

概述

特征异常检测的方法包括箱型图（Box-plot）和AVF（Attribute Value Frequency）

- 箱型图用于检测连续值类特征的数据，根据四分位数检测异常特征。
- AVF用于检测枚举值类特征的数据，根据枚举特征的取值频率及阈值检测异常特征。

箱型图异常检测

箱形图可以用来观察数据整体的分布情况，利用中位数，25%分位数，75%分位数，上边界，下边界等统计量来描述数据的整体分布情况。通过计算这些统计量，生成一个箱体图，箱体包含了大部分的正常数据，而在箱体上边界和下边界之外的，就是异常数据。

其中上下边界的计算公式如下：

UpperLimit=Q3+1.5IQR=75%分位数+（75%分位数-25%分位数）*1.5，

LowerLimit=Q1 − 1.5IQR=25%分位数-（75%分位数-25%分位数）*1.5

（将数据由小到大排序，处于中间的为中位数，即50%分位数，在75%位置的即为75%分位数或四分之三分位数——Q3，在25%位置的即为25%分位数或四分之一分位数——Q1）

AVF异常检测(Attribute Value Frequency)

版权所有 © 华为云计算技术有限公司

AVF算法全称Attribute Value Frequency，针对非数值型的数据，即类别离散数据的算法。具体步骤如下：

1. 将所有的数据点都标为非异常点；
2. 计算所有每一个属性值的频数；
3. 计算每一个点的AVF score，即样本点x的每一个属性值对应的频数之和除以属性总数，这里的属性指的都是category的属性。

AVF score值越小，样本越异常。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

参数	子参数	参数说明
outputs	pipeline_model	输出的模型文件
	output_dataframe	过滤后的数据表
	model_info_dataframe	模型信息表

参数说明

参数名称	参数描述	是否必选	默认值
selected_cols	输入特征，字段类型没有限制。	是	无
detect_strategy	系统支持Box-plot和AVF选项。Box-plot用于检测连续值类特征；AVF用于检测枚举值类特征。取值“Box-plot”、“AVF”	是	Box-plot

样例

数据样本

版权所有 © 华为云计算技术有限公司

point	1	2	3	4	5	6	7	8	9
1	1	1	1	1	2	10	3	1	1
2	2	1	1	1	2	1	2	1	1
3	1	1	1	1	2	3	3	1	1
4	4	1	1	1	2	1	2	1	1
5	4	1	1	1	2	1	3	1	1
6	6	1	1	1	2	1	3	1	1
7	7	3	2	10	5	10	5	4	4
8	3	1	1	1	2	1	2	1	1
9	1	1	1	1	2	1	3	1	1
10	3	2	1	1	1	1	2	1	1
11	5	1	1	1	2	1	2	1	1
12	2	5	3	3	6	7	7	5	1

配置流程

运行流程

运行示例

- Box_plot

```
params = {
"inputs": inputs,
"selected_cols": "1,2,3,4,5,6,7,8,9",
"detect_strategy": "Box_plot"
}
```

过滤后数据表

过滤掉了第1、7、10、12行

point	1	2	3	4	5	6	7	8	9
2	2	1	1	1	2	1	2	1	1

版权所有 © 华为云计算技术有限公司

point	1	2	3	4	5	6	7	8	9
3	1	1	1	1	2	3	3	1	1
4	4	1	1	1	2	1	2	1	1
5	4	1	1	1	2	1	3	1	1
6	6	1	1	1	2	1	3	1	1
8	3	1	1	1	2	1	2	1	1
9	1	1	1	1	2	1	3	1	1
11	5	1	1	1	2	1	2	1	1

模型表

model_key	model_value
detect_model	{'featureName': '1', 'mid': 3.0, 'q1': 1.0, 'q3': 4.0, 'upper_bound': 8.5, 'lower_bound': -3.5}
detect_model	{'featureName': '2', 'mid': 1.0, 'q1': 1.0, 'q3': 1.0, 'upper_bound': 1.0, 'lower_bound': 1.0}
detect_model	{'featureName': '3', 'mid': 1.0, 'q1': 1.0, 'q3': 1.0, 'upper_bound': 1.0, 'lower_bound': 1.0}
detect_model	{'featureName': '4', 'mid': 1.0, 'q1': 1.0, 'q3': 1.0, 'upper_bound': 1.0, 'lower_bound': 1.0}
detect_model	{'featureName': '5', 'mid': 2.0, 'q1': 2.0, 'q3': 2.0, 'upper_bound': 2.0, 'lower_bound': 2.0}
detect_model	{'featureName': '6', 'mid': 1.0, 'q1': 1.0, 'q3': 3.0, 'upper_bound': 6.0, 'lower_bound': -2.0}
detect_model	{'featureName': '7', 'mid': 3.0, 'q1': 2.0, 'q3': 3.0, 'upper_bound': 4.5, 'lower_bound': 0.5}
detect_model	{'featureName': '8', 'mid': 1.0, 'q1': 1.0, 'q3': 1.0, 'upper_bound': 1.0, 'lower_bound': 1.0}
detect_model	{'featureName': '9', 'mid': 1.0, 'q1': 1.0, 'q3': 1.0, 'upper_bound': 1.0, 'lower_bound': 1.0}

版权所有 © 华为云计算技术有限公司

- AVF

```
params = {
"inputs": inputs,
"selected_cols": "1,2,3,4,5,6,7,8,9",
"detect_strategy": "AVF"}
```

过滤后数据表

过滤掉了第12行

point	1	2	3	4	5	6	7	8	9
1	1	1	1	1	2	10	3	1	1
2	2	1	1	1	2	1	2	1	1
3	1	1	1	1	2	3	3	1	1
4	4	1	1	1	2	1	2	1	1
5	4	1	1	1	2	1	3	1	1
6	6	1	1	1	2	1	3	1	1
8	3	1	1	1	2	1	2	1	1
9	1	1	1	1	2	1	3	1	1
10	3	2	1	1	1	1	2	1	1
11	5	1	1	1	2	1	2	1	1

模型表

model_key	model_value
detect_model	{'featureName': ['1', '2', '3', '4', '5', '6', '7', '8', '9'], 'frequency_info': {'mid': 73.0, 'q1': 58.0, 'q3': 74.0, 'threshold': 34.0}}

7.5.1.3.18 特征异常平滑

概述

特征异常平滑算子用于将数据中的异常数据平滑到一定的区间，可选择采用箱线图、阈值、百分位和z-score的方法确定平滑区间。

- z-score方式：计算所需要平滑的特征的均值mean和标准差std，并引入置信因子cl

$$mean = \frac{\sum_{i=1}^{n} x_i}{n}$$

版权所有 © 华为云计算技术有限公司

$$std = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}}$$

平滑区间上界：$mean + cl * std$

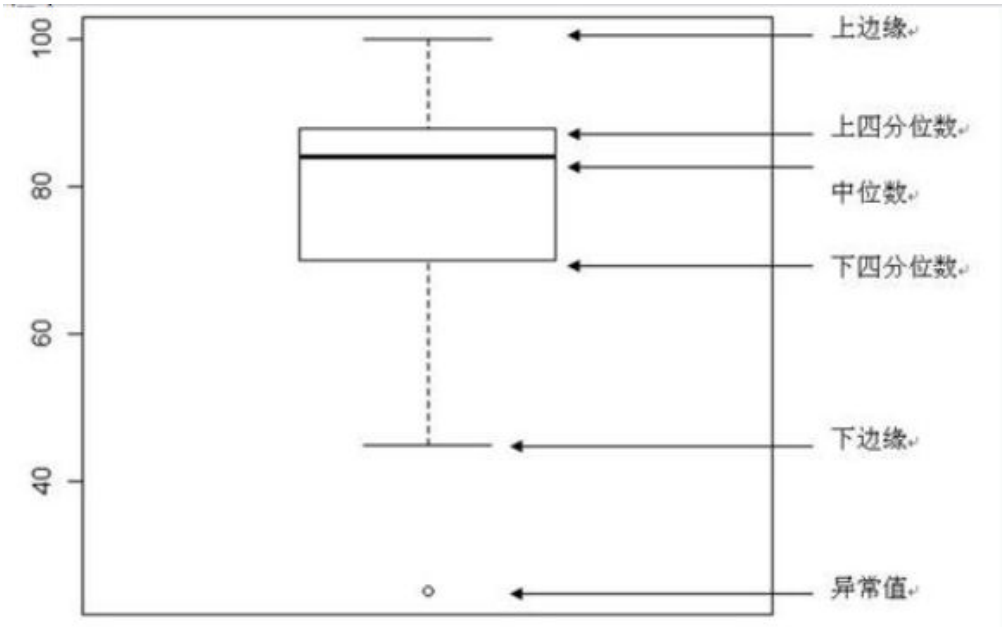
平滑区间下界：$mean - cl * std$

- min-max-per方式：通过上下百分位计算，假设需要平滑的特征列最大值为max，上百分位为max_per，下百分位为min_per

  平滑区间上界：$min+(max-min)*max\_per$

  平滑区间下界：$min+(max-min)*min\_per$

- min-max-thresh方式：直接指定平滑区间的上界和下界
- boxplot方式：通过箱线图的方式计算平滑区间



中位数（Q2 / 50th百分位数）：数据集的中间值；

下四分位数（Q1 / 25百分位数）：最小数（不是“最小值”）和数据集的中位数之间的中间数；

上四分位数（Q3 / 75th Percentile）：数据集的中位数和最大值之间的中间值（不是“最大值”）；

四分位间距（IQR）：第25至第75个百分点的距离

上边缘：Q3 + 1.5 * IQR

下边缘：Q1 -1.5 * IQR

平滑区间上界：上边缘

版权所有 © 华为云计算技术有限公司

平滑区间下界：下边缘

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象。

输出

参数	子参数	参数说明
output	output_port_1	output为字典类型，output_port_1为pyspark中的PipelineModel类型对象，特征异常平滑模型。
output	output_port_2	output_port_2为pyspark中的DataFrame类型，为特征异常平滑结果。

参数说明

参数	是否必选	参数说明	默认值
soften_cols	是	需要进行特征异常平滑处理的列，逗号分隔。	无
soften_method	是	特征平滑方法，可选z-score,min-max-per,min-max-thresh,boxplot。	"z-score"
keep_original	是	是否保留原始列，如果保留则新增列，列名为原始列前加'soften_'。	False
cl	否	置信水平，当选择z-score方法时需要配置此参数。	1
min_per	否	最低百分位。当平滑方法为min-max-per时需要配置该参数。	0.0

版权所有 © 华为云计算技术有限公司

参数	是否必选	参数说明	默认值
max_per	否	最高百分位。当平滑方法为min-max-per时需要配置该参数。	0.1
min_thresh	否	阈值最小值。当平滑方法为min-max-thresh时需要配置该参数。	-9999
max_thresh	否	阈值最大值。当平滑方法为阈值平滑时需要配置该参数。	9999
is_sparse	是	是否为k:v的稀疏特征, 如果指定该列，soften_cols参数只支持选择稀疏特征列kv_col中的列名。	False
kv_col	否	如果为稀疏特征，指定稀疏特征列名。	"kv"
item_spliter	否	稀疏特征的分隔符。	","
kv_spliter	否	稀疏特征key和value的分隔符。	":"

样例

数据样本

样例1 非稀疏数据

value1	value2	index
-100	-80	0
1	2	1
1	2	2
3	4	3
4	6	4
100	50	5

样例2 稀疏数据

版权所有 © 华为云计算技术有限公司

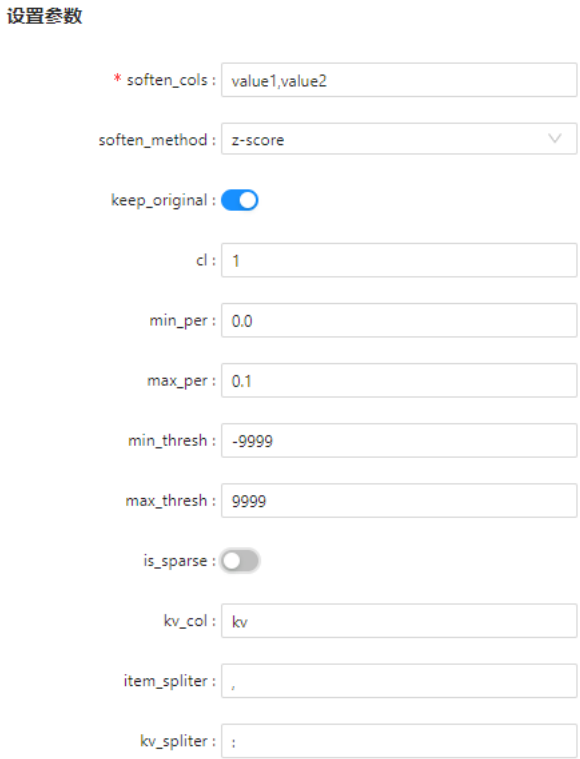
kv	rowid
f0:-100, f1:-80	0
	1
f0:3	2
f0:3, f1:4	3
f0:100, f1:50	4

配置流程

运行流程

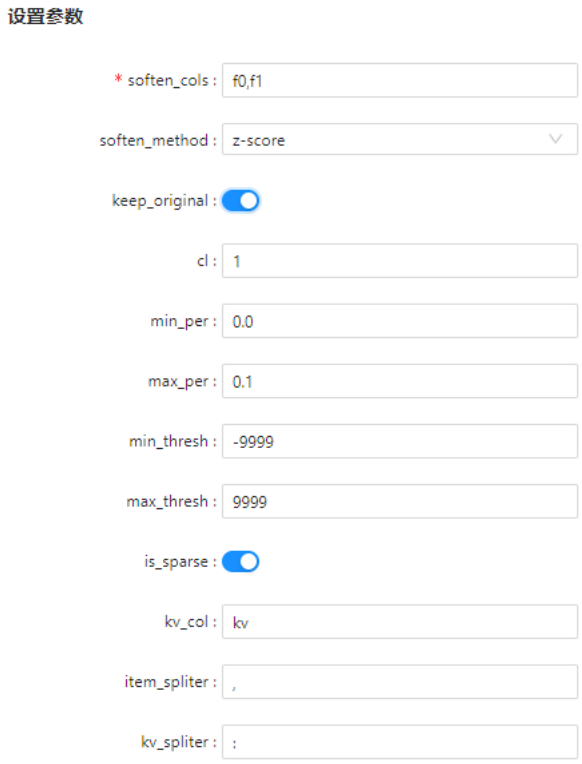
参数设置

图 7-66 样例 1 数据参数设置



版权所有 © 华为云计算技术有限公司

图 7-67 样例 2 数据参数设置



结果查看

图 7-68 样例 1 数据运行结果

value1	value2	index	soften_value1	soften_value2
-100	-80	0	-56.25450920346682	-41.21821945280256
1	2	1	1.0	2.0
1	2	2	1.0	2.0
3	4	3	3.0	4.0
4	6	4	4.0	6.0
100	50	5	59.25450920346682	35.88488611946923

图 7-69 样例 2 数据运行结果

kv	rowid	soften_kv
f0:-100, f1:-80	0	l,f1:-62.48942104517176
	1	
f0:3	2	f0:3
f0:3, f1:4	3	f0:3,f1:4
f0:100, f1:50	4	l1,f1:45.15608771183843

版权所有 © 华为云计算技术有限公司

7.5.1.3.19 gbdt 编码模型训练

概述

利用训练好的gbdt分类模型对输入的特征进行离散化处理。对每棵树的叶子节点进行编码，预测的时候遍历到叶子节点对应位置的编码为1，该树其余节点的编码为0。该节点主要用于生产gbdt的分类模型，并存储到输入参数对应的位置上。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象，用于生成gbdt编码的模型

输出

参数	子参数	参数说明
outputs	output_port_1	指向一个pyspark的DataFrame类型对象，该对象为一个空的dataframe

参数说明

参数	是否必选	参数说明	默认值
input_features	是	输入的特征（需要编码的特征）	"feature"
label_column	是	预测结果类别的字段名	"label"
model_saved_path	是	模型保存的路径	""
max_iter	是	最大迭代次数（树的棵数）	4
max_depth	是	树的最大深度	5
subsampling_rate	是	构建单棵树的采样比例	1.0
feature_subset_strategy	是	构建单棵树的特征抽取策略，取值为"auto"、"all"	"auto"

版权所有 © 华为云计算技术有限公司

样例

数据样本

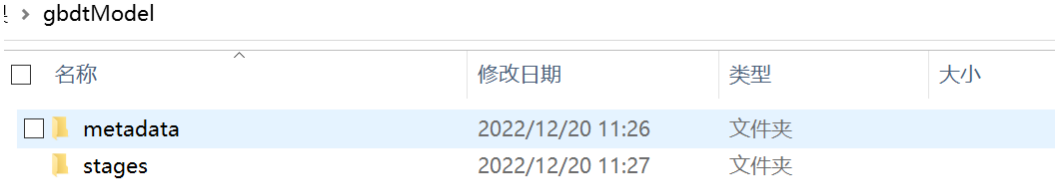
```
label,age,count
1,20,23
0,19,33
0,21,24
1,7,24
0,11,43
1,32,12
0,21,43
1,32,45
```

配置流程

运行流程

参数设置

查看结果



版权所有 © 华为云计算技术有限公司

7.5.1.3.20 gbdt 编码模型应用

概述

利用训练好的gbdt分类模型对输入的特征进行离散化处理。对每棵树的叶子节点进行编码，预测的时候遍历到叶子节点对应位置的编码为1，该树其余节点的编码为0。该节点主要用于读取gbdt编码模型训练阶段保存的模型，并对数据进行离散化编码。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象，用于进行gbdt特征编码

输出

表 7-7

参数	子参数	参数说明
outputs	output_port_1	指向一个pyspark的DataFrame类型对象，该对象为原始的输入数据

参数说明

表 7-8

参数	是否必选	参数说明	默认值
model_saved_path	是	模型存储的位置，需要和对应的gbdt编码模型训练节点对应参数保持一致	""

样例

数据样本

```
label,age,count
1,20,23
0,19,33
0,21,24
1,7,24
0,11,43
1,32,12
```

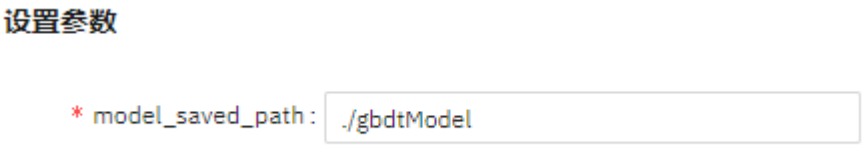

版权所有 © 华为云计算技术有限公司

```
0,21,43
1,32,45
```

配置流程

运行流程

参数设置



查看结果

gbdt编码后的特征为：encodedKVFeature

```
label,age,count,gbt_assembled_features,standard_scale_feature,label_index,encodedKVFeature
1,20,23,"[20.0,23.0]","[2.277363769238288,1.9139391581953824]",1.0,"3:1,4:1,11:1,12:1"
0,19,33,"[19.0,33.0]","[2.1634955807763734,2.7460866182803314]",0.0,"1:1,7:1,9:1,14:1"
0,21,24,"[21.0,24.0]","[2.391231957700202,1.9971539042038775]",0.0,"2:1,7:1,10:1,14:1"
1,7,24,"[7.0,24.0]","[0.7970773192334007,1.9971539042038775]",1.0,"3:1,5:1,11:1,15:1"
0,11,43,"[11.0,43.0]","[1.2525500730810584,3.57823407836528]",0.0,"1:1,7:1,9:1,14:1"
1,32,12,"[32.0,12.0]","[3.6437820307812605,0.9985769521019388]",1.0,"0:1,4:1,8:1,12:1"
0,21,43,"[21.0,43.0]","[2.391231957700202,3.57823407836528]",0.0,"1:1,7:1,9:1,14:1"
1,32,45,"[32.0,45.0]","[3.6437820307812605,3.74466357038227]",1.0,"0:1,6:1,8:1,13:1"
```

7.5.2 输入输出

7.5.2.1 输入

7.5.2.1.1 读取 DLI 表

概述

读取用户于DLI服务创建的外表（OBS表）。

输入

无

版权所有 © 华为云计算技术有限公司

输出

数据集

参数说明

参数	参数说明
DLI_database	用户的目标DLI数据库名称
DLI_table	用户的目标DLI数据库中目标DLI外表（OBS表）的名称

样例

```
params = {
    "DLI_database": None,  # @param {"label":"DLI_database","type":"string","required":"true","helpTip":""}
    "DLI_table": None  # @param {"label":"DLI_table","type":"string","required":"true","helpTip":""}
}
read_DLI_table____id___ = MLSReadDLITable(**params)
read_DLI_table____id___.run()
# @output {"label":"dataframe","name":"read_DLI_table____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.2.1.2 读取数据

概述

读取格式化的数据，支持csv、json、parquet等。

输入

无

输出

数据集

参数说明

参数	参数说明
input_file_path	数据文件的绝对路径、相对路径、目录路径或者文件路径均可
format	文件格式，支持csv等
has_header	是否包含表头
delimiter	csv数据每行的分隔符，默认为逗号

样例

```
params = {
    "input_file_path": "",  # @param {"label":"input_file_path","type":"path","required":"true","helpTip":""}
```

 版权所有 © 华为云计算技术有限公司

```
    "format": "csv",  # @param {"label":"format","type":"string","required":"false","helpTip":""}
    "has_header": True,  # @param {"label":"has_header","type":"boolean","required":"false","helpTip":""}
    "delimiter": ","  # @param {"label":"delimiter","type":"string","required":"false","helpTip":""}
}
read_data____id___ = MLSReadData(**params)
read_data____id___.run()
# @output {"label":"dataframe","name":"read_data____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.2.1.3 读取模型

概述

读取spark pipeline model类型的模型文件。

输入

无

输出

spark pipeline model类型的模型对象

参数说明

参数	参数说明
input_model_path	模型文件所在的路径

样例

```
params = {
    "input_model_path": ""  # @param
{"label":"input_model_path","type":"path","required":"true","helpTip":""}
}
read_model____id___ = MLSReadPipelineModel(**params)
read_model____id___.run()
# @output {"label":"pipeline_model","name":"read_model____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
```

7.5.2.1.4 从 OBS 读取 CSV 数据

概述

从OBS（对象存储服务）中读取csv格式的数据。

输入

无

输出

数据集

 版权所有 © 华为云计算技术有限公司

参数说明

参数	参数说明
obs_data_path	OBS中csv数据文件或数据所在文件夹的绝对路径，例如：obs://桶名称/文件夹/数据文件，obs://桶名称/文件夹
has_header	是否包含表头
delimiter	csv数据每行的分隔符，默认为逗号

样例

```
params = {
    "obs_data_path": "",  # @param {"label":"obs_data_path","type":"string","required":"true","helpTip":""}
    "has_header": True,  # @param {"label":"has_header","type":"boolean","required":"false","helpTip":""}
    "delimiter": ","  # @param {"label":"delimiter","type":"string","required":"true","helpTip":""}
}
mls_read_obs_data____id___ = MLSReadOBSData(**params)
mls_read_obs_data____id___.run()
# @output {"label":"dataframe","name":"mls_read_obs_data____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.2.1.5 从 OBS 读取模型

概述

从OBS（对象存储服务）中读取模型文件。

输入

无

输出

模型

参数说明

参数	参数说明
obs_model_path	OBS中模型文件的绝对路径，模型文件必须是spark pipeline model文件

样例

```
params = {
    "obs_model_path": ""  # @param {"label":"obs_model_path","type":"string","required":"true","helpTip":""}
}
mls_read_obs_model____id___ = MLSReadOBSPipelineModel(**params)
mls_read_obs_model____id___.run()
# @output {"label":"pipeline_model","name":"mls_read_obs_model____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
```

 版权所有 © 华为云计算技术有限公司

7.5.2.1.6 读取 parquet 数据

概述

读取parquet格式的数据。

输入

无

输出

数据集

参数说明

参数	参数说明
input_file_path	parquet数据文件所在的路径

样例

```
params = {
    "input_file_path": ""   # @param {"label":"input_file_path","type":"path","required":"true","helpTip":""}
}
read_parquet_data____id___ = MLSReadParquetData(**params)
read_parquet_data____id___.run()
# @output {"label":"dataframe","name":"read_parquet_data____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.2.1.7 读取文本数据

概述

读取文本格式的数据。

输入

无

输出

数据集

参数说明

参数	参数说明
input_path	文本数据文件所在的路径
line_separator	分隔符，默认为换行符"\n"

版权所有 © 华为云计算技术有限公司

参数	参数说明
columns_str	列名以逗号分隔的字符串，默认一行为一列，列名为"text_col"

样例

```
params = {
    "input_path": "",  # @param {"label":"input_path","type":"path","required":"true","helpTip":""}
    "line_separator": "\n",  # @param {"label":"line_separator","type":"string","required":"false","helpTip":""}
    "columns_str": "text_col"  # @param {"label":"columns_str","type":"string","required":"false","helpTip":""}
}
read_text_data____id___ = MLSReadTextData(**params)
read_text_data____id___.run()
# @output {"label":"dataframe","name":"read_text_data____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.2.1.8 读 CSV 文件

概述

读CSV文件支持从LOCAL、OBS、HDFS读取CSV类型的文件数据。

输入

无

输出

表 7-9

参数	子参数	参数说明
output	output_port_1	output为字典类型，output_port_1为pyspark中的DataFrame类型对象，为算子读取的结果。

版权所有 © 华为云计算技术有限公司

参数说明

参数名称	是否必选	参数说明	默认值
schema_str	是	非空字符串 schema：配置每一列对应的数据类型，格式为colname0 coltype0[, colname1 coltype1[, ...]]。例如：f0 string,f1 bigint,f2 double。 注意：配置的数据类型需要与CSV文件每一列的数据类型保持一致，否则该列内容会读取失败。	无
local_file_path	否	本地文件路径 非必须，可通过文件夹选取；仅当file_source为LOCAL时，该路径有效。	无
file_path	否	读取CSV文件的路径 1. 当文件来源选择OBS时，支持输入OBS文件路径,此时路径必须以OBS://开头。 2. 当文件来源选择OTHERS时，支持输入HDFS文件路径。	无
file_source	否	支持LOCAL、OBS和OTHERS。范围：['LOCAL','OBS','OTHERS']	LOCAL
field_delimiter	否	字段分隔符；如果输入则必须为字符	,
handle_invalid_method	否	处理无效值的方法(无效值表示schema_str中设置的数据类型和csv中的不符)，取值如下： 1.ERROR：抛出异常 2.SKIP：使用csv中的格式替换	ERROR
ignore_first_line	否	是否忽略第一行的数据。 如果原表中已有表头，则需要开启此开关，否则会报错。	FALSE
quote_string	否	引号字符，设置用于转义引号值的单个字符。	"
row_delimiter	否	行分隔符。	\n

版权所有 © 华为云计算技术有限公司

参数名称	是否必选	参数说明	默认值
skip_blank_line	否	是否忽略空行。 如果为True，该行数据全空时忽略；否则不忽略。	TRUE

说明

1. schema_str这个参数，相当于增加列名（如果csv没有列名，则增加列名，ignore_first_line需置为False）或 重命名列名（如果csv有列名，可以改列名，ignore_first_line需置为True）。

2. 只支持string，bigint，double类型，之后如果是想改变数据类型，需使用新算子做类型转换；其中tinyint、smallint、int均为bigint类型，char、varchar、date等其他类型均为string类型。

3. 该算子默认以"\n"作为行分隔符，如果某一字段内部存在"\n",需要提前处理；例如；将"\n"提前替换为空格，防止读取失败。示例如下：

```
import pandas as pd
df = pd.read_csv("test.csv",index_col=0)
df = df.replace(to_replace=r'[\n\r]', value=' ', regex=True, inplace=True)
df.to_csv("output.csv")
```

样例

数据样本

```
5.1,3.5,1.4,0.2,Iris-setosa
5.0,2.0,3.5,1.0,Iris-versicolor
5.1,3.7,1.5,0.4,Iris-setosa
6.4,2.8,5.6,2.2,Iris-virginica
6.0,2.9,4.5,1.5,Iris-versicolor
4.9,3.0,1.4,0.2,Iris-setosa
5.7,2.6,3.5,1.0,Iris-versicolor
4.6,3.6,1.0,0.2,Iris-setosa
5.9,3.0,4.2,1.5,Iris-versicolor
6.3,2.8,5.1,1.5,Iris-virginica
4.7,3.2,1.3,0.2,Iris-setosa
5.1,3.3,1.7,0.5,Iris-setosa
5.5,2.4,3.8,1.1,Iris-versicolor
```

配置流程

运行流程

算法参数设置

schema_str: sepal_length double, sepal_width double, petal_length double, petal_width double, category string

版权所有 © 华为云计算技术有限公司

查看结果

	sepal_length	sepal_width	petal_length	petal_width	category
1	5.1	3.5	1.4	0.2	Iris-setosa
2	5.0	2.0	3.5	1.0	Iris-versicolor
3	5.1	3.7	1.5	0.4	Iris-setosa
4	6.4	2.8	5.6	2.2	Iris-virginica
5	6.0	2.9	4.5	1.5	Iris-versicolor
6	4.9	3.0	1.4	0.2	Iris-setosa
7	5.7	2.6	3.5	1.0	Iris-versicolor
8	4.6	3.6	1.0	0.2	Iris-setosa
9	5.9	3.0	4.2	1.5	Iris-versicolor
10	6.3	2.8	5.1	1.5	Iris-virginica
11	4.7	3.2	1.3	0.2	Iris-setosa
12	5.1	3.3	1.7	0.5	Iris-setosa
13	5.5	2.4	3.8	1.1	Iris-versicolor

7.5.2.2 输出

7.5.2.2.1 保存为 DLI OBS 表

概述

保存数据到用户的DLI外表（OBS表）。

版权所有 © 华为云计算技术有限公司

说明

保存为DLI外表时，对vector类型的数据转换为array类型，如果无需保存vector数据，可在该算子前添加数据集选择列算子。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

无

参数说明

参数	子参数	参数说明
DLI_database	-	用户的目标DLI数据库名称
DLI_table	-	用户的目标DLI数据库中目标DLI外表或要新建DLI外表的名称
file_format	-	DLI外表使用的数据格式
mode	-	数据的写入类型（追加或覆盖，默认为覆盖模式）。使用PySpark insertInto函数，因此追加或者覆盖都要保证特征列数量和顺序一致
OBS_path	-	用户目标DLI外表的OBS存储路径

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "DLI_database": None,  # @param {"label":"DLI_database","type":"string","required":"true","helpTip":""}
    "DLI_table": None,  # @param {"label":"DLI_table","type":"string","required":"true","helpTip":""}
    "file_format": "parquet",  # @param
{"label":"file_format","type":"enum","options":"orc,parquet,json,csv,carbon,avro","required":"true","helpTip":"
"}
    "mode": "overwrite",  # @param
{"label":"mode","type":"enum","options":"overwrite,append","required":"true","helpTip":""}
    "OBS_path": ""  # @param {"label":"OBS_path","type":"string","required":"true","helpTip":""}
}
save_DLI_table____id___ = MLSSaveDLITable(**params)
save_DLI_table____id___.run()
```


版权所有 © 华为云计算技术有限公司

7.5.2.2.2 保存数据

概述

保存数据到本地文件系统。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

无

参数说明

参数	子参数	参数说明
output_file_path	-	输出路径
save_mode	-	保存模式，默认为overwrite
has_header	-	是否包含表头，默认为True

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "output_file_path": "",  # @param {"label":"output_file_path","type":"path","required":"true","helpTip":""}
    "save_mode": "overwrite",  # @param {"label":"save_mode","type":"string","required":"true","helpTip":""}
    "has_header": True  # @param {"label":"has_header","type":"boolean","required":"true","helpTip":""}
}
save_data____id___ = MLSSaveData(**params)
save_data____id___.run()
```

7.5.2.2.3 保存 CSV 数据到 OBS

概述

将csv格式的数据保存到OBS

版权所有 © 华为云计算技术有限公司

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

无

参数说明

参数	子参数	参数说明
output_file_path	-	保存数据到OBS的路径
encoding	-	编码，默认为utf-8

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "output_file_path": "",  # @param
{"label":"output_file_path","type":"string","required":"true","helpTip":""}
    "encoding": "utf-8"  # @param {"label":"encoding","type":"string","required":"true","helpTip":""}
}
mls_save_data____id___ = MLSSaveDataToOBS(**params)
mls_save_data____id___.run()
```

7.5.2.2.4 保存模型

概述

保存spark pipeline类型的模型到本地文件系统。

输入

参数	子参数	参数说明
inputs	pipeline_model	inputs为字典类型，pipeline_model为pyspark中的PipelineModel对象

输出

无

版权所有 © 华为云计算技术有限公司

参数说明

参数	子参数	参数说明
output_model_path	-	输出路径

样例

```
inputs = {
    "pipeline_model": None  # @input {"label":"pipeline_model","type":"PipelineModel"}
}
params = {
    "inputs": inputs,
    "output_model_path": ""  # @param
{"label":"output_model_path","type":"path","required":"true","helpTip":""}
}
save_model____id___ = MLSSavePipelineModel(**params)
save_model____id___.run()
```

7.5.2.2.5 保存模型到 OBS

概述

将训练出来的spark标准pipeline类型的模型保存到OBS里面

输入

参数	子参数	参数说明
inputs	pipeline_model	inputs为字典类型，pipeline_model为pyspark中的PipelineModel对象

输出

无

参数说明

参数	子参数	参数说明
obs_model_path	-	保存模型到obs的obs路径

样例

```
inputs = {
    "pipeline_model": None  # @input {"label":"pipeline_model","type":"PipelineModel"}
}
params = {
    "inputs": inputs,
    "obs_model_path": ""  # @param {"label":"obs_model_path","type":"string","required":"true","helpTip":""}
}
```


版权所有 © 华为云计算技术有限公司

```
mls_save_model____id___ = MLSSavePipelineModelToOBS(**params)
mls_save_model____id___.run()
```

7.5.2.2.6 保存 parquet 数据

概述

保存parquet格式的数据到本地文件系统。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

无

参数说明

参数	子参数	参数说明
output_file_path	-	输出路径
save_mode	-	保存模式，默认为"overwrite"

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "output_file_path": "",  # @param {"label":"output_file_path","type":"path","required":"true","helpTip":""}
    "save_mode": "overwrite"  # @param {"label":"save_mode","type":"string","required":"false","helpTip":""}
}
save_parquet_data____id___ = MLSSaveParquetData(**params)
save_parquet_data____id___.run()
```

7.5.2.2.7 数据压缩

概述

将数据压缩后到本地文件系统。

版权所有 © 华为云计算技术有限公司

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

无

参数说明

参数	子参数	参数说明
data_delimeter	-	数据分割符
compression_type	-	数据压缩的格式，当前支持Bzip2，deflate，Gzip三种方式
data_partition	-	数据保存的分区
data_path	-	数据保存路径

样例

```
inputs = {
    "dataframe": None,  # @input {"label":"dataDF","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "data_delimeter": ",",  # @param {"label":"data_delimeter","type":"string","required":"false","helpTip":""}
    "compression_type": "Bzip2",  # @param {"label":"task","type":"enum",
"options":"Bzip2,deflate,Gzip","required":"true","helpTip":""}
    "data_partition": 1,  # @param {"label":"data_partition","type":"int","required":"false","helpTip":""}
    "data_path": "",  # @param {"label":"data_path","type":"path","required":"true","helpTip":""}
}
execute_compress____id___ = MLSSaveWithCompression(**params)
execute_compress____id___.run()
```

7.5.3 模型工程

7.5.3.1 分类

7.5.3.1.1 决策树分类

概述

“决策树分类”节点用于产生二分类或多分类模型。

决策树是附加概率结果的一个树状的决策图，是直观的运用统计概率分析的图法，树中的每一个节点表示对象属性的判断条件，其分支表示符合节点条件的对象，树的叶

版权所有 © 华为云计算技术有限公司

子节点表示对象所属的预测结果。其通过基尼不纯度（Gini impurity）或熵（Entropy）来对一个集合的有序程度进行量化，并对一次拆分进行量化评价。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

参数说明

参数	子参数	参数说明
b_use_default_encoder	-	是否使用默认编码，默认为True
input_features_str	-	输入的特征列名以逗号分隔组成的格式化字符串，例如: "column_a" "column_a,column_b"
label_col	-	目标列
classifier_label_index_col	-	目标列经过标签编码后的新的列名，默认为"label_index"
classifier_feature_vector_col	-	算子输入的特征向量列的列名，默认为"model_features"
prediction_index_col	-	算子输出的预测label对应的标签列，默认为"prediction_index"
prediction_col	-	算子输出的预测label的列名，默认为"prediction"
max_depth	-	树的最大深度，默认为5
max_bins	-	最大分箱数，默认为32
min_instances_per_node	-	树节点分割时要求子节点包含的最小实例数，默认为1
min_info_gain	-	最小信息增益，默认为0
impurity	-	不纯度，支持entropy、gini，默认为"gini"

版权所有 © 华为云计算技术有限公司

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "b_use_default_encoder": True,  # @param {"label": "b_use_default_encoder", "type": "boolean", "required": "true", "helpTip": ""}
    "input_features_str": "",  # @param {"label": "input_features_str", "type": "string", "required": "false", "helpTip": ""}
    "outer_pipeline_stages": None,
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": ""}
    "classifier_label_index_col": "label_index",  # @param {"label": "classifier_label_index_col", "type": "string", "required": "true", "helpTip": ""}
    "classifier_feature_vector_col": "model_features",  # @param {"label": "classifier_feature_vector_col", "type": "string", "required": "true", "helpTip": ""}
    "prediction_index_col": "prediction_index",  # @param {"label": "prediction_index_col", "type": "string", "required": "true", "helpTip": ""}
    "prediction_col": "prediction",  # @param {"label": "prediction_col", "type": "string", "required": "true", "helpTip": ""}
    "max_depth": 5,  # @param {"label": "max_depth", "type": "integer", "required": "true", "range":"(0,2147483647]", "helpTip": ""}
    "max_bins": 32,  # @param {"label": "max_bins", "type": "integer", "required": "true", "range":"(0,2147483647]", "helpTip": ""}
    "min_instances_per_node": 1,  # @param {"label": "min_instances_per_node", "type": "integer", "required": "true", "range": "[1,2147483647]", "helpTip": ""}
    "min_info_gain": 0.0,  # @param {"label": "min_info_gain", "type": "number", "required": "true", "range": "[0,none)", "helpTip": ""}
    "impurity": "gini"  # @param {"label": "impurity", "type": "enum", "required": "true", "options": "entropy,gini", "helpTip": ""}
}
dt_classifier____id___ = MLSDecisionTreeClassifier(**params)
dt_classifier____id___.run()
# @output {"label":"pipeline_model","name":"dt_classifier____id___.get_outputs()['output_port_1']","type":"PipelineModel"}
```

7.5.3.1.2 梯度提升树分类

概述

“梯度提升树分类”节点用于生成二分类模型，是一种基于决策树的迭代分类算法。该算法采用迭代的思想不断地构建决策树模型，每棵树都是通过梯度优化损失函数而构建，从而达到从基准值到目标值的逼近。算法思想可简单理解成：后一次模型都是针对前一次模型预测出错的情况进行修正，模型随着迭代不断地改进，从而获得比较好的预测效果。

梯度提升树分类的损失函数为对数似然损失函数，如下所示：

$$2\sum_{i=1}^{N}\log(1+\exp(-2y_iF(x_i)))$$

式中，N 表示样本数量，x_i 表示样本 i 的特征，y_i 表示样本 i 的标签，$F(x_i)$ 表示样本 i 预测的标签。

 版权所有 © 华为云计算技术有限公司

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

参数说明

参数	子参数	参数说明
input_features_str	-	输入的列名以逗号分隔组成的字符串，例如： "column_a" "column_a,column_b"
label_col	-	目标列
classifier_label_index_col	-	目标列经过标签编码后的新的列名，默认为"label_index"
classifier_feature_vector_col	-	算子输入的特征向量列的列名，默认为"model_features"
prediction_index_col	-	算子输出的预测label对应的标签列，默认为"prediction_index"
prediction_col	-	算子输出的预测label的列名，默认为"prediction"
max_depth	-	树的最大深度，默认为5
max_bins	-	最大分箱数，默认为32
min_instances_per_node	-	树节点分割时要求子节点包含的最小实例数，默认为1
min_info_gain	-	最小信息增益，默认为0
max_iter	-	最大迭代次数，默认为20
step_size	-	步长，默认为0.1
subsampling_rate	-	训练每棵树时对训练集的抽样率，默认为1.0

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
```


版权所有 © 华为云计算技术有限公司

```
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "b_use_default_encoder": True,
    "input_features_str": "",  # @param {"label": "input_features_str", "type": "string", "required": "false", "helpTip": ""}
    "outer_pipeline_stages": None,
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": "target label column"}
    "classifier_label_index_col": "label_index",  # @param {"label": "classifier_label_index_col", "type": "string", "required": "true", "helpTip": ""}
    "classifier_feature_vector_col": "model_features",  # @param {"label": "classifier_feature_vector_col", "type": "string", "required": "true", "helpTip": ""}
    "prediction_index_col": "prediction_index",  # @param {"label": "prediction_index_col", "type": "string", "required": "true", "helpTip": ""}
    "prediction_col": "prediction",  # @param {"label": "prediction_col", "type": "string", "required": "true", "helpTip": ""}
    "max_depth": 5,  # @param {"label": "max_depth", "type": "integer", "required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "max_bins": 32,  # @param {"label": "max_bins", "type": "integer", "required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "min_instances_per_node": 1,  # @param {"label": "min_instances_per_node", "type": "integer", "required": "true", "range":"(0,2147483647]", "helpTip": ""}
    "min_info_gain": 0.0,  # @param {"label": "min_info_gain", "type": "number", "required": "true", "range": "[0,none)", "helpTip": ""}
    "loss_type": "logistic",
    "max_iter": 20,  # @param {"label": "max_iter", "type": "integer", "required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "step_size": 0.1,  # @param {"label": "step_size", "type": "number", "required": "true", "range": "(0,none)", "helpTip": ""}
    "subsampling_rate": 1.0  # @param {"label": "subsampling_rate", "type": "number", "required": "true", "range": "(0,1.0]", "helpTip": ""}
}
gbt_classifier____id___ = MLSGBTClassifier(**params)
gbt_classifier____id___.run()
# @output {"label":"pipeline_model","name":"gbt_classifier____id___.get_outputs()['output_port_1']","type":"PipelineModel"}
```

7.5.3.1.3 LightGBM 分类

概述

对mmlspark python包中LightGBM分类的封装

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

 版权所有 © 华为云计算技术有限公司

参数说明

参数	子参数	参数说明
input_features_str	-	输入的列名以逗号分隔组成的字符串，例如： "column_a" "column_a,column_b"
label_col	-	目标列
classifier_label_index_col	-	目标列经过标签编码后的新的列名，默认为"label_index"
classifier_feature_vector_col	-	算子输入的特征向量列的列名，默认为"model_features"
prediction_index_col	-	算子输出的预测label对应的标签列，默认为"prediction_index"
prediction_col	-	算子输出的预测label的列名，默认为"prediction"
probability_col	-	算子输出的概率列的列名，默认为"probability"
is_unbalance	-	数据集是否不平衡，默认为False
timeout	-	超时时间，默认为1200秒
objective	-	目标函数，支持binary,multiclass,multiclassova，默认为"binary"
max_depth	-	树的最大深度，默认为-1
num_iteration	-	迭代次数，默认为100
learning_rate	-	学习率，默认为0.1
num_leaves	-	叶子数目，默认为31
max_bin	-	最大分箱数，默认为255
bagging_fraction	-	bagging的比例，默认为1
bagging_freq	-	bagging的频率，默认为0
bagging_seed	-	bagging时的随机数种子，默认为3
early_stopping_round	-	提前结束迭代的轮数，默认为0
feature_fraction	-	特征的比例，默认为1.0
min_sum_hessian_in_leaf	-	一个叶子上最小hessian和。取值区间为[0, 1]，默认为1e-3
boost_from_average	-	是否将初始分数调整为标签的平均值，以加快收敛速度,，默认为True

版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
boosting_type	-	提升方法的提升类型。 可选值有：gbdt、gbrt、rf、dart、goss，默认为"gbdt"
lambda_l1	-	L1正则化系数，默认为0.0
lambda_l2	-	L2正则化系数,，默认为0.0
num_batches	-	如果大于0，在训练中将数据集分割成不同的批次，默认为0
parallelism	-	学习树时的并行方法，支持data_parallel, voting_parallel，默认为"data_parallel"
thresholds_str	-	多分类时使用，表示每个类别对应的概率值预置的数组，字符串用逗号隔开

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "outer_pipeline_stages": None,
    "input_features_str": "",  # @param
{"label":"input_features_str","type":"string","required":"false","helpTip":""}
    "label_col": "",  # @param {"label":"label_col","type":"string","required":"true","helpTip":""}
    "classifier_label_index_col": "label_index",  # @param
{"label":"classifier_label_index_col","type":"string","required":"false","helpTip":""}
    "classifier_feature_vector_col": "model_features",  # @param
{"label":"classifier_feature_vector_col","type":"string","required":"false","helpTip":""}
    "prediction_index_col": "prediction_index",  # @param
{"label":"prediction_index_col","type":"string","required":"false","helpTip":""}
    "prediction_col": "prediction",  # @param
{"label":"prediction_col","type":"string","required":"false","helpTip":""}
    "probability_col": "probability",  # @param
{"label":"probability_col","type":"string","required":"false","helpTip":""}
    "is_unbalance": False,  # @param {"label":"is_unbalance","type":"boolean","required":"false","helpTip":""}
    "timeout": 1200.0,  # @param {"label":"timeout","type":"number","required":"false","helpTip":""}
    "objective": "binary",  # @param {"label":"objective","type":"string","required":"false","helpTip":""}
    "max_depth": -1,  # @param
{"label":"max_depth","type":"integer","required":"false","range":"[-1,2147483647]","helpTip":""}
    "num_iteration": 100,  # @param
{"label":"num_iteration","type":"integer","required":"false","range":"(0,2147483647]","helpTip":""}
    "learning_rate": 0.1,  # @param {"label":"learning_rate","type":"number","required":"false","helpTip":""}
    "num_leaves": 31,  # @param
{"label":"num_leaves","type":"integer","required":"false","range":"(0,2147483647]","helpTip":""}
    "max_bin": 255,  # @param
{"label":"max_bin","type":"integer","required":"false","range":"(0,2147483647]","helpTip":""}
    "bagging_fraction": 1.0,  # @param
{"label":"bagging_fraction","type":"number","required":"false","helpTip":""}
    "bagging_freq": 0,  # @param
{"label":"bagging_freq","type":"integer","required":"false","range":"[0,2147483647]","helpTip":""}
    "bagging_seed": 3,  # @param
{"label":"bagging_seed","type":"integer","required":"false","range":"[0,2147483647]","helpTip":""}
    "early_stopping_round": 0,  # @param
{"label":"early_stopping_round","type":"integer","required":"false","range":"[0,2147483647]","helpTip":""}
    "feature_fraction": 1.0,  # @param
{"label":"feature_fraction","type":"number","required":"false","helpTip":""}
```


版权所有 © 华为云计算技术有限公司

```
    "min_sum_hessian_in_leaf": 1e-3,  # @param
{"label":"min_sum_hessian_in_leaf","type":"number","required":"false","helpTip":""}
    "boost_from_average": True,  # @param
{"label":"boost_from_average","type":"boolean","required":"false","helpTip":""}
    "boosting_type": "gbdt",  # @param
{"label":"boosting_type","type":"string","required":"false","helpTip":""}
    "lambda_l1": 0.0,  # @param {"label":"lambda_l1","type":"number","required":"false","helpTip":""}
    "lambda_l2": 0.0,  # @param {"label":"lambda_l2","type":"number","required":"false","helpTip":""}
    "num_batches": 0,  # @param
{"label":"num_batches","type":"integer","required":"false","range":"[0,2147483647]","helpTip":""}
    "parallelism": "data_parallel",  # @param
{"label":"parallelism","type":"string","required":"false","helpTip":""}
    "thresholds_str": ""   # @param {"label":"thresholds_str","type":"string","required":"false","helpTip":""}
}
lightgbm_classifier____id___ = MLSLightGBMClassifier(**params)
lightgbm_classifier____id___.run()
# @output {"label":"pipeline_model","name":"lightgbm_classifier____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
```

7.5.3.1.4 线性支持向量机分类

概述

“支持向量机分类”节点构造一个线性支持向量机模型，支持二分类和多分类。该节点采用Trust Region Newton Method（TRON）算法优化L2-SVM模型，更适用于大规模数据的建模，模型训练效率更高。

算法实现方式的简介如下：

- 二分类

  给定训练集(x_i,y_i), $i = 1,...n$, $x_i \in R^p$, $y_i \in \{-1, +1\}$，惩罚系数C，通过TRON优化方法求解以下非约束优化问题，得出权值向量w和偏置量b：

  $$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \left(\max(0, 1 - y_i(w^T x_i + b))\right)^2 \quad (1)$$

  并通过以下决策函数对新样本x_{new}预测出类别标签y_{new}。

  $$y_{new} = \operatorname{sgn}(w^T x_{new} + b) \quad (2)$$

- 多分类

  通过one-vs-the-rest策略实现多分类任务。训练时依次把某个类别的样本归为一类，其他剩余的样本归为另一类，转化为k个二分类问题，构造出了k个二分类SVM分类器。分类时将未知样本分类为具有最大分类函数值的那一类。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

 版权所有 © 华为云计算技术有限公司

参数说明

参数	子参数	参数说明
b_use_default_encoder	-	是否使用默认编码，默认为True
input_features_str	-	输入的列名以逗号分隔组成的字符串，例如： "column_a" "column_a,column_b"
label_col	-	目标列
classifier_label_index_col	-	目标列经过标签编码后的新的列名，默认为"label_index"
classifier_feature_vector_col	-	算子输入的特征向量列的列名，默认为"model_features"
prediction_index_col	-	算子输出的预测label对应的标签列，默认为"prediction_index"
prediction_col	-	算子输出的预测label的列名，默认为"prediction"
max_iter	-	最大迭代次数，默认为100
reg_param	-	正则化系数，默认为0.0
tol	-	收敛阈值，默认为1e-6
fit_intercept	-	默认为True
standardization	-	训练模型之前是否对训练特征标准化，默认为True
aggregation_depth	-	聚合时的深度，默认为2

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "b_use_default_encoder": True,  # @param {"label": "b_use_default_encoder", "type": "boolean", "required": "true", "helpTip": ""}
    "input_features_str": "",  # @param {"label": "input_features_str", "type": "string", "required": "false", "helpTip": ""}
    "outer_pipeline_stages": None,
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": "target label column"}
    "classifier_label_index_col": "label_index",  # @param {"label": "classifier_label_index_col", "type": "string", "required": "true", "helpTip": ""}
    "classifier_feature_vector_col": "model_features",  # @param {"label": "classifier_feature_vector_col", "type": "string", "required": "true", "helpTip": ""}
    "prediction_index_col": "prediction_index",  # @param {"label": "prediction_index_col", "type": "string", "required": "true", "helpTip": ""}
    "prediction_col": "prediction",  # @param {"label": "prediction_col", "type": "string", "required": "true",
```


版权所有 © 华为云计算技术有限公司

```
"helpTip": ""}
    "max_iter": 100,  # @param {"label": "max_iter", "type": "integer", "required": "true", "range":
"(0,2147483647]", "helpTip": ""}
    "reg_param": 0.0,  # @param {"label": "reg_param", "type": "number", "required": "true", "range":
"[0,none)", "helpTip": ""}
    "tol": 1e-6,  # @param {"label": "tol", "type": "number", "required": "true", "range": "(0,none)", "helpTip":
""}
    "fit_intercept": True,  # @param {"label": "fit_intercept", "type": "boolean", "required": "true", "helpTip":
""}
    "standardization": True,  # @param {"label": "standardization", "type": "boolean", "required": "true",
"helpTip": ""}
    "aggregation_depth": 2  # @param {"label": "aggregation_depth", "type": "integer", "required": "true",
"range": "(0,2147483647]", "helpTip": ""}
}
linear_svc_classifier____id___ = MLSLinearSVCClassifier(**params)
linear_svc_classifier____id___.run()
# @output {"label":"pipeline_model","name":"linear_svc_classifier____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
```

7.5.3.1.5 逻辑回归分类

概述

“逻辑回归”节点用于数据二分类，支持自动化建模。它可以根据输入训练集高效地完成参数自动调优，并通过LOGISTIC函数将线性回归的输出映射到[0,1]区间，最后根据阈值判断完成数据二分类。

逻辑回归本质上是一种线性分类方法，因此在考虑使用逻辑回归模型前，要保证所提出的特征与目标变量之间的关系可以使用线性模型来表达。特征与目标变量之间的线性关系越强，逻辑回归的模型性能越好。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

参数说明

参数	子参数	参数说明
b_use_default_encoder	-	是否使用默认编码，默认为True
input_features_str	-	输入的列名以逗号分隔组成的字符串，例如： "column_a" "column_a,column_b"
label_col	-	目标列

 版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
classifier_label_index_col	-	目标列经过标签编码后的新的列名，默认为"label_index"
classifier_feature_vector_col	-	算子输入的特征向量列的列名，默认为"model_features"
prediction_col	-	算子输出的预测label对应的标签列，默认为"prediction_index"
prediction_index_col	-	算子输出的预测label的列名，默认为"prediction"
max_iter	-	最大迭代次数，默认为100
reg_param	-	正则化参数，默认为0.0
elastic_net_param	-	弹性网络参数，默认为0.0
tol	-	迭代算法的收敛阈值，默认为1e-6
fit_intercept	-	是否要使用截距，默认为True
standardization	-	是否正则化特征，默认为True
aggregation_depth	-	聚合的深度，默认为2
family	-	模型训练中使用哪种标签分布，支持auto、binomial、multinomial，默认为"auto"

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "b_use_default_encoder": True,  # @param {"label": "b_use_default_encoder", "type": "boolean", "required": "true", "helpTip": ""}
    "input_features_str": "",  # @param {"label": "input_features_str", "type": "string", "required": "false", "helpTip": ""}
    "outer_pipeline_stages": None,
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": "target label column"}
    "classifier_label_index_col": "label_index",  # @param {"label": "classifier_label_index_col", "type": "string", "required": "true", "helpTip": ""}
    "classifier_feature_vector_col": "model_features",  # @param {"label": "classifier_feature_vector_col", "type": "string", "required": "true", "helpTip": ""}
    "prediction_col": "prediction",  # @param {"label": "prediction_col", "type": "string", "required": "true", "helpTip": ""}
    "prediction_index_col": "prediction_index",  # @param {"label": "prediction_index_col", "type": "string", "required": "true", "helpTip": ""}
    "max_iter": 100,  # @param {"label": "max_iter", "type": "integer", "required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "reg_param": 0.0,  # @param {"label": "reg_param", "type": "number", "required": "true", "range": "[0,none)", "helpTip": ""}
    "elastic_net_param": 0.0,  # @param {"label": "elastic_net_param", "type": "number", "required": "true", "range": "[0,none)", "helpTip": ""}
    "tol": 1e-6,  # @param {"label": "tol", "type": "number", "required": "true", "range": "(0,none)", "helpTip":
```


版权所有 © 华为云计算技术有限公司

```
""}
    "fit_intercept": True,  # @param {"label": "fit_intercept", "type": "boolean", "required": "true", "helpTip":
""}
    "standardization": True,  # @param {"label": "standardization", "type": "boolean", "required": "true",
"helpTip": ""}
    "aggregation_depth": 2,  # @param {"label": "aggregation_depth", "type": "integer", "required": "true",
"range": "(0,2147483647]", "helpTip": ""}
    "family": "auto",  # @param {"label": "family", "type": "enum", "required": "true",
"options":"auto,binomial,multinomial", "helpTip": ""}
    "lower_bounds_on_coefficients": None,
    "upper_bounds_on_coefficients": None,
    "lower_bounds_on_intercepts": None,
    "upper_bounds_on_intercepts": None
}
lr_classifier____id___ = MLSLogisticRegressionClassifier(**params)
lr_classifier____id___.run()
# @output {"label":"pipeline_model","name":"lr_classifier____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
```

7.5.3.1.6 多层感知机分类

概述

“多层感知机分类”节点可用于建立一个基于前馈人工神经网络的分类模型。

前馈人工神经网络采用一种单向多层结构。其中每一层包含若干个神经元，同一层的神经元之间没有互相连接，层间信息的传送只沿一个方向进行。其中第一层称为输入层。最后一层为输出层，中间为隐层。K+1层前馈神经网络矩阵形式如下表示，其中*X*为特征集，*w*为权重值，b为偏置量，*y*为预测值。

$$y(X)=f_K\left(\ldots f_2\left(w_2^T f_1\left(w_1^T X+b_1\right)+b_2\right)\ldots+b_K\right)$$

中间层的节点使用sigmod函数：

$$f(z_i)=\frac{1}{1+e^{-z_i}}$$

输出层的节点使用softmax函数：

$$f(z_i)=\frac{e^{z_i}}{\sum_{k=1}^{N} e^{z_k}}$$

输出层中的节点个数对应类别数量。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

版权所有 © 华为云计算技术有限公司

输出

spark pipeline类型的模型

参数说明

参数	子参数	参数说明
b_use_default_encoder	-	是否使用默认编码，默认为True
input_features_str	-	输入的列名以逗号分隔组成的字符串，例如： "column_a" "column_a,column_b"
label_col	-	目标列
classifier_label_index_col	-	目标列经过标签编码后的新的列名，默认为"label_index"
classifier_feature_vector_col	-	算子输入的特征向量列的列名，默认为"model_features"
prediction_col	-	算子输出的预测label的列名，默认为"prediction"
prediction_index_col	-	算子输出的预测label对应的标签列，默认为"prediction_index"
max_iter	-	最大迭代次数，默认为100
tol	-	收敛阈值，默认为1e-6
seed	-	随机数种子，默认为0
layers_str	-	层的个数用逗号分隔组成的字符串，例如: "2,3,4" "3"
step_size	-	步长，默认为0.03
solver	-	用来优化的处理算法，支持l-bfgs、gd，默认为"l-bfgs"
initial_weights_str	-	初始化权重用逗号分隔组成的字符串，例如: "0.01" "0.01,0.02,0.04"

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
```


版权所有 © 华为云计算技术有限公司

```
    "inputs": inputs,
    "b_output_action": True,
    "b_use_default_encoder": True,  # @param {"label": "b_use_default_encoder", "type": "boolean",
"required": "true", "helpTip": ""}
    "input_features_str": "",  # @param {"label": "input_features_str", "type": "string", "required": "false",
"helpTip": ""}
    "outer_pipeline_stages": None,
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": ""}
    "classifier_label_index_col": "label_index",  # @param {"label": "classifier_label_index_col", "type":
"string", "required": "true", "helpTip": ""}
    "classifier_feature_vector_col": "model_features",  # @param {"label": "classifier_feature_vector_col",
"type": "string", "required": "true", "helpTip": ""}
    "prediction_col": "prediction",  # @param {"label": "prediction_col", "type": "string", "required": "true",
"helpTip": ""}
    "prediction_index_col": "prediction_index",  # @param {"label": "prediction_index_col", "type": "string",
"required": "true", "helpTip": ""}
    "max_iter": 100,  # @param {"label": "max_iter", "type": "integer", "required": "true", "range":
"(0,2147483647]", "helpTip": ""}
    "tol": 1e-6,  # @param {"label": "tol", "type": "number", "required": "true", "range": "(0,none)", "helpTip":
""}
    "seed": 0,  # @param {"label": "seed", "type": "integer", "required": "false", "range": "[0,2147483647]",
"helpTip": ""}
    "layers_str": "",  # @param {"label": "layers_str", "type": "string", "required": "false", "helpTip": ""}
    "block_size": 128,
    "step_size": 0.03,  # @param {"label": "step_size", "type": "number", "required": "true", "range":
"(0,none)", "helpTip": ""}
    "solver": "l-bfgs",  # @param {"label": "solver", "type": "enum", "required": "true", "options": "gd,l-bfgs",
"helpTip": ""}
    "initial_weights_str": ""  # @param {"label": "initial_weights_str", "type": "string", "required": "false",
"helpTip": ""}
}
multilayer_perception_classifier____id___ = MLSMultilayerPerceptronClassifier(**params)
multilayer_perception_classifier____id___.run()
# @output {"label":"pipeline_model","name":"multilayer_perception_classifier____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
```

7.5.3.1.7 朴素贝叶斯分类

概述

“朴素贝叶斯”节点用于产生多分类模型，用户在使用时需要指定数据的“Role”字段，默认支持“Input”、“Target”、“Rejected”、“ID”四种类型，且只能选择其一种。

朴素贝叶斯算法是基于贝叶斯定理与特征条件独立假设的分类方法。

朴素贝叶斯法实现简单，学习与预测的效率都很高，是一种常用的方法。对于给定的训练数据集：

1. 首先基于特征条件独立假设学习输入/输出的联合概率分布。
2. 然后基于此模型，对给定的输入x，利用贝叶斯定理求出后验概率最大的输出y。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

 版权所有 © 华为云计算技术有限公司

输出

spark pipeline类型的模型

参数说明

参数	子参数	参数说明
b_use_default_encoder	-	是否使用默认编码，默认为True
input_features_str	-	输入的列名以逗号分隔组成的字符串，例如： "column_a" "column_a,column_b"
label_col	-	目标列
classifier_label_index_col	-	目标列经过标签编码后的新的列名，默认为"label_index"
classifier_feature_vector_col	-	算子输入的特征向量列的列名，默认为"model_features"
prediction_col	-	算子输出的预测label的列名，默认为"prediction"
prediction_index_col	-	算子输出的预测label对应的标签列，默认为"prediction_index"
smoothing	-	平滑参数，默认为1.0
model_type	-	模型类型，支持multinomial、bernoulli，默认为"multinomial"

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "b_use_default_encoder": True,  # @param {"label": "b_use_default_encoder", "type": "boolean", "required": "true", "helpTip": ""}
    "input_features_str": "",  # @param {"label": "input_features_str", "type": "string", "required": "false", "helpTip": ""}
    "outer_pipeline_stages": None,
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": ""}
    "classifier_label_index_col": "label_index",  # @param {"label": "classifier_label_index_col", "type": "string", "required": "true", "helpTip": ""}
    "classifier_feature_vector_col": "model_features",  # @param {"label": "classifier_feature_vector_col", "type": "string", "required": "true", "helpTip": ""}
    "prediction_col": "prediction",  # @param {"label": "prediction_col", "type": "string", "required": "true", "helpTip": ""}
    "prediction_index_col": "prediction_index",  # @param {"label": "prediction_index_col", "type": "string", "required": "true", "helpTip": ""}
    "smoothing": 1.0,  # @param {"label": "smoothing", "type": "number", "required": "true", "range": "[0,none)", "helpTip": ""}
```


版权所有 © 华为云计算技术有限公司

```
    "model_type": "multinomial"  # @param {"label": "model_type", "type": "enum", "required": "true",
"options": "multinomial,bernoulli", "helpTip": ""}
}
naive_bayes_classifier____id___ = MLSNaiveBayesClassifier(**params)
naive_bayes_classifier____id___.run()
# @output {"label":"pipeline_model","name":"naive_bayes_classifier____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
```

7.5.3.1.8 随机森林分类

概述

“随机决策森林分类”节点用于产生二分类或多分类模型。随机决策森林是用随机的方式建立一个森林模型，森林由很多的决策树组成，每棵决策树之间没有关联。当有一个新的样本输入时，森林中的每一棵决策树分别进行判断，哪一类被选择最多，就预测这个样本属于那一类。

随机决策森林分类中的决策树算法通过基尼不纯度（Gini impurity）或熵（Entropy）来对一个集合的有序程度进行量化，并对一次拆分进行量化评价。

- 基尼不纯度是指将来自集合中的某种结果随机应用于集合中某一数据项的预期误差率，计算公式如下：

  $$\sum_{i=1}^{C} f_i(1-f_i)$$

- 熵是信息论中的概念，用来表示集合的无序程度，熵越大表示集合越混乱，反之则表示集合越有序，计算公式如下：

  $$\sum_{i=1}^{C} -f_i \log(f_i)$$

  *f*i表示类别*i*样本数量占所有样本的比例，*C*表示数据类别数。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

参数说明

参数	子参数	参数说明
b_use_default_encoder	-	是否使用默认编码，默认为True

 版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
input_features_str	-	输入的列名以逗号分隔组成的字符串，例如： "column_a" "column_a,column_b"
label_col	-	目标列
classifier_label_index_col	-	传给分类器的目标列，必须为数值列
classifier_feature_vector_col	-	传给分类器的特征列，必须为向量列
prediction_col	-	算子输出的预测label的列名，默认为"prediction"
prediction_index_col	-	算子输出的预测label对应的标签列，默认为"prediction_index"
max_depth	-	树的最大深度，默认为5
max_bins	-	最大分箱数，默认为32
min_instances_per_node	-	节点分割时，要求子节点必须包含的最少实例数，默认为1
min_info_gain	-	节点是否分割要求的最小信息增益，默认为0
impurity	-	计算信息增益的方法，支持entropy、gini，默认为"gini"
num_trees	-	树的个数，默认为20
feature_subset_strategy	-	节点分割时考虑用到的特征列的策略，支持auto、all、onethird、sqrt、log2、n，默认为"all"
subsampling_rate	-	学习每棵决策树用到的训练集的比例，默认为1.0
seed	-	随机数种子，默认为0

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "b_use_default_encoder": True,  # @param {"label": "b_use_default_encoder", "type": "boolean", "required": "true", "helpTip": ""}
    "input_features_str": "",  # @param {"label": "input_features_str", "type": "string", "required": "false", "helpTip": ""}
    "outer_pipeline_stages": None,
```


版权所有 © 华为云计算技术有限公司

```
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": "target
label column"}
    "classifier_label_index_col": "label_index",  # @param {"label": "classifier_label_index_col", "type":
"string", "required": "true", "helpTip": ""}
    "classifier_feature_vector_col": "model_features",  # @param {"label": "classifier_feature_vector_col",
"type": "string", "required": "true", "helpTip": ""}
    "prediction_index_col": "prediction_index",  # @param {"label": "prediction_index_col", "type": "string",
"required": "true", "helpTip": ""}
    "prediction_col": "prediction",  # @param {"label": "prediction_col", "type": "string", "required": "true",
"helpTip": ""}
    "max_depth": 5,  # @param {"label": "max_depth", "type": "integer", "required": "true", "range":
"(0,2147483647]", "helpTip": ""}
    "max_bins": 32,  # @param {"label": "max_bins", "type": "integer", "required": "true", "range":
"(0,2147483647]", "helpTip": ""}
    "min_instances_per_node": 1,  # @param {"label": "min_instances_per_node", "type": "integer",
"required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "min_info_gain": 0.0,  # @param {"label": "min_info_gain", "type": "number", "required": "true", "range":
"[0,none)", "helpTip": ""}
    "impurity": "gini",  # @param {"label": "impurity", "type": "enum", "required": "true", "options":
"entropy,gini", "helpTip": ""}
    "num_trees": 20,  # @param {"label": "num_trees", "type": "integer", "required": "true", "range":
"(0,2147483647]", "helpTip": ""}
    "feature_subset_strategy": "all",  # @param {"label": "feature_subset_strategy", "type": "enum",
"required": "true", "options":"auto,all,onethird,sqrt,log2", "helpTip": ""}
    "subsampling_rate": 1.0,  # @param {"label": "subsampling_rate", "type": "number", "required": "true",
"range": "(0,1.0]", "helpTip": ""}
    "seed": 0  # @param {"label": "seed", "type": "integer", "required": "true",
"range":"[0,2147483647]","helpTip": "seed"}
}
rf_classifier____id___ = MLSRandomForestClassifier(**params)
rf_classifier____id___.run()
# @output {"label":"pipeline_model","name":"rf_classifier____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
```

7.5.3.1.9 FM 算法

概述

FM主要是解决稀疏数据下的特征组合问题，并且其预测的复杂度是线性的，对于连续和离散特征有较好的通用性。

公式为：

$$\tilde{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} < v_i, v_j > x_i x_j$$

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

 版权所有 © 华为云计算技术有限公司

参数说明

参数	参数含义	默认值
tensor_col_name	特征列名称。 数据格式为key:value，多个特征使用英文逗号（,）分隔。例如1:1.0,3:1.0	无
label_col_name	label列名。数据必须是数值类型。如果task取值为binary_classification，则label只能取0或1。	无
task	FM算法的训练模式(分类、回归)	binary_classification
dim	使用英文逗号（,）分隔的三个整数，分别表示0次项、线性项及二次项的长度。	1,1,8
num_epochs	迭代数。	100
learn_rate	学习率。	0.01
param_lambda	使用英文逗号（,）分隔的三个浮点数，分别表示0次项、线性项及二次项的正则化系数。	0.2,0.2,0.2
init_stdev	参数初始化标准差。	0.01
mini_batch_fraction	训练过程中，最小分片大小。	1
tol	判断收敛的忍受度。	0.1
pred_result_col_name	预测结果列名。	predictResultCol
pred_score_col_name	预测得分列名(在分类模型中存在)。	predictScoreCol
keep_col_names	保存至输出结果表的列。	无

数据样例：

版权所有 © 华为云计算技术有限公司

```
label——>features
1.0>"1:5.1,2:3.5,3:1.4,4:0.2"
1.0>"1:4.9,2:3,3:1.4,4:0.2"
1.0>"1:4.7,2:3.2,3:1.3,4:0.2"
1.0>"1:4.6,2:3.1,3:1.5,4:0.2"
1.0>"1:5,2:3.6,3:1.4,4:0.2"
1.0>"1:5.4,2:3.9,3:1.7,4:0.4"
1.0>"1:4.6,2:3.4,3:1.4,4:0.3"
1.0>"1:5,2:3.4,3:1.5,4:0.2"
1.0>"1:4.4,2:2.9,3:1.4,4:0.2"
1.0>"1:4.9,2:3.1,3:1.5,4:0.1"
1.0>"1:5.4,2:3.7,3:1.5,4:0.2"
1.0>"1:4.8,2:3.4,3:1.6,4:0.2"
1.0>"1:4.8,2:3,3:1.4,4:0.1"
1.0>"1:4.3,2:3,3:1.1,4:0.1"
1.0>"1:5.8,2:4,3:1.2,4:0.2"
```

算链样例

说明

当训练特征维度高于1w的样本，建议使用CU64G以上的资源。

7.5.3.1.10 GBDT PMML 模型预测

概述

读取由scikit-learn等平台生产的GBDT的PMML模型文件，并对新的数据进行预测。当前只支持GBDT的分类模型。预测的结果包含预测的类别及其概率，以及一个包含各个类别，及其概率的详细信息字段。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象，用于最终预测的数据

版权所有 © 华为云计算技术有限公司

输出

参数	子参数	参数说明
outputs	output_port_1	指向一个pyspark的DataFrame类型对象，该对象中包含GBDT分类PMML模型预测结果

参数说明

参数	是否必选	参数说明	默认值
model_path	是	PMML模型所在的位置	""
append_cols	是	需要输出的列，如col1，col2。如果不设置则默认输出所有的输入列	""
result_col	是	预测结果类别的字段名	"prediction_result"
score_col	是	预测结果类别概率的字段名	"prediction_score"
detail_col	是	预测结果的详细信息	"prediction_detail"

样例

数据样本

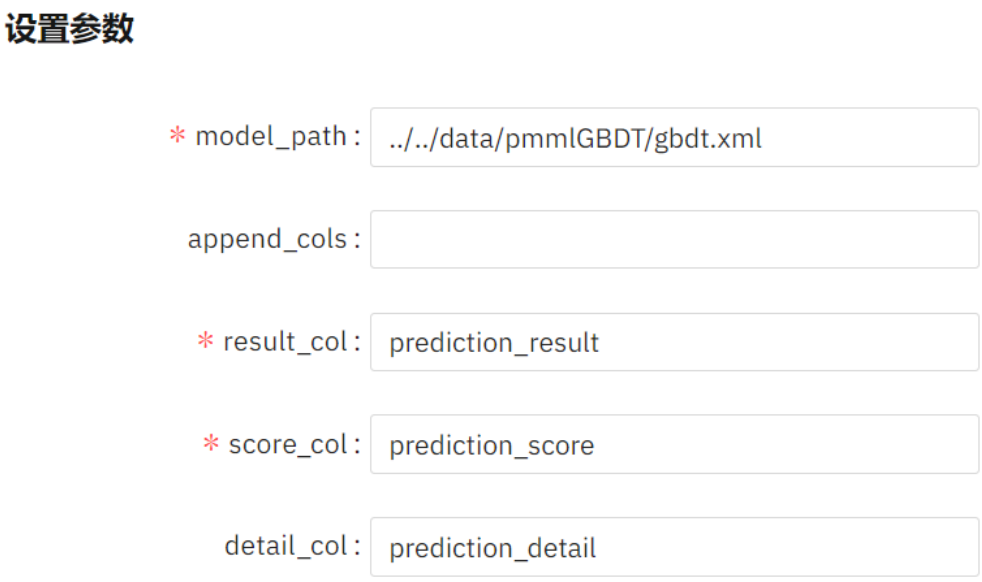
```
sepallengthcm,sepalwidthcm,petallengthcm,petalwidthcm
4.6,3.1,1.5,0.2,0
```

配置流程

运行流程

参数设置

版权所有 © 华为云计算技术有限公司



查看结果

```
sepallengthcm,sepalwidthcm,petallengthcm,petalwidthcm,prediction_result,prediction_score,prediction_detail
4.6,3.1,1.5,0.2,0,0.8451708330082154,"{""p_0"":0.8451708330082154,""p_1"":0.15482916699178462}"
```

7.5.3.1.11 多层感知机分类(pytorch)

概述

使用pytorch实现的多层感知机分类算法，可运行于异构资源池上。

该算子通过cuda自动判断gpu是否可用。如果gpu可用，优先使用gpu训练；否则使用cpu训练。

输入

参数	子参数	参数说明
data_url	_	data_url为输入数据存储的obs文件夹路径。例如obs://test/data/

输出

参数	子参数	参数说明
train_url	_	train_url 为模型训练结果保存的obs文件夹路径，用于保存输出模型文件。例如obs://test/train_out/

版权所有 © 华为云计算技术有限公司

参数说明

参数	是否必选	参数说明	默认值
feature_index_list	是	feature列的index，不同index之间以','分割。举例： 1,2,3,4	""
label_index	是	label列的index。举例：5	""
hidden_layer_list	是	隐藏层神经元的个数,不同数值之间以','分割，每个数值代表每一层神经元的个数。 int类型，范围[]	"10,5"
hidden_layer_activation	是	隐藏层激活函数，可选；范围['Sigmoid','ReLU']	Sigmoid
epochs	是	训练迭代次数,int类型，范围[]	4
batch_size	是	batch_size大小，int类型，范围[]	24
learning_rate	是	学习率，double类型，范围[0,10]	0.01

样例

数据样本

公开数据集: ionosphere数据集。

配置流程

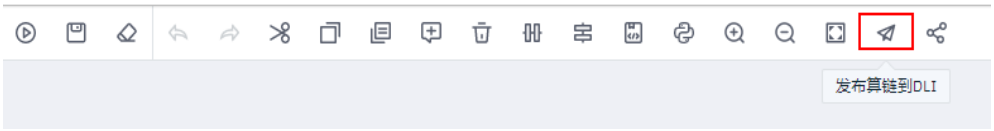
运行流程

1. 将多层感知机分类(pytorch)算子拖入画布，并进行参数配置。

版权所有 © 华为云计算技术有限公司

2. 单击界面上面发布算链到DLI按钮，将配置好的算子发布到DLI。

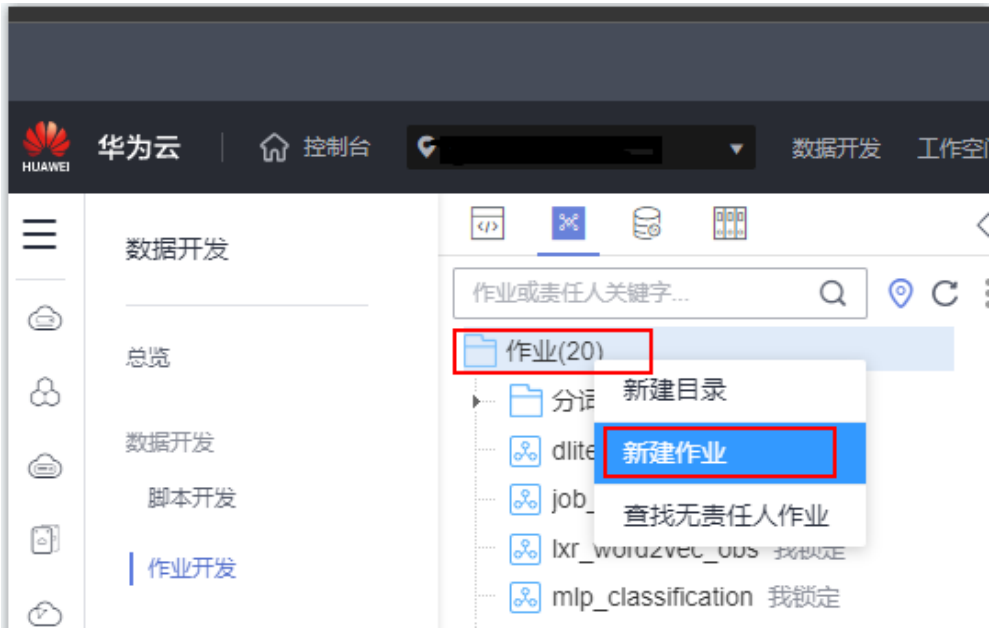
版权所有 © 华为云计算技术有限公司



3. 配置对应的DLI工作空间、名称、队列及OBS存放路径。

4. 在DLI界面的数据开发-作业开发页签下，单击作业新建作业。

版权所有 © 华为云计算技术有限公司

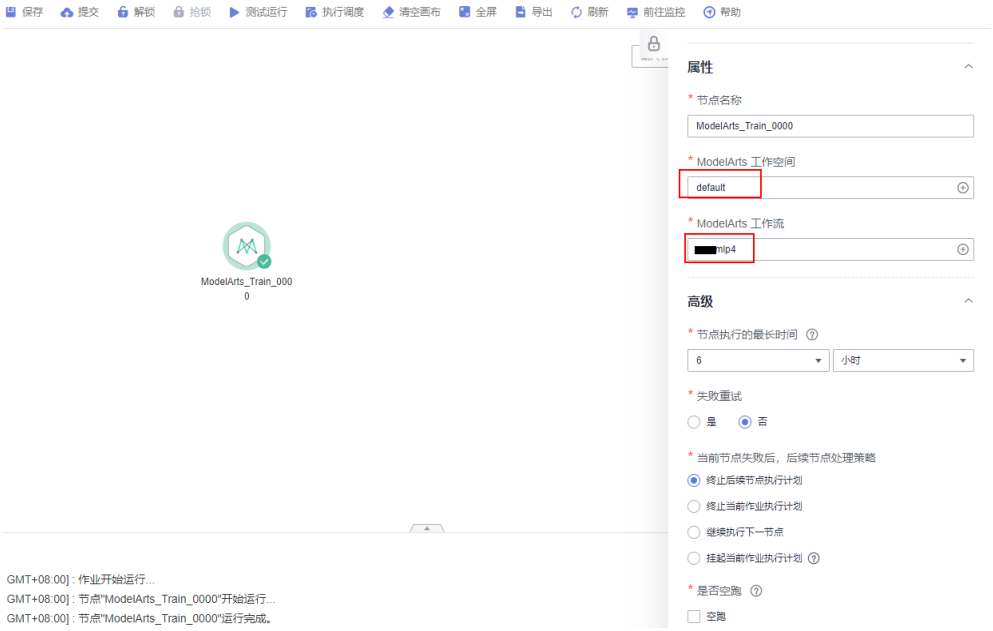


5. 配置作业名称等相关参数，单击“确定”保存。

6. 作业列表中找到4中新建的作业，并在右侧选项中选择ModelArts Train并拖入右侧画布中。

版权所有 © 华为云计算技术有限公司

7. 配置节点名称等信息，工作流选择上述MLS界面提交发布的工作流。

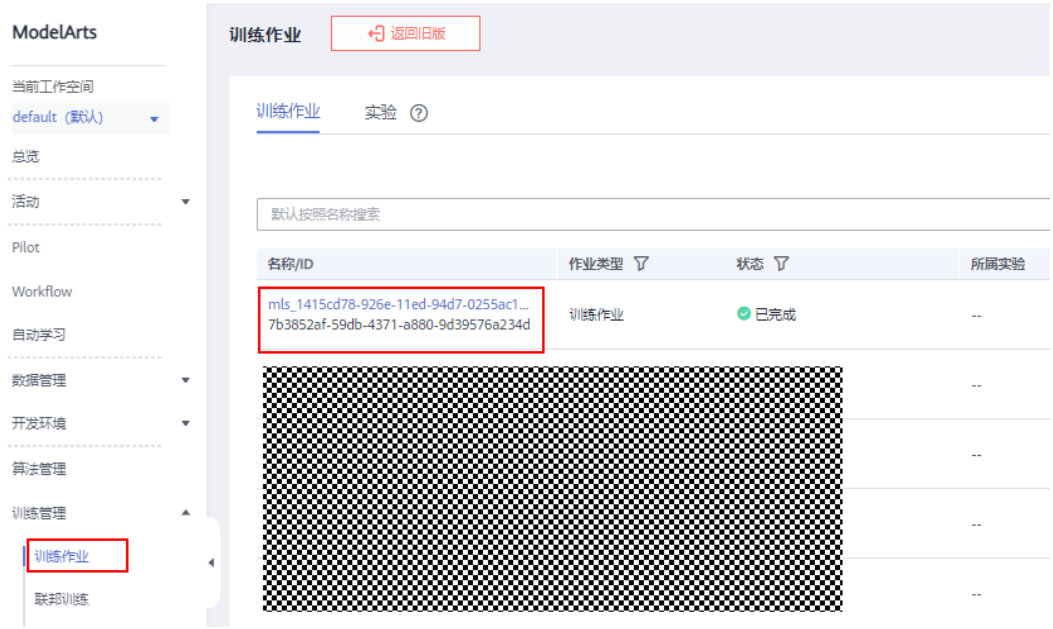


8. 依次上方保存，提交按钮，并右键单击测试运行，将DLI作业下发至ModelArts运行。

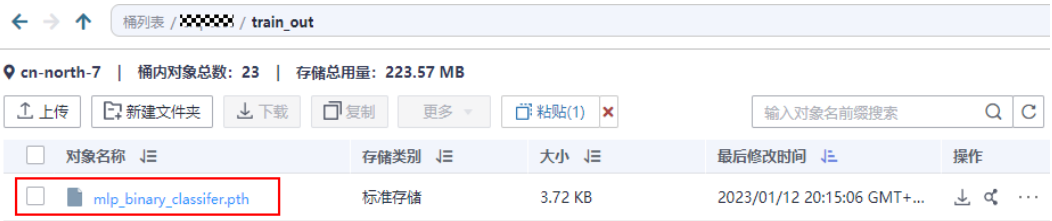
版权所有 © 华为云计算技术有限公司

查看结果

在ModelArts训练作业界面可查看刚刚下发的DLI作业，单击进入可查看相关日志及运行详情。



训练所得模型保存在配置流程**1.将多层感知机分类(pytorch)算子拖入画...**中所设置的obs目录中。



版权所有 © 华为云计算技术有限公司

7.5.3.1.12 多层感知机预测(PyTorch)

概述

使用PyTorch实现的多层感知机分类算法，可运行于异构资源池上。

该算子通过cuda自动判断GPU是否可用。如果GPU可用，优先使用GPU训练；否则使用CPU训练。

输入

参数	参数说明
train_url	train_url为存储模型文件的obs文件夹路径。例如“obs://test/data/”
predict_data_url	predict_data_url为保存预测结果的obs文件夹路径

输出

参数	参数说明
train_url	train_url为模型训练结果保存的obs文件夹路径，用于保存输出模型文件。例如“obs://test/train_out/”

参数说明

参数	是否必选	参数说明	默认值
feature_index_list	是	feature列的index,不同index之间以','分割。eg. 1,2,3,4	""

样例

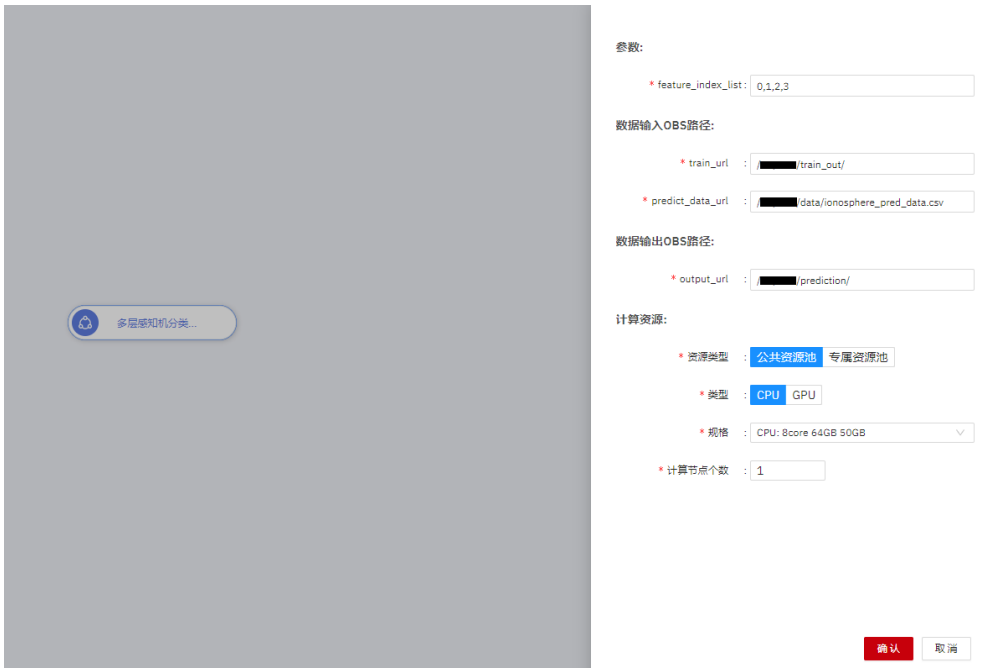
数据样本

公开数据集: ionosphere数据集.

配置流程

1. 设置多层感知机预测(pytorch)算子对应的参数信息。

版权所有 © 华为云计算技术有限公司



2. 其余步骤和**多层感知机分类算子**相同，运行完成后可在ModelArts训练界面查看运行详情。

7.5.3.2 聚类

7.5.3.2.1 二分 k 均值

概述

二分k-means算法是分层聚类（Hierarchical clustering）的一种，分层聚类是聚类分析中常用的方法。

分层聚类的策略一般有两种：

- 聚合：这是一种自底向上的方法，每一个观察者初始化本身为一类，然后两两结合。
- 分裂：这是一种自顶向下的方法，所有观察者初始化为一类，然后递归地分裂它们。二分k-means算法是分裂法的一种。

二分k-means算法是k-means算法的改进算法，相比k-means算法，它可以加速k-means算法的执行速度，因为它的相似度计算少了，能够克服k-means收敛于局部最小的缺点。

二分k-means算法的一般流程如下所示：

1. 把所有数据初始化为一个簇，将这个簇分为两个簇。
2. 选择满足条件的可以分解的簇。选择条件综合考虑簇的元素个数以及聚类代价（也就是误差平方和SSE），误差平方和的公式如下所示，其中 表示权重值， 表示该簇所有点的平均值。

$$\mathrm{SSE} = \sum_{1}^{n} w_i (y_i - \mathrm{y}^*)^2$$

版权所有 © 华为云计算技术有限公司

3. 使用k-means算法将可分裂的簇分为两簇。
4. 一直重复2、3步，直到满足迭代结束条件。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

参数说明

参数	子参数	参数说明
input_features_str	-	输入的列名以逗号分隔组成的字符串，例如： "column_a" "column_a,column_b"
cluster_feature_vector_col	-	算子输入的特征向量列的列名，默认为"model_features"
prediction_col	-	算子输出的预测label的列名，默认为"prediction"
k	-	想要聚类的个数，默认为2
max_iter	-	最大迭代次数，默认为100
min_divisible_cluster_size	-	值如果大于等于1，它表示一个可切分簇的最小点数量；如果值小于1，它表示可切分簇的点数量占总数的最小比例，该值默认为1

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "b_use_default_encoder": True,
    "outer_pipeline_stages": None,
    "input_features_str": "",  # @param {"label": "input_features_str", "type": "string", "required": "false", "helpTip": ""}
    "cluster_feature_vector_col": "model_features",  # @param {"label": "cluster_feature_vector_col", "type": "string", "required": "true", "helpTip": ""}
    "prediction_col": "prediction",  # @param {"label": "prediction_col", "type": "string", "required": "true", "helpTip": ""}
```


版权所有 © 华为云计算技术有限公司

```
    "k": 2,  # @param {"label": "k", "type": "integer", "required": "true", "range": "(0,2147483647]",
"helpTip": ""}
    "max_iter": 100,  # @param {"label": "max_iter", "type": "integer", "required": "true", "range":
"(0,2147483647]", "helpTip": ""}
    "min_divisible_cluster_size": 1.0  # @param {"label": "min_divisible_cluster_size", "type": "number",
"required": "true", "range": "(0,none)", "helpTip": ""}
}
bisecting_kmeans____id___ = MLSBisectingKmeans(**params)
bisecting_kmeans____id___.run()
# @output {"label":"pipeline_model","name":"bisecting_kmeans____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
```

7.5.3.2.2 高斯混合模型

概述

高斯混合模型（Gaussian Mixture Model）通常简称GMM，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

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

参数说明

参数	子参数	参数说明
input_features_str	-	输入的列名以逗号分隔组成的字符串，例如： "column_a" "column_a,column_b"
cluster_feature_vector_col	-	算子输入的特征向量列的列名，默认为"model_features"
prediction_col	-	算子输出的预测label的列名，默认为"prediction"
probability_col	-	算子输出的预测概率列的列名，默认为"probability"
k	-	要聚类的个数，默认为2
max_iter	-	最大迭代次数，默认为100
tol	-	收敛阈值，默认为0.01

 版权所有 © 华为云计算技术有限公司

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "b_use_default_encoder": True,
    "outer_pipeline_stages": None,
    "input_features_str": "",  # @param {"label": "input_features_str", "type": "string", "required": "false",
"helpTip": ""}
    "cluster_feature_vector_col": "model_features",  # @param {"label": "cluster_feature_vector_col", "type":
"string", "required": "true", "helpTip": ""}
    "prediction_col": "prediction",  # @param {"label": "prediction_col", "type": "string", "required": "true",
"helpTip": ""}
    "probability_col": "probability",  # @param {"label": "probability_col", "type": "string", "required": "true",
"helpTip": ""}
    "k": 2,  # @param {"label": "k", "type": "integer", "required": "true", "range": "（0,2147483647]",
"helpTip": ""}
    "max_iter": 100,  # @param {"label": "max_iter", "type": "integer", "required": "true", "range":
"(0,2147483647]", "helpTip": ""}
    "tol": 0.01  # @param {"label": "tol", "type": "number", "required": "true", "range": "(0,none)", "helpTip":
""}
}
gaussian_mixture____id___ = MLSGaussianMixture(**params)
gaussian_mixture____id___.run()
# @output {"label":"pipeline_model","name":"gaussian_mixture____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
```

7.5.3.2.3 k 均值

概述

“K-均值”节点用于产生聚类模型，用户在使用时需要指定聚类个数。K-均值算法是基于距离的算法，将所有数据归类到其最邻近的中心。

输入

参数	子参数说明	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

参数说明

参数	子参数说明	参数说明
b_use_default_encoder	-	是否使用默认编码，默认为True

 版权所有 © 华为云计算技术有限公司

参数	子参数说明	参数说明
input_features_str	-	输入的列名以逗号分隔组成的字符串，例如： "column_a" "column_a,column_b"
cluster_feature_vector_col	-	算子输入的特征向量列的列名，默认为"model_features"
prediction_col	-	pyspark kmeans聚类器输出的预测列
k	-	聚类的个数，默认为2
init_mode	-	聚类采用的初始算法，random、k-means，默认为"random"
init_steps	-	采用k-means|| 初始化模式的步数，默认为2
max_iter	-	最大迭代次数，默认为20
tol	-	迭代算法的收敛阈值，默认为1e-4

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "b_use_default_encoder": True,  # @param {"label": "b_use_default_encoder", "type": "boolean", "required": "true", "helpTip": ""}
    "outer_pipeline_stages": None,
    "input_features_str": "",  # @param {"label": "input_features_str", "type": "string", "required": "false", "helpTip": ""}
    "cluster_feature_vector_col": "model_features",  # @param {"label": "cluster_feature_vector_col", "type": "string", "required": "true", "helpTip": ""}
    "prediction_col": "prediction",  # @param {"label": "prediction_col", "type": "string", "required": "true", "helpTip": ""}
    "k": 2,  # @param {"label": "k", "type": "integer", "required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "init_mode": "random",  # @param {"label": "init_mode", "type": "string", "required": "true", "options": "random,k-means", "helpTip": ""}
    "init_steps": 2,  # @param {"label": "init_steps", "type": "integer", "required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "max_iter": 20,  # @param {"label": "max_iter", "type": "integer", "required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "tol": 1e-4  # @param {"label": "tol", "type": "number", "required": "true", "range": "(0.0,none)", "helpTip": ""}
}
kmeans____id___ = MLSKmeans(**params)
kmeans____id___.run()
# @output {"label":"pipeline_model","name":"kmeans____id___.get_outputs()['output_port_1']","type":"PipelineModel"}
```

7.5.3.3 评估

版权所有 © 华为云计算技术有限公司

7.5.3.3.1 二分类评估

概述

对二分类模型预测的结果数据集进行评估。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

二分类的评估指标：pr面积、roc面积、准确率、精确率、召回率、F1、混淆矩阵等

参数说明

参数	子参数	参数说明
label_col	-	目标列
probability_col	-	输入预测数据集的概率列的列名
prediction_index_col	-	输入预测数据集的预测label 标签列的列名
label_index_col	-	输入预测数据集的真实label标签列的列名

样例

```
inputs = {
    "predict_dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": ""}
    "probability_col": "probability",  # @param {"label": "probability_col", "type": "string", "required": "true",
"helpTip": ""}
    "prediction_index_col": "prediction_index",  # @param {"label": "prediction_index_col", "type": "string",
"required": "true", "helpTip": ""}
    "label_index_col": "label_index"  # @param {"label": "label_index_col", "type": "string", "required": "true",
"helpTip": ""}
}
binary_class_evaluation____id___ = MLSBinaryClassEvaluation(**params)
binary_class_evaluation____id___.run()
# @output {"label":"dataframe","name":"binary_class_evaluation____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```


版权所有 © 华为云计算技术有限公司

7.5.3.3.2 聚类评估

概述

对聚类模型预测的结果数据集进行评估。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

聚类的评估指标：轮廓系数silhouette等

参数说明

参数	子参数	参数说明
features_col	-	预测结果数据集中，特征列向量的列名，默认为"model_features"
prediction_col	-	预测结果数据集中，预测列的列名，默认为"prediction"

样例

```
inputs = {
    "predict_dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "features_col": "model_features",  # @param {"label": "features_col", "type": "string", "required": "true",
"helpTip": ""}
    "prediction_col": "prediction"  # @param {"label": "prediction_col", "type": "string", "required": "true",
"helpTip": ""}
}
cluster_evaluation____id___ = MLSClusterEvaluation(**params)
cluster_evaluation____id___.run()
# @output {"label":"dataframe","name":"cluster_evaluation____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.3.3.3 模型应用

概述

根据输入的spark pipeline类型的模型，对数据集进行预测。

版权所有 © 华为云计算技术有限公司

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象
	pipeline_model	inputs为字典类型，pipeline_model为spark pipeline类型的模型对象

输出

结果数据集

参数说明

无参数

样例

```
inputs = {
    "dataframe": None,  # @input {"label":"dataframe","type":"DataFrame"}
    "pipeline_model": None  # @input {"label":"pipeline_model","type":"PipelineModel"}
}
params = {
    "inputs": inputs
}
model_predict____id___ = MLSModelPredict(**params)
model_predict____id___.run()
# @output {"label":"dataframe","name":"model_predict____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.3.3.4 多分类评估

概述

对多分类模型预测的结果数据集进行评估。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

多分类的评估指标：准确率、混淆矩阵

版权所有 © 华为云计算技术有限公司

参数说明

参数	子参数	参数说明
label_col	-	目标列
prediction_index_col	-	输入预测数据集的预测label标签列的列名
label_index_col	-	输入预测数据集的真实label标签列的列名

样例

```
inputs = {
    "predict_dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": ""}
    "prediction_index_col": "prediction_index",  # @param {"label": "prediction_index_col", "type": "string",
"required": "true", "helpTip": ""}
    "label_index_col": "label_index"  # @param {"label": "label_index_col", "type": "string", "required": "true",
"helpTip": ""}
}
multi_class_evaluation____id___ = MLSMultiClassEvaluation(**params)
multi_class_evaluation____id___.run()
# @output {"label":"dataframe","name":"multi_class_evaluation____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.3.3.5 回归评估

概述

对回归模型预测的结果数据集进行评估。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

回归的评估指标：mae、mse、rmse

参数说明

参数	子参数	参数说明
label_col	-	预测结果数据集的目标列的列名

版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
prediction_col	-	预测结果数据集的预测列的列名

样例

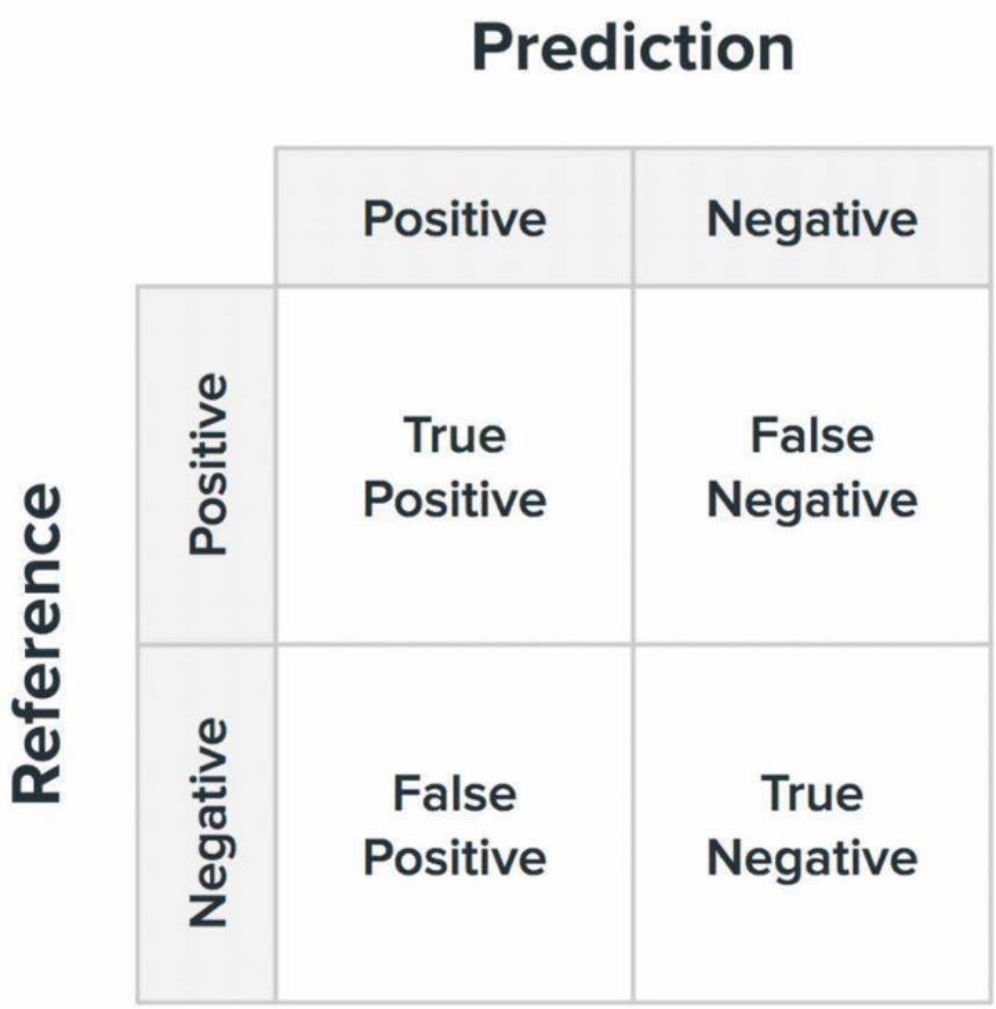
```
inputs = {
    "predict_dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": ""}
    "prediction_col": "prediction"  # @param {"label": "prediction_col", "type": "string", "required": "true",
"helpTip": ""}
}
regression_evaluation____id___ = MLSRegressionEvaluation(**params)
regression_evaluation____id___.run()
# @output {"label":"dataframe","name":"regression_evaluation____id___.get_outputs()
['output_port_1']","type":"DataFrame"}
```

7.5.3.3.6 混淆矩阵

概述

混淆矩阵是机器学习中总结分类模型预测结果的情形分析表，以矩阵形式将数据集中的记录按照真实的类别与分类模型预测的类别判断两个标准进行汇总。其中矩阵的行表示真实值，矩阵的列表示预测值。

版权所有 © 华为云计算技术有限公司



True Positive（TP）：真正类。样本的真实类别是正类，并且模型识别的结果也是正类；

False Negative（FN）：假负类。样本的真实类别是正类，但是模型将其识别为负类；

False Positive（FP）：假正类。样本的真实类别是负类，但是模型将其识别为正类；

True Negative（TN）：真负类。样本的真实类别是负类，并且模型将其识别为负类。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象，必须为分类模型的预测结果。

版权所有 © 华为云计算技术有限公司

输出

参数	子参数	参数说明
output	output_port_1	output为字典类型，output_port_1为混淆矩阵dataframe。
output	output_port_1	output为字典类型，output_port_2为具体分类指标dataframe。

参数说明

参数	是否必选	参数说明	默认值
label_col	是	数据中的标签列。	无
prediction_index_col	是	代表标签编码后的预测结果的列名，需要与mls中各种分类算子预测结果列保持一致。	"prediction_index"
label_index_col	是	经过标签编码后的标签列。	"label_index"
probability_col	否	预测结果的分类概率列。	"probability"
threshold	否	分类阈值，二分类场景下可设置，支持向量机SVM算法不支持。	0.5
positive_category	否	二分类任务的正样本。	无

样例

数据样本

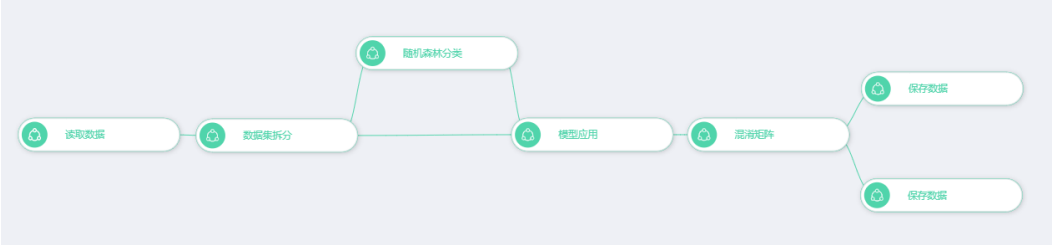
鸢尾花数据集，species列代表鸢尾花种类，共有Iris-setosa、Iris-versicolor和Iris-virginica三种类别，每种类别样本数量为50。

版权所有 © 华为云计算技术有限公司

	sepal_length	sepal_width	petal_length	petal_width	species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3	1.4	0.1	Iris-setosa
14	4.3	3	1.1	0.1	Iris-setosa
15	5.8	4	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
18	5.1	3.5	1.4	0.3	Iris-setosa
19	5.7	3.8	1.7	0.3	Iris-setosa
20	5.1	3.8	1.5	0.3	Iris-setosa

配置流程

运行流程



参数设置

参数:

* label_col : species

* prediction_index_col : prediction_index

* label_index_col : label_index

probability_col : probability

threshold : 0.5

positive_category :

结果查看

版权所有 © 华为云计算技术有限公司

confusion_matrix	Iris-versicolor	Iris-setosa	Iris-virginica
Iris-versicolor	12	0	0
Iris-setosa	0	13	0
Iris-virginica	2	0	13

Model	TruePositive	FalsePositive	Accuracy	Precision	Recall	F1 Score
Iris-versicolor	12	2	0.95	0.8571428571428571	1.0	0.923076923076923
Iris-setosa	13	0	1.0	1.0	1.0	1.0
Iris-virginica	13	0	0.95	1.0	0.8666666666666667	0.9285714285714286

7.5.3.4 推荐

7.5.3.4.1 最小二乘法

概述

ALS（交替最小二乘）是一种求解矩阵分解问题的最优化方法。

“交替最小二乘”节点用于推荐，它通过矩阵分解手段快速实现用户对物品评分的预测。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

参数说明

参数	子参数	参数说明
user_col	-	用户id所在的列名
item_col	-	项目id所在的列名
rating_col	-	评分所在的列名
recommend_nums	-	推荐物品的个数，默认为10
prediction_col	-	预测列列名，默认为"prediction"
cold_start_strategy	-	冷启动策略，默认为"nan"
alpha	-	矩阵分解的正则化系数，默认为1.0

版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
implicit_prefs	-	是否使用隐含偏好，默认为Flase
max_iter	-	最大迭代次数，默认为50
non_negative	-	是否使用非负限制，默认为False
rank	-	因子分解的秩，默认为10
reg_param	-	正则化系数，默认为0.0

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "user_col": "",  # @param {"label":"user_col","type":"string","required":"true","helpTip":""}
    "item_col": "",  # @param {"label":"item_col","type":"string","required":"true","helpTip":""}
    "rating_col": "",  # @param {"label":"rating_col","type":"string","required":"true","helpTip":""}
    "recommend_nums": 10,  # @param
{"label":"recommend_nums","type":"integer","required":"false","range":"(0,2147483647)","helpTip":""}
    "prediction_col": "prediction",  # @param
{"label":"prediction_col","type":"string","required":"false","helpTip":""}
    "cold_start_strategy": "nan",  # @param
{"label":"cold_start_strategy","type":"string","required":"false","helpTip":""}
    "alpha": 1,  # @param
{"label":"alpha","type":"number","required":"false","range":"(none,none)","helpTip":""}
    "implicit_prefs": False,  # @param {"label":"implicit_prefs","type":"boolean","required":"false","helpTip":""}
    "max_iter": 10,  # @param
{"label":"max_iter","type":"integer","required":"false","range":"(0,2147483647)","helpTip":""}
    "non_negative": False,  # @param
{"label":"non_negative","type":"boolean","required":"false","helpTip":""}
    "rank": 10,  # @param
{"label":"rank","type":"integer","required":"false","range":"(0,2147483647)","helpTip":""}
    "reg_param": 0.1  # @param
{"label":"reg_param","type":"number","required":"false","range":"(none,none)","helpTip":""}
}
als____id___ = MLSALS(**params)
als____id___.run()
# @output {"label":"pipeline_model","name":"als____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
# @output {"label":"dataframe","name":"als____id___.get_outputs()['output_port_2']","type":"DataFrame"}
```

7.5.3.4.2 向量召回评估

概述

向量召回评估算子计算召回的hitrate，用于评估召回结果的好坏，hitrate越高表示训练产出的向量去召回向量的结果越准确。支持u2i召回和i2i召回的计算。u2i召回时，拿user（用户）的向量去召回top k个items（物品），i2i召回时拿item的向量去召回top k个items。 hitrate的具体计算方法为，假设真实trigger（u2i召回时为user，i2i召回时为item）的关联item集合为M，而实际召回了top k个和trigger相似的items，如果其中落在了M里的集合为N，则top k hitrate为|N| / |M|。

版权所有 © 华为云计算技术有限公司

输入

参数	子参数	参数说明
inputs	item_embedding	inputs为字典类型，item_embedding为pyspark中的DataFrame类型对象，代表物品向量。
inputs	true_sequence	true_sequence为pyspark中的DataFrame类型对象，代表真实关联表。
inputs	user_embedding	user_embedding为pyspark中的DataFrame类型对象，代表用户向量。

输出

参数	子参数	参数说明
output	output_port_1	output为字典类型，output_port_1为总体的hitrate，类型为pyspark的DataFrame。
output	output_port_2	output为字典类型，output_port_2为详细的召回评估结果，类型为pyspark的DataFrame。

参数说明

参数	是否必选	参数说明	默认值
recall_type	是	召回类型，u2i或i2i。	u2i
emb_dim	是	向量表的维度。	5
k	是	召回数量。	3

样例

数据样本

item_embeddings

版权所有 © 华为云计算技术有限公司

item_id	embeddings
1	0.2,1.1,0.4,0.7,0.1
2	0.2,1.0,0.3,0.8,0.2
3	0.5,0.1,0.8,0.6,1.1
4	0.3,0.8,1.7,0.1,0.6
5	0.4,0.1,0.6,1.4,1.1
6	2.1,0.1,0.8,0.5,0.2
7	0.1,0.3,0.4,0.8,0.7
8	1.2,1.4,0.6,0.1,1.3

真实关联序列

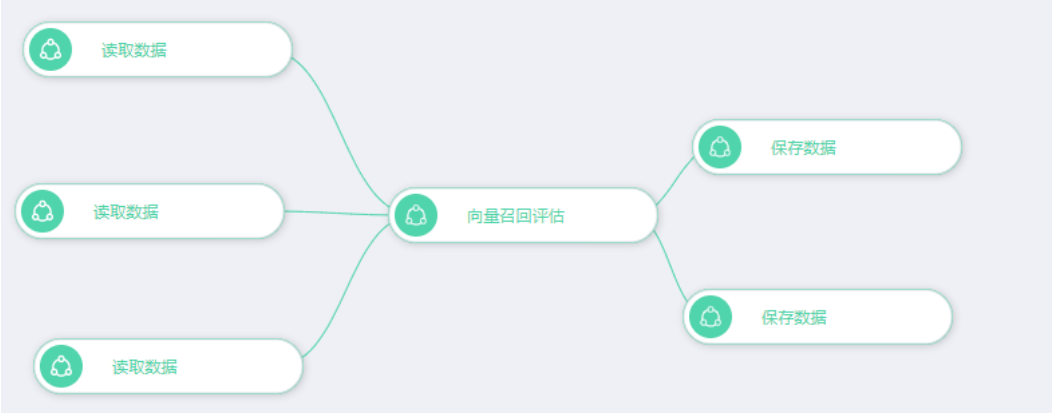
trigger_id	item_ids
10	2,4,6,5,7
20	1,3,6,7,9
30	2,8,5
40	1,2,3,5,6,7,8
50	4,7,8
60	1,2,3,4,5
70	2,3,8
80	1,5,9

user_embeddings

user_id	embeddings
10	0.2,1.3,0.4,0.7,0.1
20	0.5,1.9,0.6,0.8,0.2
30	1.5,0.1,0.2,0.6,1.1
40	2.3,0.8,1.7,0.1,0.6
50	0.4,0.7,0.9,1.4,1.1
60	1.1,0.1,0.4,0.5,0.2
70	0.1,0.3,0.8,0.8,0.7
80	1.5,0.4,0.6,0.1,1.3

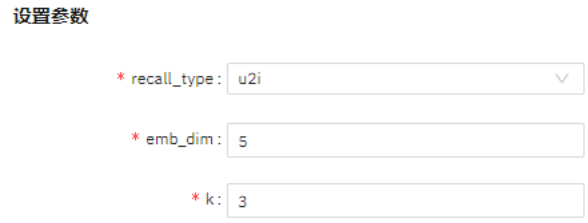
配置流程

运行流程



参数设置

版权所有 © 华为云计算技术有限公司



结果查看

hitrate
0.23529411764705882

user_id	topk_ids	topk_dists	hitrate	bad_ids	bad_dists
10	1,2,8	79,0.6920615750922011	0.2	1,8	96,0.6920615750922011
20	1,2,8	26,0.7655980604873702	0.2	2,8	26,0.7655980604873702
30	6,3,8	785,0.772283131989894	0.3333333333333333	6,3	85,0.8053698884292785
40	6,8,4	146,0.757998737004424	0.2857142857142857	4	0.757998737004424
50	7,5,3	38,0.8989480847521626	0.3333333333333333	5,3	38,0.8989480847521626
60	6,3,8	66,0.6656432011405089	0.2	6,8	21,0.6656432011405089
70	7,5,3	77,0.9166360770812138	0.3333333333333333	7,5	464,0.927615850637277
80	8,3,6	06,0.8138650886537815	0.0	8,3,6	06,0.8138650886537815

7.5.3.4.3 协同过滤-Item-based

概述

“协同过滤-Item-based”节点用于推荐场景，它通过用户和物品之间的关系计算物品之间的相似度。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象。里面存放的为用户和item之间的记录

输出

参数	子参数	参数说明
outputs	output_port_1	协同过滤的推荐结果

版权所有 © 华为云计算技术有限公司

参数说明

参数	是否必选	参数说明	默认值
user_col	是	用户id所在的列名	"user"
item_col	是	项目id所在的列名	"item"
output_table_partition	是	数据的并行度	0.5
similarity_type	是	相似度计算公式，取值为cosine，jaccard	"cosine"
topn	是	最近的n个物品	200
min_user_behavior	是	最小的用户行为数量，取值为[0,+∞）	2
max_user_behavior	是	最多的用户行为数量，取值为[0,+∞）	500
item_delimiter	是	物品之间的分隔符	" "
kv_delimiter	是	物品内部的分隔符	":"

样例

数据样本

输入数据：

```
user,item,click
1,1,1
1,2,1
1,5,1
1,7,1
2,2,1
2,6,1
2,9,1
3,4,1
3,5,1
3,7,1
3,9,1
4,2,1
5,3,1
6,6,1
6,8,1
6,9,1
7,4,1
7,5,1
8,1,1
8,3,1
8,4,1
9,5,1
9,8,1
```

配置流程

版权所有 © 华为云计算技术有限公司

运行流程

算法参数设置

查看结果

```
item,similarity
7,5:0.7071067811865475 1:0.49999999999999994 2:0.49999999999999994 9:0.408248290463863
4:0.408248290463863
3,1:0.7071067811865475 4:0.5773502691896258
8,6:0.49999999999999994 9:0.408248290463863 5:0.35355339059327373
5,7:0.7071067811865475 4:0.5773502691896258 8:0.35355339059327373 1:0.35355339059327373
2:0.35355339059327373 9:0.2886751345948129
6,9:0.816496580927726 8:0.49999999999999994 2:0.49999999999999994
9,6:0.816496580927726 7:0.408248290463863 8:0.408248290463863 2:0.408248290463863
4:0.33333333333333337 5:0.2886751345948129
1,3:0.7071067811865475 7:0.49999999999999994 2:0.49999999999999994 4:0.408248290463863
5:0.35355339059327373
4,3:0.5773502691896258 5:0.5773502691896258 7:0.408248290463863 1:0.408248290463863
9:0.33333333333333337
2,7:0.49999999999999994 6:0.49999999999999994 1:0.49999999999999994 9:0.408248290463863
5:0.35355339059327373
```


版权所有 © 华为云计算技术有限公司

7.5.3.4.4 swing

概述

swing是一个i2i的召回算法，基于User-Item-User图结构的推荐算法。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象，用于构建swing模型

输出

参数	子参数	参数说明
outputs	output_port_1	指向一个pyspark的DataFrame类型对象，swing的pipeline模型

参数说明

参数	是否必选	参数说明	默认值
user_col	是	User列的名称	user
item_col	是	Item列的名称	item
score_col	是	用户评分列名称	""
min_user_items	是	User互动的Item的最小数量	10
max_user_items	是	User互动的Item的最大数量	1000
max_item_number	是	Item参与计算的人数最大值	1000
output_score_col_name	是	预测用户评分的列名	rec_score

版权所有 © 华为云计算技术有限公司

参数	是否必选	参数说明	默认值
user_alpha	是	User的alpha参数。用于计算用户权重：user weight = 1.0/(userAlpha + userClickCount)^userBeta	0.0
user_beta	是	User的Beta参数。用于计算用户权重：user weight = 1.0/(userAlpha + userClickCount)^userBeta	0.5

样例

数据样本

```
user,item
A,t
A,r
A,p
B,t
B,r
B,p
C,p
C,q
A,h
B,h
C,h
```

配置流程

运行流程

参数设置

版权所有 © 华为云计算技术有限公司

查看结果

```
user,item,rec_score
C,t,0.1
C,r,0.1
```

7.5.3.5 回归

7.5.3.5.1 决策树回归

概述

“决策树回归”节点用于产生回归模型。

决策树算法是递归地构建决策树的过程，用平方误差最小准则，进行特征选择，生成二叉树。平方误差计算公式如下：

$$\frac{1}{N}\sum_{i=1}^{N}(y_i-\mu)^2$$

版权所有 © 华为云计算技术有限公司

$$\mu = \sum_{i=1}^{N} y_i$$

其中 是样本类标的均值，y_i 是样本的标签，N 是样本数量。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

参数说明

参数	子参数	参数说明
b_use_default_encoder	-	是否使用默认编码，默认为True
input_features_str	-	输入的列名以逗号分隔组成的字符串，例如： "column_a" "column_a,column_b"
label_col	-	目标列
regressor_feature_vector_col	-	算子输入的特征向量列的列名，默认为"model_features"
max_depth	-	树的最大深度，默认为5
max_bins	-	最大分箱数，默认为32
min_instances_per_node	-	节点分割时，要求子节点必须包含的最少实例数，默认为1
min_info_gain	-	最小信息增益，默认为0.0

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "b_use_default_encoder": True,  # @param {"label": "b_use_default_encoder", "type": "boolean", "required": "true", "helpTip": ""}
    "input_features_str": "",  # @param {"label": "input_features_str", "type": "string", "required": "false", "helpTip": ""}
```


版权所有 © 华为云计算技术有限公司

```
    "outer_pipeline_stages": None,
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": ""}
    "regressor_feature_vector_col": "model_features",  # @param {"label": "regressor_feature_vector_col",
"type": "string", "required": "true", "helpTip": ""}
    "max_depth": 5,  # @param {"label": "max_depth", "type": "integer", "required": "true", "range":
"(0,2147483647]", "helpTip": ""}
    "max_bins": 32,  # @param {"label": "max_bins", "type": "integer", "required": "true", "range":
"(0,2147483647]", "helpTip": ""}
    "min_instances_per_node": 1,  # @param {"label": "min_instances_per_node", "type": "integer",
"required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "min_info_gain": 0.0,  # @param {"label": "min_info_gain", "type": "number", "required": "true", "range":
"[0.0,none)", "helpTip": ""}
    "impurity": "variance"
}
dt_regressor____id___ = MLSDecisionTreeRegression(**params)
dt_regressor____id___.run()
# @output {"label":"pipeline_model","name":"dt_regressor____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
```

7.5.3.5.2 梯度提升树回归

概述

“梯度提升树回归”节点用于生成回归模型，是一种基于决策树的迭代回归算法。该算法采用迭代的思想不断地构建决策树模型，每棵树都是通过梯度优化损失函数而构建，从而达到从基准值到目标值的逼近。算法思想可简单理解成：后一次模型都是针对前一次模型预测出错的情况进行修正，模型随着迭代不断地改进，从而获得比较好的预测效果。

梯度提升树回归的损失函数为均方差损失函数，如下所示：

$$\sum_{i=1}^{N}(y_i - F(x_i))^2$$

其中，N 表示样本数量，x_i 表示样本 i 的特征，y_i 表示样本 i 的标签，$F(x_i)$ 表示样本 i 预测的标签。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

参数说明

参数	子参数	参数说明
b_use_default_encoder	-	是否使用默认编码，默认为True

 版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
input_features_str	-	输入的列名以逗号分隔组成的字符串，例如： "column_a" "column_a,column_b"
label_col	-	目标列
regressor_feature_vector_col	-	算子输入的特征向量列的列名，默认为"model_features"
max_depth	-	树的最大深度，默认为5
max_bins	-	最大分箱数，默认为32
min_instances_per_node	-	节点分割时，要求子节点必须包含的最少实例数，默认为1
min_info_gain	-	节点是否分割要求的最小信息增益，默认为0.0
subsampling_rate	-	学习每棵决策树用到的训练集的抽样比例，默认为1.0
loss_type	-	损失函数类型，支持squared、absolute，默认为"squared"
max_iter	-	最大迭代次数，默认为20
step_size	-	步长，默认为0.1

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "b_use_default_encoder": True,  # @param {"label": "b_use_default_encoder", "type": "boolean", "required": "true", "helpTip": ""}
    "input_features_str": "",  # @param {"label": "input_features_str", "type": "string", "required": "false", "helpTip": ""}
    "outer_pipeline_stages": None,
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": "target label column"}
    "regressor_feature_vector_col": "model_features",  # @param {"label": "regressor_feature_vector_col", "type": "string", "required": "true", "helpTip": ""}
    "max_depth": 5,  # @param {"label": "max_depth", "type": "integer", "required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "max_bins": 32,  # @param {"label": "max_bins", "type": "integer", "required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "min_instances_per_node": 1,  # @param {"label": "min_instances_per_node", "type": "integer", "required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "min_info_gain": 0.0,  # @param {"label": "min_info_gain", "type": "number", "required": "true", "range": "[0.0,none)", "helpTip": ""}
    "subsampling_rate": 1.0,  # @param {"label": "subsampling_rate", "type": "number", "required": "true", "range": "(0.0,1.0]", "helpTip": ""}
    "loss_type": "squared",  # @param {"label": "loss_type", "type": "enum", "required": "true", "options": "squared,absolute", "helpTip": ""}
```


版权所有 © 华为云计算技术有限公司

```
    "max_iter": 20,  # @param {"label": "max_iter", "type": "integer", "required": "true", "range":
"(0,2147483647]", "helpTip": ""}
    "step_size": 0.1,  # @param {"label": "step_size", "type": "number", "required": "true", "range":
"(0.0,none)", "helpTip": ""}
    "impurity": "variance"
}
gbt_regressor____id___ = MLSGBTRegression(**params)
gbt_regressor____id___.run()
# @output {"label":"pipeline_model","name":"gbt_regressor____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
```

7.5.3.5.3 LightGBM 回归

概述

对mmlspark python包中LightGBM回归的封装

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

参数说明

参数	子参数	参数说明
input_features_str	-	输入的列名以逗号分隔组成的字符串，例如： "column_a" "column_a,column_b"
label_col	-	目标列
regressor_feature_vector_col	-	算子输入的特征向量列的列名，默认为"model_features"
prediction_col	-	算子输出的预测label的列名，默认为"prediction"
objective	-	目标函数，默认为"regression"
max_depth	-	树的最大深度，默认为-1
num_iteration	-	迭代次数，默认为100
learning_rate	-	学习率，默认为0.1
num_leaves	-	叶子数目，默认为31
max_bin	-	最大分箱数，默认为255

 版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
bagging_fraction	-	bagging的比例，默认为1
bagging_freq	-	bagging的频率，默认为0
bagging_seed	-	bagging时的随机数种子，默认为3
early_stopping_round	-	提前结束迭代的轮数，默认为0
feature_fraction	-	特征的比例，默认为1.0
min_sum_hessian_in_leaf	-	一个叶子上最小hessian和。取值区间为[0, 1]，默认为1e-3
boost_from_average	-	是否将初始分数调整为标签的平均值，以加快收敛速度,，默认为True
boosting_type	-	提升方法的提升类型。 可选值有：gbdt、gbrt、rf、dart、goss，默认为gbdt
lambda_l1	-	L1正则化系数，默认为0.0
lambda_l2	-	L2正则化系数,，默认为0.0
num_batches	-	如果大于0，在训练中将数据集分割成不同的批次，默认为0
parallelism	-	学习树时的并行方法，支持data_parallel, voting_parallel，默认为"data_parallel"

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "outer_pipeline_stages": None,
    "input_features_str": "",  # @param
{"label":"input_features_str","type":"string","required":"false","helpTip":""}
    "label_col": "",  # @param {"label":"label_col","type":"string","required":"true","helpTip":""}
    "regressor_feature_vector_col": "model_features",  # @param
{"label":"regressor_feature_vector_col","type":"string","required":"false","helpTip":""}
    "prediction_col": "prediction",  # @param
{"label":"prediction_col","type":"string","required":"false","helpTip":""}
    "objective": "regression",  # @param {"label":"objective","type":"string","required":"false","helpTip":""}
    "max_depth": -1,  # @param
{"label":"max_depth","type":"integer","required":"false","range":"[-1,2147483647]","helpTip":""}
    "num_iteration": 100,  # @param
{"label":"num_iteration","type":"integer","required":"false","range":"(0,2147483647]","helpTip":""}
    "learning_rate": 0.1,  # @param {"label":"learning_rate","type":"number","required":"false","helpTip":""}
    "num_leaves": 31,  # @param
{"label":"num_leaves","type":"integer","required":"false","range":"(0,2147483647]","helpTip":""}
    "max_bin": 255,  # @param
{"label":"max_bin","type":"integer","required":"false","range":"(0,2147483647]","helpTip":""}
    "bagging_fraction": 1.0,  # @param
{"label":"bagging_fraction","type":"number","required":"false","helpTip":""}
```


版权所有 © 华为云计算技术有限公司

```
    "bagging_freq": 0,  # @param
{"label":"bagging_freq","type":"integer","required":"false","range":"[0,2147483647]","helpTip":""}
    "bagging_seed": 3,  # @param
{"label":"bagging_seed","type":"integer","required":"false","range":"[0,2147483647]","helpTip":""}
    "early_stopping_round": 0,  # @param
{"label":"early_stopping_round","type":"integer","required":"false","range":"[0,2147483647]","helpTip":""}
    "feature_fraction": 1.0,  # @param
{"label":"feature_fraction","type":"number","required":"false","helpTip":""}
    "min_sum_hessian_in_leaf": 1e-3,  # @param
{"label":"min_sum_hessian_in_leaf","type":"number","required":"false","helpTip":""}
    "boost_from_average": True,  # @param
{"label":"boost_from_average","type":"boolean","required":"false","helpTip":""}
    "boosting_type": "gbdt",  # @param
{"label":"boosting_type","type":"string","required":"false","helpTip":""}
    "lambda_l1": 0.0,  # @param {"label":"lambda_l1","type":"number","required":"false","helpTip":""}
    "lambda_l2": 0.0,  # @param {"label":"lambda_l2","type":"number","required":"false","helpTip":""}
    "num_batches": 0,  # @param
{"label":"num_batches","type":"integer","required":"false","range":"[0,2147483647]","helpTip":""}
    "parallelism": "data_parallel"  # @param
{"label":"parallelism","type":"string","required":"false","helpTip":""}
}
lightgbm_regressor____id___ = MLSLightGbmRegression(**params)
lightgbm_regressor____id___.run()
# @output {"label":"pipeline_model","name":"lightgbm_regressor____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
```

7.5.3.5.4 线性回归

概述

“线性回归”节点用于产生线性回归模型。它是利用数理统计中的回归分析，来确定两种或两种以上变数间相互依赖的定量关系的统计分析方法。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

参数说明

参数	子参数	参数说明
b_use_default_encoder	-	是否使用默认编码，默认为True
input_features_str	-	输入的列名以逗号分隔组成的字符串，例如： "column_a" "column_a,column_b"
label_col	-	目标列

 版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
regressor_feature_vector_col	-	算子输入的特征向量列的列名，默认为"model_features"
max_iter	-	最大迭代次数，默认为100
reg_param	-	正则化参数，默认为0.0
elastic_net_param	-	弹性网络参数，默认为0.0
tol	-	收敛阈值，默认为1e-6
fit_intercept	-	是否使用截距，默认为True
standardization	-	是否对特征进行正则化，默认为True
solver	-	优化时采用的处理算法，支持l-bfgs、normal、auto，默认为"auto"
aggregation_depth	-	聚合深度，默认为2
loss	-	损失函数类型，支持squaredError、huber，默认为"squaredError"
epsilon	-	默认为1.35

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "b_use_default_encoder": True,  # @param {"label": "b_use_default_encoder", "type": "boolean", "required": "true", "helpTip": ""}
    "input_features_str": "",  # @param {"label": "input_features_str", "type": "string", "required": "false", "helpTip": ""}
    "outer_pipeline_stages": None,
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": "target label column"}
    "regressor_feature_vector_col": "model_features",  # @param {"label": "regressor_feature_vector_col", "type": "string", "required": "true", "helpTip": ""}
    "max_iter": 100,  # @param {"label": "max_iter", "type": "integer", "required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "reg_param": 0.0,  # @param {"label": "reg_param", "type": "number", "required": "true", "range": "[0.0,none)", "helpTip": ""}
    "elastic_net_param": 0.0,  # @param {"label": "elastic_net_param", "type": "number", "required": "true", "range": "[0.0,none)", "helpTip": ""}
    "tol": 1e-6,  # @param {"label": "tol"", "type": "number", "required": "true", "range": "[0.0,none)", "helpTip": ""}
    "fit_intercept": True,  # @param {"label": "fit_intercept"", "type": "boolean", "required": "true", "helpTip": ""}
    "standardization": True,  # @param {"label": "standardization"", "type": "boolean", "required": "true", "helpTip": ""}
    "solver": "auto",  # @param {"label": "solver", "type": "enum", "required": "true", "options": "l-bfgs,normal,auto", "helpTip": ""}
    "aggregation_depth": 2,  # @param {"label": "aggregation_depth", "type": "integer", "required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "loss": "squaredError",  # @param {"label": "loss", "type": "enum", "required": "true", "options": "squaredError,huber", "helpTip": ""}
    "epsilon": 1.35  # @param {"label": "epsilon", "type": "number", "required": "true", "range": "(1.0,none)",
```


版权所有 © 华为云计算技术有限公司

```
"helpTip": ""}
}
linear_regression____id___ = MLSLinearRegression(**params)
linear_regression____id___.run()
# @output {"label":"pipeline_model","name":"linear_regression____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
```

7.5.3.5.5 随机森林回归

概述

“随机决策森林回归”节点用于产生回归模型。随机决策森林是用随机的方式建立一个森林模型，森林由很多的决策树组成，每棵决策树之间没有关联。当有一个新的样本输入时，该样本取值为所有决策树的预测值的平均值。

随机决策森林回归中的决策树算法是递归地构建决策树的过程，用平方误差最小准则，进行特征选择，生成二叉树。平方误差计算公式如下：

$$\frac{1}{N}\sum_{i=1}^{N}(y_i-\mu)^2$$

其中 $\mu = \sum_{i=1}^{N} y_i$ 是样本类标的均值，y_i 是样本的标签，N 是样本数量。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

参数说明

参数	子参数说明	参数说明
b_use_default_encoder	-	是否使用默认编码，默认为True
input_features_str	-	输入的列名以逗号分隔组成的字符串，例如： "column_a" "column_a,column_b"
label_col	-	目标列
regressor_feature_vector_col	-	算子输入的特征向量列的列名，默认为"model_features"

版权所有 © 华为云计算技术有限公司

参数	子参数说明	参数说明
max_depth	-	树的最大深度，默认为5
max_bins	-	最大分箱数，默认为32
min_instances_per_node	-	节点分割时，要求子节点必须包含的最少实例数，默认为1
min_info_gain	-	节点是否分割要求的最小信息增益，默认为0.0
subsampling_rate	-	学习每棵决策树用到的训练集的抽样比例，默认为1.0
num_trees	-	树的个数，默认为20
feature_subset_strategy	-	节点分割时考虑用到的特征列的策略，支持auto、all、onethird、sqrt、log2、n，默认为"all"

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "b_output_action": True,
    "b_use_default_encoder": True,  # @param {"label": "b_use_default_encoder", "type": "boolean", "required": "true", "helpTip": ""}
    "input_features_str": "",  # @param {"label": "input_features_str", "type": "string", "required": "false", "helpTip": ""}
    "outer_pipeline_stages": None,
    "label_col": "",  # @param {"label": "label_col", "type": "string", "required": "true", "helpTip": ""}
    "regressor_feature_vector_col": "model_features",  # @param {"label": "regressor_feature_vector_col", "type": "string", "required": "true", "helpTip": ""}
    "max_depth": 5,  # @param {"label": "max_depth", "type": "integer", "required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "max_bins": 32,  # @param {"label": "max_bins", "type": "integer", "required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "min_instances_per_node": 1,  # @param {"label": "min_instances_per_node", "type": "integer", "required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "min_info_gain": 0.0,  # @param {"label": "min_info_gain", "type": "number", "required": "true", "range": "[0.0,none)", "helpTip": ""}
    "impurity": "variance",
    "subsampling_rate": 1.0,  # @param {"label": "subsampling_rate", "type": "number", "required": "true", "range": "(0.0,1.0]", "helpTip": ""}
    "num_trees": 20,  # @param {"label": "num_trees", "type": "integer", "required": "true", "range": "(0,2147483647]", "helpTip": ""}
    "feature_subset_strategy": "all"  # @param {"label": "feature_subset_strategy", "type": "enum", "required": "true", "options":"auto,all,onethird,sqrt,log2", "helpTip": ""}
}
rf_regressor____id___ = MLSRandomForestRegression(**params)
rf_regressor____id___.run()
# @output {"label":"pipeline_model","name":"rf_regressor____id___.get_outputs()['output_port_1']","type":"PipelineModel"}
```

7.5.3.6 文本

版权所有 © 华为云计算技术有限公司

7.5.3.6.1 TF-IDF

概述

“词频-逆文档频率”节点主要功能是计算某个词对于所属文档的重要程度。词频-逆文档频率（Term Frequency-Inverse Document Frequency, TF-IDF）算法规定某个词语的重要性与它在一个文档中出现的次数成正比，与该词语在语料库的所有文档中出现的频率成反比。给定语料库D，则文档d_j中的词语d_j的$TFIDF_i$定义如下：

$$TFIDF_{i,j} = TF_{i,j} \times IDF_i \quad (1)$$

$$TF_{i,j} = \frac{N_{i,j}}{\sum_k N_{k,j}} \quad (2)$$

$$IDF_i = \log \frac{|D|}{|\{j : t_i \in d_j\}|} \quad (3)$$

式中，$TF_{i,j}$指词语t_i在文档d_j出现频率的归一化结果，$N_{i,j}$表示该词在文档d_j中的出现次数，$\sum_k N_{k,j}$表示文件d_j中所有词语的出现次数之和；IDF_i表示词语t_i的逆向文件频率（Inverse Document Frequency），|D|表示语料库的文件总数，$|\{j : t_i \in d_j\}|$表示包含词语t_i的文件数目。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

spark pipeline类型的模型

参数说明

参数	子参数	参数说明
text_col	-	文本列所在的列名，默认为"text_col"
tokenizer_col	-	对数据集文本列分词之后的结果列名，默认为"tokenizer_col"
tf_col	-	对数据集应用HashingTF之后的结果列名，默认为"tf_col"

版权所有 © 华为云计算技术有限公司

参数	子参数	参数说明
idf_col	-	对数据集应用IDF之后的结果列名，默认为"idf_col"
tf_binary	-	默认为False
tf_num_features	-	HashingTF中的特征个数
idf_min_doc_freq	-	最小文档频率，默认为0

样例

```
inputs = {
    "dataframe": None  # @input {"label":"dataframe","type":"DataFrame"}
}
params = {
    "inputs": inputs,
    "text_col": "text_col",  # @param {"label":"text_col","type":"string","required":"false","helpTip":""}
    "tokenizer_col": "tokenizer_col",  # @param
{"label":"tokenizer_col","type":"string","required":"false","helpTip":""}
    "tf_col": "tf_col",  # @param {"label":"tf_col","type":"string","required":"false","helpTip":""}
    "idf_col": "idf_col",  # @param {"label":"idf_col","type":"string","required":"false","helpTip":""}
    "tf_binary": False,  # @param {"label":"tf_binary","type":"boolean","required":"false","helpTip":""}
    "tf_num_features": 1 << 18,  # @param
{"label":"tf_num_features","type":"integer","required":"true","range":"(0,2147483647]","helpTip":""}
    "idf_min_doc_freq": 0  # @param
{"label":"idf_min_doc_freq","type":"integer","required":"true","range":"(0,2147483647]","helpTip":""}
}
tf_idf____id___ = MLSTFIDF(**params)
tf_idf____id___.run()
# @output {"label":"pipeline_model","name":"tf_idf____id___.get_outputs()
['output_port_1']","type":"PipelineModel"}
# @output {"label":"dataframe","name":"tf_idf____id___.get_outputs()['output_port_2']","type":"DataFrame"}
```

7.5.3.6.2 文本词向量

概述

“文本词向量”节点用于将词和句/段落映射到一个向量，可用来表示词与词之间或句与句之间的关系。该算法基于Skip-gram模型利用词语来预测它的上下文，并表示为向量形式，可应用于社交网络中的推荐系统、文本相似度等场景。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象，通常为分词算子的输出，可参考分词算子的使用。

版权所有 © 华为云计算技术有限公司

输出

参数	子参数	参数说明
output	output_port_1	output为字典类型，output_port_1为pyspark中的PipelineModel类型。
output	output_port_2	output_port_2为pyspark中的DataFrame类型，为词向量。
output	output_port_3	output_port_3为pyspark中的DataFrame类型，为文本向量。

参数说明

参数	是否必选	参数说明	默认值
text_col	是	输入数据集中文本所在列的列名。	"words"
text_id	是	文本id列，用一个id代表文本。	"id"
result_col	是	结果列的列名。	"result_col"
delimiter	是	单词间的分隔符。	" "
vector_size	是	向量长度。	10
min_count	是	词出现的最小次数，低于该值的单词会被过滤。	2
num_partitions	否	分区数目。	8
step_size	是	迭代优化时的步长，学习率。	0.025
max_iter	是	最大迭代次数。	1
window_size	是	训练过程中的窗口大小。	5
max_sentence_length	否	最大句子长度。	1000

样例

样例数据

版权所有 © 华为云计算技术有限公司

id	title
7436	【春招】面试流程 注意事项 加分点 防骗指南
7438	测试妹子现场面试，当场给出13K
7439	程序员面试一个Java实习生，展示教科书般回答，有两个大厂offer的他会来么?
7468	南邮大四Java实习生，已拿拼多多Offer
7469	面试二本实习生，疯狂问项目，他从容应对
7595	一个人的副业可以有多成功?
7597	什么是外推谬误?
7696	【启舰】大牛程序员为什么写博客，有哪些你不知道的秘密?
7706	【启舰】都说平时要多看书，可就是看不进去啊，怎么办?
7707	【启舰】马云：毕业去小公司？信你个鬼！
7821	【启舰】程序员垃圾简历长什么样?
7824	【启舰】程序员是做全栈工程师好？还是专注一个领域好?
7825	【启舰】完了，我不适合做程序员！
7827	【启舰】程序员的几层境界，你在哪一层?
7828	【启舰】程序员，就业难、工资低？很可能是你方向没选对！

该数据为分词算子的输入，分词算子的输出作为文本词向量的输入。

配置流程

运行流程

参数设置

版权所有 © 华为云计算技术有限公司

设置参数

* text_col :	words
* text_id :	id
* result_col :	result_col
* delimiter :	
* vector_size :	10
* min_count :	2
num_partitions :	8
* step_size :	0.025
* max_iter :	1
* window_size :	5
max_sentence_length :	1000

查看结果

词向量

word	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9
CV	0.010883204638957977	0.00898024346679449	0.011292695999145508	-0.07180426269769669	-0.06830473244190216	0.1625475138425827	-0.025501411408185596	0.019509041688572	-0.025308825075626373	-0.017953403294086456
辣辣	-0.1189766377210617	0.09712073206901555	0.016816649585962296	0.07983928918838501	0.08646808564662933	0.15965214371681213	-0.1896546185016632	-0.013733059167861938	-0.13680005073547363	-0.24714908003807068
货币	-0.03685355558991432	-0.024570344015955925	0.025507041772651672	-0.0064414436928927	-0.028213236480951 31	0.06280063092708588	-0.04690344259142876	-0.03860289603471756	-0.04823201894760132	0.04833373427391052
人物	-0.15489000082015 99	-0.09580201655626297	-0.12619613111019135	-0.018104899674654007	-0.037448208779096 6	0.08117342740297318	0.03286194801330566 4	0.10378102958202362	0.03925522044301033	-0.07427972555160522
可靠性	-0.09774727374315262	-0.013572181575000286	0.07456044852733612	0.08555415272712708	-0.033060070127248764	-0.03776228055357933	0.07928389310836792	-0.024892289191484 45	0.08441628515720367	0.06508749723434448
另类	-0.07828654348850253	0.00032737536821514 37	-0.032151080667972565	-0.012635922059416771	-0.01160819735378027	0.03431905806064606	0.02522701770067215	-0.062865190207958 22	0.01506272330880165 1	-0.027635058388113976
E	-0.04977769777178764	-0.07868344336748123	-0.004260015673935413	0.03184404224157334	0.14774419367313385	-0.011877093464136124	0.10342000424861908	0.05839517340064049	0.15036442875862122	-0.025633698832932949
推送	0.0003958269953727722	-0.18008549511432648	-0.11815448850393295	0.05050551146268846	-0.08367297053337097	0.013315887190401554	0.026200000196695328	0.06476093083620071	0.011224000714719296	0.021536355838179 59
其它	0.0016399472951889038	0.00044013001024723053	0.006324101239442825	-0.006756717339158058	0.07722088694572449	-0.038560669869184494	-0.042668003588914 87	0.06246673315763473 5	0.04485565423965454	0.03238717466592789
大趋势	0.015588345006108284	0.04317222908139229	0.04626947268843651	-0.024781262502074 24	0.04570148140192032	0.05749630928039551	0.010457322001457214	-0.025602299720048904	-0.004918672144412994	-0.045014858245849 61

文本向量

id	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9
7436	-0.04762900090561464	0.018899086647881914	0.20334642939269543	-0.07424078651596434	0.025668383862536687	0.2698750779605829	-0.19686289360890022	-0.06396523734124808	0.08448474983183238	-0.14655872273187226
7438	-0.03401986365206 54	0.048122692666947846	-0.007984619401395322	0.19633079357445 24	0.006655751261860132	0.2092346439138055	-0.28633290464058 52	-0.03265637108124793	-0.03414920819923282	-0.018003105651587248
7439	-0.0429987557740374	0.06332243547182191	0.009132237512279 64	0.26159311026673426	-0.041786799630658 19	0.2870950877285478	-0.44671863182024524	-0.024670647965236145	-0.11981245071034541	-0.2046080016615716
7468	0.023001750310262043	0.12244695409733 4	0.09218770312145352	0.004265353083610535	0.13633815292268991	0.14283224707469344	-0.23867425819238025	-0.02599578226060051	0.09488284913823009	-0.11354237794876099
7469	-0.15005223551027785	0.05968129262328148	-0.14190931742389995	0.07842043042182922	-0.03089523470650116 4	0.22650947670141855	-0.18820020866890746	-0.007373000184694925	-0.013535359905088412	-0.14385638370489 08
7595	0.027510073 8027085	0.03525395587914 52	-0.07897484447393152	0.40747494002183277	-0.023927158676087856	0.2158943514029185	-0.56625705626275 8	0.01677729520532 82	-0.09657547540134853	-0.23857714649703765
7597	0.18972970843315126	0.02309345453977585	0.14111225493252277	0.24818381965160372	-0.26020634472370147	0.17905344650505 07	-0.43797235488891606	0.18629448115825653	-0.34835217595100404	-0.13244842290878298
7696	-0.07522431026925058	0.12722533486564 37	-0.023399662336005884	0.3077683089410557	-0.13073187304989 38	0.2927830942413386 4	-0.52773795890457493	0.00128551813609459 83	-0.16709628728601864	-0.3044670277329929
7706	-0.06448460331088618	0.05048723442872104	-0.08337291255573 64	0.22272266736744264	-0.021671129586665252	0.1518441044088257	-0.37395851666990076	-0.056717418428314355	-0.02478713072885416	-0.25635490507671704
7707	-0.05599140142245839	0.03655387697120507 4	-0.0301076201411585	0.12620225896437962	-0.034777374177550 27	0.19881367866570 75	-0.39729661618669826	-0.075439239169160 52	0.08488320106019576	-0.2294418531159 56

7.5.3.6.3 词频统计

概述

词频统计是指统计一个字符串中，出现了多少个单词以及这些单词出现的次数。该算子一般接在分词算子后面，用以统计分词后各个单词的出现次数。

版权所有 © 华为云计算技术有限公司

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象，一般为分词后的结果。

输出

参数	子参数	参数说明
output	output_port_1	output为字典类型，output_port_1为pyspark中的DataFrame类型对象，为词频统计的结果。

参数说明

参数	是否必选	参数说明	默认值
doc_id	是	文章id	"id"
doc_content	是	文章内容（分词后的字段名）	"segment"
delimiter	否	单词之间的分隔符	" "

样例

数据样本

```
id,segment
doc001,词频统计 是指 统计 一个 字符串 中 ， 出现了 多少 个 单词 以及 这些 单词 出现 的 次数 。
```

配置流程

运行流程

参数设置

版权所有 © 华为云计算技术有限公司

结果查看

```
id,word,count
doc001,的,1
doc001,词频统计,1
doc001,一个,1
doc001,出现了,1
doc001,，,1
doc001,个,1
doc001,出现,1
doc001,多少,1
doc001,。,1
doc001,单词,2
doc001,统计,1
doc001,次数,1
doc001,这些,1
doc001,以及,1
doc001,中,1
doc001,是指,1
doc001,字符串,1
```

7.5.3.6.4 文章相似度

概述

支持cosine、levenshtein、jaccard和最长公共子序列四种方法计算文章的相似度。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象

输出

DataRame

版权所有 © 华为云计算技术有限公司

参数说明

参数名	参数类型	是否必选	参数含义	默认值
inputSelectedColName1	String	是	输入表被选第一个字段名称	无
inputSelectedColName2	String	是	输入表被选第二个字段名称	无
inputAppendColNames	String	否	输入表添加的其他字段名称， 涉及多个字段以逗号分割	无
outputColName	String	否	输出的字段名称	output
method	String	是	字符串相似度计算方法 levenshtein,levenshtein_sim,lcs,lcs_sim,cosine,hash_jaccard_sim	cosine

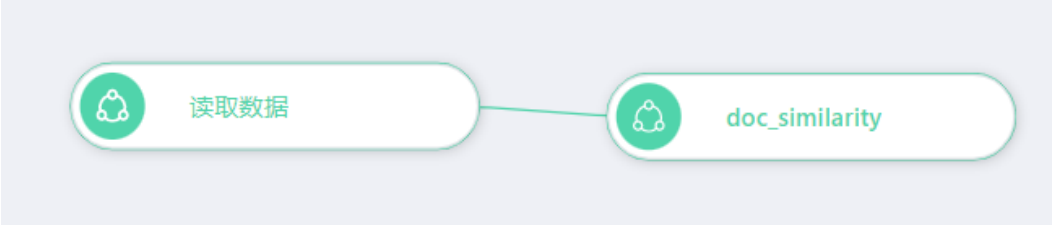
样例

数据样本

```
doc1,doc2
浩瀚 的 太平洋 潮起 潮落，见证 风云变幻 、世事 沧桑 。伴随 世界 经济 重心 逐步 东移 ， 亚太 地区 吸引 越来越多 的 全球 目光 。,浩瀚 的 太平洋 潮起 潮落，见证 风云变幻 、世事 沧桑 。伴随 世界 经济 重心 逐步 东移 ， 亚太 地区 吸引 越来越多 的 全球 目光 。
```

配置流程

运行流程



输出结果

```
+--------------------------------+--------------------------------+------+
|                            doc1|                            doc2|output|
+--------------------------------+--------------------------------+------+
|浩瀚 的 太平洋 潮起 潮落，见证...|浩瀚 的 太平洋 潮起 潮落，见证...|   1.0|
+--------------------------------+--------------------------------+------+
```


版权所有 © 华为云计算技术有限公司

7.5.3.6.5 字符串相似度

概述

支持cosine、levenshtein、jaccard、最长公共子序列、minhash_sim、ssk、simhash_hamming_sim七种方法计算字符串的相似度。

输入

参数	子参数	参数说明
inputs	dataDF	inputs为字典类型，dataDF是输入字符串集合，数据类型是pyspark中的DataFrame类型对象
inputs	paramDF	paramDF是输入的被映射的字符串集合，数据类型是pyspark中的DataFrame类型对象

输出

DataRame

参数说明

参数名	参数类型	是否必选	参数含义	默认值
inputSelectedColName1	String	是	输入表被选第一个字段名称	无
inputSelectedColName2	String	是	输入表被选第二个字段名称	无
inputAppendColNamesStr	String	否	输入表添加的其他字段名称，涉及多个字段以逗号分隔	无
inputAppendRenameColNamesStr	String	否	输入表添加的其他需要rename字段映射关系	colName1:colReName1,colName2:colReName2
outputColName	String	否	输出的字段名称	distance
method	String	是	字符串相似度计算方法levenshtein,levenshtein_sim,lcs,lcs_sim,cosine,hash_jaccard_sim	cosine

版权所有 © 华为云计算技术有限公司

参数名	参数类型	是否必选	参数含义	默认值
lambda	Double	否	SSK需要的参数	0.5
k	Int	否	SSK需要的参数	10
kVec	Int	否	SimHashHamming字符向量的大小	64
b	Int	否	minhash分桶大小	10
seed	Int	否	minhash随机hash函数的种子	0

样例

数据样本

```
str1,str2
51校园app,51校园app
51校园app,51校园app下载
51校园app,51校园app下载官网
```

配置流程

运行流程

设置参数

全部是默认参数。

输出结果

运行结果

```
+---------+-----------------+------------------+
|     str1|             str2|          distance|
+---------+-----------------+------------------+
|51校园app|        51校园app|               1.0|
|51校园app|    51校园app下载|0.9045340337332909|
|51校园app|51校园app下载官网|0.8320502943378437|
+---------+-----------------+------------------+
```


版权所有 © 华为云计算技术有限公司

7.5.3.6.6 字符串相似度 topN

概述

支持cosine、levenshtein、jaccard、最长公共子序列、minhash_sim、ssk、simhash_hamming_sim七种方法计算 文章的相似度

输入

参数	子参数	参数说明
inputs	dataDF	inputs为字典类型，dataDF是输入字符串集合，数据类型是pyspark中的DataFrame类型对象
inputs	paramDF	paramDF是输入的被映射的字符串集合，数据类型是pyspark中的DataFrame类型对象

输出

DataRame

参数说明

参数名	参数类型	是否必选	参数含义	默认值
inputSelectedColName1	String	是	输入表被选字段名称。 当该字段为空时，dataDF中第一个string类型的字段。	无
mapSelectedColName2	String	是	map表被选字段名称。 当该字段为空时，paramDF中第一个string类型的字段。	无
inputAppendColNamesStr	String	否	输入表添加的其他字段名称，涉及多个字段以逗号分隔。	无
inputAppendRenameColNamesStr	String	否	输入表添加的其他需要rename字段映射关系。	colName1:colReName1,colName2:colReName2

版权所有 © 华为云计算技术有限公司

参数名	参数类型	是否必选	参数含义	默认值
mapAppendColNamesStr	String	否	map表添加的其他字段名称，涉及多个字段以逗号分隔。	无
mapAppendRenameColNamesStr	String	否	map表添加的其他需要rename字段映射关系。	colName1:colReName1,colName2:colReName2
outputColName	String	否	输出的字段名称。	dist
method	String	是	字符串相似度计算方法 levenshtein,levenshtein_sim,lcs,lcs_sim,cosine,hash_jaccard_sim	cosine
lambda	Double	否	SSK需要的参数。	0.5
k	Int	否	SSK需要的参数。	10
kVec	Int	否	SimHashHamming字符向量的大小。	64
b	Int	否	minhash分桶大小。	100
seed	Int	否	minhash随机hash函数的种子。	0
topN	Int	否	最相似的TopN字符串。	10
subLen	Int	否	粗排时，最小子串大小。	1

样例

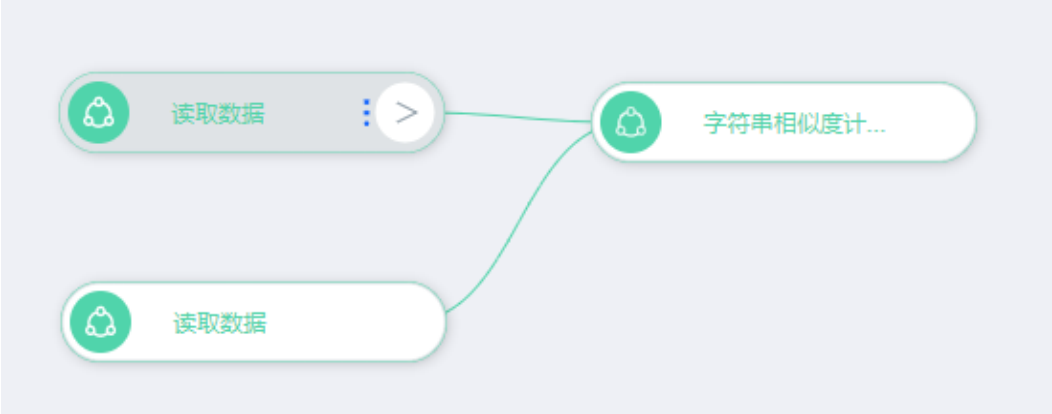
数据样本

```
str1
51校园app
51校园app下载
51校园app下载官网
```

配置流程

版权所有 © 华为云计算技术有限公司

运行流程



设置参数全部是默认参数。

输出结果

```
+-----------------+-----------------+-----------------+
|              str|       str_mapcol|             dist|
+-----------------+-----------------+-----------------+
|51校园app下载官网|51校园app下载官网|              1.0|
|51校园app下载官网|    51校园app下载|  0.9198662110078|
|51校园app下载官网|        51校园app|0.8320502943378437|
|    51校园app下载|    51校园app下载|              1.0|
|    51校园app下载|51校园app下载官网|  0.9198662110078|
|    51校园app下载|        51校园app|0.9045340337332909|
|        51校园app|        51校园app|              1.0|
|        51校园app|    51校园app下载|0.9045340337332909|
|        51校园app|51校园app下载官网|0.8320502943378437|
+-----------------+-----------------+-----------------+
```

7.5.3.6.7 NGram Count

概述

将分词后的句子生成连续N个词的NGram短语，并进行全局个数的统计，支持权重列输入。

输入

参数	子参数	参数说明
inputs	input_table	输入表表名，输入的包含分词后的句子的数据表；必填；
inputs	vocab_table	词袋词汇表；非必填；
inputs	count_table	历史ngram-count输出表；非必填；

版权所有 © 华为云计算技术有限公司

输入参数说明

参数名称	参数说明	参数要求
input_words_col_name	分词列，即进行ngram分词处理的列	string类型；必填；仅支持单列
input_words_sep	分词列中的词分隔符	string类型；必填；默认为" "
input_weight_col_name	分词行权重	string类型；表列为数值类型；非必填；
vocab_words_col_name	词袋词汇表的词汇列列名	string类型；如果词袋表不为空，此项为必填
count_gram_col_name	每个ngram短语的词个数（n），如1-gram，2-gram...，显示1-n等	string类型；表列为数值类型；如果历史输出表不为空，此项为必填
count_word_col_name	ngram短语列	string类型；如果历史输出表不为空，此项为必填
count_count_col_name	ngram统计列	string类型；表列为数值类型；如果历史输出表不为空，此项为必填
order	ngram最大单词个数，即n-gram的n	integer类型；必填；order范围为[1,3]

输出

参数	子参数	参数说明
output	output_port_1	输出表表名，标签为dataframe

输出表说明

列名	列名描述	备注
ngram	ngram短语词个数	1~n
words	ngram短语	-
count	个数统计	weight加权累计

版权所有 © 华为云计算技术有限公司

说明

1. 词袋过滤：
不在词袋中的单个词会被转为<unk>。
2. order含义：
例如order为3，则会输出1-gram 2-gram 3-gram。
3. weight列：
无weight列默认weight全为1。
4. count计算方式：
相同ngram的weight进行累加；
当前ngram-count输出表与历史ngram-count输出表相同ngram和words的count进行累加；
多列共用一列weight，如ngram相同，则对应相同weight累加作为最终count；
5. 其他：
count_gram_col_name不合法的行会被过滤掉；每行会在首尾添加<s></s>标识。

样例

数据输入

- input_table

sentence1	weight
Try your best.	1
Try to do it.	2
Try to finish it tomorrow.	2
You can try to do it.	2
	1
Why not to have a try?	1

- vocab_table

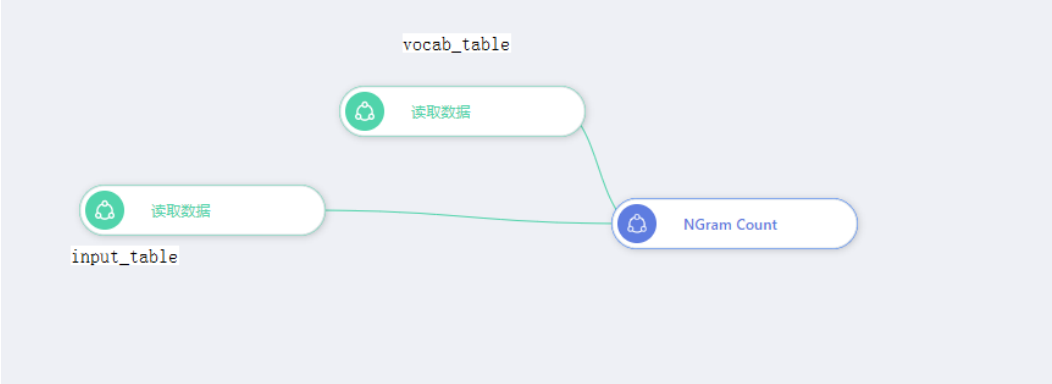
word
Try
try
to
do
your
best
best.
it

版权所有 © 华为云计算技术有限公司

not
it.
tomorrow.

配置流程

运行流程



参数设置

输出结果

 版权所有 © 华为云计算技术有限公司

ngram	words	count
1	</s>	9
1	<s>	9
1	<unk>	10
1	Try	5
1	best.	1
1	do	4
1	it	2
1	it.	4
1	not	1
1	to	7
1	tomorrow.	2
1	try	2
1	your	1
2	<s> </s>	1
2	<s> <unk>	3
2	<s> Try	5
2	<unk> </s>	1
2	<unk> <unk>	4
2	<unk> it	2
2	<unk> not	1
2	<unk> try	2
2	Try to	4
2	Try your	1
2	best. </s>	1
2	do it.	4
2	it tomorrow.	2
2	it. </s>	4
2	not to	1
2	to <unk>	3
2	to do	4

版权所有 © 华为云计算技术有限公司

2	tomorrow. </s>	2
2	try to	2
2	your best.	1
3	<s> <unk> <unk>	2
3	<s> <unk> not	1
3	<s> Try to	4
3	<s> Try your	1
3	<unk> <unk> </s>	1
3	<unk> <unk> <unk>	1
3	<unk> <unk> try	2
3	<unk> it tomorrow.	2
3	<unk> not to	1
3	<unk> try to	2
3	Try to <unk>	2
3	Try to do	2
3	Try your best.	1
3	do it. </s>	4
3	it tomorrow. </s>	2
3	not to <unk>	1
3	to <unk> <unk>	1
3	to <unk> it	2
3	to do it.	4
3	try to do	2
3	your best. </s>	1

7.5.3.6.8 PMI

概述

承接分词结果，计算一个文档里单词两两之间的互信息值（PMI）。PMI计算公式如下：

$$PMI=ln\frac{P(x,y)}{P(x)*P(y)}=ln\frac{N(x,y)*N}{N(x)*N(y)}$$

版权所有 © 华为云计算技术有限公司

相关概念解释：

共现对儿：一句话里面如果两个词在句子里的距离小于等于定义的滑动窗口大小，则这两个词共现形成共现对儿。

P(x,y)：x,y为两个词，P(x,y)指两个词的共现概率，等于两个词的共现次数N(x, y)除以所有共现对儿的个数N。

P(x)：指x与其他词共现的概率，等于x出现在所有共现对儿中的次数N(x)除以所有共现对儿的个数N。

P(y)同理。

输入

参数	子参数	参数说明
inputs	input_table	输入的包含分词后句子的数据表；必选

输入参数说明

参数名称	参数描述	参数要求
doc_col_name	分词后的文本列	string类型；必填；多列时每列当做单独的句子处理
doc_sep	分词列中的词分隔符	string类型；必填；默认为" "
min_count	最小词频	integer类型；非必填；默认为5，小于该值的词会被过滤掉，不填则识别为0，取值范围[0,2147483647]
window_size	滑动窗口大小	integer类型；非必填；默认为整行，取值范围[1, 2147483647]
partitions	数据重分区数	integer类型；非必填；取值范围[1,5000]；

版权所有 © 华为云计算技术有限公司

说明

1. partitions

大数据量情况下建议partitions重分区数取大一些，100w长文本数据建议取1000，500w长文本数据建议取2000，如果在前两种场景下用户自定义partitions小于需求值，系统会自动替换为需求值（即前面的1000，2000）。

2. 资源配置

数据量较大时建议采用更大的资源配置，可以设置executor memory大一些，参考配置如下：

cluster 32配置：

--executor-memory 8G \

--executor-cores 2 \

--num-executors 14 \

--driver-cores 4 \

--driver-memory 15G \

cluster 64配置：

--executor-memory 24G \

--executor-cores 6 \

--num-executors 10 \

--driver-cores 4 \

--driver-memory 15G \

3. 参数配置

如果运行效率过慢，可考虑增大资源配置，或修改min_count、window_size参数，min_count大一些，window_size小一些。

输出

参数	子参数	参数说明
output	output_port_1	输出表表名；标签为dataframe

输出表说明

列名	列名描述
word1	共现词对儿的第一个单词
word2	共现词对儿的第二个单词
word1_count	word1出现在所有共现词对儿中的次数
word2_count	word2出现在所有共现词对儿中的次数
co_occurrence_count	(word1, word2)共现词对儿的个数
pmi	word1与word2的PMI值

版权所有 © 华为云计算技术有限公司

样例

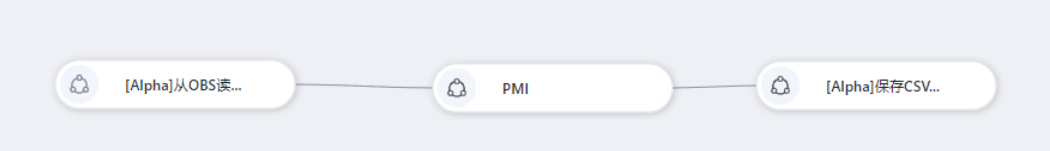
数据输入

input_table

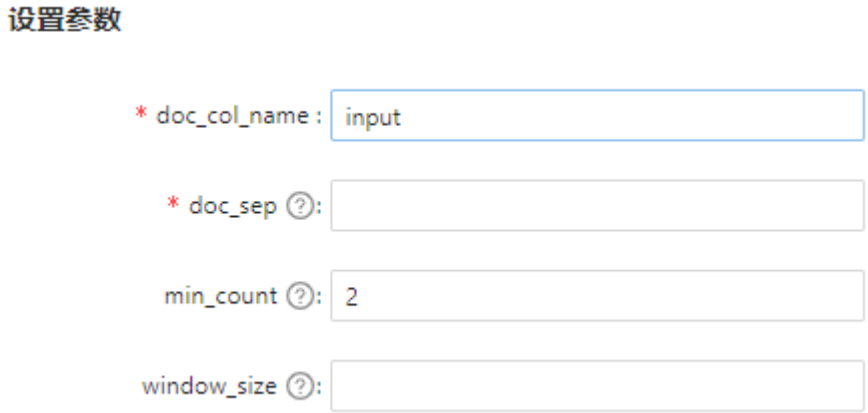
input
Try to try it how to try it
Need to try it
You try to do do something
How can you these days still not try it not do anything
It is a good chance to try also you can do it
You are right that it is a good chance to try

配置流程

运行流程



输入参数



输出结果

word1	word2	word1_count	word2_count	co_occurrences_count	pmi
You	a	11	16	1	-0.36646
You	chance	11	16	1	-0.36646

版权所有 © 华为云计算技术有限公司

You	do	11	23	2	-0.03622
You	good	11	16	1	-0.36646
You	is	11	16	1	-0.36646
You	it	11	34	1	-1.12023
You	to	11	32	2	-0.36646
You	try	11	38	2	-0.53831
a	can	16	15	1	-0.67662
a	chance	16	16	2	-0.04801
a	do	16	23	1	-1.10406
a	good	16	16	2	-0.04801
a	is	16	16	2	-0.04801
a	it	16	34	2	-0.80178
a	to	16	32	2	-0.74116
a	try	16	38	2	-0.91301
a	you	16	15	1	-0.67662
can	chance	15	16	1	-0.67662
can	do	15	23	2	-0.34638
can	good	15	16	1	-0.67662
can	is	15	16	1	-0.67662
can	it	15	34	2	-0.73724
can	not	15	12	2	0.304211
can	to	15	32	1	-1.36977
can	try	15	38	2	-0.84847
can	you	15	15	2	0.081068
chance	do	16	23	1	-1.10406
chance	good	16	16	2	-0.04801
chance	is	16	16	2	-0.04801
chance	it	16	34	2	-0.80178
chance	to	16	32	2	-0.74116
chance	try	16	38	2	-0.91301
chance	you	16	15	1	-0.67662

版权所有 © 华为云计算技术有限公司

do	do	23	23	1	-1.46697
do	good	23	16	1	-1.10406
do	is	23	16	1	-1.10406
do	it	23	34	2	-1.16469
do	not	23	12	2	-0.12323
do	to	23	32	3	-0.6986
do	try	23	38	4	-0.58276
do	you	23	15	2	-0.34638
good	is	16	16	2	-0.04801
good	it	16	34	2	-0.80178
good	to	16	32	2	-0.74116
good	try	16	38	2	-0.91301
good	you	16	15	1	-0.67662
is	it	16	34	2	-0.80178
is	to	16	32	2	-0.74116
is	try	16	38	2	-0.91301
is	you	16	15	1	-0.67662
it	it	34	34	1	-2.2487
it	not	34	12	2	-0.5141
it	to	34	32	7	-0.24217
it	try	34	38	8	-0.28048
it	you	34	15	2	-0.73724
not	not	12	12	1	-0.16579
not	try	12	38	2	-0.62532
not	you	12	15	2	0.304211
to	to	32	32	1	-2.12745
to	try	32	38	8	-0.21986
to	you	32	15	1	-1.36977
try	try	38	38	1	-2.47115
try	you	38	15	2	-0.84847

版权所有 © 华为云计算技术有限公司

7.5.3.6.9 关键词抽取

概述

承接分词结果，获取各个文档中的关键词。

原理

该算法基于TextRank，依据的PageRank算法思想，将滑动窗口内的共现词汇对儿当做相连接的节点构建网络，计算节点的价值（即单词的重要性）并排序，数值高的单词即为该文本的关键词。

TextRank公式如下，其中V_i、V_j为网络中的节点（即单词），In(V_i)表示节点V_i的所有入点，Out(V_j)表示节点V_j的全部出点，Out(V_j)表示节点V_j的所有出点（跳转指向的点，即下一个单词），N_ji表示(V_j, V_i)的个数，S(V_i)、S(V_j)表示节点V_i、V_j的价值，d为阻尼系数，默认为0.85。

$$S(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{N_{ji}}{\sum_{V_k \in Out(V_j)} N_{jk}} * S(V_j)$$

本算法基于pagerank思想，将共现词对儿AB的两条边(A,B)(B,A)添加进网络，相同元素的共现对儿不重复添加，(A,A)自指向共现对儿不添加。

输入

参数	子参数	参数说明
inputs	input_table	输入的包含分词后句子的数据表；必填

输入参数说明

参数名称	参数描述	参数要求
doc_id_col	文章id列	string类型；必填
doc_content	分词后的文本列	string类型；必填；多列时每列当做单独的句子处理
doc_content_sep	分词列中的词分隔符	string类型；必填；默认为" "
window_size	滑动窗口大小	integer类型；非必填；默认为整行，取值范围[1, 2147483647]
dumping_factor	TextRank算法的阻尼系数	double类型；非必填；默认0.85，取值范围(0, 1)

版权所有 © 华为云计算技术有限公司

参数名称	参数描述	参数要求
max_iter	TextRank算法的最大迭代次数	integer类型；非必填；默认100，取值范围[1, 5000]
epsilon	TextRank算法的收敛残差阈值	double类型；非必填；默认0.000001，取值范围(0.000001, 1)

说明

该算子直接承接分词的结果，无过滤停用词、过滤低频词等操作。

会过滤掉doc_id_col/doc_content为空的行。

输出

参数	子参数	参数说明
output	output_port_1	输出表表名；标签为dataframe

输出表说明

列名	列名描述
docId	文章id
keywords	关键词
weight	关键词权重

样例

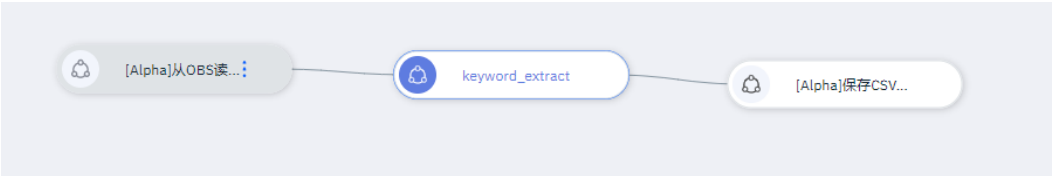
数据输入

id	text
1	A B C A A A A B D E C B B A A D E C F A F B E
2	O O P X O Y O Z Z Z X X Y O X X O Y Y
3	O O P X O
4	O O P X O Y

配置流程

版权所有 © 华为云计算技术有限公司

运行流程



输入参数

参数:

* doc_id_col:	id
* doc_content :	text
* doc_content_sep :	
window_size:	2
doc_content_sep:	0.85
max_iter:	100
epsilon:	0.000001
top_n:	5

输出结果

id	keywords	weight
1	B	0.220406
1	A	0.17985
1	C	0.17985
1	D	0.140494
1	E	0.140494
2	O	0.277862
2	X	0.277862

版权所有 © 华为云计算技术有限公司

2	P	0.148092
2	Y	0.148092
2	Z	0.148092
3	O	0.333333
3	P	0.333333
3	X	0.333333
4	O	0.366736
4	P	0.245928
4	X	0.245928
4	Y	0.141408

7.5.3.6.10 原子分词

概述

对文本数据进行分词。

该算法基于HanLP，对文本列进行分词，标注词性，并支持识别实体、机构、人名、电话号码、中英文日期、中英文时间，过滤全符号、全英文或全数字结果等，自定义词典或自定义合并的词词性标注为"nz"。

输入

参数	子参数	参数说明
inputs	input_table	输入表表名
inputs	dict_table	自定义词典表；非必选

输入参数说明

参数名称	参数描述	参数要求
input_cols	用于分词的列名	string类型；必填；可支持多列
input_cols_sep	多列分词列名分隔符	string类型；必填；默认为","
output_sep	输出表分词列分词分隔符	string类型；必填；默认为" "
remain_other_cols	输出是否保留分词列外的其他列	boolean类型；必填；默认为False

版权所有 © 华为云计算技术有限公司

参数名称	参数描述	参数要求
dict_col	词典表的word列	string类型；非必填；一列的一行代表一个词
enable_ent	是否识别简单实体	boolean类型；非必填；默认为True
enable_person	是否识别人名	boolean类型；非必填；默认为True
enable_org	是否识别机构名	boolean类型；非必填；默认为True
enable_pos	是否进行词性标注	boolean类型；非必填；默认为False
pos_sep	词性标注与单词分隔符	string类型；非必填；默认为"/"；不得与output_sep重复
enable_tel	是否识别电话号码	boolean类型；非必填；默认为True
enable_time	是否识别时间	boolean类型；非必填；默认为True
enable_date	是否识别日期	boolean类型；非必填；默认为True
enable_chn_time	是否识别中文时间	boolean类型；非必填；默认为True
enable_chn_date	是否识别中文日期	boolean类型；非必填；默认为True
filter_all_punctuation	是否过滤分词结果全为标点符号的词	boolean类型；非必填；默认为False
filter_all_en	是否过滤分词结果为全英文的词	boolean类型；非必填；默认为False
filter_all_num	是否过滤分词结果为全数字的词	boolean类型；非必填；默认为False

输出

参数	子参数	参数说明
output	output_port_1	输出表表名；标签为dataframe

版权所有 © 华为云计算技术有限公司

输出表说明

列名	列描述	备注
xxx	原输入表列	如果remain_other_cols为True，则保留input_table全列；如果为False，则只保留input_table分词列
xxx_words	分词列的分词结果列	分词结果列列名为原分词列列名 + "_words"

说明

dict_table是对默认词典的增加，不是只保留dict_table里的词；

自定义词典或自定义合并的词词性标注为"nz"；

暂不支持中文繁体简体混合的时间日期，如“陆月贰十日”。

样例

数据输入

input_table

id	sentence_en	sentence_chn
1	You can call me at 0513-1323563	我的电话是18812534124，不要打0732-1324-5634
2	It's a wrong telephone number like 071-2341. You can do it	这个电话号码1032-122233是错误的
3	It's a wrong telephone number like 12345678901	这也是个错误电话号码1236452123
4	We will go there at 13 past 10 o'clock.	我们将要在十点钟零十三分钟到那里
5	you can go there at eleven to twelve o'clock.	你可以在十一点四十九分出发
6	you can go there 52 past 20 o'clock.	你可以在二十点五十二分去
7	before 12 we finish it.	我们要在十二点之前完成它
8	We will finish it at 20:13:22	我们将在20点13分22秒完成它

版权所有 © 华为云计算技术有限公司

9	We will finish it at 13:22	我们将在十三点二十二分完成
10	We will finish it at 60:22	我们将在六十点二十二分完成（这是个错误时间）
11	aaa cafd February 1，2022 xxx	当地时间12月7日
12	dfsad February, 2022	我吃了面包在九八年五月二十八号。
13	dafdsa Feb 1st, 2nd, 3rd, 4th	1936年 3月零九日，02月八日, 1830年,
14	dsafd Feb 1	一八二五年, 三月，12月， 在九号，在五日, 公元前一五三年
15	Feb 1 in 2022	二十年后，我不知道在哪里
16	11th August, 2020	去年12月28日，我去了兰州
17	1st in/of Feb	凌晨时候我醒了一次，然后到二月拾日
18	1 in Feb in 2022	人不能，至少不应该...【陆月贰十五日记】
19	I make 2022/01/08	二零二二年一月零八日，我开始了远行
20	I make 01/08/2022	一月八日，这个活动终止了，在二〇二二年。
21	I make 2203-01-20	贰贰零叁年一月二十日，游戏内测，五月三十日，正式上线。
27	I make 2203-01-20	贰零二三年一月二十日，游戏内测，五月三十日，正式上线。
22	will you 2019.01.30	这是个正确的时间2019年01月30日
23	I got it 1988 31 05	不想写了1988年 31号 05月
24	I got it 1988.02.33	怎么还有1988年.02月.33日
25	I got it 30/05	这个是30号/05月
26	I got it 05. 12(you need know it)	这是最后一个，05月. 12号

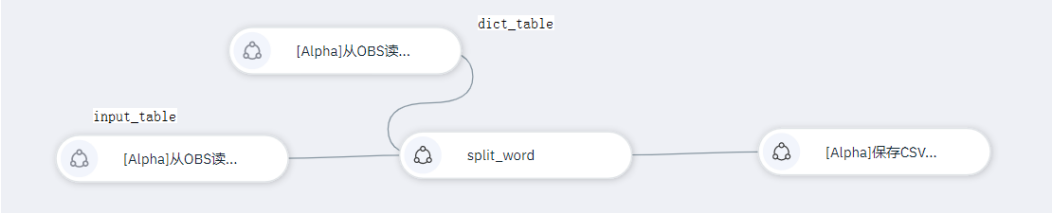
版权所有 © 华为云计算技术有限公司

dict_table

word	id
游戏内测	0
电话号码	1
正式上线	2
二十年后	3
	4
ok	5

配置流程

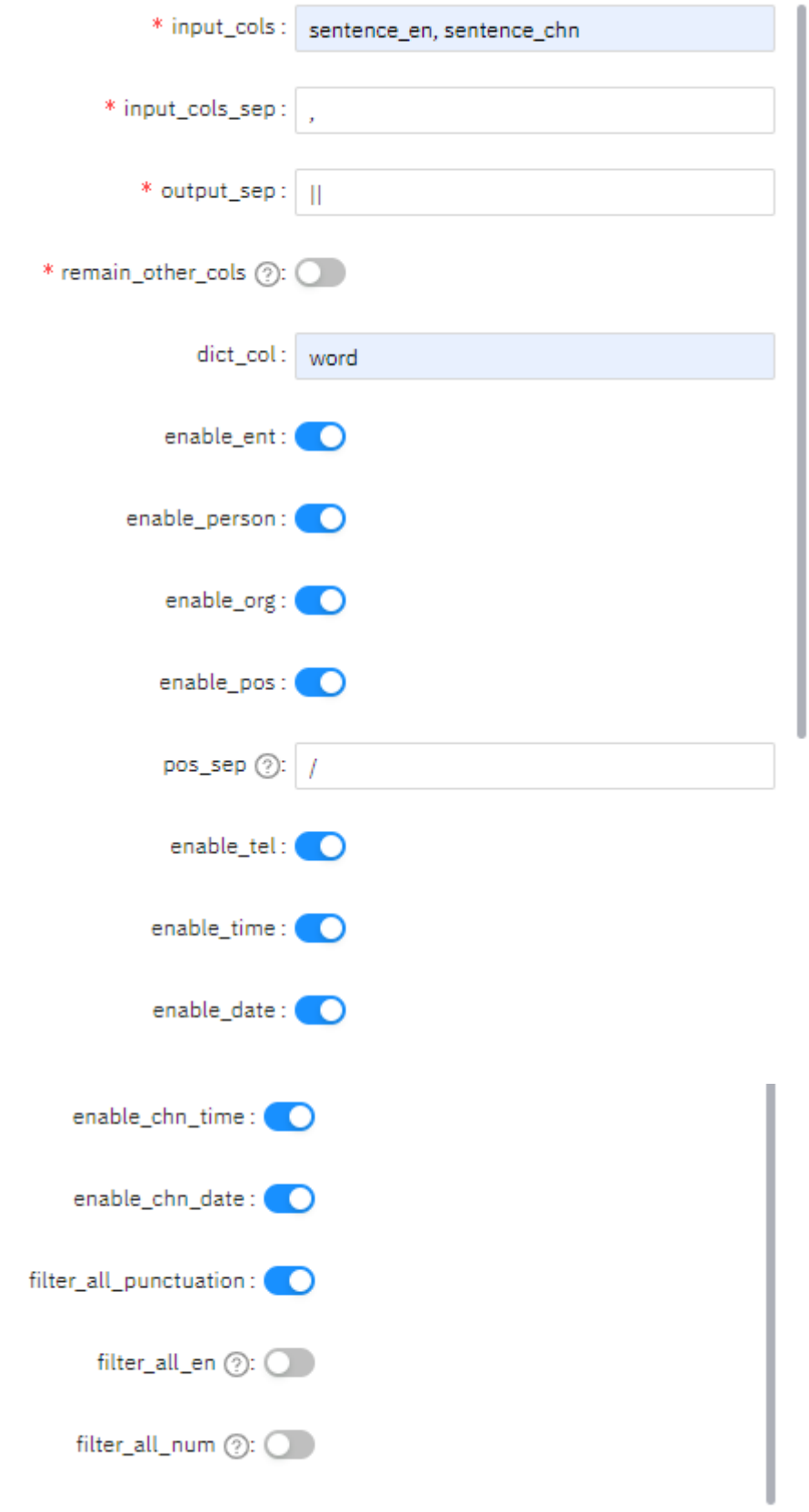
运行流程



输入参数

版权所有 © 华为云计算技术有限公司

参数:



 版权所有 © 华为云计算技术有限公司

输出结果

sentence_en	sentence_chn	sentence_en_words	sentence_chn_words
You can call me at 0513-1323563	我的电话是18812534124，不要打0732-1324-5634	You/nx||can/nx||call/nx||me/nx||at/nx||0513-1323563/nz	我/r||的/uj||电话/n||是/v||18812534124/nz||不/d||要/v||打/v||0732-1324-5634/nz
It's a wrong telephone number like 071-2341. You can do it	这个电话号码1032-122233是错误的	It's/nx||a/nx||wrong/nx||telephone/nx||number/nx||like/nx||071/m||2341/m||You/nx||can/nx||do/nx||it/nx	这个/r||电话号码/nz||1032/m||122233/m||是/v||错误/n||的/uj
It's a wrong telephone number like 12345678901	这也是个错误电话号码1236452123	It's/nx||a/nx||wrong/nx||telephone/nx||number/nx||like/nx||12345678901/m	这/r||也/d||是/v||个/q||错误/n||电话号码/nz||1236452123/m
We will go there at 13 past 10 o'clock.	我们将要在十点钟零十三分钟到那里	We/nx||will/nx||go/nx||there/nx||at 13 past 10 o'clock/nz	我们/r||将要/d||在/p||十点钟零十三分钟/nz||到/v||那里/r
you can go there at eleven to twelve o'clock.	你可以在十一点四十九分出发	you/nx||can/nx||go/nx||there/nx||at eleven to twelve o'clock/nz	你/r||可以/v||在/p||十一点四十九分/nz||出发/v
you can go there 52 past 20 o'clock.	你可以在二十点五十二分去	you/nx||can/nx||go/nx||there/nx||52 past 20 o'clock/nz	你/r||可以/v||在/p||二十点五十二分/nz||去/v
before 12 we finish it.	我们要在十二点之前完成它	before 12/nz||we/nx||finish/nx||it/nx	我们/r||要/v||在/p||十二点/nz||之前/f||完成/v||它/r

版权所有 © 华为云计算技术有限公司

We will finish it at 20:13:22	我们将在20点13分22秒完成它	We/nx||will/nx||finish/nx||it/nx||at 20:13:22/nz	我们/r||将/d||在/p||20点13分22秒/nz||完成/v||它/r
We will finish it at 13:22	我们将在十三点二十二分完成	We/nx||will/nx||finish/nx||it/nx||at 13:22/nz	我们/r||将/d||在/p||十三点二十二分/nz||完成/v
We will finish it at 60:22	我们将在六十点二十二分完成（这是个错误时间）	We/nx||will/nx||finish/nx||it/nx||at/nx||60/m||22/m	我们/r||将/d||在/p||六十/m||点/q||二十二/m||分/q||完成/v||这/r||是/v||个/q||错误/n||时间/n
aaa cafd February 1，2022 xxx	当地时间12月7日	aaa/nx||cafd/nx||February 1，2022/nz||xxx/nx	当地/s||时间/n||12月7日/nz
dfsad February, 2022	我吃了面包在九八年五月二十八号。	dfsad/nx||February, 2022/nz	我/r||吃/v||了/ul||面包/n||在/p||九八年五月二十八号/nz
dafdsa Feb 1st, 2nd, 3rd, 4th	1936年 3月零九日，02月八日, 1830年,	dafdsa/nx||Feb 1st/nz||2/m||nd/nx||3/m||rd/nx||4/m||th/nx	1936年 3月零九日/nz||02月八日/nz||1830年/nz
dsafd Feb 1	一八二五年, 三月，12月， 在九号，在五日, 公元前一五三年	dsafd/nx||Feb 1/nz	一八二五年, 三月/nz||12月/nz||在/p||九号/nz||在/p||五/m||日/j||公元前一五三年/nz
Feb 1 in 2022	二十年后，我不知道在哪里	Feb 1 in 2022/nz	二十年后/nz||我/r||不/d||知道/v||在/p||哪里/r
11th August, 2020	去年12月28日，我去了兰州	11th August, 2020/nz	去年/nz||12月28日/nz||我/r||去/v||了/ul||兰州/ns
1st in/of Feb	凌晨时候我醒了一次，然后到二月拾日	1/m||st/nx||in/nx||of Feb/nz	凌晨/nz||时候/n||我/r||醒/v||了/ul||一/m||次/q||然后/c||到/v||二月拾日/nz
1 in Feb in 2022	人不能，至少不应该...【陆月贰十五日记】	1 in Feb in 2022/nz	人/n||不能/v||至少/d||不/d||应该/v||陆月/nr||贰十五/m||日记/n
I make 2022/01/08	二零二二年一月零八日，我开始了远行	I/nx||make/nx||2022/01/08/nz	二零二二年一月零八日/nz||我/r||开始/v||了/ul||远行/vn

版权所有 © 华为云计算技术有限公司

I make 01/08/2022	一月八日，这个活动终止了，在二〇二二年。	I/nx||make/nx||01/08/2022/nz	一月八日/nz||这个/r||活动/vn||终止/v||了/ul||在/p||二〇二二年/nz
I make 2203-01-20	贰贰零叁年一月二十日，游戏内测，五月三十日，正式上线。	I/nx||make/nx||2203-01-20/nz	贰贰零叁年一月二十日/nz||游戏内测/nz||五月三十日/nz||正式上线/nz
I make 2203-01-20	贰零二三年一月二十日，游戏内测，五月三十日，正式上线。	I/nx||make/nx||2203-01-20/nz	贰零二三年一月二十日/nz||游戏内测/nz||五月三十日/nz||正式上线/nz
will you 2019.01.30	这是个正确的时间2019年01月30日	will/nx||you/nx||2019.01.30/nz	这/r||是/v||个/q||正确/a||的/uj||时间/n||2019年01月30日/nz
I got it 1988 31 05	不想写了1988年 31号 05月	I/nx||got/nx||it/nx||1988 31 05/nz	不想/v||写/v||了/ul||1988年 31号 05月/nz
I got it 1988.02.33	怎么还有1988年.02月.33日	I/nx||got/nx||it/nx||1988.02.33/nz	怎么/r||还有/v||1988/m||年/q||02/m||月/q||33/m||日/j
I got it 30/05	这个是30号/05月	I/nx||got/nx||it/nx||30/05/nz	这个/r||是/v||30号/05月/nz
I got it 05. 12(you need know it)	这是最后一个，05月. 12号	I/nx||got/nx||it/nx||05. 12/nz||you/nx||need/nx||know/nx||it/nx	这/r||是/v||最后/f||一个/mq||05月. 12号/nz

7.5.3.6.11 文本 TF-IDF

概述

文本TF-IDF是一种统计方法，用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。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出现的频率成反比下降。文本TF-IDF用于展示文本基于词频统计的输出，经TF-IDF加权的结果。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象，一般为词频统计后的结果。

版权所有 © 华为云计算技术有限公司

输出

参数	子参数	参数说明
output	output_port_1	output为字典类型，output_port_1为pyspark中的DataFrame类型对象，为TF-IDF的结果。

参数说明

参数	是否必选	参数说明	默认值
id_col	是	标识文章ID的列名，仅可指定一列	"id"
word_col	是	word列名，仅可指定一列	"word"
count_col	是	count列名，仅可指定一列	"count"
doc_count_col	否	指定doc_count列名	"doc_count"
total_word_count_col	否	指定total_word_count列名	"total_word_count"
total_doc_count_col	否	指定total_doc_count列名	"total_doc_count"
tf_col	否	指定TF列名	"tf"
idf_col	否	指定IDF列名	"idf"
tfidf_col	否	指定TF-IDF列名	"tfidf"

样例

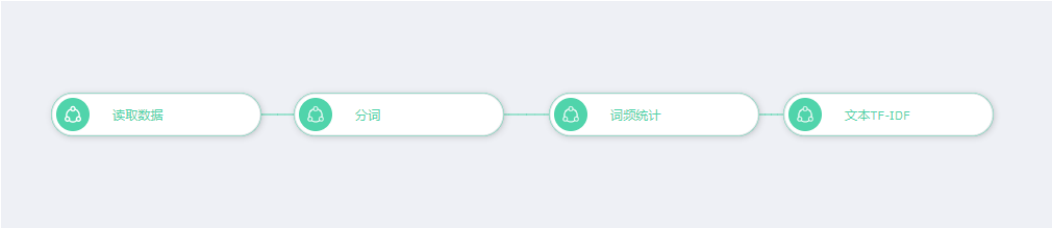
数据样本

id	sentence
1	ball ball fun planet galaxy
2	referendum referendum fun planet planet
3	planet planet planet galaxy ball
4	planet galaxy planet referendum ball

版权所有 © 华为云计算技术有限公司

配置流程

运行流程



参数设置

参数:

* id_col : id

* word_col : word

* count_col : count

doc_count_col : doc_count

total_word_count_col : total_word_count

total_doc_count_col : total_doc_count

tf_col : tf

idf_col : idf

tfidf_col : tfidf

结果查看

id	word	count	doc_count	total_word_count	total_doc_count	tf	idf	tfidf
1	galaxy	1	3	5	4	0.2	0.223144	0.044629

版权所有 © 华为云计算技术有限公司

id	word	count	doc_count	total_word_count	total_doc_count	tf	idf	tfidf
1	fun	1	2	5	4	0.2	0.510826	0.102165
1	ball	2	3	5	4	0.4	0.223144	0.089257
1	planet	1	4	5	4	0.2	0	0
2	fun	1	2	5	4	0.2	0.510826	0.102165
2	planet	2	4	5	4	0.4	0	0
2	referendum	2	2	5	4	0.4	0.510826	0.204333
3	ball	1	3	5	4	0.2	0.223144	0.044629
3	planet	3	4	5	4	0.6	0	0
3	galaxy	1	3	5	4	0.2	0.223144	0.044629
4	ball	1	3	5	4	0.2	0.223144	0.044629
4	planet	2	4	5	4	0.4	0	0
4	galaxy	1	3	5	4	0.2	0.223144	0.044629
4	referendum	1	2	5	4	0.2	0.510826	0.102165

7.5.3.6.12 三元组转 kv

概述

三元组转kv，用于将三元组表转换为kv表，三元组表为(row, key, value)的形式，kv表为(row, [key_id:value])的形式，同时会生成关于(key, key_id)的表格。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象，一般为三元组表

版权所有 © 华为云计算技术有限公司

输出

参数	子参数	参数说明
output	key_value	output为字典类型，key_value为输出的kv表
	key_id	output为字典类型，key_id为输出的key的id对照表

参数说明

参数	是否必选	参数说明	默认值
keep_col	是	不改变的列	无
key_col	是	key列	无
value_col	是	value列	无
key_value_delimiter	否	key与value间的分隔符	:
pair_delimiter	否	key-value对之间的分隔符	;

样例

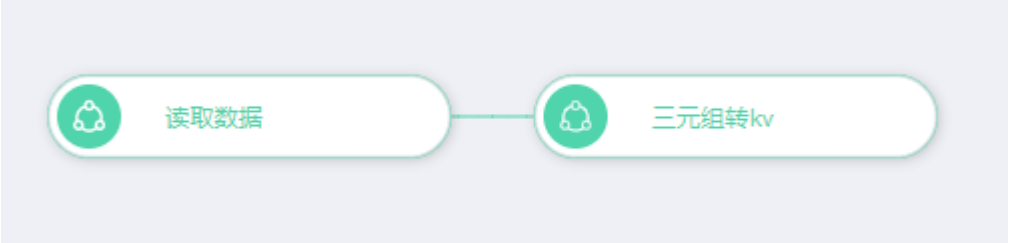
数据样本

idx	word	val
1	a	1
1	b	21
1	c	3
1	d	5
2	a	6
2	c	7
2	e	8
3	a	9
3	c	8

配置流程

版权所有 © 华为云计算技术有限公司

运行流程



参数设置

查看结果

word	key_id
d	1
e	2
c	3
a	4
b	5

idx	key_value
1	3:3;1:5;5:21;4:1
2	3:7;2:8;4:6
3	4:9;3:8

版权所有 © 华为云计算技术有限公司

7.5.3.6.13 文本分类

概述

文本分类通过TF-IDF和多项式朴素贝叶斯进行文本分类，以原始文本和标签作为输入，输出文本分类模型。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象。如果文本为中文则需要先以空格为分隔符对原始文本进行分词。

输出

参数	子参数	参数说明
output	output_port_1	output为字典类型，output_port_1为pyspark中的PipelineModel模型类型。

参数说明

参数	是否必选	参数说明	默认值
sentence_col	是	文本列	"sentence"
label_col	是	标签列（标签值需整数或浮点型）	"label"
words_col	否	用于分词后保存words的列名	"words"
feature_col	否	用于保存feature的列名	"features"
min_doc_freq	否	最小词数阈值	0
smoothing	否	平滑指数	1.0
prediction_col	否	用于保存prediction的列名	"prediction"
raw_prediction_col	否	用于保存raw_prediction的列名	"raw_prediction"

版权所有 © 华为云计算技术有限公司

参数	是否必选	参数说明	默认值
probability_col	否	用于保存probability的列名	"probability"
raw_features_col	否	用于保存raw_features的列名	"raw_features"
tf_num_features	否	tf-idf时用于保存的词的数量，建议不小于词汇种类	8000

样例

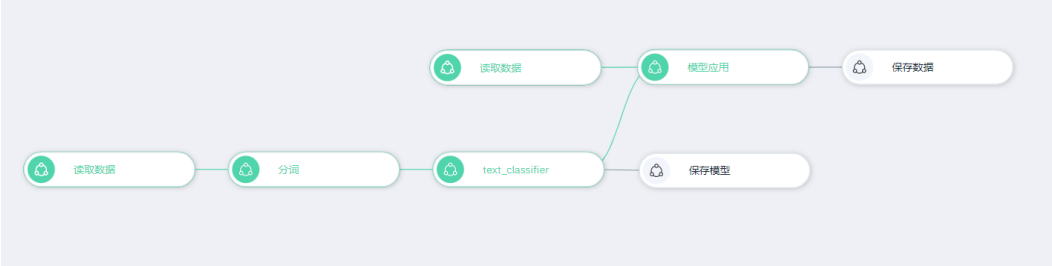
数据样本

id	sentence	label
1	ball ball fun planet galaxy	1
2	referendum referendum fun planet planet	0
3	planet planet planet galaxy ball	1
4	planet galaxy planet referendum ball	1

sentence
ball ball ball
referendum referendum
planet planet ball

配置流程

运行流程



参数设置

版权所有 © 华为云计算技术有限公司

设置参数

* sentence :	sentence
* label_col :	label
words_col :	words
feature_col :	features
min_doc_freq :	0
smoothing :	1
prediction_col :	prediction
raw_prediction_col :	rawPrediction
probability_col :	probability
raw_features_col :	rawFeatures
tf_num_features :	8000

结果查看

sentence	words	rawFeatures	features	rawPrediction	probability	prediction
ball ball ball	['ball', 'ball', 'ball']	(8000, [5492], [3.0])	(8000, [5492], [0.6694306539426294])	[-7.1150455570283399,-5.99493111918993 55]	[0.24599005712406302,0.754009942875 9369]	1
referendum referendum	['referendum', 'referendum']	(8000, [999], [2.0])	(8000, [999], [1.021651247531 9814])	[-9.5614335641019 23,-9.165 98505271 9044]	[0.40240637346162 5,0.597593 6265383749]	1

版权所有 © 华为云计算技术有限公司

sentence	words	rawFeatures	features	rawPrediction	probability	prediction
planet planet ball	['planet', 'planet', 'ball']	(8000, [5492,6309], [1.0,2.0])	(8000, [5492,6309], [0.2231435513142097 6,0.0])	[-3.1040900447882 06,-2.26862044513 54213]	[0.3024897957164007,0.6975102042835993]	1

7.5.3.6.14 LDA

概述

LDA主题分析模型(Latent Dirichlet Allocation)，由Blei等人于2003年提出的无监督学习算法，可以按照概率分布的形式给出文档集中每篇文档的主题，在文本挖掘领域，应用于文本主题识别、文本分类和文本相似度计算等方面。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象。如果文本为中文则需要先以空格为分隔符对原始文本进行分词。

输出

参数	子参数	参数说明
output	P(Z)	主题概率
	P(Z|D)	主题-文档概率
	P(D|Z)	文档-主题概率
	P(Z|W)	主题-词汇概率
	vocab	词汇表

版权所有 © 华为云计算技术有限公司

参数说明

参数	是否必选	参数说明	默认值
sentence_col	是	文本列	"sentence"
topics_k	是	主题数目>=2	2
min_doc_freq	否	最小词数阈值	0
words_col	否	分词后的words列	"words"
feature_col	否	features列	"features"
raw_features_col	否	raw features列	"rawFeatures"
topic_distribution_col	否	topic distribution列	"topicDistributionCol"
max_iter	是	最大迭代次数	50
idf_or_not	否	是否使用idf	False
topic_concentration	是	超参数\eta	1.1
doc_concentration	是	超参数\alpha	1.1

样例

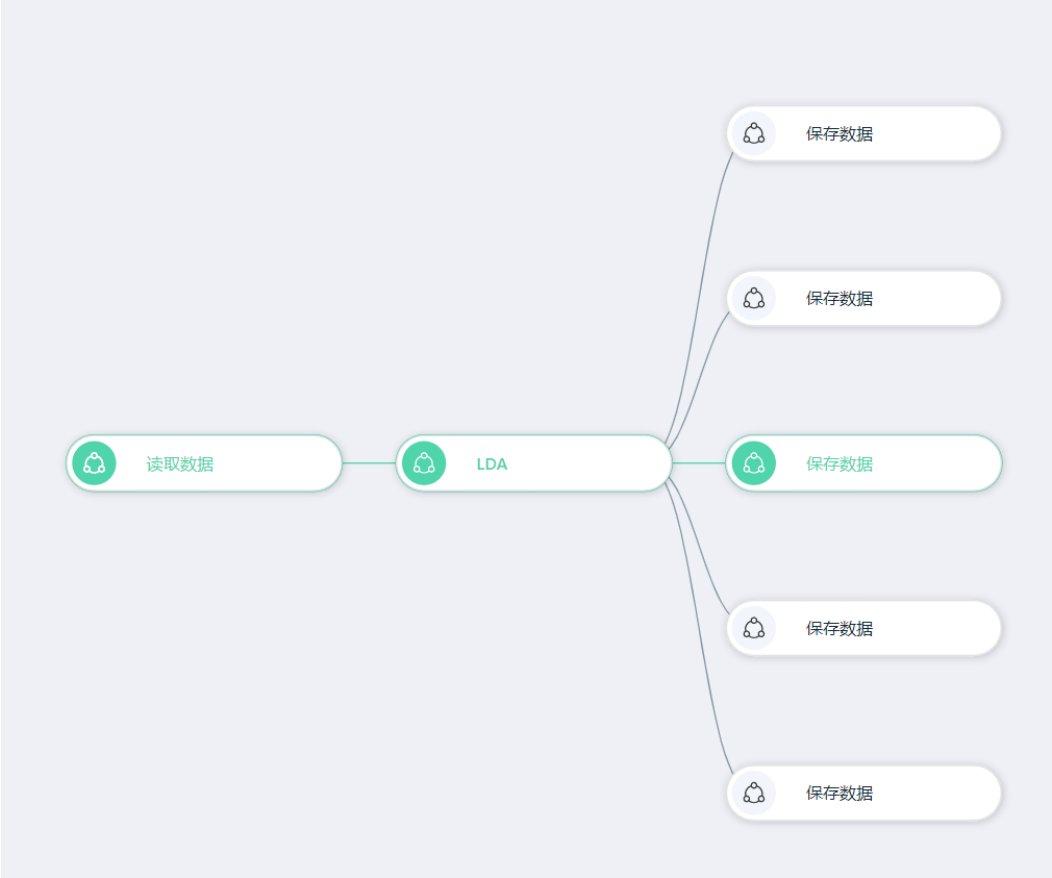
数据样本

id	sentence
1	ball ball fun planet galaxy
2	referendum referendum fun planet planet
3	planet planet planet galaxy ball
4	planet galaxy planet referendum ball

配置流程

运行流程

版权所有 © 华为云计算技术有限公司



参数设置

版权所有 © 华为云计算技术有限公司

设置参数

* sentence_col :	sentence
* topics_num_K :	3
min_doc_freq :	0
words_col :	words
feature_col :	features
raw_features_col :	rawFeatures
topic_distribution_col :	topicDistributionCol
* max_iter :	50
idf_or_not :	False
* topic_concentration(alp... :	1.1
* doc_concentration(beta) :	1.1

结果查看

P(Z)

	pz
1	0.489148687908443
2	0.29924603002570965
3	0.5449386153991806

P(Z|D)

	sentence_id	topic_0	topic_1	topic_2
1	1	0.18101253082941773	0.2580217769779673	0.5609656921926149
2	2	0.4011642489965051	0.1597167187653878	0.4391190322381071
3	3	0.37673886172691773	0.13667498421719568	0.4865861540558866
4	4	0.5085304221724884	0.3433246101165781	0.14814496771093333

版权所有 © 华为云计算技术有限公司

P(D|Z)

	sentence_id	topic_0	topic_1	topic_2
1	1	0.3700562534541624	0.86223959915458	1.0294107929600365
2	2	0.8201274150644221	0.5337304516676988	0.8058137555850063
3	3	0.7701929311878974	0.456731152642036	0.8929192028343423
4	4	1.0396234002935183	1.147298796535685	0.27185624862061497

P(Z|W)

	topic_0	topic_1	topic_2
1	0.633130951205548	0.05938280084852243	0.3074862479459296
2	0.028690480495170326	0.021644457970588137	0.9496650615342416
3	0.6681824149974718	0.31362525160543436	0.01819233339709393
4	0.06899535336534156	0.019379173429501042	0.9116254732051573
5	0.029385832608896118	0.8977773875716824	0.07283677981942145

vocab

	words
1	planet
2	ball
3	referendum
4	galaxy
5	fun

7.5.3.6.15 句子拆分

概述

对文本数据按照标点符号进行句子拆分。

该算法按照既定标点符号等进行句子拆分，并将标点符号保留在句末（给定标点符号不单独成行），一篇文章拆分成多行输出。

输入

参数	子参数	参数说明
inputs	input_table	输入表表名

版权所有 © 华为云计算技术有限公司

输入参数说明

参数名称	参数描述	参数要求
doc_id_col	标识文章的id列	string类型；必填；仅支持一列
doc_content	用于句子拆分的列名	string类型；必填；仅支持一列
delimiter	用于拆分句子的标点符号	string类型；必填；默认为"，,。：“”？?！!；;"

输出

参数	子参数	参数说明
output	output_port_1	输出表表名；标签为dataframe

输出表说明

列名	列名描述	备注
xx	非句子拆分列	保持原列名和内容
xx	句子拆分列	原列名和拆分后的单个句子

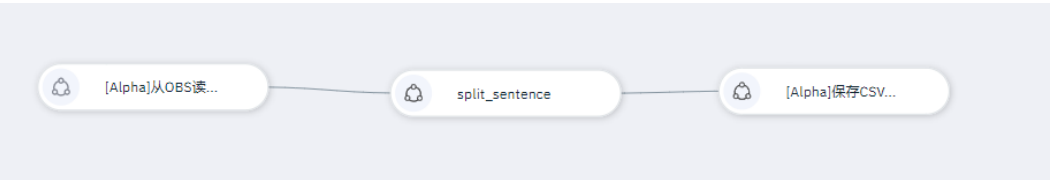
样例

数据输入

id	text
1	在周六的早晨，我喜欢煎个鸡蛋，煮一杯红底薏米粥，窝在榻榻米中慢慢苏醒。
2	在一九一三年的五月，这座边陲小镇迎来了带领他们走向未来一百年繁华的人，这一天注定不平凡。

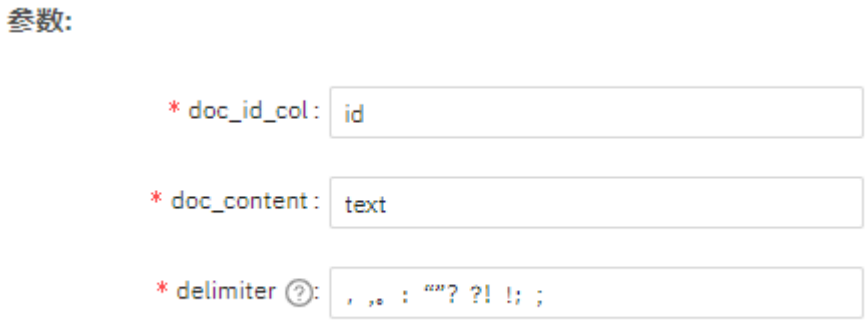
配置流程

运行流程



版权所有 © 华为云计算技术有限公司

输入参数



输出结果

id	text
1	在周六的早晨，
1	我喜欢煎个鸡蛋，
1	煮一杯红底薏米粥，
1	窝在榻榻米中慢慢苏醒。
2	在一九一三年的五月，
2	这座边陲小镇迎来了带领他们走向未来一百年繁华的人，
2	这一天注定不平凡。

7.5.3.6.16 文本摘要

概述

抽取文本中的部分原句作为文本的摘要。

该算法按照既定标点符号等进行句子拆分，基于TextRank思想求出可代表该文档的句子作为其摘要。

输入

参数	子参数	参数说明
inputs	input_table	输入表表名

版权所有 © 华为云计算技术有限公司

输入参数说明

参数名称	参数描述	参数要求
doc_id_col	标识文章的id列	string类型；必填；仅支持一列
sentence_col	原文本列列名	string类型；必填；仅支持一列
sentence_sep	用于拆分句子的标点符号	string类型；必填；默认为"，,。:: “ ”？?！!；;"
top_n	输出的摘要句子个数	integer类型；必填；默认为3
remain_other_cols	是否保留id列和原文本列外的其他列	boolean类型；必填；默认为False

输出

参数	子参数	参数说明
output	output_port_1	输出表表名；标签为dataframe

输出表说明

列名	列描述	备注
xxx	原输入表列	如果remain_other_cols为True，则保留input_table全列；如果为False，则只保留input_table原文本列
xxx_summary	原文本列的文本摘要结果列	摘要结果列列名为原文本列列名 + "_summary"

说明

输出摘要句子依据权重组合，并未保留在原文中顺序。

样例

数据输入

id	text

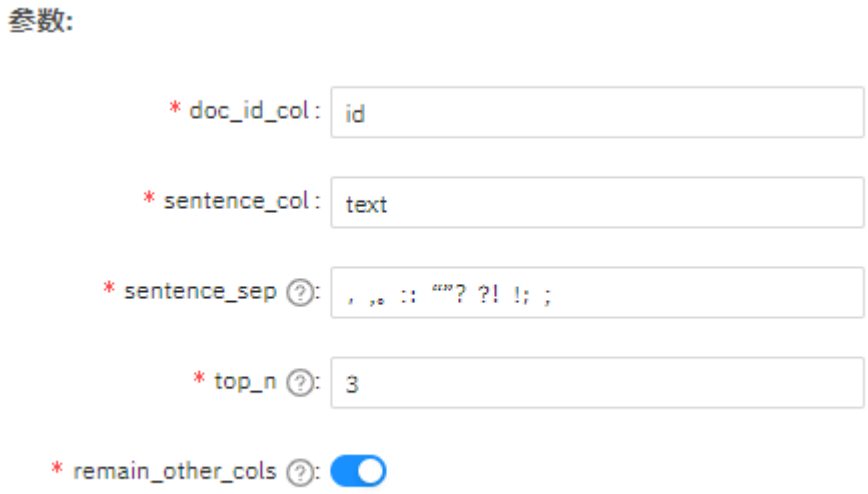
版权所有 © 华为云计算技术有限公司

1	荷兰国家旅游会议促进局亚洲区总监、中国区首席代表杨宇对《环球时报》记者表示，未来重启的中国旅游市场会更加细分，也会出现更多新的旅行需求。
2	此外，国外的酒店从业者也非常期待中国游客“回归”。禧亚酒店及度假村集团大中华区品牌负责人张章告诉记者，作为马尔代夫最大的酒店集团之一，禧亚酒店及度假村集团旗下的5家岛屿度假酒店都已重启中国管家计划，重新召回中国籍宾客关系服务人员、中文管家、中国厨师等。

配置流程

运行流程

输入参数



输出结果

id	text	text_summary
1	荷兰国家旅游会议促进局亚洲区总监、中国区首席代表杨宇对《环球时报》记者表示，未来重启的中国旅游市场会更加细分，也会出现更多新的旅行需求。	未来重启的中国旅游市场会更加细分，荷兰国家旅游会议促进局亚洲区总监、中国区首席代表杨宇对《环球时报》记者表示，也会出现更多新的旅行需求。

版权所有 © 华为云计算技术有限公司

2	此外，国外的酒店从业者也非常期待中国游客“回归”。禧亚酒店及度假村集团大中华区品牌负责人张章告诉记者，作为马尔代夫最大的酒店集团之一，禧亚酒店及度假村集团旗下的5家岛屿度假酒店都已重启中国管家计划，重新召回中国籍宾客关系服务人员、中文管家、中国厨师等。	禧亚酒店及度假村集团旗下的5家岛屿度假酒店都已重启中国管家计划，禧亚酒店及度假村集团大中华区品牌负责人张章告诉记者，作为马尔代夫最大的酒店集团之一，

7.5.3.6.17 停用词过滤

概述

停用词过滤是自然言语处理中一个重要的步骤。它可以将句子中的噪声词，和一些无关词（通常由用户指定）过滤掉。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象。里面存放的是待过滤的语句
inputs	noise_dataframe	inputs为字典类型，noise_dataframe为pyspark中的DataFrame类型对象。里面存放的用户指定的停用词

输出

参数	子参数	参数说明
output	output_port_1	dataframe类型的过滤后的结果

参数说明

参数	是否必选	参数说明	默认值
selected_filter_column	是	需要过滤停用词的字段名称	""
noise_data_column	是	停用词所在的字段名称	""

版权所有 © 华为云计算技术有限公司

参数	是否必选	参数说明	默认值
segment_output_delimiter	是	需要过滤停用词字段内部的分隔符	" "

样例

输入数据-待过滤文本

```
id,sentence
1,停用词 过滤 是 自然言语处理 中 一个 重要 的 步骤 。
```

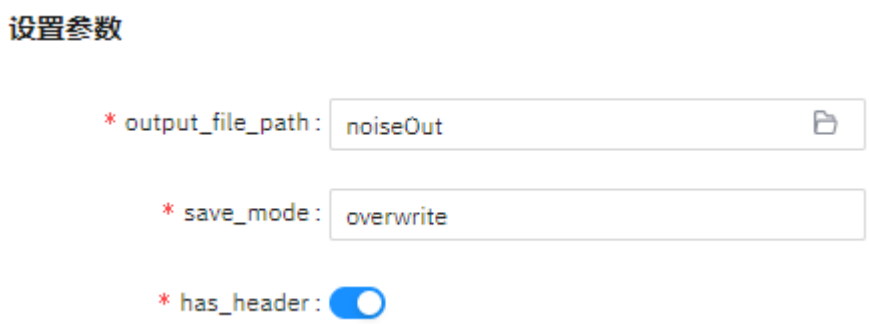
输入数据-停用词

```
noise
。
是
中
一个
```

配置流程

运行流程

算法参数设置



查看结果

```
id,sentence
1,停用词 过滤 自然言语处理 重要 的 步骤
```


版权所有 © 华为云计算技术有限公司

7.5.3.6.18 语义相似距离

概述

计算距离某个向量最近的k个向量集合。这些向量通常是通过算法生产的包含语义的向量（例如word2vec生产的词向量，或者doc2vec生产的文章向量）。可以用于寻找和一个单词或者一篇文章相似的单词或者文章。

输入

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象。用于计算距离每个向量最近的topN个向量

输出

参数	子参数	参数说明
outputs	output_port_1	指向一个pyspark的DataFrame类型对象，该对象中包含距离每个向量最近的topN个向量的结果

参数说明

参数	是否必选	参数说明	默认值
id_col	是	用户id所在的列名	"id"
vector_col	是	向量的列名列表，如col1，col2	""
topn	是	输出的距离最近的向量的数目。取值范围[1,+∞)	20
distance_type	是	距离的计算方式。取值[cosine]	"cosine"
distance_threshold	是	距离的阈值。当两个向量的距离小于此值时输出。取值范围（0,+∞）	1.0
leaf_size	是	叶子节点大小	50

版权所有 © 华为云计算技术有限公司

样例

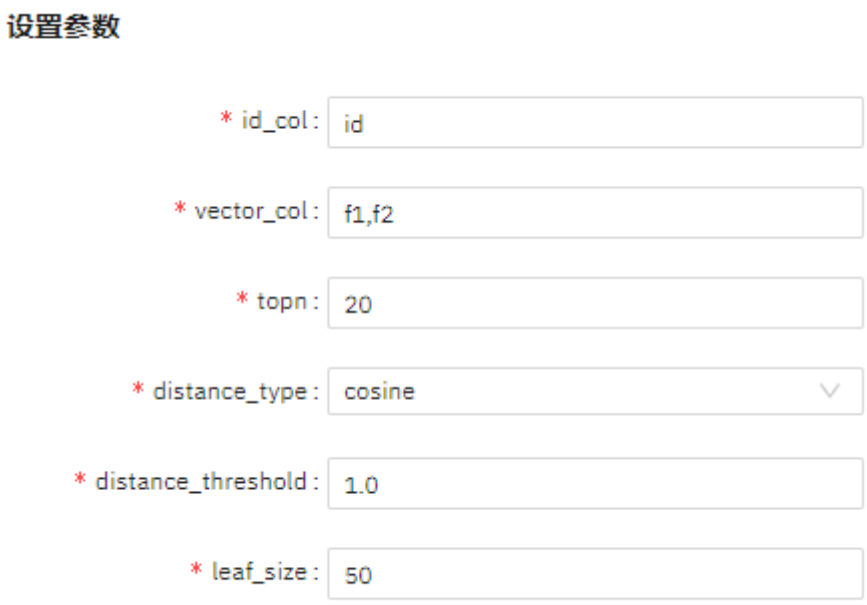
数据样本

```
id,f1,f2
1,0.1,0.2
2,0.3,0.1
3,0.7,0.2
```

配置流程

运行流程

参数设置



查看结果

```
original_id,near_id,distance,rank
3,2,0.0009438416449403242,1
3,1,0.3242753714826536,2
1,2,0.29289321881345254,1
1,3,0.3242753714826536,2
2,3,0.0009438416449403242,1
2,1,0.29289321881345254,2
```

7.5.3.7 时间序列

版权所有 © 华为云计算技术有限公司

7.5.3.7.1 ARIMA

概述

ARIMA全称为自回归积分滑动平均模型（Autoregressive Integrated Moving Average model），是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“滑动平均”，q为滑动平均项数，d为使之成为平稳序列所做的差分次数（阶数）。

说明

目前ARIMA算子只支持在Notebook环境运行，不支持DLI环境下运行。

输入

表 7-10

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象，一般为分词后的结果

输出

表 7-11

参数	子参数	参数说明
output	output_port_1	output为字典类型，output_port_1为pyspark中的DataFrame类型对象，为arima模型预测的结果。

参数说明

表3

参数	是否必选	描述	默认值
seq_col_name	是	时序列。仅用来对valueColName排序。	无
value_col_name	是	数值列。	无

版权所有 © 华为云计算技术有限公司

参数	是否必选	描述	默认值
group_col_names	否	分组列，多列用逗号分隔，例如col0,col1。每个分组会构建一个时间序列。	无
order	是	p、d和q分别表示自回归系数、差分、滑动回归系数。取值均为非负整数，范围为[0, 36]。	无
seasonal	否	sp、sd、sq分别表示季节因素的自回归，差分，移动平均系数，取值均为非负整数，范围为[0, 36]。	0,0,0
period	否	seasonal周期。数字类型，取值范围为(0, 100]。	12
predict_step	否	预测条数。数字类型，取值范围为(0, 365]。	12
confidence_level	否	预测置信水平。数字类型，取值范围为(0, 1)。	0.95

样例

数据样本

使用公开数据集AirPassengers.csv的前120行

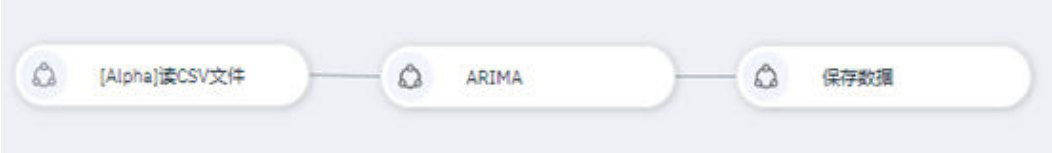
数据示例

```
Month,Passengers
1949-01,112
1949-02,118
1949-03,132
1949-04,129
1949-05,121
1949-06,135
1949-07,148
1949-08,148
1949-09,136
1949-10,119
1949-11,104
1949-12,118
1950-01,115
1950-02,126
1950-03,141
```


版权所有 © 华为云计算技术有限公司

配置流程

运行流程：



算法参数设置：

设置参数

* seq_col_name	Month
* value_col_name	passger
group_col_names	
* order	3,1,1
seasonal	0,0,0
period	12
predict_step	12
confidence_level	0.95

查看结果：

版权所有 © 华为云计算技术有限公司

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                          Passengers   No. Observations:                  119
Model:             SARIMAX(3, 1, 1)x(0, 1, 1, 12)   Log Likelihood                -393.376
Date:                            Wed, 28 Dec 2022   AIC                            798.751
Time:                                    11:17:25   BIC                            814.732
Sample:                                         0   HQIC                           805.228
                                              - 119
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.5342      0.182      2.935      0.003       0.177       0.891
ar.L2          0.1852      0.148      1.249      0.212      -0.106       0.476
ar.L3         -0.1834      0.150     -1.219      0.223      -0.478       0.111
ma.L1         -0.7876      0.214     -3.674      0.000      -1.208      -0.367
ma.S.L12      -0.0776      0.100     -0.775      0.438      -0.274       0.119
sigma2        97.5878     13.736      7.104      0.000      70.665     124.509
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 3.90
Prob(Q):                              0.97   Prob(JB):                         0.14
Heteroskedasticity (H):               1.65   Skew:                            -0.15
Prob(H) (two-sided):                  0.14   Kurtosis:                         3.89
===================================================================================
```

说明

上图所示模型信息不输出到表格中，可在日志中查看。

pdate	forecast	lower	upper
1	347.5533560531952	328.19162914538344	366.915082961007
2	351.7764184099141	327.61316018230735	375.9396766375208
3	332.8913265880475	304.22619513720167	361.5564580388933
4	377.7956130817893	346.9178247499129	408.6734014136658
5	364.8461913375799	332.2481157289312	397.4442669462286
6	379.0863154704068	345.2705335664142	412.90209737439943
7	450.6364517467664	415.70762508073017	485.5652784128026
8	505.4873412995408	469.5308706086275	541.4438119904542
9	518.4523519528448	481.47166547949314	555.4330384261964
10	420.4426508674792	382.45085930699236	458.434442427966
11	374.5553761533195	335.5578396271561	413.5529126794829
12	326.1342855793034	286.1471301790137	366.12144097959316

7.5.3.7.2 Auto ARIMA

概述

该算子可以帮助我们自动确定 A R I M A (p , d , q) (P , D , Q) m。

Auto ARIMA选择过程

版权所有 © 华为云计算技术有限公司

1. 执行差分测试 决定差分d的大小（ KPSS检测和ADF检测 ）

2. 拟合模型: 通过限制start_p、max_p、start_q max_q，在该范围内搜索最优参数；如果启用了季节性可选项，则还会执行Canova-Hansen来确定季节性差分的最佳阶数 D ，并之后基于此确定最佳 P 和 Q 超参数。

3. Auto ARIMA基于给定的information_criterion进行模型优化，范围('aic', 'aicc', 'bic', 'hqic', 'oob')；A并通过生成AIC和BIC值来确定参数的最佳组合。AIC和BIC值是用于比较模型的评估器。这些值越低，模型就越好。

说明

目前 Auto ARIMA算子只支持在Notebook环境运行，不支持DLI环境下运行。

输入

表 7-12

参数	子参数	参数说明
inputs	dataframe	inputs为字典类型，dataframe为pyspark中的DataFrame类型对象，一般为分词后的结果

输出

表 7-13

参数	子参数	参数说明
output	output_port_1	output为字典类型，output_port_1为pyspark中的DataFrame类型对象，为Auto Arima模型预测的结果。

参数说明

表3

参数	是否必选	描述	默认值
seq_col_name	是	时序列。用来对valueColName排序。	无
value_col_name	是	数值列	无

版权所有 © 华为云计算技术有限公司

参数	是否必选	描述	默认值
group_col_names	否	分组列，多列用逗号分隔，如“col0,col1”。每个分组会构建一个时间序列	无
frequency	否	时序频率，正整数，范围为(0, 12]。	12**说明** 12表示12月/年。
max_order	否	p，q最大值，正整数，范围为[0,4]。	2
max_seasonal_order	否	季节性p，q最大值，正整数，范围为[0,2]。	1
max_diff	否	差分d最大值。正整数，范围为[0,2]。	2
max_seasonal_diff	否	季节性差分d最大值。正整数，范围为[0,1]。	1
diff	否	差分d，正整数，范围为[0,2]。**diff与maxDiff同时设置时，maxDiff被忽略。diff与seasonalDiff要同时设置**。	-1**说明** 取值为-1表示不指定diff。
seasonal_diff	否	季节性差分d。正整数，范围为[0,1]。**seasonalDiff与maxSeasonalDiff同时设置时，maxSeasonalDiff被忽略。**	-1**说明** 取值为-1表示不指定seasonalDiff。
max_iter	否	最大迭代次数，正整数	1500
tol	否	容忍度，double类型。	1e-5
predict_step	否	预测条数，数字，范围为(0, 365]。	12
confidence_level	否	预测置信水平，数字，范围为(0, 1)。	0.95

版权所有 © 华为云计算技术有限公司

样例

数据样本

使用公开数据集AirPassengers.csv的前120行

数据示例：

```
Month,Passengers
1949-01,112
1949-02,118
1949-03,132
1949-04,129
1949-05,121
1949-06,135
1949-07,148
1949-08,148
1949-09,136
1949-10,119
1949-11,104
1949-12,118
1950-01,115
1950-02,126
1950-03,141
```

配置流程

运行流程

算法参数设置

版权所有 © 华为云计算技术有限公司

设置参数

* seq_col_name:	Month
* value_col_name:	Passengers
group_col_names:	
frequency:	12
max_order:	2
max_seasonal_order:	1
max_diff:	2
max_seasonal_diff:	1
diff:	-1
seasonal_diff:	-1
predict_step:	12
confidence_level:	0.95
max_iter:	1500

查看结果

版权所有 © 华为云计算技术有限公司

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                                 y   No. Observations:                  119
Model:             SARIMAX(2, 0, 0)x(0, 1, 0, 12)   Log Likelihood                -396.839
Date:                            Wed, 28 Dec 2022   AIC                            801.679
Time:                                    18:54:34   BIC                            812.370
Sample:                                         0   HQIC                           806.013
                                              - 119
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      4.5597      2.054      2.220      0.026       0.533       8.586
ar.L1          0.6701      0.100      6.694      0.000       0.474       0.866
ar.L2          0.1562      0.097      1.614      0.106      -0.033       0.346
sigma2        96.5155     11.915      8.101      0.000      73.163     119.868
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):                 1.62
Prob(Q):                              0.92   Prob(JB):                         0.45
Heteroskedasticity (H):               1.47   Skew:                             0.00
Prob(H) (two-sided):                  0.26   Kurtosis:                         3.60
===================================================================================
```

说明

上图所示模型信息不输出到表格中，可在日志中查看。

pdate	forecast	lower	upper
1	345.7846777396164	326.52953555922875	365.0398199200041
2	351.8973665467896	328.71896646478643	375.0757666287928
3	332.060461831226	306.1173089391258	358.0036147233262
4	377.8399452648552	350.09885422341995	405.58103630629046
5	365.370249238134	336.38421548155287	394.3562829947152
6	381.67365763182994	351.8149902497976	411.5323250138623
7	454.78610213258906	424.30779671914235	485.2644075460358
8	511.73514104649183	480.81362355679283	542.6566585361909
9	526.5448536936494	495.304667237773	557.7850401495259
10	426.23568017310225	394.7655603628543	457.70579998335023
11	381.82507853956827	350.1886382068814	413.4615188722551
12	333.3279401390349	301.57098081397106	365.0848994640988

7.6 算链操作

7.6.1 查看算链

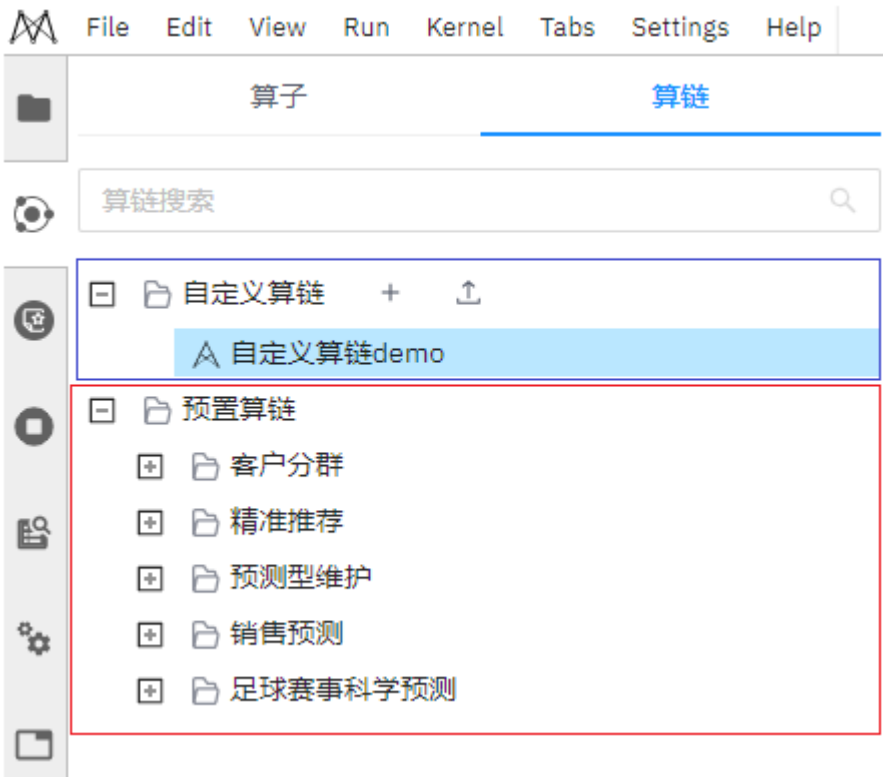
MLS提供了部分预置算链，方便用户直接使用。当预置算链不满足用户使用场景时，用户可以自行编排创建算链。

版权所有 © 华为云计算技术有限公司

单击Jupyter Lab界面左侧导航条上的图标，可快速进入MLS资产管理界面，当前版本支持算子和算链两种资产。

在算链页签中，包括预置算链和自定义算链两种，如图7-70所示。

图 7-70 算链列表



预置算链

资产管理界面中，预置算链如图7-70中红色框所示，开发者可以直接打开这些预置算链，并对这些算链进行运行或编辑等操作。

自定义算链

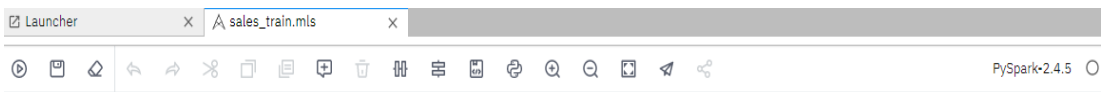
自定义算链类别中保存着开发者自行开发的或通过预置算链创建的算链，如图7-70中蓝色框所示，初始为空。

7.6.2 算链编排界面说明

导航按钮

算链编排界面导航栏提供丰富的界面操作，包括运行、保存、清除、撤销、恢复等功能。

图 7-71 算链编排界面导航栏

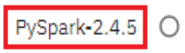
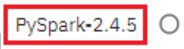
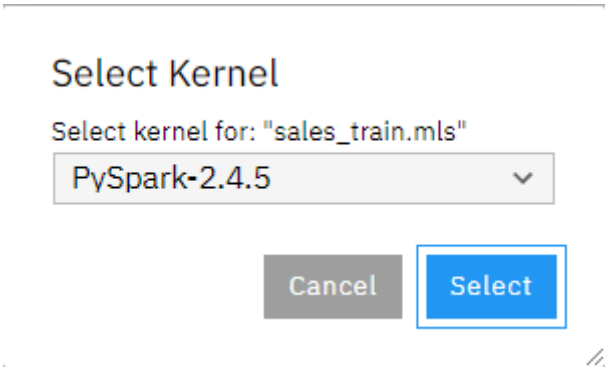


版权所有 © 华为云计算技术有限公司

表 7-14 导航栏功能说明

图标	操作名称	功能说明
	运行算链	运行界面上的算链。
	保存算链	保存算链。
	清除算链	一键清除画布所有算子及连线。
	撤销	撤销上一步的操作。键盘撤销操作也可实现该功能。
	恢复	恢复撤销的操作。
	剪切	选中界面上的算子，单击剪切图标即删除该算子。 可以使用键盘Shift选中多个算子，同时剪切。 剪切后可粘贴算子。
	复制	选中界面上的算子，单击复制图标或键盘复制操作Ctrl+C，即复制该算子。 可同时复制多个算子。
	粘贴	单击粘贴图标或键盘粘贴操作Ctrl+V，即可将复制算子粘贴在画布中。
	添加评论	选中界面上的算子，单击添加评论，可以在画布空白处为该算子添加标注或注释。可以同时选中多个算子添加评论。
	删除	选中界面上的算子，删除该算子。可删除多个算子。
	水平排列	将画布中的算链进行水平方向的重排列。
	垂直排列	将画布中的算链进行垂直方向的重排列。
	转换至Notebook	将画布中的所有算链按照顺序转化为一个ipynb后缀格式的Notebook文件。
	转换至Python	将画布中的所有算链按照顺序转化为一个py后缀格式的Python脚本。
	算链发布	将算链一键发布至ModelArst Workflow。 Workflow是邀测功能暂未上线。
	放大	将画布放大。

版权所有 © 华为云计算技术有限公司

图标	操作名称	功能说明
	缩小	将画布缩小。
	自适应屏幕	将画布重定位到画布中间。
	kernel切换	单击右上角kernel信息框，如  ，可切换kernel。 
	kernel状态	 表示kernel处于空闲状态，  表示kernel处于运行状态。

算链编辑区

算链编辑区是一个以画布形式展现的编排区域，可在该区域对算子进行算链的全部操作，包括拖拽建模、算子连线、编辑代码、设置参数等。

界面菜单

在算链编辑区右键界面空白处，出现界面菜单，包含添加评论、全选、编辑、撤销、恢复、取消高亮功能。

版权所有 © 华为云计算技术有限公司

图 7-72 界面菜单

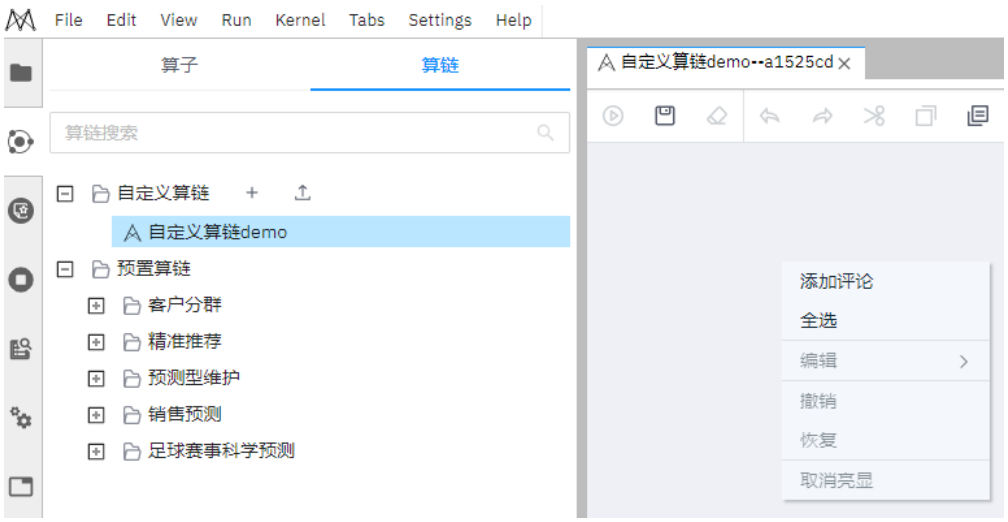


表 7-15 界面菜单说明

操作名称	功能说明
添加评论	同导航栏添加评论功能。
全选	选中界面上所有算子和评论。 使用键盘全选键Ctrl+A也可实现该功能。
编辑	选中算子或评论，可进行剪切、复制操作；选择粘贴，对之前剪切、复制的算子或评论进行粘贴操作。
撤消	同导航栏撤消功能。
恢复	同导航栏恢复功能。
取消亮显	消除界面上所有高亮的算子和评论。

算子结点

被拖拽到算链画布中的一个算子叫做一个算子结点。算子结点共有五种状态，如表7-16所示。

表 7-16 算子结点状态说明

结点状态	状态说明	图片
正常	呈现灰色。	
选中	呈现蓝色。	

版权所有 © 华为云计算技术有限公司

结点状态	状态说明	图片
运行中	呈现蓝色，右侧居中具有运行中标志。	
运行成功	呈现绿色。	
运行失败	呈现红色	

算子菜单

鼠标右键单击算子结点出现算子菜单，包含编辑、删除、高亮显示、设置参数、编辑代码、运行至此算子、运行当前算子、展示运行结果功能，如图7-73所示。算子菜单说明如表7-17所示。

图 7-73 算子菜单

表 7-17 算子菜单说明

菜单名称	操作说明
编辑	可对选中算子进行剪切、复制操作；选择粘贴，对之前复制的算子进行粘贴操作。
删除	对选中算子进行删除操作。

版权所有 © 华为云计算技术有限公司

菜单名称	操作说明
高亮显示	• 选择分支亮显，对选中算子所在算链分支高亮。 • 选择上行亮显，对选中算子前（包含该算子）的分支高亮。 • 选择下行亮显，对选中算子后（包含该算子）的分支高亮。 • 选择取消高亮，使其恢复正常状态。
设置参数	单击“设置参数”，算链界面右侧滑出参数编辑框。 修改参数后，单击“确定”，保存参数设置；单击“取消”则不保存。
编辑代码	单击“编辑代码”，算链界面右侧滑出编辑代码框，如图7-74所示。 上方为自定义编辑算子框，可编辑代码； 单击“执行代码”，结果将展现在代码执行结果框； 单击“保存”，则将修改后的代码保存并退出边界代码界面； 单击“取消”，则不保存且退出； 右上角为控制选择项，选中“代码”即展示代码；选中“结果”即显示结果；单击“全屏”，编辑界面将铺满算链编辑界面；单击“退出全屏”则取消全屏。
运行至此算子	算链运行至该算子。
运行当前算子	算链运行当前选中算子。
展示运行结果	展示当前选中算子的运行结果，如果该算子为未运行状态，则运行至当前算子后，展示结果。

版权所有 © 华为云计算技术有限公司

图 7-74 编辑算子代码

算子参数设置

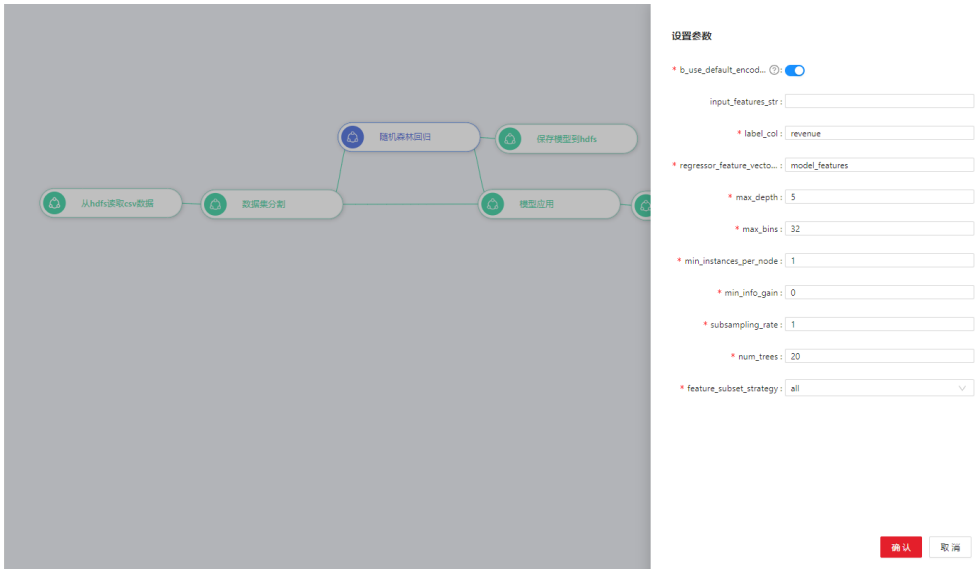
右键单击算子，选择“设置参数”，右侧滑出设置参数界面，如图7-75所示。

红色的*表示该参数为必填项，单击 显示该参数的说明。

单击“确认”，保存参数修改；单击“取消”，则不保存。

版权所有 © 华为云计算技术有限公司

图 7-75 参数设置框



算子连线

算子之间的连线具有两种意义，分为控制流和数据流。

- 控制流表示连线两端算子具有控制关系，即算子运行顺序。
- 数据流表示连线两端算子之间具有数据交换关系（简称数据关系），具有数据关系的两个算子，源算子的某个输出为目标算子的某个输入。

MLS中未刻意区分这两种关系，一般而言，存在数据交换的算子同时具有控制关系和数据关系，而不存在数据交换的算子之间则仅具有控制关系。

表 7-18 算子连线说明

流状态	状态说明	限制说明
数据流-单个输入输出	源算子具有单输出，目标算子具有单输入	如果源算子输出数据的类型和目标算子输入数据的类型不一致，则连线失败。
数据流-多个输入输出	源算子或目标算子具有多个输出或输入	会出现输入输出选择框，如果选择的源算子输出数据的类型和目标算子输入数据的类型不一致，则连线失败。
控制流	源算子和目标算子均无输出或输入	无。

连线菜单

选中连线，展示连线菜单。右键单击删除，可删除该连线，如图7-76所示。

版权所有 © 华为云计算技术有限公司

图 7-76 删除连线

Kernel 切换及状态

在Jupyter体系结构中，Kernel是由服务器启动的独立进程，不同的Kernel具有不同的编程语言和环境，用户可通过kernel运行代码。

目前，MLS的一个Editor对应一个Kernel，Editor中的所有算子会在此Kernel中运行。

Kernel切换及状态

同算链编排界面导航栏kernel切换及状态。

支持的Kernel

目前MLS仅支持PySpark-2.4.5。

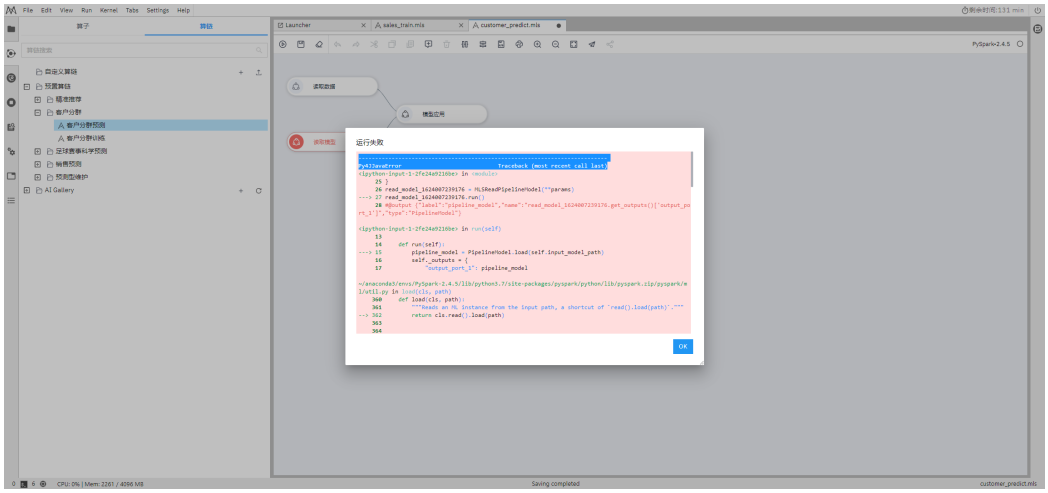
算链状态

算链运行成功，则算子和连线均转变为绿色。

算链运行失败，则弹出运行失败框，展示报错日志，如图7-77所示。

通过鼠标拖动运行失败框右下角（红色箭头）可放大缩小该框。运行失败算子及其连线转为红色。可使用算子菜单编辑代码功能对运行失败的算子进行调试。

图 7-77 算链运行失败

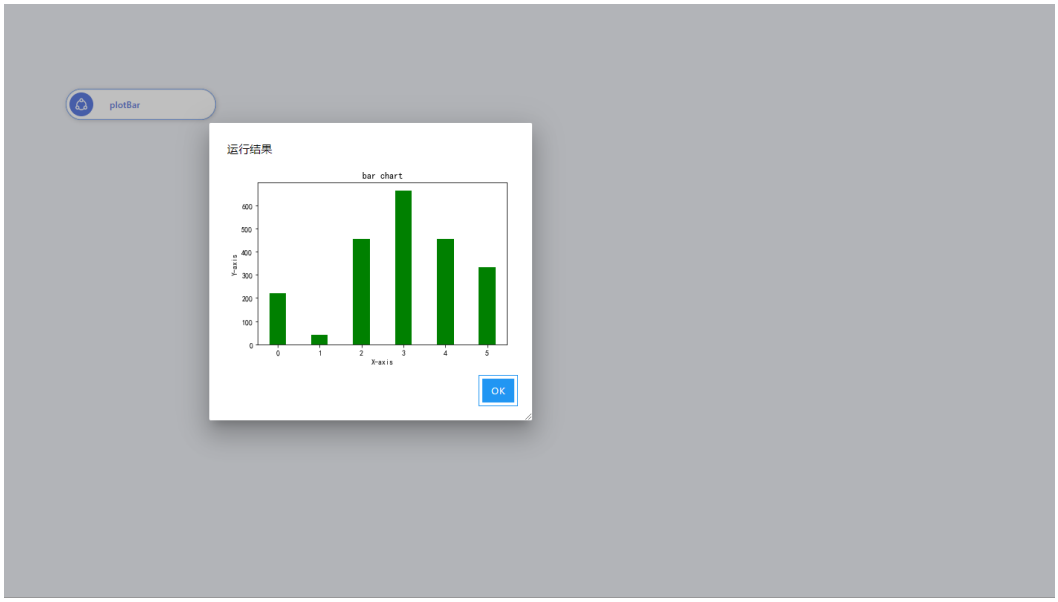


结果展示界面

展示当前选中算子的运行结果，展示打印输出，如图7-78所示。

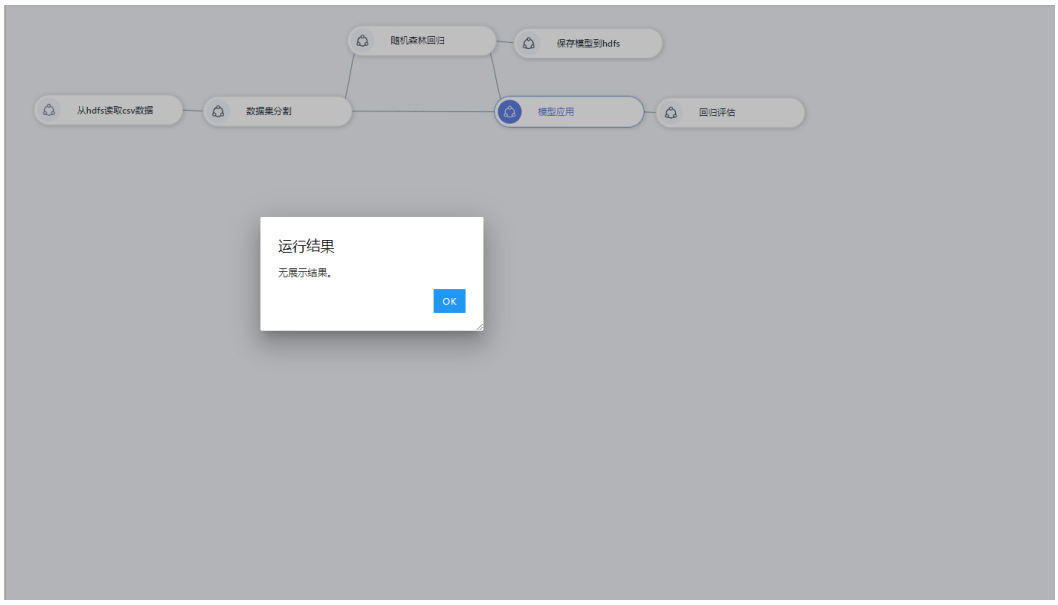
版权所有 © 华为云计算技术有限公司

图 7-78 结果展示（有输出）



如果无输出结果，则弹出无运行结果框，如图7-79所示。

图 7-79 结果展示（无输出）



如果该算子为未运行状态，则运行至当前算子后，展示结果。

说明

展示结果展示的是上一次运行的结果，在编辑代码后如需查看最新的结果，请先运行。

版权所有 © 华为云计算技术有限公司

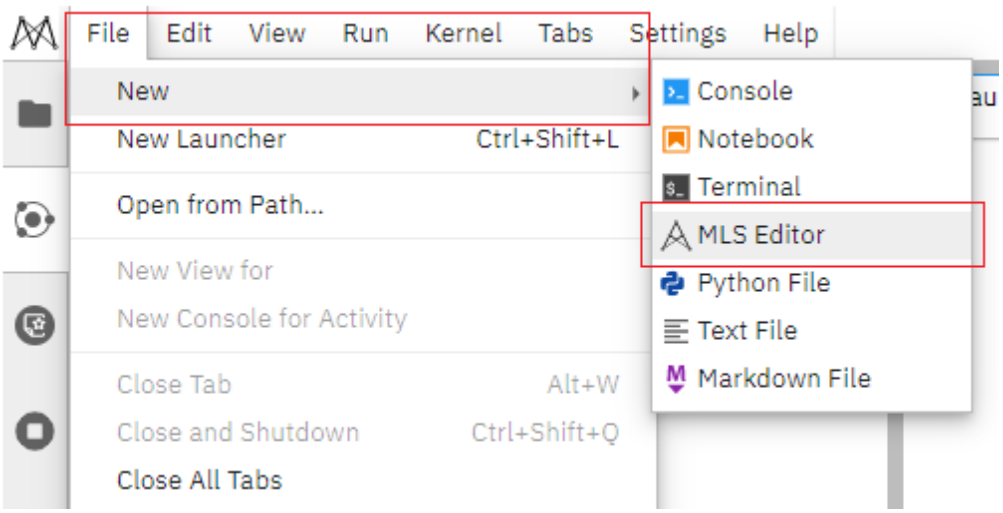
7.6.3 算链编排操作

新建算链

单击Launcher界面的MLS Editor图标，或者单击JupyterLab导航栏“File >New >MLS Editor”菜单，在新弹出的Kernel选择框中选择相应Kernel，即可创建一个新的算链。

创建算链后，左侧界面自动跳转到算子预览界面。具体操作同快速入门。

图 7-80 创建算链

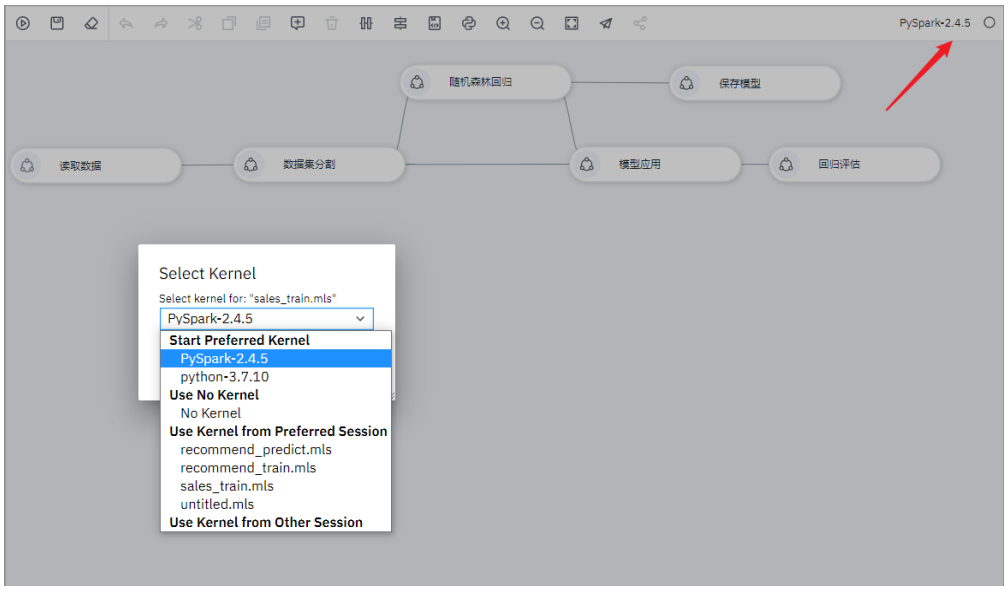


设置算链 Kernel

开发者可以通过两种方式设置算链Kernel：

- 新建MLS Editor时选择Kernel，如2介绍。
- 单击界面右上角切换Kernel，同导航栏切换Kernel功能，如图7-81所示。

图 7-81 切换 Kernel

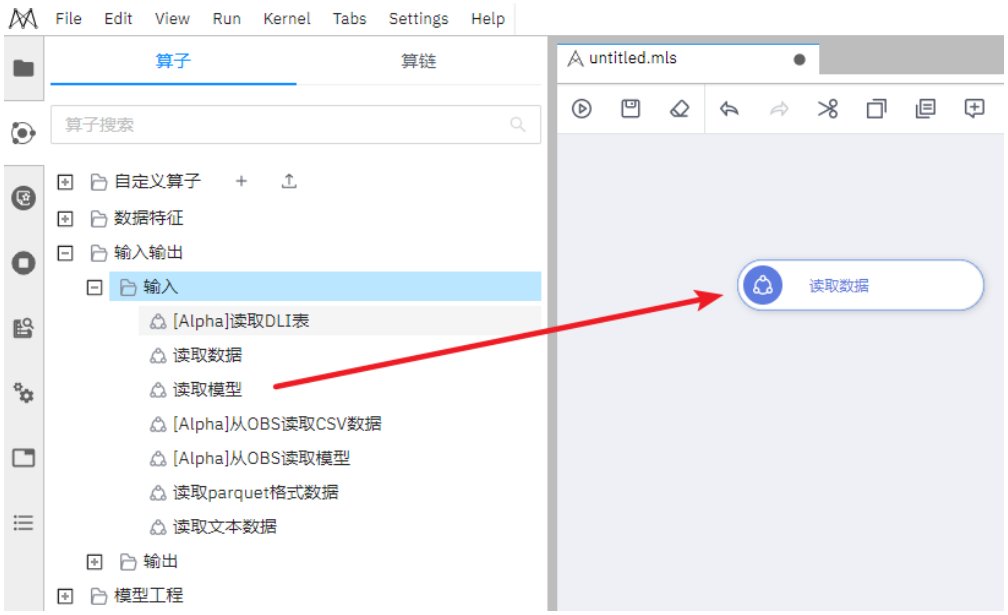


版权所有 © 华为云计算技术有限公司

创建算子结点（拖拽）

从左侧资产管理界面拖拽预置算子或自定义算子至右侧算链编辑界面，即可在算链中创建算子结点，如图7-82所示。

图 7-82 拖拽算子生成算子结点



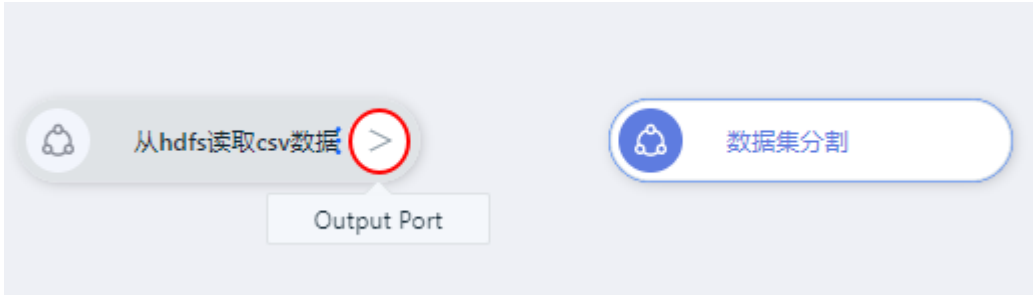
说明

如果算子不是从算子浏览界面拖拽至画布，比如从Jupyterlab文件浏览界面拖拽算子，则无法拖动，生成算子结点失败。

建立算子结点关联（连线）

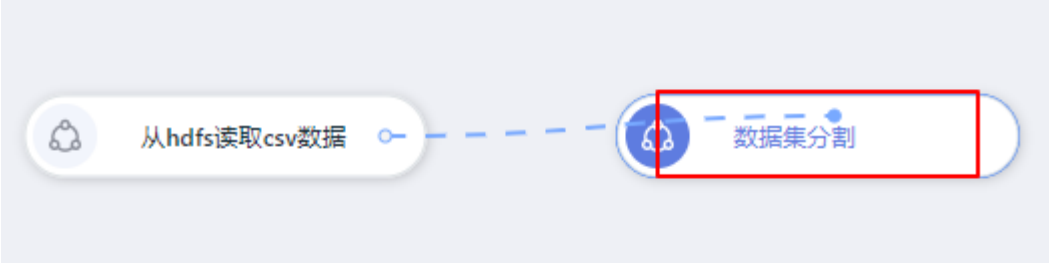
鼠标移至算子结点，从右侧输出端口，如图7-83所示，拖动连线至下一个算子结点，鼠标尽量放置至如图7-84红框位置，如果不在此区域，将会导致算子连线失败。

图 7-83 从输出端口移动至下一结点



版权所有 © 华为云计算技术有限公司

图 7-84 连线结束位置



如果源算子与目标算子之间存在输入输出关系，分为两种情况，如表7-19介绍。

表 7-19 输入输出关系

输入输出说明	操作
源算子单输出，目标算子单输入	如果输出数据类型和输入类型一致，则连接成功；否则连接失败，如图7-84所示。
源算子和目标算子中有一个及以上具有多输入输出关系	弹出输入输出选择框，选择输入输出端口，单击OK，如果输出数据类型和输入类型一致，则连接成功；否则连接失败。

图 7-85 算子连线失败

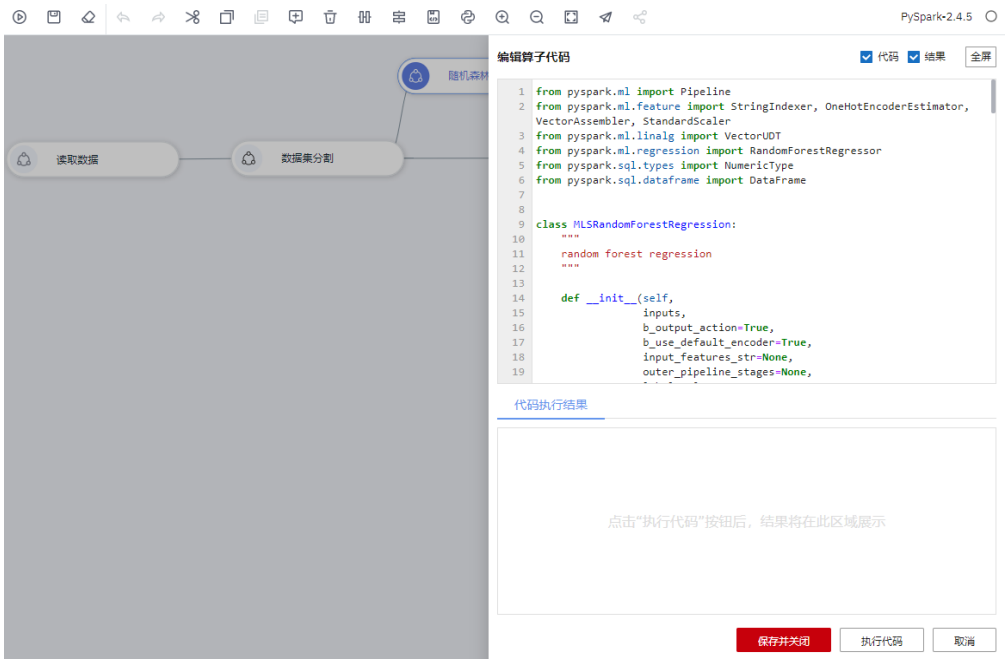
编辑算链中算子

- 算子代码修改

  双击界面中的算子节点或在算子结点上右键选择“编辑代码”功能，即可在算链编排界面右侧滑出算子编辑器进行算子代码编辑。

版权所有 © 华为云计算技术有限公司

图 7-86 算子编辑器



- 算子代码调试

  算子代码进行编辑后，单击下方“执行代码”，即可调试该算子，算子结果展示在“代码执行结果”界面。

- 算子代码保存

  单击下方“保存并关闭”，保存算子代码并关闭算子编辑器；单击取消则不保存并关闭算子编辑器。

- 算子编辑控制

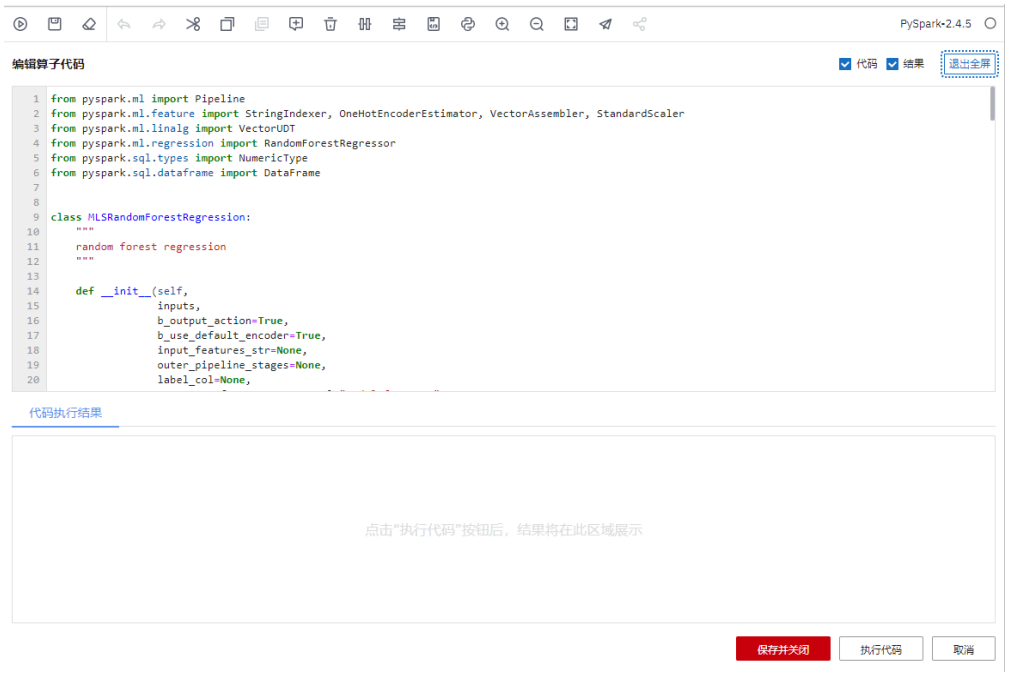
  算子编辑控制具有是否选择展示代码、是否选择展示结果、是否全屏。

  选择展示代码、展示结果、非全屏，如图7-86所示。

  选择展示代码、展示结果、全屏，如图7-87所示，单击“退出全屏”按钮则恢复。

版权所有 © 华为云计算技术有限公司

图 7-87 展示代码、结果、全屏



算链转换

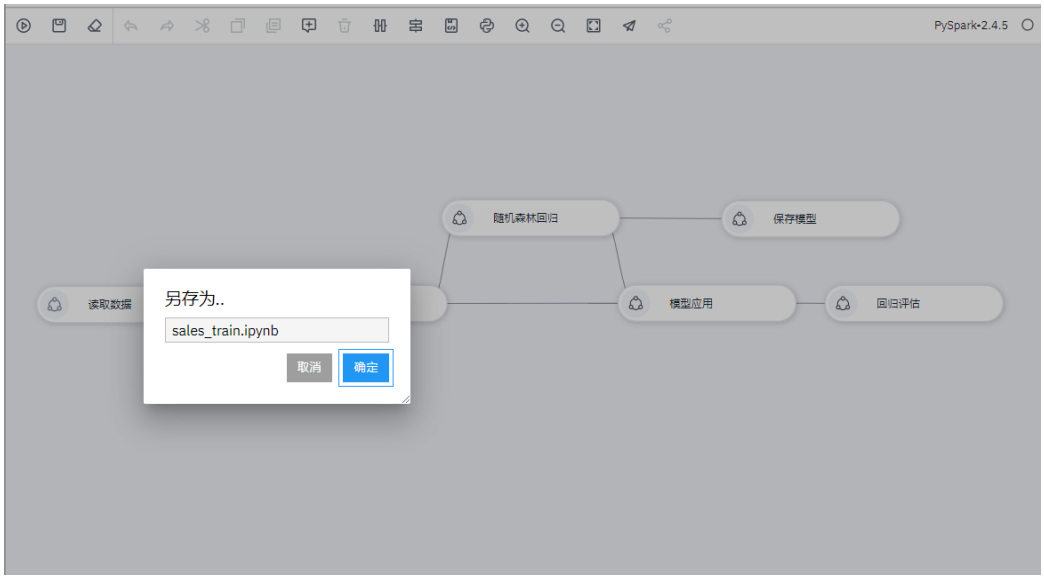
转换成Notebook

算链界面所有算子按照DAG顺序转换为一个Notebook文件并打开。

单击导航栏转化为Notebook按钮，选择另存为路径，默认路径为当前MLS Editor所在路径，如图7-88所示。

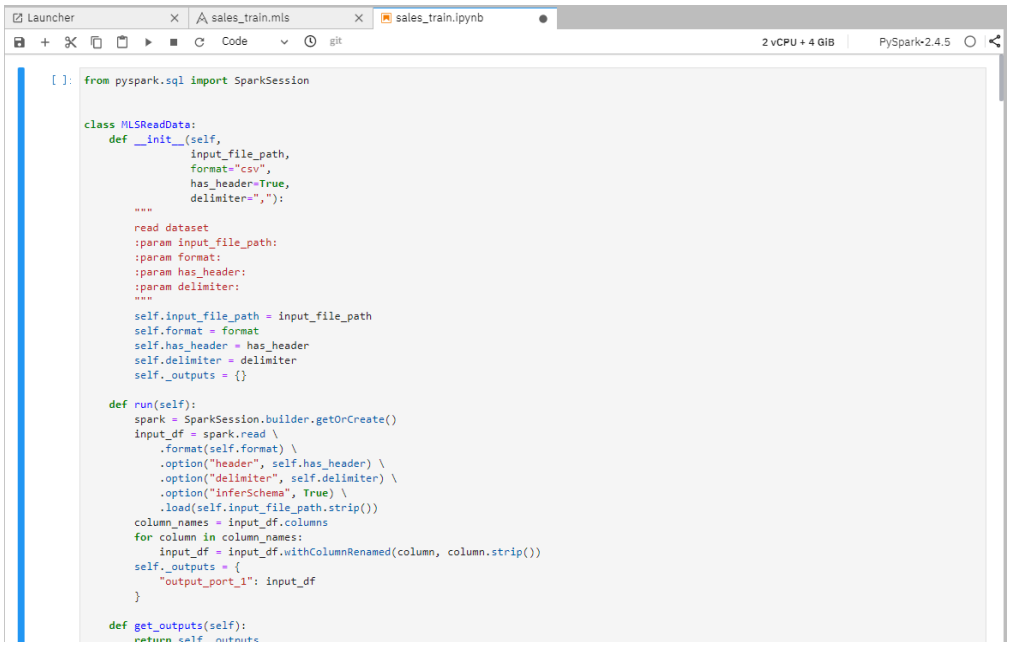
单击保存后，自动打开转换后的Notebook文件，如图7-89所示。

图 7-88 选择 Notebook 文件保存路径



版权所有 © 华为云计算技术有限公司

图 7-89 自动打开 Notebook 文件



转换成python

算链界面所有算子按照顺序转换为一个python脚本并打开。

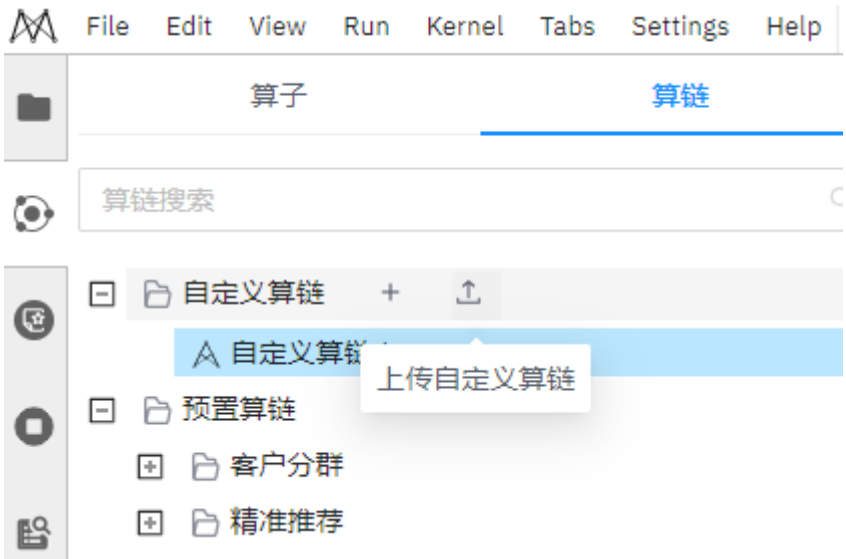
操作过程同转换为Notebook。

7.6.4 上传/下载算链

上传算链

单击上传自定义算链图标 ，从本地上传算链，如图7-90所示，当前版本仅支持上传mls文件。

图 7-90 上传自定义算链



版权所有 © 华为云计算技术有限公司

下载算链

对算链单击右键，选择下载，即可将算链下载到本地，如图7-91所示。算子会以mls文件（后缀为.mls）下载到本地。

图 7-91 下载算链

7.6.5 运行算链

算链运行分为三种模式：运行算链、运行到此处、运行当前算子。

由于一个MLS Editor对应一个Kernel（进程），所有运行模式皆可获取前续阶段运行后的所有变量。

表 7-20 算链运行模式

运行模式	操作	说明
运行算链	单击算链编排界面导航栏运行按钮。	运行整个算链，如图7-92所示。对应Notebook的Run All功能。
运行至此算子	右击算链编排界面算子，选择运行至此算子。	运行选中算子的前面分支（包含该算子），如图7-93所示。对应Notebook的Run Above功能。
运行当前算子	右击算链编排界面算子，选择运行当前算子。	运行选中算子，可使用前续阶段运行后的所有变量，如图7-94所示。对应Notebook的Run Selected Cell功能。

版权所有 © 华为云计算技术有限公司

图 7-92 运行算链

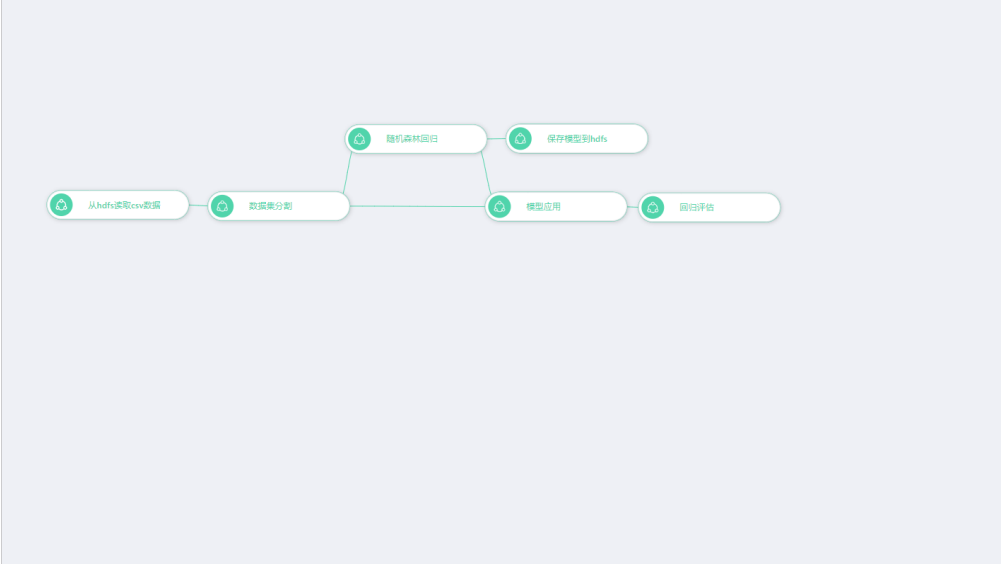


图 7-93 运行至模型应用算子

版权所有 © 华为云计算技术有限公司

图 7-94 运行回归评估算子

7.7 常见问题

7.7.1 ML Studio 错误码

ML Studio在使用中出现的错误码及含义详见下表。

错误码	错误描述
1001	预置资产不可删除
1002	预置资产不可修改
1003	算子分组不存在
1004	缺少必要参数
1005	资产类别不存在
1006	名字中不允许使用字符< > ' \" ; \ ` = # $ % ^ & ( )
1007	资产不存在
1008	资产id冲突
1009	资产名字冲突
1010	资产资源不存在
1011	返回无效内容
1012	保存资产失败
1201	无效的OBS路径
1202	无法访问目标OBS路径

版权所有 © 华为云计算技术有限公司

错误码	错误描述
1211	文件不可用
1212	OBS路径不可用
1213	无法上传脚本至指定OBS资源地址
1301	请指定DLI队列
1302	无效的DLI资源规格类型
1303	非法的Driver与Executor资源分配策略
1304	请指定工作空间名称
1305	请指定工作空间id
1306	不支持发布包含该节点类型的Workflow
1307	无法获取ModelArts工作空间信息
1321	无法获取ModelArts Workflow信息
1401	无法获取DLI队列信息

版权所有 © 华为云计算技术有限公司

8 使用 Notebook 开发 Ascend 算子

概述

训练、推理场景下，使用第三方框架时遇到不支持的算子，需要自己开发；网络调优时，发现一些算子组合性能较低，需重新开发高性能算子替换低性能的算子，此时可以通过VS Code一键连接云上Notebook，使用云上资源，在VS Code端进行算子开发、调试等。Notebook中已经配置好环境，您无需进行CANN软件安装和环境变量配置就可以进行工程化开发。

说明

本文档提供了一套算子工程样例代码，您可以直接使用。如果需要了解Ascend算子的编程模型等，请参见**昇腾文档**。Notebook中已经配置好环境，无需进行CANN软件安装和环境变量配置，直接在VS Code远端环境中直接进行算子分析及后续操作。

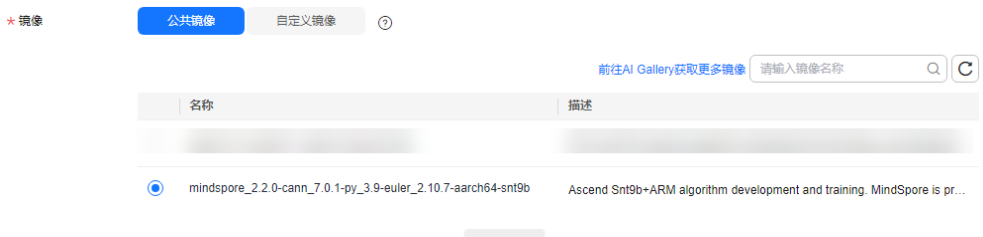
准备工作

- 单击**链接**下载算子样例并上传到OBS桶。
- 创建基于mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b引擎的Notebook实例，并打开SSH远程开发开关。且该Notebook实例状态必须为“运行中”。具体操作参考**创建Notebook实例**。

  说明

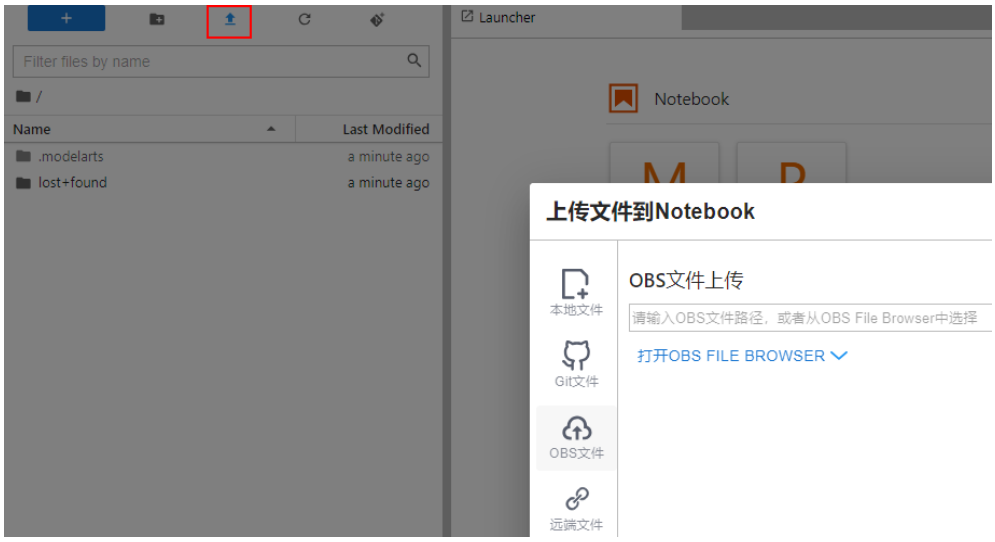
  本文档只针对选用“mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b”的引擎进行算子调测，如果使用其它AI引擎可能会报错。

  图 8-1 创建基于 mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b 引擎的 Notebook 实例

- 打开JupyterLab，单击文件上传按钮  ，将OBS桶的样例文件传至Notebook。详细操作请参考**上传OBS文件到JupyterLab**。

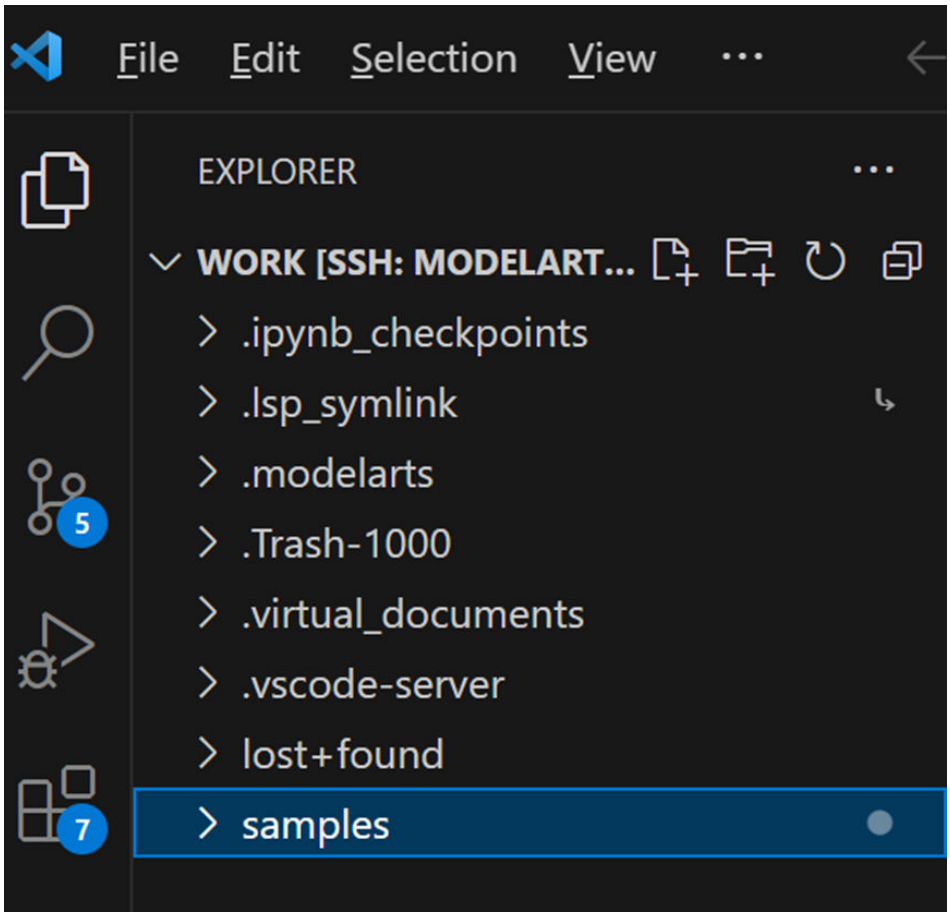
版权所有 © 华为云计算技术有限公司



通过 VS Code 连接云端 Notebook

1. Notebook创建完成后处于运行状态，单击操作列的“更多 > VS Code接入”，参考**VS Code一键连接Notebook**连接云上开发环境。
2. 成功连接云上开发环境后，VS Code界面上会显示云上已下载的工程如**图8-2**所示。

图 8-2 工程目录



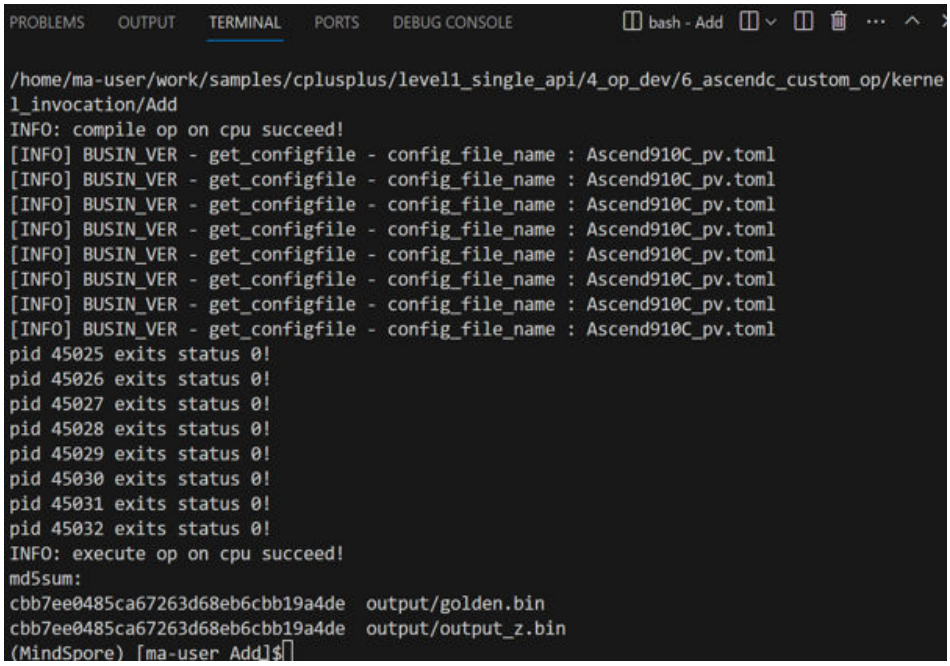
版权所有 © 华为云计算技术有限公司

在 VS Code 中调试 Add 算子

1. 在Terminal中执行如下命令进入Add算子所在目录。
   ```
   cd samples/cplusplus/level1_single_api/4_op_dev/6_ascendc_custom_op/kernel_invocation/Add
   ```
2. 执行如下命令编译和运行脚本。
   1. CPU模式下执行如下命令。
      ```
      bash run.sh add_custom ascend910B1 VectorCore cpu
      ```
      其中，add_custom表示需要运行的算子，ascend910B1表示算子运行的AI处理器型号，VectorCore表示在VectorCore上运行，cpu表示算子以cpu模式运行。

      运行结果如下，当前使用md5sum对比了所有输出bin文件，md5值一致表示实际的输出数据和真值数据相符合。

      图 8-3 CPU 模式运行结果

      ```
      /home/ma-user/work/samples/cplusplus/level1_single_api/4_op_dev/6_ascendc_custom_op/kerne
      l_invocation/Add
      INFO: compile op on cpu succeed!
      [INFO] BUSIN_VER - get_configfile - config_file_name : Ascend910C_pv.toml
      [INFO] BUSIN_VER - get_configfile - config_file_name : Ascend910C_pv.toml
      [INFO] BUSIN_VER - get_configfile - config_file_name : Ascend910C_pv.toml
      [INFO] BUSIN_VER - get_configfile - config_file_name : Ascend910C_pv.toml
      [INFO] BUSIN_VER - get_configfile - config_file_name : Ascend910C_pv.toml
      [INFO] BUSIN_VER - get_configfile - config_file_name : Ascend910C_pv.toml
      [INFO] BUSIN_VER - get_configfile - config_file_name : Ascend910C_pv.toml
      [INFO] BUSIN_VER - get_configfile - config_file_name : Ascend910C_pv.toml
      pid 45025 exits status 0!
      pid 45026 exits status 0!
      pid 45027 exits status 0!
      pid 45028 exits status 0!
      pid 45029 exits status 0!
      pid 45030 exits status 0!
      pid 45031 exits status 0!
      pid 45032 exits status 0!
      INFO: execute op on cpu succeed!
      md5sum:
      cbb7ee0485ca67263d68eb6cbb19a4de  output/golden.bin
      cbb7ee0485ca67263d68eb6cbb19a4de  output/output_z.bin
      (MindSpore) [ma-user Add]$
      ```
   2. NPU模式下执行如下命令
      ```
      bash run.sh add_custom ascend910B1 VectorCore npu
      ```
      运行结果如下，当前使用md5sum对比了所有输出bin文件，md5值一致表示实际的输出数据和真值数据相符合。

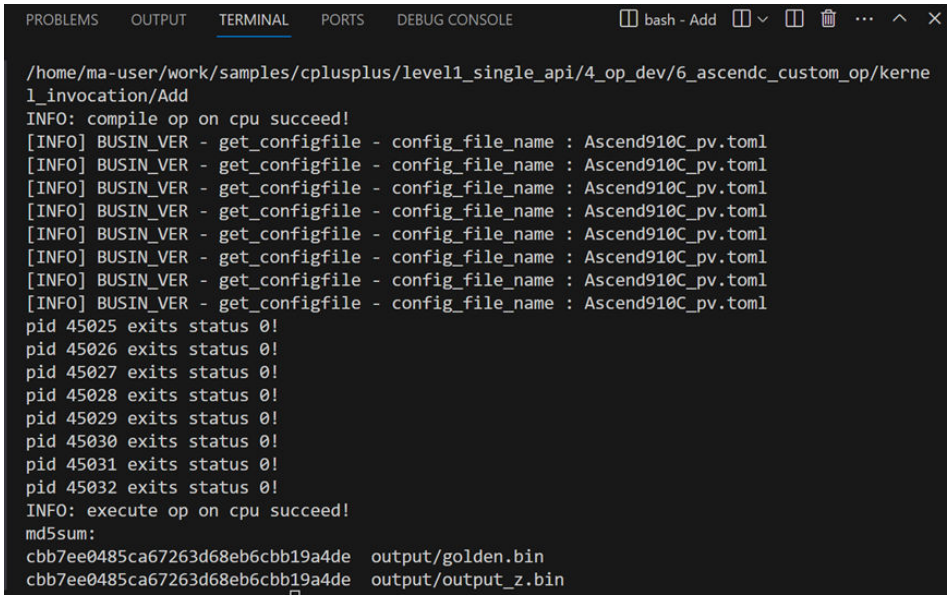


图 8-4 NPU 模式运行结果

```
/home/ma-user/work/samples/cplusplus/level1_single_api/4_op_dev/6_ascendc_custom_op/kerne
l_invocation/Add
INFO: compile op on cpu succeed!
[INFO] BUSIN_VER - get_configfile - config_file_name : Ascend910C_pv.toml
[INFO] BUSIN_VER - get_configfile - config_file_name : Ascend910C_pv.toml
[INFO] BUSIN_VER - get_configfile - config_file_name : Ascend910C_pv.toml
[INFO] BUSIN_VER - get_configfile - config_file_name : Ascend910C_pv.toml
[INFO] BUSIN_VER - get_configfile - config_file_name : Ascend910C_pv.toml
[INFO] BUSIN_VER - get_configfile - config_file_name : Ascend910C_pv.toml
[INFO] BUSIN_VER - get_configfile - config_file_name : Ascend910C_pv.toml
[INFO] BUSIN_VER - get_configfile - config_file_name : Ascend910C_pv.toml
pid 45025 exits status 0!
pid 45026 exits status 0!
pid 45027 exits status 0!
pid 45028 exits status 0!
pid 45029 exits status 0!
pid 45030 exits status 0!
pid 45031 exits status 0!
pid 45032 exits status 0!
INFO: execute op on cpu succeed!
md5sum:
cbb7ee0485ca67263d68eb6cbb19a4de  output/golden.bin
cbb7ee0485ca67263d68eb6cbb19a4de  output/output_z.bin
```

在 VS Code 中调试 matmul 算子

1. matmul算子所在目录为“samples/cplusplus/level1_single_api/4_op_dev/6_ascendc_custom_op/kernel_invocation/Matmul”。
2. 在work目录下执行如下命令。
```
cd samples/cplusplus/level1_single_api/4_op_dev/6_ascendc_custom_op/kernel_invocation/Matmul
```
3. 执行如下命令修改main.cpp文件，此文件为调用算子的应用程序文件。
```
vim main.cpp
```
将param4FileSize的值改为192

图 8-5 修改 param4FileSize 为 192

```
int32_t main(int32_t argc, char* argv[])
{
    size_t param1FileSize = 512 * 512 * sizeof(uint16_t);  // uint16_t represent half
    size_t param2FileSize = 512 * 1024 * sizeof(uint16_t);  // uint16_t represent half
    size_t param3FileSize = 512 * 1024 * sizeof(float);
    size_t param4FileSize = 192 * sizeof(uint32_t);
"main.cpp" 100L, 4472B                                              1,1           Top
```

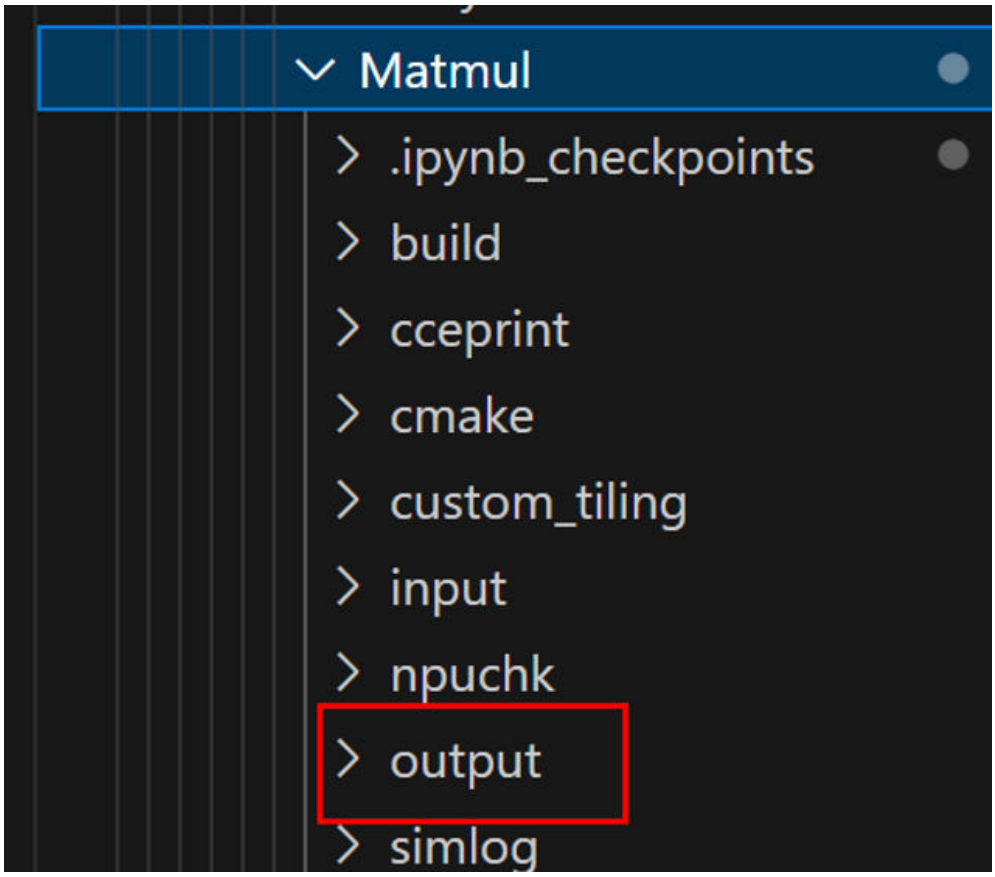
4. 执行如下命令修改vim matmul_custom.cpp文件。
```
vim matmul_custom.cpp
```
将 matmul_custom.cpp中的tiling.K更改成tiling.Ka

图 8-6 tiling.K 更改成 tiling.Ka

```
    if (GetBlockIdx() >= tiling.usedCoreNum) {
        return;
    }

    GlobalTensor<A_T> aGlobal;
    GlobalTensor<B_T> bGlobal;
    GlobalTensor<C_T> cGlobal;

    aGlobal.SetGlobalBuffer(reinterpret_cast<__gm__ A_T*>(a), tiling.M * tiling.Ka);
    bGlobal.SetGlobalBuffer(reinterpret_cast<__gm__ B_T*>(b), tiling.Ka * tiling.N);
    cGlobal.SetGlobalBuffer(reinterpret_cast<__gm__ C_T*>(c), tiling.M * tiling.N);
```

5. 手动在Matmul目录下创建名为“output”的文件夹。

图 8-7 创建 output 文件夹



6. 执行如下命令编译和运行脚本。

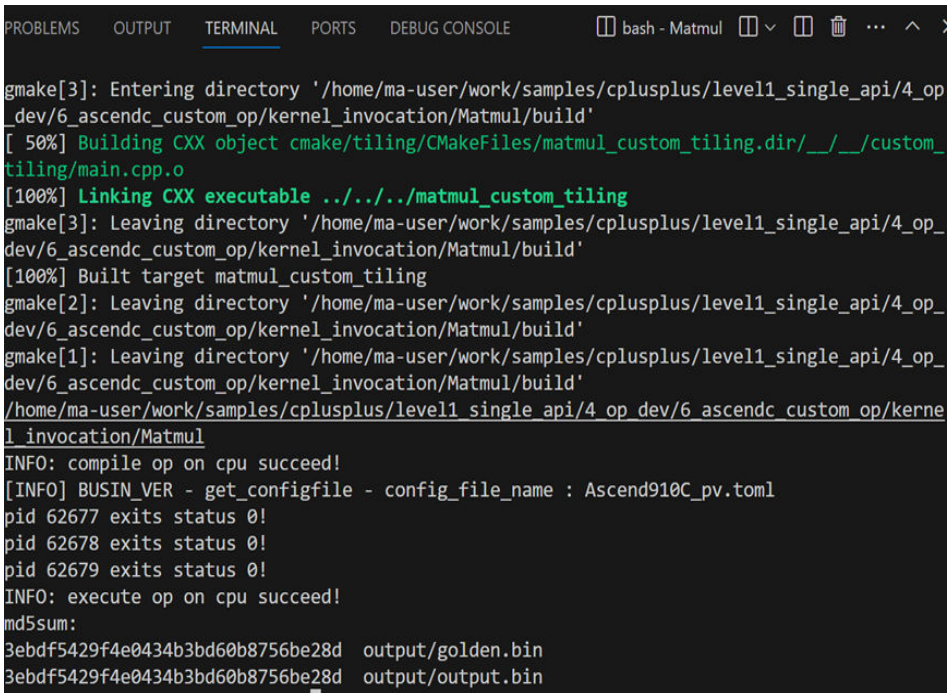
   a. CPU模式下执行如下命令

   ```
   bash run.sh matmul_custom ascend910B1 AiCore cpu ONBOARD CUSTOM_TILING
   ```

   运行结果如下，当前使用md5sum对比了所有输出bin文件，md5值一致表示实际的输出数据和真值数据相符合。

版权所有 © 华为云计算技术有限公司

图 8-8 CPU 模式运行结果

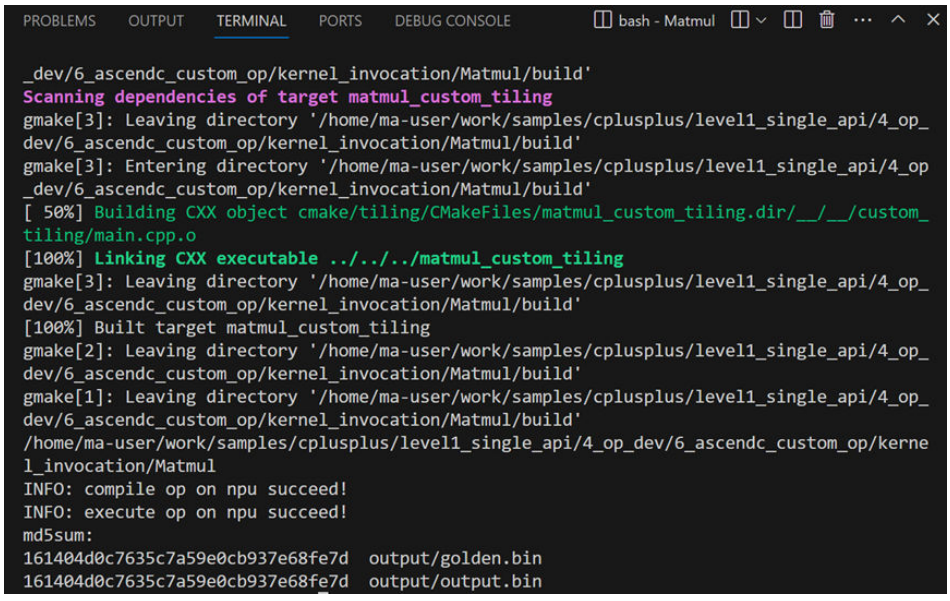


b. NPU模式下执行如下命令

```
bash run.sh matmul_custom ascend910B1 AiCore npu ONBOARD CUSTOM_TILING
```

运行结果如下，当前使用md5sum对比了所有输出bin文件，md5值一致表示实际的输出数据和真值数据相符合。

图 8-9 NPU 模式运行结果



生成 profile

NPU调试后，会在工程目录下生成matmul_custom_npu可执行文件，执行如下命令生成profiling。

 版权所有 © 华为云计算技术有限公司

```
msprof --application="matmul_custom_npu" --output="./output"
```

图 8-10 生成 profile

```
[INFO] Export all data in PROF_000001_20240126103713957_OBBJCJNMKIJGEACC done.
[INFO] Start query data in PROF_000001_20240126103713957_OBBJCJNMKIJGEACC.
Job Info        Device ID       Dir Name        Collection Time                 Model IDI
teration Number Top Time Iteration      Rank ID

NA              0               device_0        2024-01-26 10:37:13.959103      N/A     N
/A              N/A                     0

NA                              host            2024-01-26 10:37:13.959103      N/A     N
/A              N/A                     0

[INFO] Query all data in PROF_000001_20240126103713957_OBBJCJNMKIJGEACC done.
[INFO] Profiling finished.
[INFO] Process profiling data complete. Data is saved in /home/ma-user/work/samples/cplus
plus/level1_single_api/4_op_dev/6_ascendc_custom_op/kernel_invocation/Matmul/output/PROF_
000001_20240126103713957_OBBJCJNMKIJGEACC
```

停止 Notebook 实例前备份文件

Notebook实例停止时，后端对应的容器环境会被删除，只有“/home/ma-user/work”目录下的内容会持久化保存，其他目录下的修改都会丢失。

备份方法

可以在停止Notebook实例前手工复制文件到/home/ma-user/work目录下。

需要复制的目录内容包括：

1. /home/ma-user/ AscendProjects目录下的自建工程
2. /home/ma-user/modelzoo/目录下的模型转换后的om文件、配置文件、评估报告
3. /home/ma-user/.mindstudio目录下的ssh配置
4. 其他用户自己修改的内容

当Notebook实例再次启动时，用户将手工备份的目录内容复制回原始目录后即可正常继续使用。

版权所有 © 华为云计算技术有限公司

9 算法开发套件

9.1 算法开发套件简介

ModelArts算法开发套件提供了一个全流程和白盒化的云原生算法开发工具，支持通过本地VS Code/PyCharm远程连接到云上开发环境后使用。Notebook作为算法工程开发入口，便于用户无感调用云上计算存储资源进行开发，同时提供丰富的、可扩展的算法套件便于用户使用或二次开发。

算法开发套件支持的主要特性：

- 工程管理：用户可通过ma-cli createproject创建工程。

- 资产管理：用户可通过python manage.py list列举或python manage.py install来安装、升级、覆盖已发布的算法包、模型包、数据集。

- 数据拷贝：用户可通过python manage.py copy命令来实现OBS和本地数据的快速传输。

- 本地训练、验证、推理：用户可通过python manage.py run命令。

- 本地部署：用户可通过python manage.py deploy命令。

- ModelArts训练、模型转换：用户可通过python manage.py run --launch_remote命令提交远程训练作业。

- ModelArts部署：用户可通过python manage.py deploy --launch_remote命令提交远程部署任务。

- 模块化设计：用户可基于算法框架规范构建自定义的算法资产并发布到ModelArts。

算法开发套件还支持用户在Notebook中用Python API进行交互式、参数化、低代码的开发方式快速完成算法验证与实践，使用指导请参考**通过Python API使用算法套件**。

9.2 准备开发环境

创建 Notebook 实例

1. 登录ModelArts控制台。选择控制台区域，以“华北-北京四”为例。

版权所有 © 华为云计算技术有限公司

2. 在开发环境Notebook中创建基于pytorch1.4-cuda10.1-cudnn7-ubuntu18.04镜像的Notebook，具体操作请参见**创建Notebook实例**章节。

   如果需要使用本地IDE（PyCharm或VS Code）远程连接Notebook，此处需要开启SSH远程开发。

打开 Notebook 实例

提供以下2种方式：

一、使用JupyterLab打开

1. 在ModelArts控制台 “开发环境 > Notebook”页面的列表中，单击Notebook操作栏的“打开”，进入JupyterLab页面。JupyterLab操作请参见**JupyterLab简介及常用操作**。
2. 创建一个ipynb文件或者打开Terminal窗口，用于交互使用命令。

图 9-1 创建一个 ipynb 文件

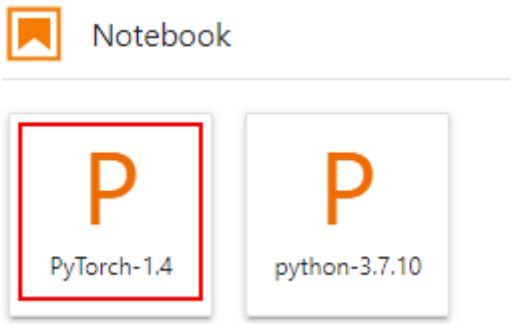
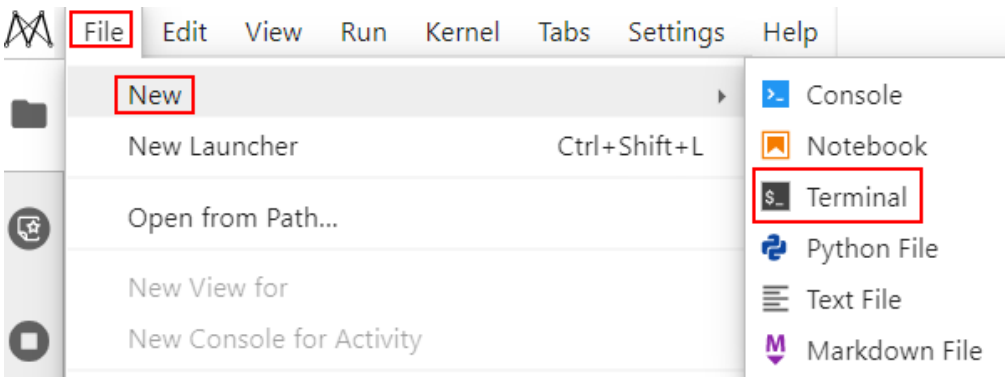


图 9-2 打开 Terminal



二、使用本地IDE（PyCharm或VS Code）远程连接到Notebook。

具体操作请参见**配置本地IDE（PyCharm ToolKit连接）**或**VS Code一键连接Notebook**。

9.3 创建算法工程

ma-cli是基于cookiecutter开发的用于管理工程的命令行工具，它支持创建一个算法模板工程并一键式安装ModelArts算法套件等。

 版权所有 © 华为云计算技术有限公司

了解算法工程模板

算法工程模板结构如下：

```
your-project-name
├── algorithms            -- algorithm toolkit folder
│   ├──custom
├── data                -- data folder,
│   └── raw              -- the original, immutable data dump.
├── docs                -- doc for your project
├── model_zoo             -- pretrained model zoo folder
├── project-slug          -- user's source code in this project.
├── config_ma.py          -- ModelArts related configuration, authentications, temp obs
bucket, proxy, etc.
├── manage.py             -- CLI entry
├── README.md
├── requirement.txt       -- The requirements file for reproducing the  environment
└── setup.py             -- makes project pip installable (pip install -e .) so project-slug can be
imported
```

其中：

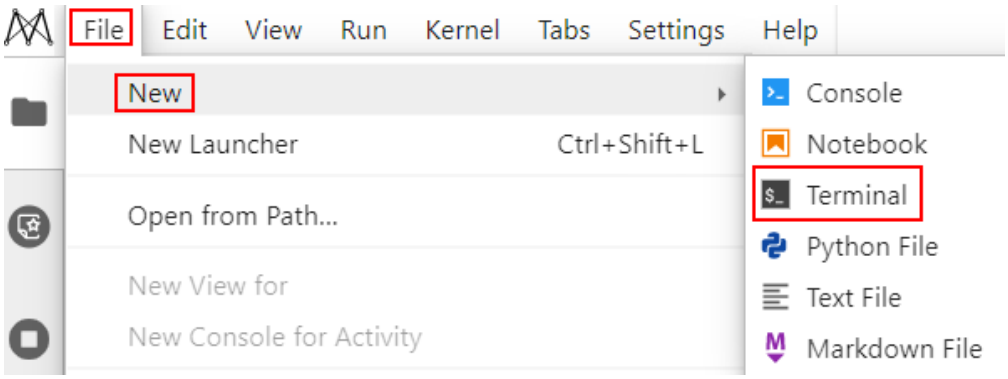
- algorithms：算法默认下载路径。
- data：数据集默认下载路径。
- docs：使用文档目录。
- model_zoo：模型默认下载路径。
- config_ma.py：创建工程时输入的配置信息，如果在算法内的config.py里填写了同样的信息，那么算法config.py的优先级高于config_ma.py。
- manage.py：算法工程命令行入口。

使用 ma-cli 创建工程

场景一：在JupyterLab的Terminal里使用ma-cli创建工程

1. 打开JupyterLab的Terminal。

   图 9-3 打开 Terminal

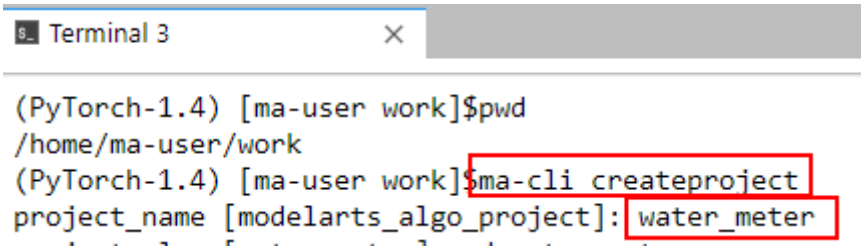
   

2. 执行ma-cli createproject命令创建工程，用户可按照提示交互式输入project_name，此处以water_meter为例。

   ```
   ma-cli createproject
   ```

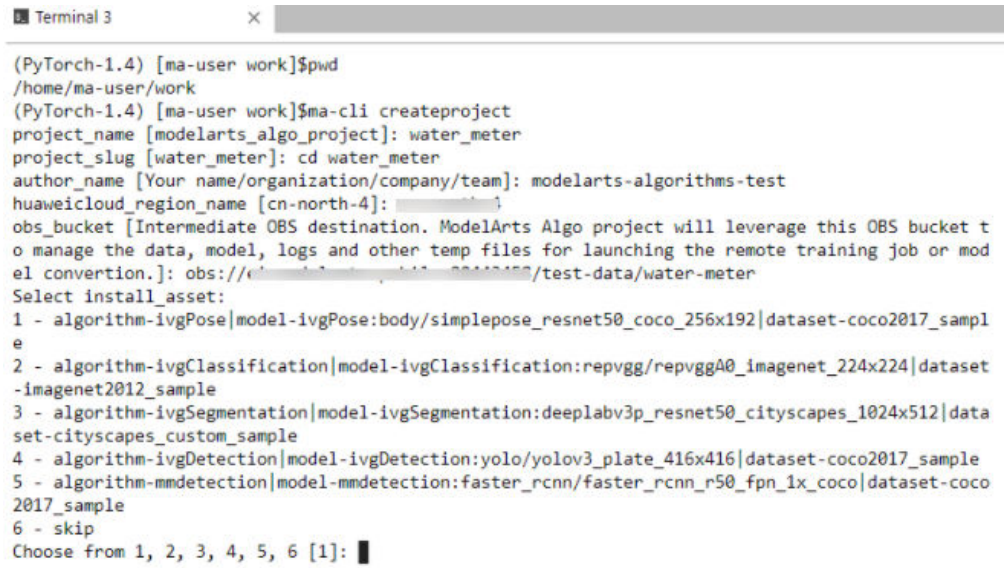

版权所有 © 华为云计算技术有限公司

图 9-4 创建算法工程



3. 执行cd {project_name}命令切换到工程目录下。
   ```
   cd {project_name}
   ```
4. 根据界面提示输入相关参数。
   ```
   author_name [Your name/organization/company/team]: 自定义，例如：modelarts-algorithms-test
   huaweicloud_region_name [cn-north-4]: 此参数可以忽略，系统会自动识别Notebook所在区域。
   obs_bucket ： 自定义OBS桶路径，该OBS桶必须与Notebook在同一个区域。
   Select install_asset:
   ```

图 9-5 创建算法工程参数设置



场景二：在JuputerLab的Ipynb文件里使用ma-cli创建工程

说明

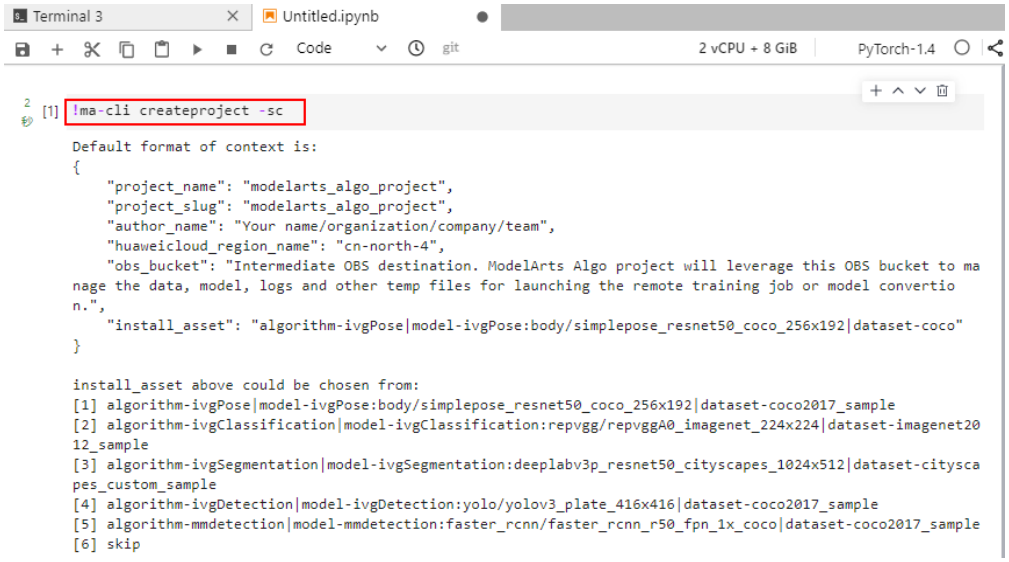
在JuputerLab的Ipynb文件里执行命令，需要在命令前加！符号，例如：

```
!ma-cli createproject -sc
```

1. 输入如下命令并执行，获取需要填写的参数信息。
   ```
   !ma-cli createproject -sc
   ```


版权所有 © 华为云计算技术有限公司

图 9-6 创建算法工程



2. 在上方导航栏单击 ＋ ，新建代码行，并输入如下命令创建工程，样例代码如下。

```
context = {
    "project_name": "modelarts_algo_project",
    "project_slug": "modelarts_algo_project",
    "author_name": "Your name/organization/company/team",
    "huaweicloud_region_name": "cn-north-4",
    "obs_bucket": "obs://bucket_name/bucket_directory",
    "install_asset": "skip"
}
!ma-cli createproject --context f"{context}"
```

“author_name” 替换成自定义名称。

“obs_bucket”替换成实际的OBS桶路径。

图 9-7 创建工程

3. 切换工作目录为创建的工程目录。此命令没有结果输出。

```
import os
# set project_dir to your project path
project_dir = '/home/ma-user/work/modelarts_algo_project'
os.chdir(project_dir)
```

 版权所有 © 华为云计算技术有限公司

图 9-8 切换工作目录

```
[11] import os
     # set project_dir to your project path
     project_dir = '/home/ma-user/work/modelarts_algo_project'
     os.chdir(project_dir)
```

4. 通过python manage.py -h查看算法开发套件支持的命令及其用法。

```
!python manage.py -h
```

图 9-9 查看算法开发套件支持的命令及其用法

```
[ ] !python manage.py -h

Usage: manage.py [OPTIONS] COMMAND [ARGS]...

Options:
  -h, --help  Show this message and exit.

Commands:
  convert  Convert model.
  copy     Transfer data between OBS and ModelArts Notebook.
  deploy   deploy model.
  export   To export certain contents.
  flow     To execute the task by your flow config.
  install  Install algorithm/model/dataset.
  list     List algorithm/model/dataset.
  query    To query ModelArts info.
  run      Train Evaluate Infer.
```

查看如何使用 ma-cli

系统提供了内置的help，通过以下命令可以查看如何使用ma-cli和ma-cli createproject相关命令。

查看如何使用ma-cli

```
ma-cli -h
```

查看如何使用ma-cli createproject

```
ma-cli createproject -h
```

9.4 使用算法套件快速完成水表读数识别

本示例围绕真实AI需求场景，介绍算法开发套件在水表表盘读数识别算法开发任务上的使用流程。

算法开发套件中目前提供自研(ivg系列)和开源(mm系列)共两套算法资产，可应用于分类、检测、分割和OCR等任务中。本示例中将组合使用自研分割算法(ivgSegmentation)和开源OCR算法(mmOCR)完成水表读数识别项目，并使用算法开发套件将其部署为华为云在线服务。

版权所有 © 华为云计算技术有限公司

> **说明**
>
> 本案例教程仅适用于“华北-北京四”区域Notebook。

准备数据

1. 登录OBS控制台，创建OBS对象桶，区域选择“华北-北京四”。
2. 登录ModelArts控制台，选择控制台区域为“华北-北京四”。
3. 在“全局配置”页面查看是否已经配置授权，允许ModelArts访问OBS。如果没有配置授权，请参考**配置访问授权（全局配置）**添加授权。
4. 分别下载本案例的数据集，**水表表盘分割数据集**和**水表表盘读数OCR识别数据集**到OBS桶中，单击数据集右侧的“下载”，弹出“选择云服务区域”，选择区域后单击“确定”进入下载详情页面。下载方式选择“对象存储服务（OBS）”，填写OBS路径信息等，详细步骤请参考**下载数据集**。

   OBS路径示例如下：

   obs://{OBS桶名称}/water_meter_segmentation 水表表盘分割数据集

   obs://{OBS桶名称}/water_meter_crop 水表表盘读数OCR识别数据集

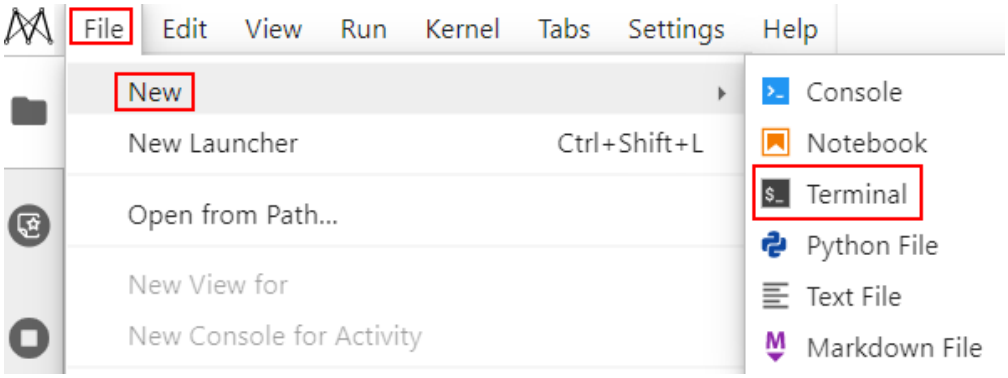
   > **说明**
   >
   > 从AI Gallery下载数据集免费，但是数据集存储在OBS桶中会收取少量费用，具体计费请参见**OBS价格详情页**，案例使用完成后请及时清除资源和数据。

准备开发环境

1. 在ModelArts控制台的“ 开发环境 > Notebook”页面中，创建基于pytorch1.8-cuda10.2-cudnn7-ubuntu18.04镜像，类型为GPU，规格选择**Pnt1**或**Vnt1**系列的Notebook，具体操作请参见**创建Notebook实例**章节。

   如果需要使用本地IDE（PyCharm或VS Code）远程连接Notebook，需要开启SSH远程开发。本案例以在线的JupyterLab为例介绍整个过程。
2. Notebook创建完成后，状态为“运行中”。单击“操作”栏的“打开”，进入JupyterLab页面。
3. 打开JupyterLab的Terminal。此处以Terminal为例介绍整个过程。JupyterLab更多操作请参见**JupyterLab简介及常用操作**。

图 9-10 打开 Terminal

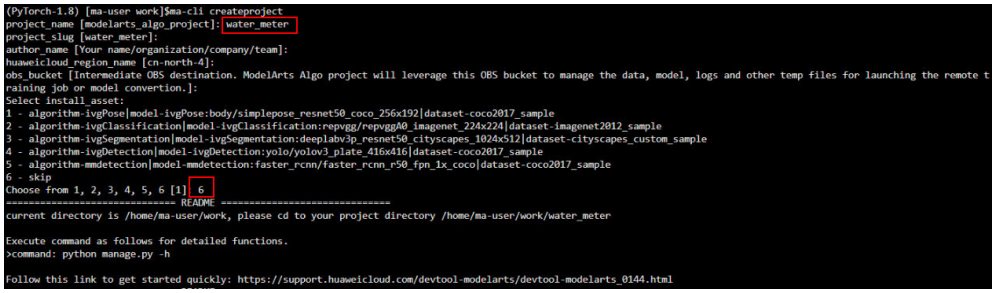


 版权所有 © 华为云计算技术有限公司

Step1 创建算法工程

1. 在JupyterLab的Terminal中，在work目录下执行**ma-cli createproject**命令创建工程，根据提示输入工程名称，例如：water_meter。然后按回车键选择默认参数（连续按五次回车），并选择跳过资产安装步骤（选择6）。

   图 9-11 创建工程

2. 执行以下命令进入工程目录。

   ```
   cd water_meter
   ```

3. 执行以下命令复制项目数据到Notebook中。

   ```
   python manage.py copy --source {obs_dataset_path} --dest ./data/raw/water_meter_crop
   python manage.py copy --source {obs_dataset_path} --dest ./data/raw/
   water_meter_segmentation
   ```

   说明

   {obs_dataset_path}路径为**Step1 准备数据**中下载到OBS中的数据集路径，比如“obs://{OBS桶名称}/water_meter_segmentation”和“obs://{OBS桶名称}/water_meter_crop”

   图 9-12 复制数据集到 Notebook 中

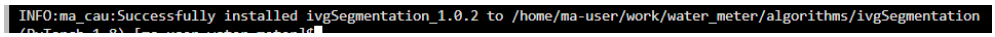
   

Step2 使用 deeplabv3 完成水表区域分割任务

1. 执行如下命令安装ivgSegmentation套件。

   ```
   python manage.py install algorithm ivgSegmentation==1.0.2
   ```

   图 9-13 ivgSegmentation 套件安装成功

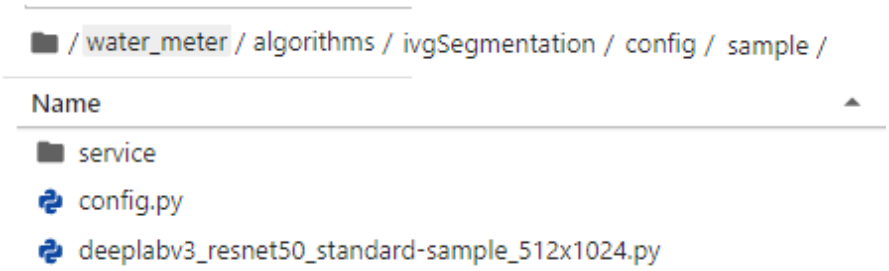
   

   如果提示ivgSegmentation版本不正确，可以通过命令**python manage.py list algorithm**查询版本。

2. 安装ivgSegmentation套件后，在JupyterLab界面左侧的工程目录中进入“./algorithms/ivgSegmentation/config/sample”文件夹中查看目前支持的分割模型，以sample为例（sample默认的算法就是deeplabv3），文件夹中包括config.py（算法外壳配置）和deeplabv3_resnet50_standard-sample_512x1024.py（模型结构）。

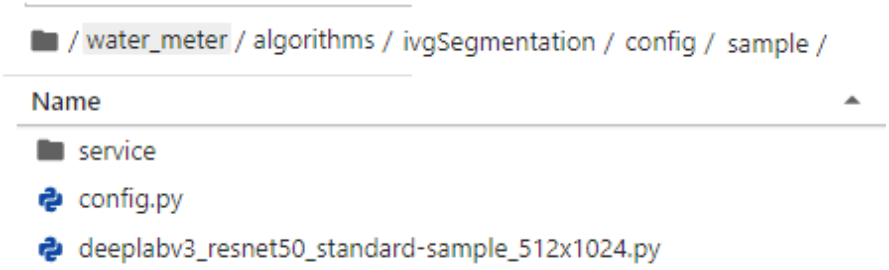
版权所有 © 华为云计算技术有限公司

图 9-14 进入 sample 文件夹



3. 表盘分割只需要区分背景和读数区域，因此属于二分类，需要根据项目所需数据集对配置文件进行修改，如下所示：
   修改“config.py”文件。

   图 9-15 修改 sample 文件夹下的 config.py 文件

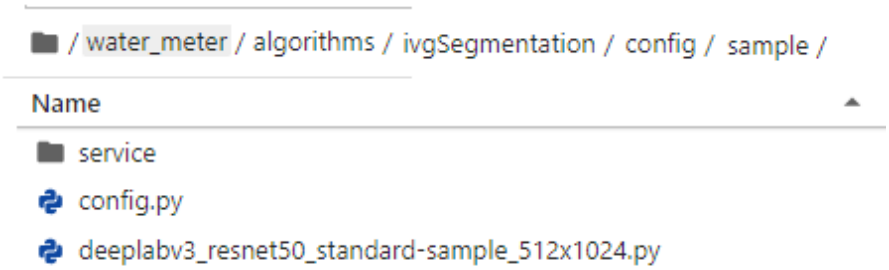
   

   # config.py

   ```
   ...
   alg_cfg = dict(
   ...
   data_root='data/raw/water_meter_segmentation',  # 修改为真实路径本地分割数据集路径
   ...
   )
   ```

   修改完后按Ctrl+S保存。
4. 修改“deeplabv3_resnet50_standard-sample_512x1024.py”文件。

   图 9-16 修改 deeplabv3_resnet50_standard-sample_512x1024.py 文件

   # deeplabv3_resnet50_standard-sample_512x1024.py

 版权所有 © 华为云计算技术有限公司

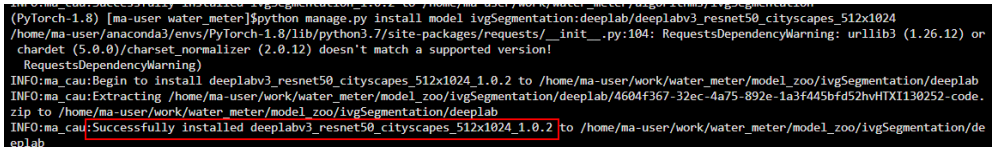
```
gpus=[0]
...
data_cfg = dict(
    ...    num_classes=2,  # 修改为2类
    ...
    ...    train_scale=(512, 512),  # (h, w)#size全部修改为(512, 512)
    ...    train_crop_size=(512, 512),  # (h, w)
    ...    test_scale=(512, 512),  # (h, w)
    ...    infer_scale=(512, 512),  # (h, w)
)
```

修改完按Ctrl+S保存。

5. 在water_meter工程目录下，执行如下命令安装deeplabv3预训练模型。

```
python manage.py install model ivgSegmentation:deeplab/
deeplabv3_resnet50_cityscapes_512x1024
```

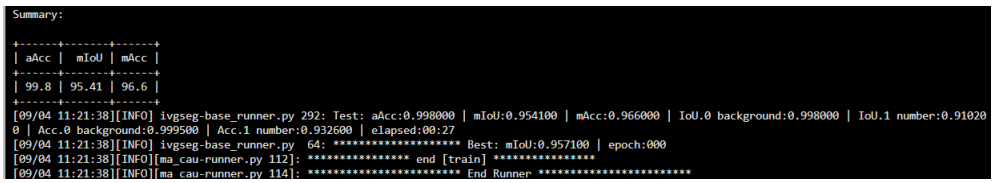
图 9-17 安装 deeplabv3 预训练模型



6. 执行如下命令训练分割模型。（推荐使用GPU进行训练）

```
python manage.py run --cfg algorithms/ivgSegmentation/config/sample/config.py --gpus 0
```

图 9-18 分割模型训练结果



训练好的模型会保存在指定位置中，默认为“./output/deeplabv3_resnet50_standard-sample_512x1024/checkpoints/”中。

7. 验证模型效果。

模型训练完成后，可以在验证集上计算模型的指标，首先修改配置文件的模型位置。

修改“config.py”文件，修改完按Ctrl+S保存。

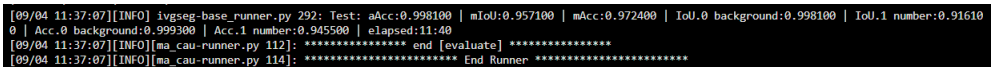
config.py

```
    ...
alg_cfg = dict(
    ...
    load_from='./output/deeplabv3_resnet50_standard-sample_512x1024/checkpoints/
checkpoint_best.pth.tar',  # 修改训练模型的路径
    ...
)
```

执行如下命令计算模型指标。

```
python manage.py run --cfg algorithms/ivgSegmentation/config/sample/config.py --
pipeline evaluate
```

图 9-19 模型指标计算结果



 版权所有 © 华为云计算技术有限公司

8. 模型推理。

   模型推理能够指定某一张图片，并且推理出图片的分割区域，并进行可视化，首先需要指定需要推理的图片路径。

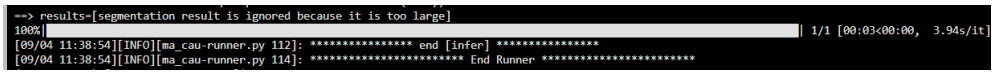
   修改“config.py”文件，修改完按Ctrl+S保存。

   ```
   alg_cfg = dict(
       ...
     img_file='./data/raw/water_meter_segmentation/image/train_10.jpg'  # 指定需要推理的图片路径
     ...
   )
   ```

   执行如下命令推理模型。

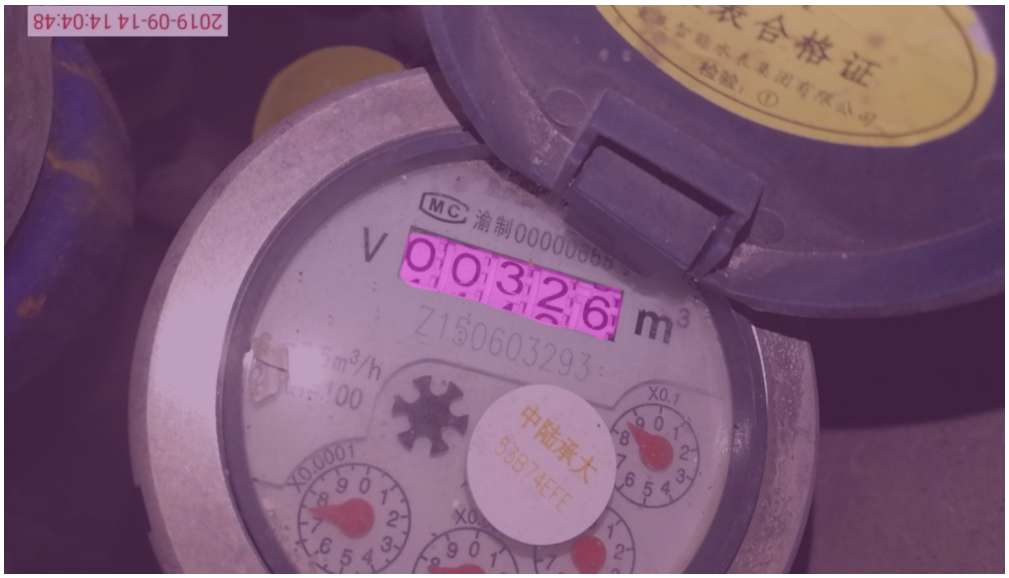
   ```
   python manage.py run --cfg algorithms/ivgSegmentation/config/sample/config.py --pipeline infer
   ```

   图 9-20 表盘分割模型推理结果

   推理输出的图片路径在“./output/deeplabv3_resnet50_standard-sample_512x1024”下。

   图 9-21 水表表盘分割结果可视化

9. 执行如下命令导出算法SDK。

   ```
   python manage.py export --cfg algorithms/ivgSegmentation/config/sample/config.py --is_deploy
   ```

   算法开发套件支持将模型导出成一个模型SDK，方便进行模型部署等下游任务。SDK导出的路径为“./export/deeplabv3_resnet50_standard-sample_512x1024/Linux_x86_64_GPU_PyTorch_Common_py”

版权所有 © 华为云计算技术有限公司

图 9-22 SDK 导出路径

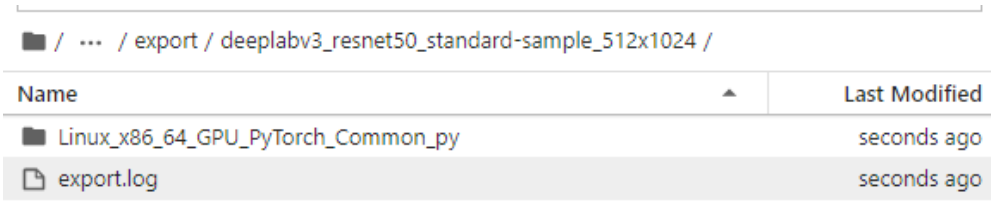
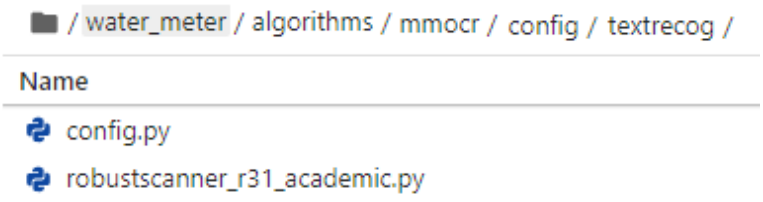


图 9-23 SDK 导出示意图

Step3 水表读数识别

1. 执行如下命令安装mmocr套件。
   ```
   python manage.py install algorithm mmocr==0.2.1
   ```
2. 安装mmocr套件后，“./algorithms/mmocr/config/textrecog”文件夹中包括config.py（算法外壳配置）和robustscanner_r31_academic.py（模型结构），需要根据所需算法和数据集路径修改配置文件。以下以robust_scanner算法为例。

   图 9-24 进入 textrecog 文件夹

   修改“robustscanner_r31_academic.py”，如下所示：

   ```
   ...
   train_prefix = 'data/raw/water_meter_crop/'  # 修改数据集路径改为水表ocr识别数据集路径
   train_img_prefix1 = train_prefix + 'train'
   train_ann_file1 = train_prefix + 'train.txt'

   ...
   ```

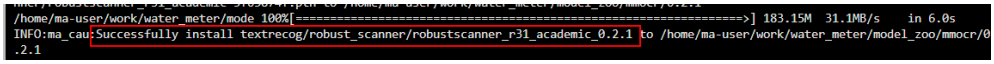

版权所有 © 华为云计算技术有限公司

```
test_prefix = 'data/raw/water_meter_crop/'
test_img_prefix1 = test_prefix + 'val/'
test_ann_file1 = test_prefix + 'val.txt'
```

3. 执行如下命令安装robust_scanner预训练模型。

```
python manage.py install model mmocr:textrecog/robust_scanner/
robustscanner_r31_academic
```

图 9-25 安装 robust_scanner 模型



4. 训练OCR模型。

初次使用mmcv时需要编译mmcv-full，该过程较慢，可以直接使用官方预编译的依赖包。

预编译包URL： **https://download.openmmlab.com/mmcv/dist/cu102/torch1.6.0/index.html**

```
pip uninstall mmcv -y
pip install https://download.openmmlab.com/mmcv/dist/cu102/torch1.6.0/mmcv_full-1.3.9-cp37-cp37m-manylinux1_x86_64.whl
pip install rapidfuzz==2.15.1
pip install numpy -U
```

修改“./algorithms/mmocr/config/textrecog/config.py”，将EPOCHS（迭代数量）改为2。

图 9-26 修改 textrecog 文件夹下的 config.py 文件

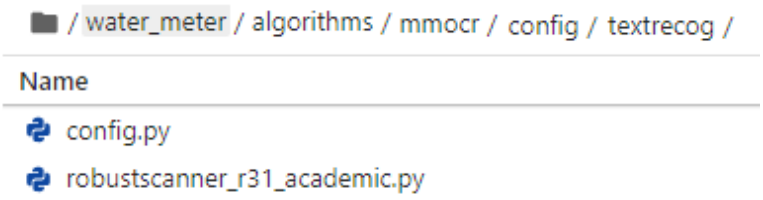


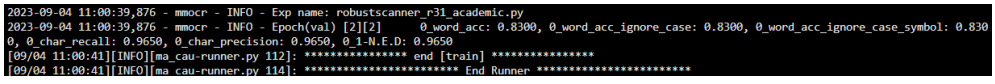
图 9-27 迭代数量修改为 2

```
BATCH_SIZE = 128
EPOCHS = 2
LR = 1e-3

# /cache is recommended for data root
```

执行如下命令训练OCR模型。（仅使用GPU进行训练，大概需要几分钟）

```
python manage.py run --cfg algorithms/mmocr/config/textrecog/config.py
```

图 9-28 OCR 模型训练结果



训练好的模型会保存在指定位置中，默认为output/robustscanner_r31_academic/文件夹中。

 版权所有 © 华为云计算技术有限公司

5. 验证模型效果。

   模型训练完成后，可以在验证集上计算模型的指标，首先修改配置文件的模型位置。

   修改“./algorithms/mmocr/config/textrecog/config.py”

   #config.py

   ```
   ...
   model_path = './output/robustscanner_r31_academic/latest.pth'
   ...
   ```

   执行如下命令验证模型。

   ```
   python manage.py run --cfg algorithms/mmocr/config/textrecog/config.py --pipeline evaluate
   ```

   图 9-29 计算模型的指标

   ```
   Evaluateing data/raw/water_meter_crop/val.txt with 200 images now
   {'0_word_acc': 0.83, '0_word_acc_ignore_case': 0.83, '0_word_acc_ignore_case_symbol': 0.83, '0_char_recall': 0.965, '0_char_precision': 0.965, '0_1-N.E
   D': 0.965}
   [09/04 11:03:39][INFO][ma_cau-runner.py 112]: ********** end [evaluate] **********
   [09/04 11:03:39][INFO][ma_cau-runner.py 114]: ********** End Runner **********
   ```

6. **可选:** 模型推理。

   模型推理能够指定某一张图片，并且推理出图片的分割区域，并进行可视化。首先需要指定待推理的图片路径，修改“./algorithms/mmocr/config/textrecog/config.py”，具体如下。

   ```
   ...
   infer_img_file='./data/raw/water_meter_crop/val/train_10.jpg'  # 指定需要推理的图片路径
   ...
   ```

   执行如下命令推理。

   ```
   python manage.py run --cfg algorithms/mmocr/config/textrecog/config.py --pipeline infer
   ```

   图 9-30 模型推理结果

   ```
   [09/04 11:06:27][INFO][ma_cau-runner.py 186]: Command:  python  algorithms/mmocr/algorithm/tools_adapter/infer.py  --config algorithms/mmocr/config/text
   recog/robustscanner_r31_academic.py --img ./data/raw/water_meter_crop/val/train_10.jpg --checkpoint ./output/robustscanner_r31_academic/latest.pth --out
   _file output/robustscanner_r31_academic/vis/vis_train_10.jpg

   Use load_from_local loader
   result: {'text': '00326', 'score': 0.9999996185302734}
   [09/04 11:06:38][INFO][ma_cau-runner.py 112]: ********** end [infer] **********
   [09/04 11:06:38][INFO][ma_cau-runner.py 114]: ********** End Runner **********
   ```

   推理输出的图片路径在“output/robustscanner_r31_academic/vis”文件夹下。

   图 9-31 表盘读数识别结果图

7. 执行如下命令导出算法SDK。

 版权所有 © 华为云计算技术有限公司

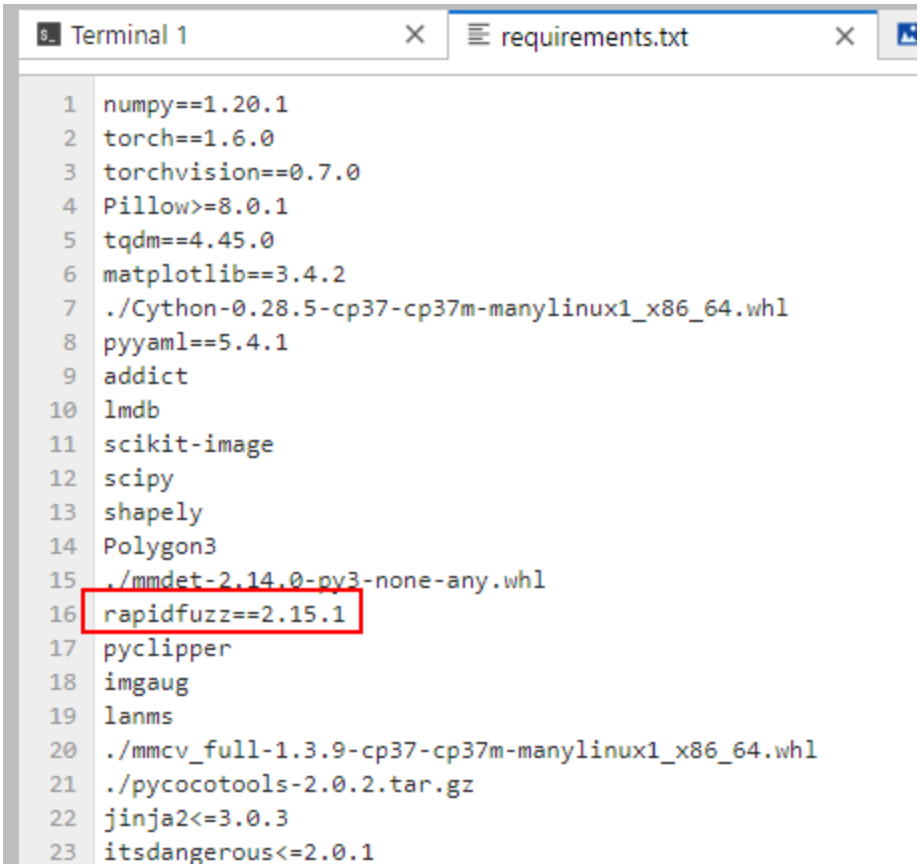
```
python manage.py export --cfg algorithms/mmocr/config/textrecog/config.py
```

Step4 部署为在线服务

本次展示仅部署OCR服务， 包括本地部署和线上部署， 部署上线后调用部署服务进行本地图片的推理，获取水表的预测读数。部署在线服务，需要指定OBS桶以便保存部署所需要的文件。

1. 修改“./algorithms/mmocr/config/textrecog/config.py”文件，配置为用户的OBS桶。
   ```
   # 替换为用户自己的OBS桶信息
   obs_bucket = 'obs://{your_obs_bucket_path}'
   ```
2. 导出模型文件并修改rapidfuzz版本，然后本地部署。
   ```
   python manage.py export --cfg algorithms/mmocr/config/textrecog/config.py --is_deploy
   # 导出部署模型所需文件
   ```
   修改“./export/robustscanner_r31_academic/Linux_x86_64_GPU_PyTorch_Common_py/res/requirements.txt”，将rapidfuzz版本修改为2.15.1

   图 9-32 修改 rapidfuzz 版本为 2.15.1


   ```
   python manage.py deploy --cfg algorithms/mmocr/config/textrecog/config.py  # 本地部署调试
   ```

   本地部署成功后的输出结果

   ```
   #
   ...
   ```

 版权所有 © 华为云计算技术有限公司

```
[Conda environment created successfully.]
local_service_port is 127.0.0.1:42153
Deploying the local service ...
Successfully deployed the local service. You can check the log in /home/ma-user/work/water_meter/export/robustscanner_r31_academic/Linux_x86_64_GPU_PyTorch_Common_py/log.txt
[07/05 09:40:14][INFO][ma_cau-deployer.py 49]: {
 "text": "00326",
 "score": 0.9999999046325684
}
[07/05 09:40:14][INFO][ma_cau-deployer.py 59]: ********** End Deployer **********
```

3. 本地部署成功后执行如下命令进行在线部署，大约需要十几分钟。

   ```
   python manage.py deploy --cfg algorithms/mmocr/config/textrecog/config.py --launch_remote
   ```

   部署成功后，任务会提交至ModelArts控制台的默认工作空间，在“部署上线 > 在线服务”页面会看到此任务。

Step5 清除资源和数据

通过此示例学习完成创建算法套件流程后，如果不再使用，建议您清除相关资源，避免造成资源浪费和不必要的费用。

- 删除Notebook：在ModelArts“开发环境 > Notebook”界面，单击对应实例“操作”列的“更多 > 删除”。
- 删除数据：前往OBS，删除上传的数据，然后删除文件夹及OBS桶。
- 停止在线服务：在ModelArts“部署上线 > 在线服务”界面，单击对应在线服务“操作”列的“更多 > 停止”。

9.5 使用样例

9.5.1 使用现有模型进行推理

本小节以姿态估计ivgPose 的预训练模型simplepose_resnet50_coco_256x192为例进行图片人物的姿势估计推理。

Step0 准备环境

使用前需要先准备开发环境。

1. 参考**准备开发环境**章节创建并打开Notebook实例。
2. 参考**创建算法工程**章节创建算法工程。

Step1 安装算法套件

安装ivgPose算法套件。

```
python manage.py install algorithm ivgPose
```

Step2 安装预训练模型

默认下载到当前工程的./model_zoo/{算法类型}/{算法版本}目录下。

```
python manage.py install model ivgPose:body/simplepose_resnet50_coco_256x192
```


版权所有 © 华为云计算技术有限公司

Step3 推理

1. 准备好待推理的图片，本小节以算法套件里内置的推理图片为例，原图片如下。

   图 9-33 待推理的图片

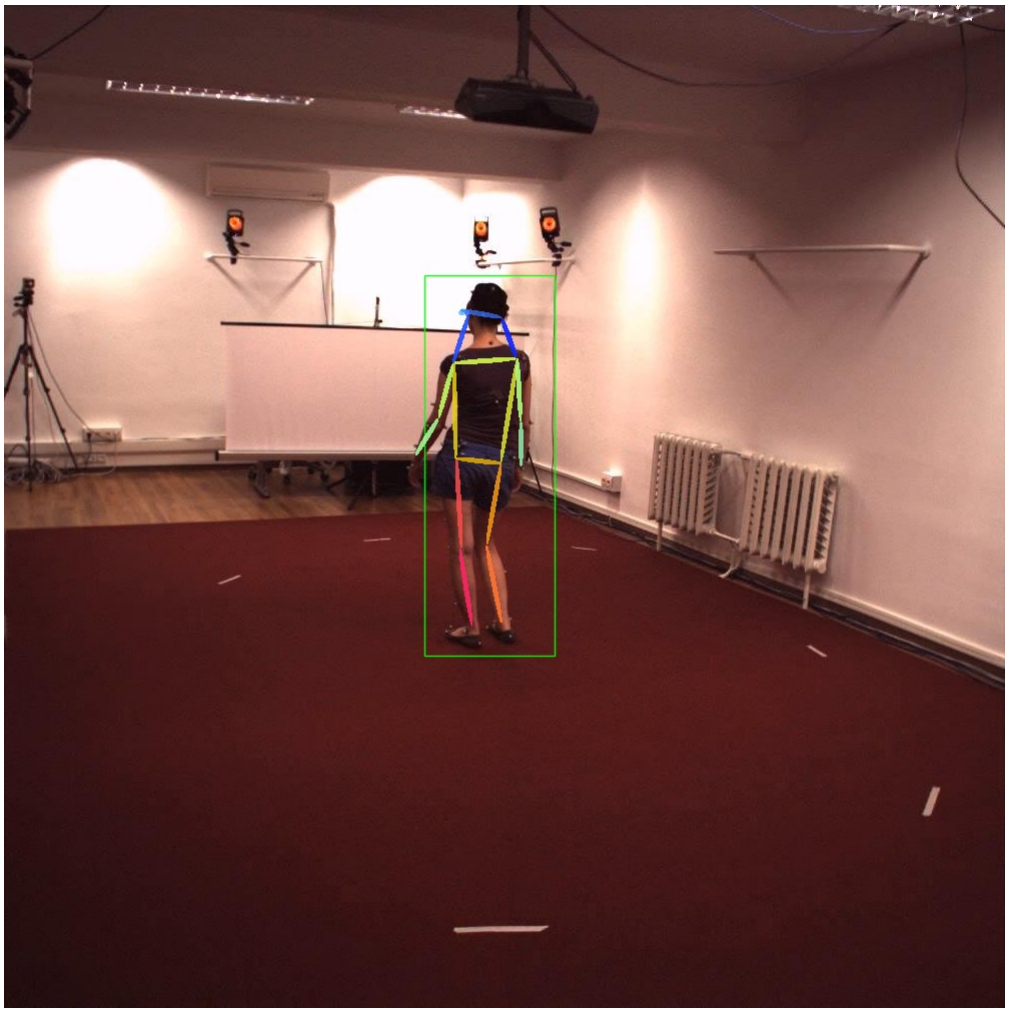
2. 在Terminal中执行下述推理命令，其中--img_file的值为待推理图片的路径。

   ```
   python manage.py run --cfg algorithms/ivgPose/config/sample/config.py --pipeline infer --demo_type image_demo --load_from model_zoo/ivgPose/body/simplepose_resnet50_coco_256x192.pth.tar --img_path algorithms/ivgPose/algorithm/examples/images/body/human36m_s_01_act_02_subact_01_ca_01_000001.jpg  --is_show
   ```

3. 运行完毕后，在当前目录的output/simplepose_resnet50_coco_256x192/下可以看到推理后的文件human36m_s_01_act_02_subact_01_ca_01_000001_vis.jpg，打开后显示如下。

 版权所有 © 华为云计算技术有限公司

图 9-34 推理后的文件



9.5.2 使用现有数据集测试现有模型

为了评估模型的准确性，通常会基于标准数据集来测试模型。算法套件当前发布了五种mini数据集，包括coco2017_sample，imagenet2012_sample，cityscapes_custom_sample，coco_stuff和icdar_det。

本章节将介绍如何基于已有数据集测试现有模型。

Step0 准备环境

使用前需要先准备开发环境。

1. 参考**准备开发环境**章节创建并打开Notebook实例。
2. 参考**创建算法工程**章节创建算法工程。

Step1 安装算法套件

安装ivgPose算法套件。

```
python manage.py install algorithm ivgPose
```


版权所有 © 华为云计算技术有限公司

Step2 安装数据集

安装coco数据集。

```
python manage.py install dataset coco2017_sample
```

Step3 安装预训练模型

默认下载到当前工程的./model_zoo/{算法类型}目录下。

```
python manage.py install model ivgPose:body/simplepose_resnet50_coco_256x192
```

Step4 修改配置文件

打开预训练模型对应算法资产的配置文件路径，如./algorithms/ivgPose/config/sample/config.py。按需进行验证相关参数的修改，如修改数据集路径data_root为下载的mini数据集路径./data/raw/coco2017_sample、指定验证阶段使用的验证数据集路径等。如下述代码所示。

```
alg_cfg = dict(
    cfg=f'{work_dir}/config/sample/{alg_name}.py',
    data_root='data/raw/coco2017_sample',
    load_from=f'model_zoo/{alg_type}/body/{alg_name}.pth.tar',
    pretrained=f'model_zoo/{alg_type}/backbone/resnet50_imagenet_224x224.pth',
    img_file=f'{work_dir}/algorithm/examples/images/body/coco_000000013729.jpg',
    bbox_file=f'{work_dir}/algorithm/examples/images/body/bboxes.json'
)
    evaluate_args=dict(
        cfg=alg_cfg['cfg'],
        output_dir=run_dir,
        data_root=alg_cfg['data_root'],
        load_from=alg_cfg['load_from'],
        gpus=None,
        samples_per_gpu=None,
    ),
```

Step5 本地验证

在Terminal中执行下述测试命令，其中：--cfg为该预训练模型对应算法资产的配置文件路径，--load_from的值为待测试的模型路径。

```
python manage.py run --cfg algorithms/ivgPose/config/sample/config.py --pipeline evaluate --load_from ./model_zoo/ivgPose/body/simplepose_resnet50_coco_256x192.pth.tar
```

运行完毕后，在交互式输出界面或config.py配置的{run_dir}目录下的test.log可以看到验证过程的日志和结果。如下述信息所示：

```
ivgpose-utils.py 133: Test: AP:0.735012 | AP .5:0.925143 | AP .75:0.813647 | AP (M):0.705874 | AP (L):0.778985 | AR:0.764798 | AR .5:0.933879 | AR .75:0.832966 | AR (M):0.732641 | AR (L):0.813304 | elapsed:00:28
```

9.5.3 使用现有数据集训练预训练模型

本章节将介绍如何基于已有数据集训练一个预训练模型。

算法套件当前发布了五种mini数据集，包括coco2017_sample，imagenet2012_sample，cityscapes_custom_sample，coco_stuff和icdar_det。

版权所有 © 华为云计算技术有限公司

Step0 准备环境

使用前需要先准备开发环境。

1. 参考**准备开发环境**章节创建并打开Notebook实例。
2. 参考**创建算法工程**章节创建算法工程。

Step1 安装算法套件

安装ivgPose算法套件。

```
python manage.py install algorithm ivgPose
```

Step2 安装数据集

安装coco数据集。

```
python manage.py install dataset coco2017_sample
```

Step3 安装预训练模型

默认下载到当前工程的./model_zoo/{算法类型}/{算法版本}目录下。

```
python manage.py install model ivgPose:body/simplepose_resnet50_coco_256x192
python manage.py install model ivgPose:backbone/resnet50_imagenet_224x224
```

Step4 修改配置文件

配置文件主要包括算法配置文件和算法参数文件，例如：

- 算法配置文件：./algorithms/ivgPose/config/sample/config.py
- 算法参数文件：./algorithms/ivgPose/config/sample/simplepose_resnet50_coco_256x192.py

根据需要修改训练相关参数，如数据集路径data_root、runner中的train_args参数以及train参数等。如下述代码所示。

config.py文件

```
train_args=dict(
    cfg=alg_cfg['cfg'],
    output_dir=run_dir,
    data_root=alg_cfg['data_root'],
    load_from=None,
    gpus=None,
    lr=None,
    max_epoch=5,
    samples_per_gpu=None,
    resume_from=None,
    pretrained=alg_cfg['pretrained'],
    train_data_root=None,
    val_data_root=None,
),
```

simplepose_resnet50_coco_256x192.py文件

```
train = dict(
```

 版权所有 © 华为云计算技术有限公司

```
    save_period=-1,
    val_period=5,
    best_saver=dict(key_indicator='AP', rule='max'),
    ema=dict(type='ExpEMA')
)
```

Step5 本地训练

在Terminal中执行下述训练命令，其中：--cfg为该预训练模型对应算法资产的配置文件路径。

```
python manage.py run --cfg algorithms/ivgPose/config/sample/config.py --gpus 0
```

运行完毕后，在交互式输出界面或config.py配置的{run_dir}目录下的train_{timestamp}.log可以看到训练过程的日志和结果。如下述信息所示。

```
Train: 000/003 | loss:0.002145 | acc_pose:0.029999 | lr:{'0.000001', '0.000005'}: 100%|
| 397/397 [00:36<00:00, 10.73it/s]
[07/19 15:36:30][INFO] ivgpose-utils.py  52: Train: 000/003 | loss:0.002145 | acc_pose:0.029999 | lr:
{'0.000001', '0.000005'} | elapsed:00:36
Val: 000/003 | loss:0.002451 | acc_pose:0.060385 | : 100%|
| 397/397 [00:16<00:00, 24.09it/s]
[07/19 15:36:53][INFO] ivgpose-utils.py 101: Val: 000/003 | loss:0.002451 | acc_pose:0.060385 | elapsed:00:16
[07/19 15:36:55][INFO] ivgpose-utils.py 167: ******************** Best: acc_pose:0.060385 | epoch:000
Train: 001/003 | loss:0.002144 | acc_pose:0.043690 | lr:{'0.000170', '0.000017'}: 100%|
| 397/397 [00:35<00:00, 11.10it/s]
[07/19 15:37:31][INFO] ivgpose-utils.py  52: Train: 001/003 | loss:0.002144 | acc_pose:0.043690 | lr:
{'0.000170', '0.000017'} | elapsed:00:35
Train: 002/003 | loss:0.002148 | acc_pose:0.053395 | lr:{'0.000034', '0.000335'}: 100%|
| 397/397 [00:35<00:00, 11.13it/s]
[07/19 15:38:13][INFO] ivgpose-utils.py  52: Train: 002/003 | loss:0.002148 | acc_pose:0.053395 | lr:
{'0.000034', '0.000335'} | elapsed:00:35
Val: 002/003 | loss:0.002407 | acc_pose:0.088161 | : 100%|
| 397/397 [00:16<00:00, 23.41it/s]
[07/19 15:38:36][INFO] ivgpose-utils.py 101: Val: 002/003 | loss:0.002407 | acc_pose:0.088161 | elapsed:00:16
[07/19 15:38:39][INFO] ivgpose-utils.py 167: ******************** Best: acc_pose:0.088161 | epoch:002
```

9.5.4 使用自定义数据集进行训练

ivgSegmentation支持cityscapes_custom_sample、[Pascal VOC](下载地址：**http://host.robots.ox.ac.uk/pascal/VOC/index.html**)和自定义数据集。

Step0 准备环境

使用前需要先准备开发环境。

1. 参考**准备开发环境**章节创建并打开Notebook实例。
2. 参考**创建算法工程**章节创建算法工程。

Step1 上传数据集到开发环境中

按照不同算法定义的数据结构和格式准备好数据集。

- 如果数据集在OBS桶里，在Terminal里可执行下述命令将OBS里的数据下载到Notebook环境中。
  ```
  python manage.py copy --source obs://my_bucket/my_data_dir/ --dest /home/ma-user/work/my_data_dir
  ```
- 如果数据集比较小且存放在PC上。

  如果使用的本地IDE是VS Code，可以通过拖放方式复制到VS Code连接的Notebook中。

版权所有 © 华为云计算技术有限公司

自定义数据集结构如下：

```
├── my_dataset
│   ├── image
│   │   ├── train
│   │   │   ├── xxx{img_suffix}
│   │   │   ├── yyy{img_suffix}
│   │   │   ├── zzz{img_suffix}
│   │   │   ├── ...
│   │   ├── val
│   ├── annotation
│   │   ├── train
│   │   │   ├── xxx{seg_map_suffix}
│   │   │   ├── yyy{seg_map_suffix}
│   │   │   ├── zzz{seg_map_suffix}
│   │   │   ├── ...
│   │   ├── val
```

Step2 修改配置文件

配置文件主要包括算法配置文件和算法参数文件，例如：

- 算法配置文件：./algorithms/ivgPose/config/sample/config.py
- 算法参数文件：./algorithms/ivgPose/config/sample/simplepose_resnet50_coco_256x192.py

根据需要修改训练相关参数，如数据集路径data_root、runner中的train_args参数以及train参数等。如下述代码所示。

config.py文件

```
train_args=dict(
    cfg=alg_cfg['cfg'],
    output_dir=run_dir,
    data_root=alg_cfg['data_root'],
    load_from=None,
    gpus=None,
    lr=None,
    max_epoch=5,
    samples_per_gpu=None,
    resume_from=None,
    pretrained=alg_cfg['pretrained'],
    train_data_root=None,
    val_data_root=None,
),
```

simplepose_resnet50_coco_256x192.py文件

```
train = dict(
    save_period=-1,
    val_period=5,
    best_saver=dict(key_indicator='AP', rule='max'),
    ema=dict(type='ExpEMA')
)
```

Step3 在开发环境中进行训练

在Terminal中执行下述训练命令，其中：--cfg为该预训练模型对应算法资产的配置文件路径。

```
python manage.py run --cfg algorithms/ivgPose/config/sample/config.py --gpus 0
```


版权所有 © 华为云计算技术有限公司

运行完毕后，在交互式输出界面或config.py配置的{run_dir}目录下的train_{timestamp}.log可以看到训练过程的日志和结果。如下述信息所示。

```
Train: 000/003 | loss:0.002145 | acc_pose:0.029999 | lr:{'0.000001', '0.000005'}: 100%|████████████████████| 397/397 [00:36<00:00, 10.73it/s]
[07/19 15:36:30][INFO] ivgpose-utils.py  52: Train: 000/003 | loss:0.002145 | acc_pose:0.029999 | lr:{'0.000001', '0.000005'} | elapsed:00:36
Val: 000/003 | loss:0.002451 | acc_pose:0.060385 | : 100%|████████████████████████████████████| 397/397 [00:16<00:00, 24.09it/s]
[07/19 15:36:53][INFO] ivgpose-utils.py 101: Val: 000/003 | loss:0.002451 | acc_pose:0.060385 | elapsed:00:16
[07/19 15:36:55][INFO] ivgpose-utils.py 167: ******************** Best: acc_pose:0.060385 | epoch:000
Train: 001/003 | loss:0.002144 | acc_pose:0.043690 | lr:{'0.000170', '0.000017'}: 100%|████████████████████| 397/397 [00:35<00:00, 11.10it/s]
[07/19 15:37:31][INFO] ivgpose-utils.py  52: Train: 001/003 | loss:0.002144 | acc_pose:0.043690 | lr:{'0.000170', '0.000017'} | elapsed:00:35
Train: 002/003 | loss:0.002148 | acc_pose:0.053395 | lr:{'0.000034', '0.000335'}: 100%|████████████████████| 397/397 [00:35<00:00, 11.13it/s]
[07/19 15:38:13][INFO] ivgpose-utils.py  52: Train: 002/003 | loss:0.002148 | acc_pose:0.053395 | lr:{'0.000034', '0.000335'} | elapsed:00:35
Val: 002/003 | loss:0.002407 | acc_pose:0.088161 | : 100%|████████████████████████████████████| 397/397 [00:16<00:00, 23.41it/s]
[07/19 15:38:36][INFO] ivgpose-utils.py 101: Val: 002/003 | loss:0.002407 | acc_pose:0.088161 | elapsed:00:16
[07/19 15:38:39][INFO] ivgpose-utils.py 167: ******************** Best: acc_pose:0.088161 | epoch:002
```

Step4 在开发环境中进行验证

在Terminal中执行下述测试命令，其中：--cfg为该预训练模型对应算法资产的配置文件路径，--load_from的值为待测试的模型路径，config.py为：

```
python manage.py run --cfg algorithms/ivgPose/config/sample/config.py --pipeline evaluate --load_from ./model_zoo/ivgPose/body/simplepose_resnet50_coco_256x192.pth.tar
```

运行完毕后，在交互式输出界面或config.py配置的{run_dir}目录下的test.log可以看到验证过程的日志和结果。如下述信息所示：

```
ivgpose-utils.py 133: Test: AP:0.735012 | AP .5:0.925143 | AP .75:0.813647 | AP (M):0.705874 | AP (L):0.778985 | AR:0.764798 | AR .5:0.933879 | AR .75:0.832966 | AR (M):0.732641 | AR (L):0.813304 | elapsed:00:28
```

Step5 推理

1. 准备好待推理的图片，本小节以算法套件里内置的推理图片为例，原图片如下：

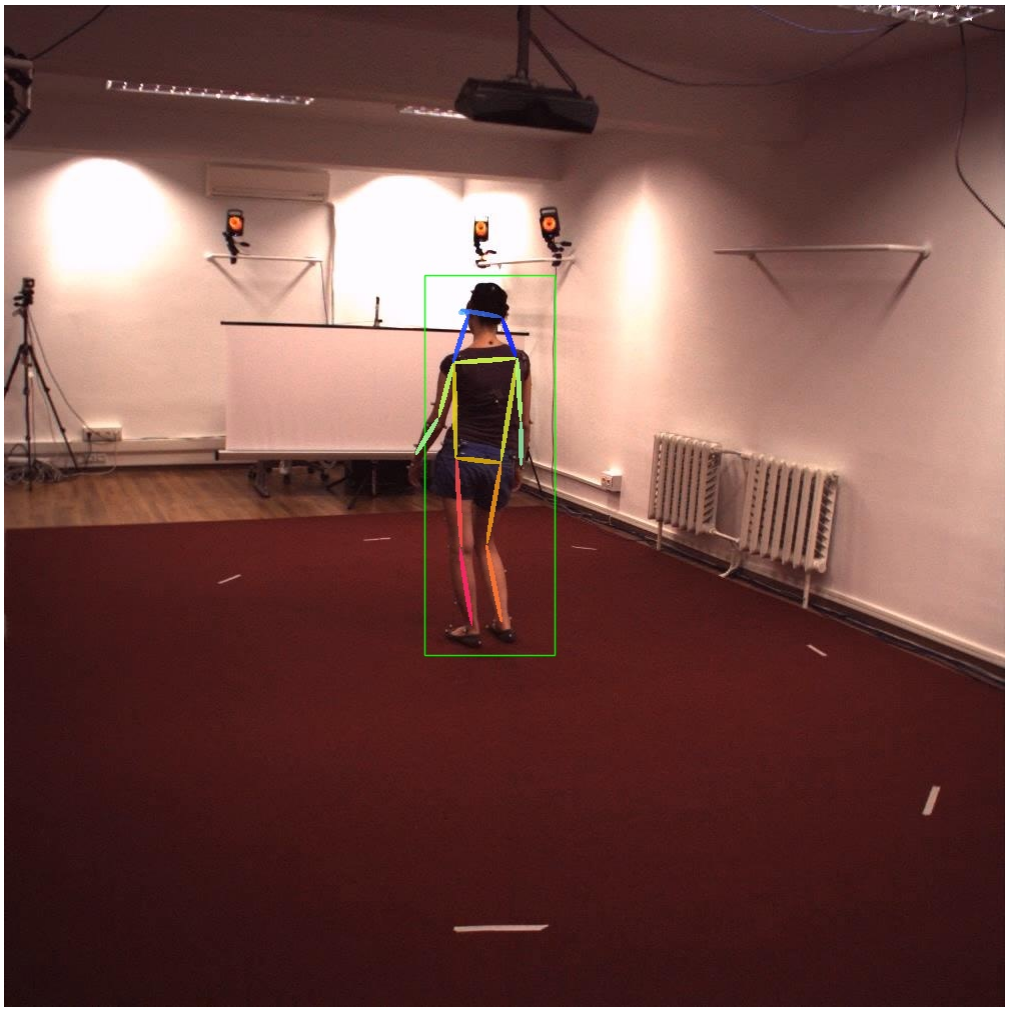
 版权所有 © 华为云计算技术有限公司

图 9-35 待推理的图片

2. 在Terminal中执行下述推理命令，其中--img_file的值为待推理图片的路径：
```
python manage.py run --cfg algorithms/ivgPose/config/sample/config.py --pipeline infer --demo_type image_demo --load_from model_zoo/ivgPose/body/simplepose_resnet50_coco_256x192.pth.tar --img_path algorithms/ivgPose/algorithm/examples/images/body/human36m_s_01_act_02_subact_01_ca_01_000001.jpg  --is_show
```
3. 运行完毕后，在当前目录的export/exp_tmp下可以看到推理后的文件，打开后显示如下。

 版权所有 © 华为云计算技术有限公司

图 9-36 推理后的文件



Step6 提交 ModelArts 训练作业

1. 参考Step2，修改外壳的配置文件。

   a. 填写OBS桶信息

   ```
   obs_bucket = 'obs://my_bucket/my_object'
   ```

   b. 按需修改runner里的**Adapter**参数。

   ```
   adapter=dict(
       requirements=f'{work_dir}/algorithm/requirements.txt',

       framework_type='PyTorch',
       framework_version='PyTorch-1.4.0-python3.6',
       instance_type='modelarts.p3.large.public',
       pool_id=None,

       downloads=dict(
           src=[f'{obs_bucket}/{alg_cfg["data_root"]}',
                f'{obs_bucket}/{alg_cfg["pretrained"]}'],
           dst=[alg_cfg['data_root'], alg_cfg['pretrained']],
       ),
       uploads=dict(
           src=[run_dir],
           dst=[f'{obs_bucket}/{run_dir}'],
   ```


版权所有 © 华为云计算技术有限公司

```
        ),
    ),
```

2. 在Terminal中输入下述命令来提交ModelArts训练作业完成训练。

```
python manage.py run --launch_remote --cfg algorithms/ivgPose/config/sample/config.py --gpus 0
```

提交完训练作业后，可以在ModelArts控制台交互式界面看到当前训练作业的状态（如排队中、运行中等），可以在config.py配置的{run_dir}/{训练作业名称目录}下看到ModelArts上的训练日志。

9.6 算法开发套件命令说明

9.6.1 列举

每个内置的算法资产内包含了算法套件、数据集、预训练模型，可以通过list命令查看内置的资产。

查询内置算法

```
python manage.py list algorithm
```

查询内置数据集

```
python manage.py list dataset
```

查询内置预训练模型

```
# 列举所有的内置预训练模型
> python manage.py list model

# 根据关键字过滤
> python manage.py list model --filter mmDetection
```

9.6.2 安装

安装内置算法

```
# 安装ivgPose最新版本
> python manage.py install algorithm ivgPose

# 安装ivgPose==1.0.0
> python manage.py install algorithm ivgPose==1.0.0

# 强制安装最新版本的ivgPose,将覆盖已有的ivgPose目录
> python manage.py install algorithm ivgPose --force

# 升级ivgPose到最新版本，将新建ivgPose_{version}目录
> python manage.py install algorithm ivgPose --upgrade
```

安装内置数据集

```
安装最新版本的内置coco2017_sample数据集
> python manage.py install dataset coco2017_sample

强制安装coco2017_sample==1.0.0
> python manage.py install dataset coco2017_sample==1.0.0 --force
```


版权所有 © 华为云计算技术有限公司

安装内置预训练模型

```
# 安装预训练模型ivgPose:backbone/resnet50_imagenet_224x224
> python manage.py install model ivgPose:backbone/resnet50_imagenet_224x224

# 强制安装预训练模型ivgPose:backbone/resnet50_imagenet_224x224
> python manage.py install model ivgPose:backbone/resnet50_imagenet_224x224 --force
```

9.6.3 复制

本地和 OBS 数据交互

```
# 复制本地目录到OBS
> python manage.py copy --source /home/ma-user/work/my_dir --dest obs://my_bucket/my_dir/

# 复制OBS目录到本地
> python manage.py copy --source obs://my_bucket/my_dir/ --dest /home/ma-user/work/my_dir

# 复制本地文件到OBS
> python manage.py copy --source /home/ma-user/work/my_file --dest obs://my_bucket/
my_dir/my_file

# 复制OBS文件到本地
> python manage.py copy --source obs://my_bucket/my_dir/my_file --dest /home/ma-user/
work/my_file

# 加速文件传输
当存在大量小文件时，逐个文件上传
> python manage.py copy --source obs://my_bucket/my_dir/my_file --dest /home/ma-user/
work/my_file
```

9.6.4 运行

本地模式

```
# 基于CPU训练
> python manage.py run --cfg algorithms/ivgSegmentation/config/sample/config.py

# 基于单卡GPU训练
> python manage.py run --cfg algorithms/ivgSegmentation/config/sample/config.py --gpus 0

# 基于单卡GPU验证模型
> python manage.py run --cfg algorithms/ivgSegmentation/config/sample/config.py --gpus 0 --
pipeline evaluate  --load_from output/deeplabv3_resnet50_standard-sample_1024x512/
checkpoints/checkpoint_best.pth.tar

# 基于模型simplepose_resnet50_coco_256x192.pth.tar进行图片推理
> python manage.py run --cfg algorithms/ivgPose/config/sample/config.py --pipeline infer --
demo_type image_demo --load_from model_zoo/ivgPose/body/
simplepose_resnet50_coco_256x192.pth.tar --img_path algorithms/ivgPose/algorithm/examples/
images/body/human36m_s_01_act_02_subact_01_ca_01_000001.jpg  --is_show
```

远程模式

```
# 基于四卡GPU进行训练
# 单机单进程
> python manage.py run --launch_remote --cfg algorithms/ivgPose/config/sample/config.py --
gpus 0,1,2,3
# 单机多进程
> python manage.py run --launch_remote --cfg algorithms/ivgPose/config/sample/config.py --
```

 版权所有 © 华为云计算技术有限公司

```
gpus 0,1,2,3 --local_size 4

# 基于双节点训练
# 两节点，每个节点4卡，单进程
> python manage.py run --cfg algorithms/ivgSegmentation/config/sample/config.py --gpus
0,1,2,3 --world_size 2
# 两节点，每个节点4卡，多进程
> python manage.py run --cfg algorithms/ivgSegmentation/config/sample/config.py --gpus
0,1,2,3 --local_size 4 --world_size 2
```

9.6.5 导出

```
# 导出源码、config.json和model
> python manage.py export --cfg algorithms/ivgSegmentation/config/sample/config.py

# 导出SDK和部署所需文件
> python manage.py export --cfg algorithms/ivgSegmentation/config/sample/config.py --
is_deploy
```

9.6.6 部署

9.6.6.1 部署命令

```
# 利用导出的源码、配置文件和模型，进行模型本地部署
> python manage.py deploy --cfg algorithms/ivgSegmentation/config/sample/config.py

# 云端在线部署模型
> python manage.py deploy --cfg algorithms/ivgSegmentation/config/sample/config.py --
launch_remote
```

9.6.6.2 部署脚本参考

模型部署主要由config.json和customize_service.py共同组成，其中，config.json的功能是管理模型运行所需的依赖环境，customize_service.py的功能是模型前向推理的逻辑。本篇文档将对于自研和第三方算法套件的customize_sevice.py进行介绍。

一、基础结构

如下所示是模型推理的基本逻辑，由_preprocess、_inference、_postprocess三部分组成。

表 9-1 模型推理函数解析

函数	输入	返回值	功能
_preprocess	request请求原始数据， Type(map)	numpy类型数据列表， Type(map)	接收request数据，并转换为模型可以接受的输入格式
_inference	numpy类型数据列表， Type(map)	numpy类型推理结果列表，Type(map)	对于输入数据进行前向推理，得到推理结果

版权所有 © 华为云计算技术有限公司

函数	输入	返回值	功能
_postprocess	numpy类型数据列表， Type(map)	json类型输出结果， Type(map)	将推理的结果进行后处理，得到预期的输出格式，该结果就是最终的返回值

```
class SingleNodeService(ModelService):
    '''SingleNodeModel defines abstraction for model service which loads a
    single model.
    '''

    def inference(self, data):
        '''
        Wrapper function to run preprocess, inference and postprocess functions.

        Parameters
        ----------
        data : map of object
            Raw input from request.

        Returns
        -------
        list of outputs to be sent back to client.
            data to be sent back
        '''
        data = self._preprocess(data)
        ...
        data = self._inference(data)
        ...
        data = self._postprocess(data)
        ...
        return data

    @abstractmethod
    def _inference(self, data):
        '''
        Internal inference methods. Run forward computation and
        return output.

        Parameters
        ----------
        data : map of NDArray
            Preprocessed inputs in NDArray format.

        Returns
        -------
        list of NDArray
            Inference output.
        '''
        return data

    @abstractmethod
    def _preprocess(self, data):
        '''
        Internal preprocess methods. Do transformation on raw
        inputs and convert them to NDArray.
```

 版权所有 © 华为云计算技术有限公司

```
        Parameters
        ----------
        data : map of object
            Raw inputs from request.

        Returns
        -------
        list of NDArray
            Processed inputs in NDArray format.
        '''
        return data

    @abstractmethod
    def _postprocess(self, data):
        '''
        Internal postprocess methods. Do transformation on inference output
        and convert them to MIME type objects.

        Parameters
        ----------
        data : map of NDArray
            Inference output.

        Returns
        -------
        list of object
            list of outputs to be sent back.
        '''
        return data
```

二、自研套件

ivgDetection是自研的目标检测算法套件之一，此处以预置的yolov3车牌检测模型（pytorch）为例，介绍ivgDetection套件的推理文件书写方式。

2.1 模型初始化

模型初始化只会在模型初次部署时调用，主要包括模型准备工作，包含模型构建和模型导入等内容。

```
...
# PyTorch runtime
from model_service.pytorch_model_service import PTServingBaseService
from base.config import CfgNode
from infer import Controller
...

class ImageClassificationService(PTServingBaseService):
    def __init__(self, model_name, model_path, **kwargs):
        cfg = CfgNode()
        cfg.model_dir = osp.join(osp.dirname(osp.abspath(__file__)), 'res')
        cfg.devices = list(range(torch.cuda.device_count())) if torch.cuda.is_available() else []
        cfg.max_batch_size = 1
        cfg.is_vis = False

        controller = Controller(cfg=cfg)
        # 初始化模型并载入训练好的模型，模型结构和模型路径都是在算法开发套件的config.py中指
定的
        self.infer = controller.create_ruinner(infer_type='ImageDetRunner')

        if torch.cuda.is_available():
```

 版权所有 © 华为云计算技术有限公司

```
            logger.info('Using GPU for inference')
        else:
            logger.info('Using CPU for inference')
```

2.2 _pre_process

```
    def _preprocess(self, data: dict) -> dict:
        imgs = []
        for k, v in data.items():
            if k == 'images':
                for file_name, file_content in v.items():

                    img = np.asarray(Image.open(file_content), dtype=np.uint8)

                    imgs.append(img)
            # 返回一个字典，其中字典的键可以是任意的，对应的值就是numpy类型的图像列表
        preprocess_data = dict(
            inputs=(imgs,)
        )
        return preprocess_data
```

2.3 _inference

```
    def _inference(self, data: dict) -> dict:
    # 这里的input就是_pre_process中返回的字典的键，需要进行对应
        inputs = data['inputs']
        # 前向推理
        results, _ = self.infer(*inputs)

        inference_data = dict(
            results=results
        )
        return inference_data
```

2.4 _post_process

```
    def _postprocess(self, data: dict) -> dict:
        results = data['results']
        # 将推理结果results进行后处理，如下的后处理方式可以直接在网页端显示bbox可视化结果
        bboxes, scores, classes = [], [], []
        work_dir = osp.dirname(osp.abspath(__file__))

        # 类别索引对照表，存放在模型导出后的res/config.json文件中
        label_list = json.load(open(osp.join(work_dir, 'res','config.json')))['infer']['visualizer']
['label_list']
        for item in results['preds'][0]:
            bboxes.append([item[1], item[0], item[3], item[2]])
            scores.append(item[-2])
            classes.append(label_list[int(item[-1])])

        detection = dict(
            detection_classes=classes,
            detection_boxes=bboxes,
            detection_scores=scores
        )

        return detection
```

 版权所有 © 华为云计算技术有限公司

图 9-37 ivgDetection 在线部署


```
{
  "detection_classes": [
    "plate"
  ],
  "detection_boxes": [
    [
      264.01228502061633,
      311.3196867058076,
      343.07130951693404,
      546.5949329054499
    ]
  ],
  "detection_scores": [
    0.8014615178108215
  ]
}
```


上述可视化后处理方式还需要配合config.json一起指定输出的数据结构，用户需要在config.json中的response中写成：

```
...
...
    "response": {
        "Content-type": "multipart/form-data",
        "data": {
            "type": "object",
            "properties": {
                "detection_classes": {
                    "type": "array",
                    "items": [{
                        "type": "string"
                    }]
                },
                "detection_boxes": {
                    "type": "array",
                    "items": [{
                        "type": "array",
                        "minItems": 4,
                        "maxItems": 4,
                        "items": [{
                            "type": "number"
                        }]
                    }]
                },
                "detection_scores": {
                    "type": "array",
                    "items": [{
                        "type": "number"
                    }]
                }
            }
        }
    }
```

三、开源套件

mmdetection是算法开发套件支持的第三方目标检测算法套件之一，我们以预置的faster_rcnn目标检测模型（pytorch）为例，介绍mmdetection套件的推理文件书写方式。

3.1 模型初始化

版权所有 © 华为云计算技术有限公司

构建模型结构，载入模型权值，完成模型初始化构建工作。

```
from model_service.pytorch_model_service import PTServingBaseService
from mmdet.apis import (inference_detector, init_detector)
...
class ImageClassificationService(PTServingBaseService):
    def __init__(self, model_name, model_path, **kwargs):
        model_dir = osp.join(osp.dirname(osp.abspath(__file__)), 'res')
        # 模型导出生成的config.py、训练模型、类别索引文件路径
        config = osp.join(model_dir, 'config.py')
        checkpoint = osp.join(model_dir, 'model.pth')
        class_txt_path = osp.join(model_dir, 'classes.txt')

        device = 'cuda:0' if torch.cuda.is_available() else 'cpu'
        cfg_options = {
            'data.test.classes':f'class_txt_path',
        }

        # 模型初始化，并载入训练模型
        self.model = init_detector(config, checkpoint, device=device, cfg_options=cfg_options)


        if torch.cuda.is_available():
            logger.info('Using GPU for inference')
        else:
            logger.info('Using CPU for inference')
```

3.2 _pre_process

从request请求中读取图片流，并转换为模型可以接受的输入。

```
    def _preprocess(self, data: dict) -> dict:
        imgs = []
        for k, v in data.items():
            if k == 'images':
                for file_name, file_content in v.items():
                    img = cv2.imdecode(np.array(bytearray(file_content.read()), dtype=np.uint8),
cv2.IMREAD_COLOR)
                    imgs.append(img)

        preprocess_data = dict(
            inputs=imgs
        )
        return preprocess_data
```

3.3 _inference

```
    def _inference(self, data: dict) -> dict:
        inputs = data['inputs']
        # 前向推理，得到推理结果
        result = inference_detector(self.model, inputs)

        inference_data = dict(
            results=result
        )
        return inference_data
```

3.4 _post_process

```
    def _postprocess(self, data: dict) -> dict:
        results = data['results']
        # 将推理结果转换成最终的json格式
        label_list = self.model.CLASSES
```

 版权所有 © 华为云计算技术有限公司

```
    classes = []
    bboxes = []
    scores = []
    for index, res in enumerate(results[0]):
        category = label_list[index]
        for item in res:
            if len(item) < 5:
                continue
            bbox = item[:4].tolist()
            score = float(item[4])
            if score >= SCORE_THR:
                classes.append(category)
                bboxes.append([bbox[1], bbox[0], bbox[3], bbox[2]])
                scores.append(score)

    detection = dict(
            detection_classes=classes,
            detection_boxes=bboxes,
            detection_scores=scores
    )
    return detection
```

四、用户自定义模型

如果用户需要部署的模型是基于自研套件或者第三方套件进行二次开发的，那么推理脚本和内置的算法是一致的，同一个套件中的所有算法理论上是可以共享同一个部署推理脚本的。如果是完全自创的或者不属于算法开发套件内置套件的任何一种，用户可以参考**模型推理代码编写说明**。

9.6.6.3 调用部署服务进行推理

推理本地部署服务，本地服务的端口信息（如127.0.0.1:1234）将会在部署成功后打印在控制台中，推理本地服务，只需提供本地服务的地址即可。

> python manage.py run --cfg algorithms/ivgSegmentation/config/sample/config.py --pipeline infer --demo_type image_service --url http://127.0.0.1:1234

推理在线服务，需要提供用户的AK和SK信息进行验证，URL可以在ModelArts的部署上线->在线服务->指定服务->调用指南中获取。

> python manage.py run --cfg algorithms/ivgSegmentation/config/sample/config.py --pipeline infer --demo_type image_service --url {URL} --auth algorithms/ivgSegmentation/config/sample/config.py

服务请求成功后会在./output/deeplabv3_resnet50_standard-sample_1024x512/vis文件夹中保存请求的可视化结果。

9.6.6.4 删除已发布模型和在线服务

已经部署的模型和在线服务可以在ModelArts控制台上手动删除，也可以在Notebook中通过ModelArts SDK删除。

⚠ 注意

执行删除操作不可逆，请谨慎删除模型和在线服务！

版权所有 © 华为云计算技术有限公司

1. 根据模型ID删除已部署模型。

```
from modelarts.session import Session
from modelarts.model import Model
def delete_model_by_id(model_id):
    sess = Session()
    model_instance = Model(session=sess, model_id="").model_instance
    model_instance.delete_model(model_id)
if __name__ == '__main__':
    model_id = '************************'
    delete_model_by_id(auth, model_id)
```

2. 根据部署服务ID删除已部署服务。

```
from modelarts.session import Session
from modelarts.model import Model
def delete_service_by_id(service_id):
    sess = Session()
    model_instance = Model(session=sess, model_id="").model_instance
    model_instance.delete_service(service_id)
if __name__ == '__main__':
    service_id = '*************************'
    delete_service_by_id(service_id)
```

9.6.7 编排

```
# 根据flow config，一键运行所有任务
> python manage.py flow --cfg algorithms/ivgSegmentation/config/sample/config.py
```

9.6.8 查询

查询训练参数

```
# 查询训练规格列表
> python manage.py query train flavor --cfg algorithms/ivgSegmentation/config/sample/
config.py

# 查询训练引擎列表
> python manage.py query train framework --cfg algorithms/ivgSegmentation/config/sample/
config.py

# 查询专属池列表
> python manage.py query train pool --cfg algorithms/ivgSegmentation/config/sample/config.py
```

查询已部署模型和服务

```
# 查询已发布的模型
> python manage.py query deploy model --cfg algorithms/ivgSegmentation/config/sample/
config.py

# 查询已发布的在线服务
> python manage.py query deploy service --cfg algorithms/ivgSegmentation/config/sample/
config.py
```

查询在线服务详情

```
from modelarts.session import Session
from modelarts.predictor import Predictor
```


版权所有 © 华为云计算技术有限公司

```
def query_service(service_id):
    sess = Session()
    model_instance = Predictor(session=sess, service_id=service_id)
    return model_instance.get_service_info(service_id)


if __name__ == '__main__':
    service_id = '*************************'
    service_info = query_service(service_id)
    print(service_info)
```

9.7 配置参数说明

9.7.1 全局参数

本小节介绍了算法开发套件配置文件中涉及到的全局参数。

平台参数

- obs_bucket: 表示使用的远程存储根目录。
- auth: 表示账户信息及配置。详见鉴权参数。

鉴权参数

- hwc: 是否是华为云环境，默认是True。
- region_name: 华为云区域，默认是cn-north-4。

运行参数

- alg_type: 表示使用的算法类型，基于`algorithms/${alg_type}`得到算法的目录。
- alg_name: 表示使用的算法名称。
- run_dir: 表示runner的工作目录，放在此处便于runner内部复用。
- convert_dir: 表示converter的工作目录，放在此处便于converter内部复用。
- export_dir: 表示exporter的工作目录，放在此处便于exporter内部复用。

复用参数

- alg_cfg: 表示算法使用的共性参数，便于多处共享。

9.7.2 Runner

本小节介绍了基于算法开发套件进行算法的训练、测试和推理等阶段的相关参数。

其中，训练、测试支持单节点或多节点分布式运行，单节点分布式启动时只需给定local_size，即会在一个机器上启动多次算法脚本，从而实现单节点多进程；多节点分布式启动则需要在各个node上指定int_method（当前仅支持TCPmethod）、rank（节点序号）、world_size（节点数）、local_size（单节点进程数）。

启动ModelArt远程分布式作业时，仅需传递world_size和local_size参数，其余参数ModelArts平台会自动传递。

版权所有 © 华为云计算技术有限公司

通过算法外壳启动分布式会给算法脚本传递如下分布式参数，其中rank和world_size分别表示总进程序号和总进程数，local_rank表示单节点进程序号，需要脚本自行解析和配置，配置方式可以参考领域套件代码。

```
--launcher=utility  # 分布式启动方法，基于算法外壳启动称为utility
--init_method=${init_method}  # TCP初始化方法，指定IP和Port
--rank=${rank}  # 总进程序号
--world_size=${world_size}  # 总进程数
--local_rank=${local_rank}  # 单节点进程序号
--local_size=${local_size}  # 单节点进程数
```

公共参数

- alg_type：表示使用的算法类型，继承Global参数。

- run_dir：表示runner的工作目录，继承Global参数。

- pipeline：表示不同算法管控模式构成的流水线，允许command传入`--pipeline mode1,mode2,...`进行覆盖。如train， evaluate，infer。

- pipeline_maps：表示算法支持的管控模式及其对应的算法入口脚本。

算法参数

- common_args：表示算法的共性参数，所有mode的算法参数会基于其进行update，不存在时默认为空字典。

- ${mode}_args：表示pipeline_maps中各mode对应的特性参数，在特定mode进行运行时，会和common_args进行组装（common_args.update(mode_args)）以构成mode最终的参数，并且允许command传入算法的任意参数进行覆盖（通过argparse解析时的无法识别参数进行传递，因此不能与common_algorithm_utility的参数重叠）。

说明

value为None时不传递，value为''（空字符串）时作为`store_arg`传递，采用`position_args=[arg1, arg2, ...]`的形式传递位置参数，采用`arg=[val1, val2, ...]`的形式传递列表参数。

远程训练参数

adapter：对算法运行过程中的所有环境切换、依赖安装、数据下载、结果上传等功能进行管理与实现。对本地启动远程作业的方式进行管控。

9.7.3 Converter

本小节基于算法开发套件进行模型转换的相关参数。

公共参数

- alg_type: 表示使用的算法类型，继承Global参数。

- convert_dir: 表示converter的工作目录，继承Global参数。

- pipeline: 表示不同模型转换模式构成的流水线，允许command传入`--pipeline mode1,mode2,...`进行覆盖。

版权所有 © 华为云计算技术有限公司

- pipeline_maps: 表示算法支持的模型转换模式及对应的类型及脚本，所有涉及code-based model的模型转换均由用户实现custom脚本调用套件能力进行转换，其余采用default的套件能力即可。

转换参数

- common_args: 表示模型转换的共性参数，使用方式同runner；不识别的参数不会被使用。

- ${mode}_args: 表示pipeline_maps中各mode对应的特性参数，使用方式同runner；针对custom模式的模型转换，通过`script_args`传入算法脚本所需的参数，同样可以通过command中unknown_args的形式进一步传递算法脚本所需的其他参数。

- config_file：表示配置文件的路径，None表示`${convert_dir}/config.json`。会将转换onnx模型的配置写入config。

- do_comparison：表示是否进行转换前后模型的输出进行比较，部分情况不支持比较。

模型转换支持的类型有：

- torch2onnx (custom)

- script_args：表示传递给算法自定义脚本的命令行参数，算法自定义脚本中，先构建好torch model，而后调用功能外壳的能力进行模型转换。具体使用方式请参见领域套件。

- input_shapes: 表示模型输入的形状，list[tuple]

- input_names: 表示模型输入的名字，list[str]

- output_names: 表示模型输出的名字，list[str]

- onnx_path：表示生成的onnx模型路径，None表示`${convert_dir}/model.onnx`

- opset_version：表示转换onnx的版本，默认使用当前torch的默认版本

- onnx2rt (default， only support TensorRT5.1.5 for now)

- onnx_path：表示生成的onnx模型路径，None表示`${convert_dir}/model.onnx`

- rt_path: 表示生成的tensorRT模型路径，None表示`${convert_dir}/model.rt`

- mode: 表示生成的tensorRT模型的类型，目前支持`fp32`和`int8`(int8目标仅支持单图像输入)

- output_names: 表示模型输出的名字，list[str]。如果设置，则会将忽略list之外的输出

- calib_cfg：`int8`模式下进行参数校正的配置，详见后续说明

- torch2caffe (custom，support pytorch>=1.1.0,<1.6.0])

- script_args：表示传递给算法自定义脚本的命令行参数，算法自定义脚本中，先构建好torch model，而后调用功能外壳的能力进行模型转换

- input_shapes: 表示模型输入的形状，list[tuple]

- input_names: 表示模型输入的名字，list[str]

 版权所有 © 华为云计算技术有限公司

- caffe_prototxt：表示生成的caffe结构文件路径，None表示`${convert_dir}/model.prototxt`

- caffe_caffemodel：表示生成的caffe权重文件路径，None表示`${convert_dir}/model.caffemodel`

- caffe2ascend (default, only support low version for now)

- caffe_prototxt：表示生成的caffe结构文件路径，None表示`${convert_dir}/model.prototxt`

- caffe_caffemodel：表示生成的caffe权重文件路径，None表示`${convert_dir}/model.caffemodel`

- om_path: 表示生成的D模型路径，None表示`${convert_dir}/model.om`

- insert_op_cfg：表示插入新opetator（aipp）的配置文件，默认不使用

- input_shapes: 表示模型输入的形状，list[tuple]。只在进行输出比较时需要

远程训练参数

- adapter: 对算法运行过程中的所有环境切换、依赖安装、数据下载、结果上传等功能进行管理与实现。对本地启动远程作业的方式进行管控。

9.7.4 Exporter

本小节介绍了基于算法开发套件进行内容导出的相关参数。

公共参数

- type: 表示导出的平台，目前仅支持local模式导出。

- alg_type: 表示使用的算法类型，继承Global参数。

- export_dir: 表示exporter的工作目录，继承Global参数。

- pipeline: 表示导出的模式构成的流水线，允许command传入`--pipeline mode1,mode2,...`进行覆盖。

- pipeline_maps: 表示算法支持的内容导出模式及对应的类型及脚本，暂时仅支持导出模型SDK；支持用户实现custom脚本进行自定义导出，也可以采用default的形式进行runtime-based的模型sdk导出。`default`形式仅支持领域套件，自定义算法请采用`custom`形式并自行实现相应脚本。

导出参数

- common_args: 表示内容导出的共性参数，使用方式同runner。

- ${mode}_args: 表示pipeline_maps中各mode对应的特性参数，使用方式同runner；针对custom的模型转换，通过script_args传入算法脚本所需的参数，同样可以通过command中unknown_args的形式进一步传递算法脚本所需的其他参数。

9.7.5 Deployer

本小节介绍了基于算法开发套件进行模型部署的相关参数。

版权所有 © 华为云计算技术有限公司

公共参数

- requirements: 表示部署模型的依赖列表文件路径。

- service_name: 表示需要部署在线服务名称及模型发布的名称。

- deploy_dir: 表示deployer的工作目录，继承Global参数。

- model_type: 表示需要部署的模型依赖的运行框架。

- pred_img_path: 表示模型部署完成后需要进行测试的图片路径，如果省略这个参数，将会跳过预测过程。

- instance_type: 表示在线部署使用的规格。

9.7.6 Adapter

本小节介绍了基于算法开发套件进行环境管控的相关参数。

adapter主要负责：

1. 对代码运行环境进行配置，目前支持的环境配置内容包括：conda环境切换、pip依赖安装。
2. 基于功能外壳进行无感远程作业的相关参数，可以自动完成启动前数据上传、远程作业启动、远程作业监控和运行中/后数据下载等。

参数说明

- requirements: 依赖项，可以为指定的requirements文件，也可以是一个list，list可以放置requirements路径或者指定的package名称。

- framework_type：训练作业选择的引擎规格，必选，如PyTorch。

- framework_version：训练作业选择的引擎版本，必选，如PyTorch-1.4.0-python3.6-v2。

- instance_type：训练作业选择的资源规格，如modelarts.p3.large.eco。

- world_size：训练作业计算节点个数。可选，默认为1。

- pool_id：专属池ID，默认为True。

- downloads: 表示数据下载内容的源路径（ src ）和目标路径（ dst ），其中src和dst需一一对应。不下载可以置为None。

- uploads：表示启动远程作业前数据上传的源路径和目标路径。不上传可以置为None。

9.7.7 Flow

本小节介绍了基于算法开发套件进行自动化流水线构建的相关参数。目前仅支持串行流，以list进行表示，其中的node以dict进行表示，node一般对应于tools中的一个脚本，运行node本质上是直接运行相应脚本。

参数说明

- type: 表示node的类型，目前支持launch和default形式。launch类型会启动远程作业运行脚本，default类型则直接本地运行脚本。

 版权所有 © 华为云计算技术有限公司

- mode: 表示node的运行模式，launch type会采用对应mode进行launch，default type则直接执行对应的脚本。

- args: 表示node的参数，参数解析方式同runner，所有参数必须在此进行设定。

9.8 通过 Python API 使用算法套件

算法开发套件支持用户在Notebook中用python API进行交互式、参数化、低代码的开发方式快速完成算法验证与实践。当前目标检测类套件（"mmdetection", "ivgDetetcion"）支持通过API的方式使用。

9.8.1 算法工程环境管理

EnvManager是针对于AI开发套件工程化管理的抽象，主要用于创建算法工程、查看算法/模型/数据集等资产、安装依赖环境、切换工作路径等。

创建算法工程

```
from modelarts.algo_kits import EnvManager
env = EnvManager()
env.init_env(install_path=None, project_name=None)  # 初始化工程
```

查看资产

查看模型信息，可以传入filter_keyword进行信息过滤。

```
env.show_asset(mode="model", filter_keyword=["coco", "fcos"])
```

查看数据集信息。

```
env.show_asset("dataset")
```

查看套件信息。

```
env.show_asset("algorithm")
```

安装资产

安装算法套件，Notebook实例启动后，首次安装套件后可能需要重新启动kernel以使用更新后的依赖包。

```
env.install(mode="algorithm", asset_name="mmdetection", version="2.17.0")
```

安装数据集。

```
env.install(mode="dataset", asset_name="coco2017_sample")
```

安装预训练模型。

```
env.install('model', 'mmdetection:fcos/fcos_r50_caffe_fpn_gn-head_1x_coco')
```

其它功能

切换工作路径。

```
env.change_work_path('your_work_path')
```

安装套件依赖环境。

版权所有 © 华为云计算技术有限公司

```
env.change_work_path('your_work_path')
```

安装预编译的mmcv-full，减少编译mmcv-full的时间。

```
env.install_mmcv_full(version="1.3.9",
                      torch="1.8.0",
                      cuda="10.2",
                      python="3.7.0")
```

9.8.2 创建数据集

DataBlock是AI开发套件数据集的统一抽象，针对不同的套件场景定义了不同的DataBlock类，以目标检测为例，目前支持加载coco格式的数据集加载模块COCODetDataBlock。

voc以及yolo格式的数据集可以通过COCOConverter进行数据集转化，请参考**数据集转换**。

加载数据集

```
from modelarts.algo_kits import DetDataBlock
"""
加载coco格式数据集，并指定训练集和验证集文件路径，类别数和类别信息默认会自动读取。
"""
db = DetDataBlock(env,
                  batch_size=8,
                  data_root="./data/raw/coco2017_sample",
                  num_classes=80,
                  train_img_prefix="val2017",
                  train_ann_file="annotations/instances_val2017.json",
                  val_img_prefix="val2017",
                  val_ann_file="annotations/instances_val2017.json",
                  model_name="mmdetection:fcos/fcos_r50_caffe_fpn_gn-head_1x_coco"
                  )
```

表 9-2 DetDataBlock 参数说明

参数名称	可选/必选	参数类型	参数描述
env	必选	EnvManager Object	上下文管理对象，初始化方法请参考**算法工程环境管理**。
batch_size	可选	int	批处理大小，默认为8。
workers	可选	int	数据加载进程数，默认为4。
data_type	可选	string	数据集类型，目前只支持“coco”，其他数据集格式可以通过COCOConverter进行转换。
data_root	必选	string	数据集路径。

版权所有 © 华为云计算技术有限公司

参数名称	可选/必选	参数类型	参数描述
seed	可选	int	随机数种子。
num_classes	可选	int	类别数，默认会从标注信息中读取。
categories	可选	list / tuple / string	类别信息，默认会从标注信息中读取，如果指定类别信息，则只加载对应类别的数据。
train_img_prefix	必选	string	训练集图片的相对路径（相对于data_root）。
train_ann_file	必选	string	训练集标注文件的相对路径（相对于data_root）。
val_img_prefix	可选	string	验证集图片的相对路径（相对于data_root），默认不加载验证集信息。
val_ann_file	可选	string	验证集标注文件的相对路径（相对于data_root），默认不加载验证集信息。
test_img_prefix	可选	string	测试集图片的相对路径（相对于data_root），默认不加载测试集信息。
test_ann_file	可选	string	测试集标注文件的相对路径（相对于data_root），默认不加载测试集信息。
model_name	必选	string	模型名称。

数据集统计信息可视化

数据集API支持自动统计数据集信息，比如目标、尺寸等信息，帮助用户更好的理解数据集，同时还可以动态查看每一个batch经过pipeline之后的输入图像，确保数据增强的正确性。

db.plot_dataset_stats() 可以绘制数据集的统计信息，图像的显示大小可以由figsize参数控制；

 版权所有 © 华为云计算技术有限公司

db.print_dataset_stats() 能够打印出具体的统计信息数值；
db.show_batch() 可以动态展示内存中的经过增强后的图片信息，可以通过rows（显示行数）和figsize（显示大小）来控制输出。

注：show_batch返回一个生成器，可以使用next进行访问，比如：

```
plotter = db.show_batch(rows=2, figsize=(14, 8))
next(plotter)
```

表 9-3 show_batch 参数说明

参数名称	可选/必选	参数类型	参数描述
rows	可选	int	batch绘制行数，默认为1。
figsize	可选	tuple	画布尺寸。

数据集转换

“COCOConverter”支持目标检测数据集的转换，包括：

- VOC转coco-> xml to json
- Manifest转coco -> xml to json
- YOLO转coco -> txt to json

COCOConverter使用方式：

```
from modelarts.algo_kits import COCOConverter

COCOConverter(data_root="./data", # 需要转换的原始数据集根目录
                        out_root=None, # 将会保存在data_root的同级目录

              data_type="xml",
              bbox_format="xyxy",
              img_prefix="", # ""表示图片在根目录
              ann_prefix="", # ""表示标注文件在根目录
              split_file=None,
              stage="train",
              is_norm_bbox=False)
```

表 9-4 COCOConverter 参数配置说明

参数名称	可选/必选	参数类型	参数描述
data_root	必选	string	待转换的数据集路径。
classes	可选	list / tuple / string	数据集类别，xml数据格式可以缺省，系统会自动收集，yolo格式数据集需要指定。

 版权所有 © 华为云计算技术有限公司

参数名称	可选/必选	参数类型	参数描述
out_root	可选	string	转换得到的json文件保存路径，默认为data_root。
data_type	可选	string	待转换的数据集格式，支持xml或者txt，默认是xml。
bbox_format	可选	string	数据集标注格式，支持 "xyxy"、"xywh"和"xywh_c"，默认是"xyxy"，表示目标框标注信息为左上角坐标和右下角坐标。，表注：yolo的格式为"xywh_c" "xyxy" -> (x_lt, y_lt, w, h), "xywh" -> (x_lt, y_lt, x_rd, y_rd), "xywh_c" -> (x_center, y_center, w, h)
img_prefix	必选	string	数据集中图片的路径，相对于data_root，如果和data_root同级目录，可以写为""。
ann_prefix	必选	string	数据集中标注文件的路径，相对于data_root，如果和data_root同级目录，可以写为""。
split_file	可选	string	数据集划分信息文本的路径，默认使用整个数据集。
stage	可选	string	生成的coco数据集标记，取值为"train"、"val"、"test"，默认为"train"，生成的标注文件名为"instances_train2017.json"。

版权所有 © 华为云计算技术有限公司

参数名称	可选/必选	参数类型	参数描述
is_norm_bbox	可选	bool	数据集标注是否经过归一化，YOLO格式数据集一般为True，xml一般为False。

9.8.3 构建模型

Model对象是AI开发套件模型结构的统一抽象，能够加载预训练模型权重，用于后续的训练、推理和评估任务，支持将模型放在不同的资源设备上。

```
from modelarts.algo_kits import Model

model = Model(
    env,
    model_name="mmdetection:fcos/fcos_r50_caffe_fpn_gn-head_1x_coco",
    num_classes=80,
    checkpoint="your_local_path_of_pretrained_model",
    load_default_backbone=False
)




# 将模型加载至cpu上
model.to_device(-1)
```

表 9-5 Model 参数说明

参数名称	可选/必选	参数类型	参数描述
env	必选	EnvManager Object	上下文管理对象，初始化方法请参考**通过Python API使用算法套件**。
model_name	必选	string	模型名称。
num_classes	可选	int	类别数，默认会从标注信息中读取。
checkpoint	可选	string	本地预训练模型路径，默认为None，使用默认值时随机生成网络参数。

版权所有 © 华为云计算技术有限公司

参数名称	可选/必选	参数类型	参数描述
load_default_backbone	可选	boolean	是否加载默认的预训练骨干网络，如resnet50，默认为False，该参数设置为True时模型自动从open-mmlab中拉取，可与checkpoint参数二选一。

9.8.4 构建学习器

Learner对象是AI开发套件学习器的统一抽象，用于构建模型训练、评估和推理的通用学习器。

```
from modelarts.algo_kits import Learner

learner = Learner(
    model=model,
    datablock=db,
    output_dir="./output",
    optimizer="SGD",
    momentum=0.999,
    checkpoint="your_local_path_of_pretrained_model",
    warmup_policy="linear",
    warmup_iters=500,
    epoch_based_eval_interval=1,
    save_ckpt_interval=1,
    log_params={"log_type":["table", "graph"], "table_highlight_method":"best",
"log_interval":10}

)
```

表 9-6 Learner 初始化参数说明

参数名称	可选/必选	参数类型	参数描述
model	必选	Model object	模型对象，初始化方法参考Model模块。
datablock	必选	DataBlock object	数据集对象，初始化方法参考DataBlock模块。
output_dir	必选	string	训练时checkpoint，日志等的输出路径。
optimizer	可选	string	训练优化器，支持SGD和Adam，默认为SGD。

版权所有 © 华为云计算技术有限公司

参数名称	可选/必选	参数类型	参数描述
momentum	可选	float	动量，默认为0.999。
checkpoint	可选	string	预训练模型路径，默认为None，优先级高于Model的checkpoint入参，指定checkpoint入参后会覆盖初始化Model时对应checkpoint的模型参数。
warmup_policy	可选	string	warmup策略，默认为linear。其中，open-mmlab系列模型支持linear和polynomial两种，ivg系列模型支持cosine。
warmup_iters	可选	int/float	warmup的迭代次数，默认为500。
epoch_based_eval_interval	可选	int/float	训练过程中评估间隔数，以epoch为单位，默认为1。
iter_based_eval_interval	可选	int/float	训练过程中评估间隔数，以iteration为单位，默认为100。
save_ckpt_interval	可选	int/float	训练过程中保存模型的间隔数，以epoch为单位，默认为1。

版权所有 © 华为云计算技术有限公司

参数名称	可选/必选	参数类型	参数描述
log_params	可选	dict	训练日志相关参数，包含“log_type”，“table_highlight_method”和“log_interval”三个字段，分别代表日志类型，训练损失高亮方法，日志间隔时间。log_type支持text(原始日志)，table(表格)，graph(动态曲线图)和tensorboard四中，其中除text外其余日志类型可任意组合；table_highlight_method支持best(高亮最优)和gradient(高亮渐变)；log_interval的单位为iteration。具体见上述代码示例。

基于 learner 进行模型训练

```
learner.fit(
    lr=0.001,
    max_epochs=5,
    weight_decay=0.005,
    seed=None,
    checkpoint=None,
    gpu_ids=[0],
    world_size=1,
    local_size=1,
    launch_remote=False
)
```

表 9-7 learner.fit 参数

参数名称	可选/必选	参数类型	参数描述
lr	可选	float	学习率参数，默认为0.001。
max_epochs	可选	int/float	最大训练epoch数，默认为5。

版权所有 © 华为云计算技术有限公司

参数名称	可选/必选	参数类型	参数描述
weight_decay	可选	float	权重衰减，默认为0.005。
seed	可选	int	训练随机数，默认为None。
checkpoint	可选	string	预训练模型路径，优先级高于初始化Learner和Model时的路径参数，指定checkpoint入参后会覆盖初始化Learner或Model时对应checkpoint的模型参数。
gpu_ids	可选	int / list	open-mmlab系列默认使用0号卡进行训练。ivg系列可指定，如0或者[0]或者[0,1]。
world_size	可选	int	分布式训练节点数量，默认为1，暂不支持分布式训练。
local_size	可选	int	单个节点中的GPU数量，默认为1，暂不支持分布式训练。
launch_remote	可选	boolean	提交训练作业，默认为False，暂不支持基于API提交训练作业。

基于 learner 进行模型评估

```
learner.validate(
    data=db,
    checkpoint="your_local_path_of_pretrained_model",
    gpu_ids=[0]
)
```


版权所有 © 华为云计算技术有限公司

表 9-8 learner.validate 参数

参数名称	可选/必选	参数类型	参数描述
data	可选	DataBlock object	数据集对象，默认为None，初始化方法参考DataBlock模块。默认值时会基于构建好的DataBlock对象中的validation data进行模型评估。
checkpoint	可选	string	预训练模型路径，默认为None。当基于learner.fit完成训练且该参数为None，则基于训练后的模型参数进行评估。如果指定checkpoint路径，则加载对应路径的模型参数进行评估。
gpu_ids	可选	int/list	模型评估时使用的GPU序号。open-mmlab系列模型默认使用0号卡。ivg系列可指定，如0或者[0, 1]。
show_score_thr	可选	float	评估时预测结果置信度阈值，默认为0.5，仅用于open-mmlab系列模型。
out	可选	string	评估结果pkl文件保存路径，默认为None，仅用于open-mmlab系列模型。

基于 learner.predict 进行模型推理

```
learner.predict(
    img_path='your_local_path_of_image',
    checkpoint='your_local_path_of_pretrained_model',
    gpu_ids=None,
    save_dir='your_local_path_for_saving_output'
)
```


版权所有 © 华为云计算技术有限公司

表 9-9 learner.predict 参数

参数名称	可选/必选	参数类型	参数描述
img_path	必选	string	图片路径，当前predict仅支持推理图片。
checkpoint	可选	string	预训练模型路径，默认为None。当基于learner.fit完成训练且该参数为None，则基于训练后的模型参数进行推理。如果指定checkpoint路径，则加载对应路径的模型参数进行推理。
gpu_ids	可选	int/list	推理时使用的GPU，默认为None（使用cpu进行推理）。
save_dir	可选	string	默认为初始化Learner时指定的work_dir，可指定其他本地路径。
model	可选	Model object	自定义Model对象，仅用于open-mmlab系列模型，默认为None。默认值时使用基于learner.fit训练好的模型进行推理。
score_thr	可选	float	推理时结果置信度阈值，默认为0.3，仅用于open-mmlab系列模型。
ret_vis	可选	boolean	是否可视化推理结果，默认为False，仅用于open-mmlab系列模型。

9.8.5 本地交互式推理

VisPlatform提供了在Notebook中交互式推理的能力，提供了在线图片/视频的实时推理、训练结果离线可视化推理、数据集可视化等能力，帮助用户更加快速、简便地对于数据和训练的模型进行理解和评估。

 版权所有 © 华为云计算技术有限公司

以目标检测场景为例，用户可以通过鼠标勾选需要推理的图片、视频，并且可以选择不同的iou和score参数进行动态比较。

VisPlatform

```
from modelarts.algo_kits import VisDetPlatform

VisDetPlatform(learner, stage="val", det_box_color=(0, 0, 255))
```

表 9-10 Valplatform 参数说明

参数名称	可选/必选	参数类型	参数描述
learner	可选	Learner object	Learner对象，用于在线实时推理，在离线推理或数据集可视化场景可省略。
stage	可选	string	推理图片所属的DataBlock数据集类别，包括"train"、"val"、"test"，默认为"train"，只对在线推理有效。
classes	可选	list	需要展示的类别列表。
ann_json	可选	string	标注文件路径，主要用于离线展示。
det_file	可选	string	验证输出的检测结果文件路径，主要用于离线展示。
det_box_color	可选	tuple	检测框颜色，默认为(0, 0, 255)
gt_box_color	可选	string	GT标注框颜色，默认为(80, 127, 255)
without_gt	可选	bool	离线推理是否包含gt信息。
mask_palette	可选	list	mask蒙版颜色列表，如果不设置，则使用随机颜色，只对mask类算法生效。

版权所有 © 华为云计算技术有限公司

参数名称	可选/必选	参数类型	参数描述
infer_param	可选	dict	推理相关参数，如不指定则使用配置文件默认参数，与推理速度相关，目前只对mmdetection类模型生效，主要包含： 1. nms_pre：nms操作前生成的bbox数量 2. max_per_img：每张图片最大目标数 3. img_scale：图像resize尺寸。

基于 VisPlatform 进行图片/视频推理

图片推理

方式1：在线推理

```
VisDetPlatform(learner, stage="val", det_box_color=(0, 0, 255))
```

- 单击“Select”按钮可以交互式选择需要预测的文件、文件夹，会自动过滤出文件夹支持的图片及视频。
- 当检测框置信度大于“Score Thr”时会在图片中显示，当检测框和标注框的iou值大于“Iou Thr”时会显示蓝色框，当iou值小于“Iou Thr时”会显示红色框。
- “Category”默认选择“ALL”，表示会显示所有类别信息，也可以选择只显示某一种类别。
- “Pred_bbox”表示是否显示检测框，“Pred_label”表示是否显示对应预测的标签和置信度，“GT_bbox”表示是否显示标注框，“GT_label”表示是否显示对应的标签。

方式2： 离线推理

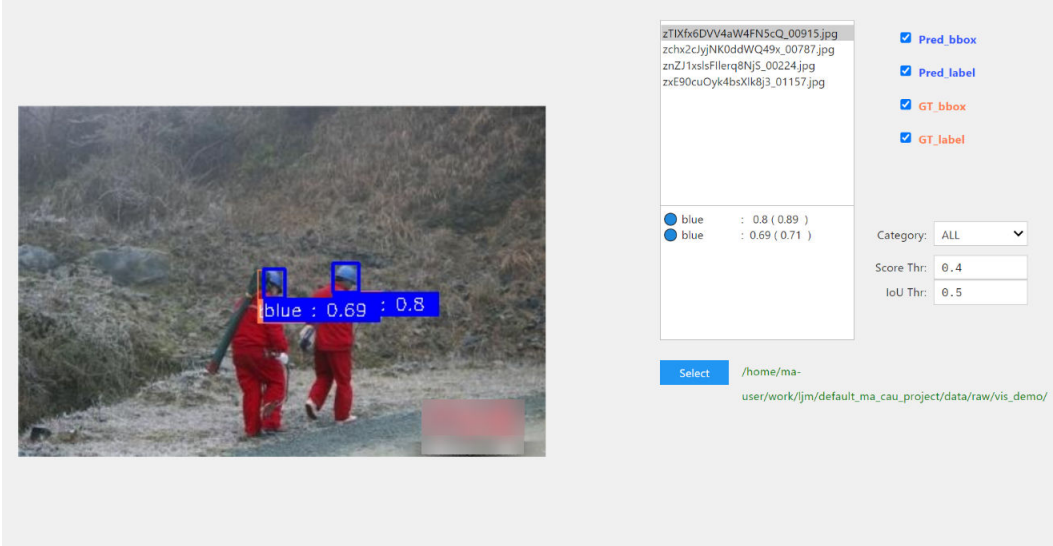
版权所有 © 华为云计算技术有限公司

在验证集进行评估，并保存验证集检测结果到文件中。

```
learner.validate(out="./output/mmdetection/fcos/prediction.pkl")
```

使用检测结果文件和标注进行离线推理

```
VisDetPlatform(ann_json="./data/raw/helmet/annotations/instances_val2017.json", det_file="./output/mmdetection/fcos/prediction.pkl")
```

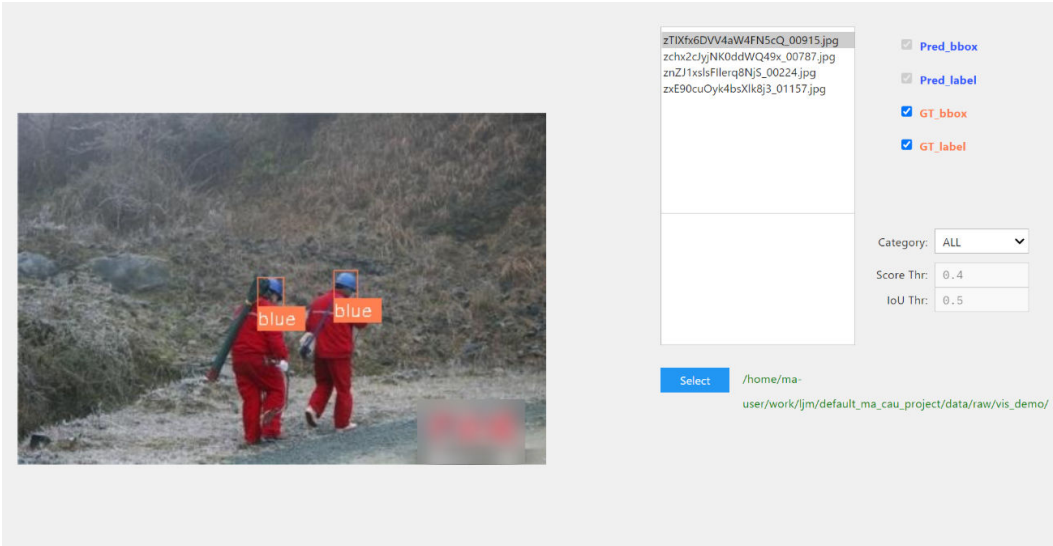


此时的交互式方式和在线推理类似，但是不支持视频的推理。

方式3： 数据集可视化

只展示数据集标注信息。

```
VisDetPlatform(ann_json="./data/raw/helmet/annotations/instances_val2017.json")
```

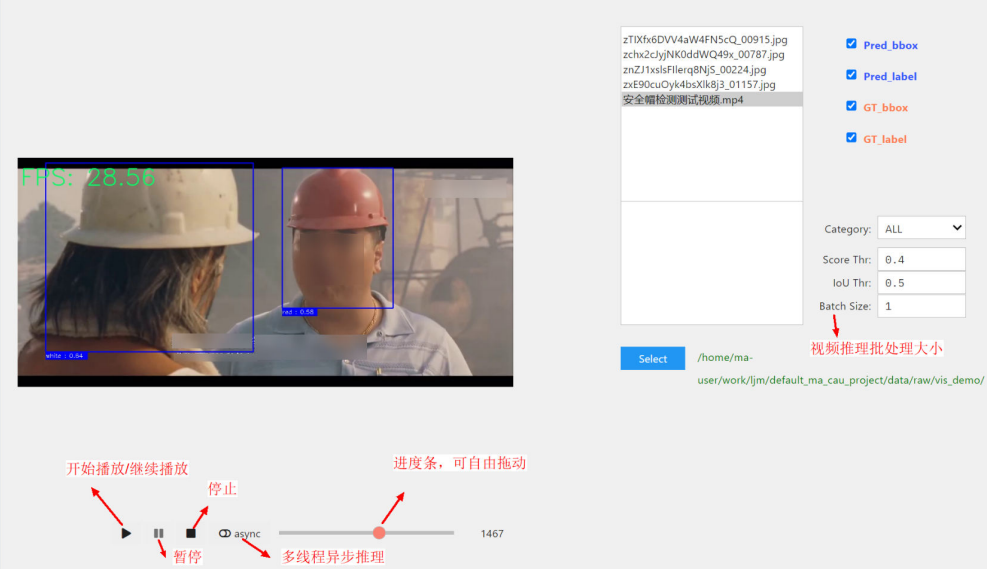


- 如果只传入ann_json路径，那么可以使用VisDetPlatform对数据集进行可视化，此时pred_bbox、pred_label、Score Thr和IoU Thr被锁定，无法进行编辑操作；
- 只会对于ann_json中具有标注信息的图片进行可视化。

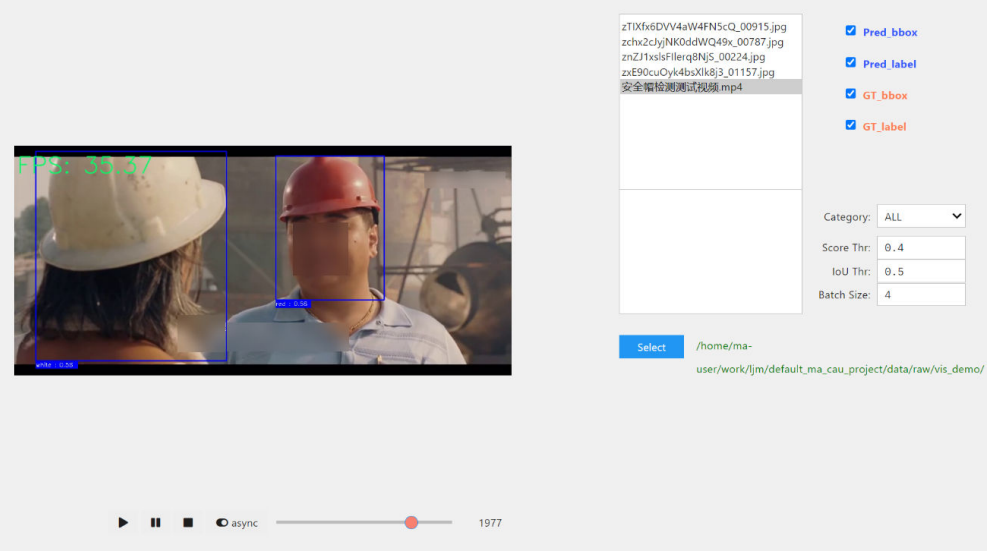
视频推理

 版权所有 © 华为云计算技术有限公司

```
infer_param = {"nms_pre": 100, "max_per_img": 20, "img_scale": (800, 600)}  # 默认为None
VisDetPlatform(learner, infer_param=infer_param, stage="val", det_box_color=(0, 0, 255))
```



- infer_param参数取值默认为None，则会使用默认的推理配置参数，设置infer_param后（接收nms_pre, max_per_img, img_scale），能够提高视频推理的FPS；
- 视频推理模式下，pred_bbox、pred_label、GT_bbox、GT_label和IoU Thr无效；
- 开启async模式后，对于小模型而言，适量增大“batch size”的大小能够提高视频推理的FPS，如下图所示：



版权所有 © 华为云计算技术有限公司

10 ModelArts CLI 命令参考

10.1 ModelArts CLI 简介

功能介绍

ModelArts CLI，即ModelArts命令行工具，是一个跨平台命令行工具，用于连接ModelArts服务并在ModelArts资源上执行管理命令。用户可以使用交互式命令行提示符或脚本通过终端执行命令。为了方便理解，下面将ModelArts CLI统称为ma-cli。ma-cli支持用户在ModelArts Notebook及线下虚拟机中与云端服务交互，使用ma-cli命令可以实现命令自动补全、鉴权、镜像构建、提交ModelArts训练作业、提交DLI Spark作业、OBS数据复制等。

使用场景

- ma-cli已经集成在ModelArts开发环境Notebook中，可以直接使用。

  登录ModelArts控制台，在“开发环境 > Notebook”中创建Notebook实例，打开Terminal，使用ma-cli命令。
- ma-cli在本地Windows/Linux环境中需要安装后在本地Terminal中使用。安装步骤具体可参考**（可选）本地安装ma-cli**。

  说明

  - ma-cli不支持在git-bash上使用。
  - 推荐使用Linux Bash、ZSH、Fish，WSL或PowerShell等Terminal。在使用过程中，注意您的敏感信息数据保护，避免敏感信息泄露。

命令预览

```
$ ma-cli -h
Usage: ma-cli [OPTIONS] COMMAND [ARGS]...

Options:
  -V, -v, --version       1.2.1
  -C, --config-file TEXT  Configure file path for authorization.
  -D, --debug             Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT      CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help          Show this message and exit.

Commands:
```


版权所有 © 华为云计算技术有限公司

```
  configure        Configures authentication and endpoints info for the CLI.
  image            Support get registered image list、register or unregister image、debug image, build image
in Notebook.
  obs-copy         Copy file or directory between OBS and local path.
  ma-job           ModelArts job submission and query job details.
  dli-job          DLI spark job submission and query job details.
  auto-completion  Auto complete ma-cli command in terminal, support "bash(default)/zsh/fish".
```

其中，-C、-D、-P，-h参数属于全局可选参数。

- -C表示在执行此命令时可以手动指定鉴权配置文件，默认使用~/.modelarts/ma-cli-profile.yaml配置文件；
- -P表示鉴权文件中的某一组鉴权信息，默认是DEFAULT；
- -D表示是否开启debug模式（默认关闭），当开启debug模式后，命令的报错堆栈信息将会打印出来，否则只会打印报错信息；
- -h表示显示命令的帮助提示信息。

命令说明

表 10-1 ma-cli 支持的命令

命令	命令详情
configure	ma-cli鉴权命令，支持用户名密码、AK/SK
image	ModelArts镜像构建、镜像注册、查询已注册镜像信息等
obs-copy	本地和OBS文件/文件夹间的相互复制
ma-job	ModelArts训练作业管理，包含作业提交、资源查询等
dli-job	DLI Spark任务提交及资源管理
auto-completion	命令自动补全

10.2 （可选）本地安装 ma-cli

使用场景

本文以Windows系统为例，介绍如何在Windows环境中安装ma-cli。

Step1：安装 ModelArts SDK

参考**本地安装ModelArts SDK** 完成SDK的安装。

Step2：下载 ma-cli

1. **下载ma-cli软件包**。
2. 完成软件包签名校验。
   a. **下载软件包签名校验文件**。

版权所有 © 华为云计算技术有限公司

b. 安装openssl并执行如下命令进行签名校验。

```
openssl cms -verify -binary -in D:\ma_cli-latest-py3-none-any.whl.cms -inform DER -content
D:\ma_cli-latest-py3-none-any.whl  -noverify > ./test
```

说明

本示例以软件包在D:\举例，请根据软件包实际路径修改。

Step3：安装 ma-cli

1. 在本地环境cmd中执行命令**python --version**，确认环境已经安装完成Python。（Python版本需大于3.7.x且小于3.10.x版本，推荐使用3.7.x版本）

   ```
   C:\Users\xxx>python --version
   Python *.*.*
   ```

2. 执行命令**pip --version**，确认Python通用包管理工具pip已经存在。

   ```
   C:\Users\xxx>pip --version
   pip **.*.* from c:\users\xxx\appdata\local\programs\python\python**\lib\site-packages\pip (python *.*)
   ```

3. 执行如下命令，安装ma-cli。

   pip install *{ma-cli软件包路径}*\ma_cli-latest-py3-none-any.whl

   ```
   C:\Users\xxx>pip install C:\Users\xxx\Downloads\ma_cli-latest-py3-none-any.whl
   ......
   Successfully installed ma_cli.*.*.*
   ```

   在安装ma-cli时会默认同时安装所需的依赖包。当显示“Successfully installed”时，表示ma-cli安装完成。

   说明

   如果在安装过程中报错提示缺少相应的依赖包，请根据报错提示执行如下命令进行依赖包安装。

   pip install xxxx

   其中，xxxx为依赖包的名称。

10.3 ma-cli auto-completion 自动补全命令

命令行自动补全是指用户可以在Terminal中输入命令前缀通过Tab键自动提示支持的ma-cli命令。ma-cli自动补全功能需要手动在Terminal中激活。执行**ma-cli auto-completion**命令，用户根据提示的补全命令，复制并在当前Terminal中执行，就可以自动补全ma-cli的命令。目前支持Bash、Fish及Zsh三种Shell，默认是Bash。

以Bash命令为例：在Terminal中执行**eval "$(_MA_CLI_COMPLETE=bash_source ma-cli)"**激活自动补全功能。

```
eval "$(_MA_CLI_COMPLETE=bash_source ma-cli)"
```

此外，可以通过“ma-cli auto-completion Fish”或“ma-cli auto-completion Fish”命令查看“Zsh”、“Fish”中的自动补全命令。

命令概览

```
$ ma-cli auto-completion -h
Usage: ma-cli auto-completion [OPTIONS] [[Bash|Zsh|Fish]]

  Auto complete ma-cli command in terminal.

  Example:
```


版权所有 © 华为云计算技术有限公司

```
  # print bash auto complete command to terminal
  ma-cli auto-completion Bash

Options:
  -H, -h, --help  Show this message and exit.
```

```
# 默认显示Bash Shell自动补全命令

$ ma-cli auto-completion

Tips: please paste following shell command to your terminal to activate auto complation.

[ OK ] eval "$(_MA_CLI_COMPLETE=bash_source ma-cli)"

# 执行上述命令，此时Terminal已经支持自动补全

$ eval "$(_MA_CLI_COMPLETE=bash_source ma-cli)"
```

```
# 显示Fish Shell自动补全命令
$ ma-cli auto-completion Fish
Tips: please paste following shell command to your terminal to activate auto complation.

[ OK ] eval (env _MA_CLI_COMPLETE=fish_source ma-cli)
```

10.4 ma-cli configure 鉴权命令

鉴权信息说明

- 在虚拟机及个人PC场景，需要配置鉴权信息，目前支持用户名密码鉴权（默认）和AK/SK鉴权；
- 在使用账号认证时，需要指定username和password；在使用IAM用户认证时，需要指定account、username和password；
- 在ModelArts Notebook中可以不用执行鉴权命令，默认使用委托信息，不需要手动进行鉴权操作；
- 如果用户在ModelArts Notebook中也配置了鉴权信息，那么将会优先使用用户指定的鉴权信息。

  说明

  在鉴权时，注意您的敏感信息数据保护，避免敏感信息泄露。

命令参数总览

```
$ ma-cli configure -h
Usage: ma-cli configure [OPTIONS]

Options:
  -auth, --auth [PWD|AKSK|ROMA]  Authentication type.
  -rp, --region-profile PATH      ModelArts region file path.
  -a, --account TEXT              Account of an IAM user.
  -u, --username TEXT             Username of an IAM user.
  -p, --password TEXT             Password of an IAM user
  -ak, --access-key TEXT          User access key.
  -sk, --secret-key TEXT          User secret key.
  -r, --region TEXT               The region you want to visit.
  -pi, --project-id TEXT          User project id.
  -C, --config-file TEXT          Configure file path for authorization.
  -D, --debug                     Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT              CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help                  Show this message and exit.
```


版权所有 © 华为云计算技术有限公司

表 10-2 鉴权命令参数说明

参数名	参数类型	是否必选	参数说明
-auth / --auth	String	否	鉴权方式，支持PWD（用户名密码）、AKSK（access key和secret key），默认是PWD。
-rp / --region-profile	String	否	指定ModelArts region配置文件信息。
-a / --account	String	否	IAM租户账号，在使用IAM用户认证场景时需要指定，属于PWD鉴权的一部分。
-u / --username	String	否	用户名，在使用账号认证时表示账号名，IAM认证时表示IAM用户名，在云星账号场景不需要指定，属于PWD鉴权的一部分。
-p / --password	String	否	密码，属于PWD鉴权的一部分。
-ak / --access-key	String	否	access key，属于AKSK鉴权的一部分。
-sk / --secret-key	String	否	secret key，属于AKSK鉴权的一部分。
-r / --region	String	否	region名称，如果不填会默认使用REGION_NAME环境变量的值。
-pi / --project-id	String	否	项目ID，如果不填会默认使用对应region的值，或者使用PROJECT_ID环境变量。
-P / --profile	String	否	鉴权配置项，默认是DEFAULT。
-C / --config-file	String	否	配置文件本地路径，默认路径为~/.modelarts/ma-cli-profile.yaml。

配置用户名密码鉴权

以在虚拟机上使用**ma-cli configure**为例，介绍如何配置用户名密码进行鉴权。

说明

以下样例中所有以${}装饰的字符串都代表一个变量，用户可以根据实际情况指定对应的值。

比如${your_password}表示输入用户自己的密码信息。

```
# 默认使用DEFAULT鉴权配置项，默认提示账号、用户名及密码（其中账号和用户名如果不需要填写可以使用Enter跳过）
$ ma-cli configure --auth PWD --region ${your_region}
account: ${your_account}
username: ${your_username}
password: ${your_password}  # 输入在控制台不会回显
```


版权所有 © 华为云计算技术有限公司

AKSK 鉴权

如下命令表示使用AKSK进行鉴权，需要交互式输入AK及SK信息。默认提示AK和SK，且输入在控制台不会回显。

⚠ 注意

以下样例中所有以${}装饰的字符串都代表一个变量，用户可以根据实际情况指定对应的值。

比如${access key}表示输入用户自己的access key。

```
ma-cli configure --auth AKSK
access key [***]: ${access key}
secret key [***]: ${secret key}
```

执行完鉴权命令后，将会在~/.modelarts/ma-cli-profile.yaml配置文件中保存相应的鉴权信息。

10.5 ma-cli image 镜像构建命令

10.5.1 ma-cli image 镜像构建命令概述

ma-cli image命令支持：查询用户已注册的镜像、查询/加载镜像构建模板、Dockerfile镜像构建、查询/清理镜像构建缓存、注册/取消注册镜像、调试镜像是否可以在Notebook中使用等。具体命令及功能可执行**ma-cli image -h**命令查看。

镜像构建命令总览

```
$ ma-cli image -h
Usage: ma-cli image [OPTIONS] COMMAND [ARGS]...
  Support get registered image list, register or unregister image, debug image, build image in Notebook.

Options:
  -H, -h, --help  Show this message and exit.

Commands:
  add-template, at  List build-in dockerfile templates.
  build             Build docker image in Notebook.
  debug             Debug SWR image as a Notebook in ECS.
  df                Query disk usage.
  get-image, gi     Query registered image in ModelArts.
  get-template, gt  List build-in dockerfile templates.
  prune             Prune image build cache.
  register          Register image to ModelArts.
  unregister        Unregister image from ModelArts.
```

表 10-3 镜像构建支持的命令

命令	命令详情
get-template	查询镜像构建模板。

版权所有 © 华为云计算技术有限公司

命令	命令详情
add-template	加载镜像构建模板。
get-image	查询ModelArts已注册镜像。
register	注册SWR镜像到ModelArts镜像管理。
unregister	取消注册ModelArts镜像管理中的已注册镜像。
build	基于指定的Dockerfile构建镜像 （只支持ModelArts Notebook里使用）。
df	查询镜像构建缓存（只支持ModelArts Notebook里使用）。
prune	清理镜像构建缓存 （只支持ModelArts Notebook里使用）。
debug	在ECS上调试SWR镜像是否能在ModelArts Notebook中使用 （只支持已安装docker环境的ECS）。

10.5.2 查询镜像构建模板

ma-cli提供了一些常用的镜像构建模板，模板中包含了在ModelArts Notebook上进行Dockerfile开发的牵引指导。

```
$ ma-cli image get-template -h
Usage: ma-cli image get-template [OPTIONS]

  List build-in dockerfile templates.

  Example:

  # List build-in dockerfile templates
  ma-cli image get-template [--filer <filter_info>]  [--page-num <yourPageNum>] [--page-size
<yourPageSize>]

Options:
  --filter TEXT                   filter by keyword.
  -pn, --page-num INTEGER RANGE   Specify which page to query.  [x>=1]
  -ps, --page-size INTEGER RANGE  The maximum number of results for this query.  [x>=1]
  -D, --debug                     Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT              CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help                  Show this message and exit.
(PyTorch-1.4) [ma-user work]$
```

表 10-4 参数说明

参数名	参数类型	是否必选	参数说明
--filter	String	否	根据模板名称关键字过滤模板列表。
-pn / --page-num	Int	否	镜像页索引，默认是第1页。

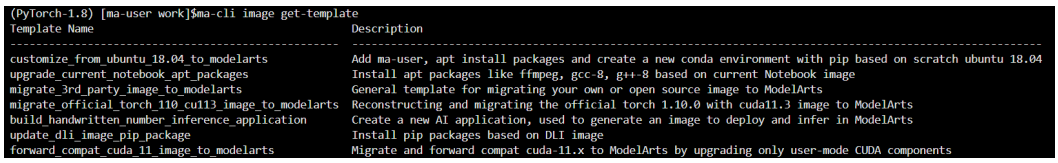
版权所有 © 华为云计算技术有限公司

参数名	参数类型	是否必选	参数说明
-ps / --page-size	Int	否	每页显示的镜像数量，默认是20。

示例

查看镜像构建模板。

```
ma-cli image get-template
```



10.5.3 加载镜像构建模板

ma-cli可以使用**add-template**命令将镜像模板加载到指定文件夹下，默认路径为当前命令所在的路径。

比如${current_dir}/.ma/${template_name}/。也可以通过--dest命令指定保存的路径。当保存的路径已经有同名的模板文件夹时，可以使用--force | -f参数进行强制覆盖。

```
$ ma-cli image add-template -h
Usage: ma-cli image add-template [OPTIONS] TEMPLATE_NAME

  Add buildin dockerfile templates into disk.

  Example:

  # List build-in dockerfile templates
  ma-cli image add-template customize_from_ubuntu_18.04_to_modelarts --force

Options:
  --dst TEXT          target save path.
  -f, --force         Override templates that has been installed.
  -D, --debug         Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT  CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help      Show this message and exit.
```

表 10-5 参数说明

参数名	参数类型	是否必选	参数说明
--dst	String	否	加载模板到指定路径，默认是当前路径。
-f / --force	Bool	否	是否强制覆盖已存在的同名模板，默认不覆盖。

版权所有 © 华为云计算技术有限公司

示例

加载**customize_from_ubuntu_18.04_to_modelarts**镜像构建模板。

```
ma-cli image add-template customize_from_ubuntu_18.04_to_modelarts
```

```
(PyTorch-1.8) [ma-user work]$ma-cli image add-template customize_from_ubuntu_18.04_to_modelarts
[ OK ] Successfully add configuration template [ customize_from_ubuntu_18.04_to_modelarts ] under folder [ /home/ma-user/work/.ma/customize_from_ubuntu_18.04_to_modelarts ]
```

10.5.4 查询 ModelArts 已注册镜像

Dockerfile一般需要提供一个基础镜像的地址，目前支持从docker hub等开源镜像仓拉取公开镜像，以及SWR的公开或私有镜像。其中ma-cli提供了查询ModelArts预置镜像和用户已注册镜像列表及SWR地址。

```
$ma-cli image get-image -h
Usage: ma-cli image get-image [OPTIONS]

  Get registered image list.

  Example:

  # Query images by image type and only image id, show name and swr_path
  ma-cli image get-image --type=DEDICATED

  # Query images by image id
  ma-cli image get-image --image-id ${image_id}

  # Query images by image type and show more information
  ma-cli image get-image --type=DEDICATED -v

  # Query images by image name
  ma-cli image get-image --filter=torch

Options:
  -t, --type [BUILD_IN|DEDICATED|ALL]
                                  Image type(default ALL)
  -f, --filter TEXT               Image name to filter
  -v, --verbose                   Show detailed information on image.
  -i, --image-id TEXT             Get image details by image id
  -n, --image-name TEXT           Get image details by image name
  -wi, --workspace-id TEXT        The workspace where you want to query image(default "0")
  -pn, --page-num INTEGER RANGE   Specify which page to query  [x>=1]
  -ps, --page-size INTEGER RANGE  The maximum number of results for this query  [x>=1]
  -C, --config-file PATH          Configure file path for authorization.
  -D, --debug                     Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT              CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help                  Show this message and exit.
```

表 10-6 参数说明

参数名	参数类型	是否必选	参数说明
-t / --type	String	否	查询的镜像类型，支持BUILD_IN、DEDICATED和ALL三种查询类型。 ● BUILD_IN：预置镜像 ● DEDICATED：用户已注册的自定义镜像 ● ALL：所有镜像

版权所有 © 华为云计算技术有限公司

参数名	参数类型	是否必选	参数说明
-f / --filter	String，	否	镜像名关键字。根据镜像名关键字过滤镜像列表。
-v / --verbose	Bool	否	显示详细的信息开关，默认关闭。
-i / --image-id	String	否	查询指定镜像ID的镜像详情。
-n / --image-name	String	否	查询指定镜像名称的镜像详情。
-wi / --workspace-id	String	否	查询指定工作空间下的镜像信息。
-pn / --page-num	Int	否	镜像页索引，默认是第1页。
-ps / --page-size	Int	否	每页显示的镜像数量，默认是20。

示例

查询ModelArts已注册的自定义镜像。

```
ma-cli image get-image --type=DEDICATED
```

```
(PyTorch-1.8) [ma-user work]$ma-cli image get-image --type=DEDICATED
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
| INDEX |              IMAGE ID                |    NAME    |                                  SWR PATH                                   |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   1   | c857e5a8                 fc5e3d002f  |  0314test  |          huaweicloud.com/notebook_test/0314test:1.0.0                       |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   2   | 193b2557                 d39093a811  |    0328    |          7.myhuaweicloud.com/notebook_test/0328:1                           |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   3   | 171fe036                 3b37e9aa7c  |    0926    |          aweicloud.com/ei_modelarts_y00218826_05/0926:1                      |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   4   | 1b48bb0a                 689b0a7267  |    0926    |          weicloud.com/ei_modelarts_y00218826_05/0926:111                    |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   5   | c8667cf0                 d2e3563107  |     1      |          huaweicloud.com/ei_modelarts_y00218826_05/1:6                      |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   6   | 3e6cda6a                 a360eea80e  |     1      |          huaweicloud.com/ei_modelarts_y00218826_05/1:1                      |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   7   | 42e86ca5                 ec198be968  |    111     |          .myhuaweicloud.com/notebook_test/111:1227                          |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   8   | 0f349cef                 c411011ef2  | 11111110801|          aweicloud.com/notebook_test/11111110801:111111                     |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   9   | 3a082e32                 4f485aadb6  |   112121   |          eicloud.com/ei_modelarts_y00218826_05/112121:123                   |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   10  | db0d02f6                 74eb00e1ce  |    1203    |          myhuaweicloud.com/notebook_test/1203:1.2.3                         |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   11  | 031dc02e                 d92cd457d8  |    1227    |          .myhuaweicloud.com/notebook_test/1227:111                          |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   12  | f7d95648                 7aaec8b1cc  |    1227    |          .myhuaweicloud.com/notebook_test/1227:888                          |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   13  | 2f720610                 a1d1db9d7d  |    1227    |          .myhuaweicloud.com/notebook_test/1227:6666                         |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   14  | 42221bf2                 22d726d270  |    1229    |          .myhuaweicloud.com/notebook_test/1229:123                          |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   15  | 70deea1e                 70b2414ae7  |    123     |          myhuaweicloud.com/mindspore-dis-train/123:2                        |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   16  | e6cc5414                 ce318069f4  |    123     |          .myhuaweicloud.com/notebook_test/123:45678                         |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   17  | 6e7a86c9                 319fb3bb28  |    1234    |          .myhuaweicloud.com/notebook_test/1234:666                          |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   18  | ec036306                 8c9dc6b391  |    1234    |          7.myhuaweicloud.com/notebook_test/1234:1                           |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   19  | b37f8f3b                 7a9941c978  |   441211   |          .myhuaweicloud.com/notebook_test/441211:11                        |
+-------+--------------------------------------+------------+-----------------------------------------------------------------------------+
|   20  | d5acd51b                 ef16534d68  |    aaa     |          .myhuaweicloud.com/notebook_test/aaa:1.1.1                         |
```


版权所有 © 华为云计算技术有限公司

10.5.5 在 ModelArts Notebook 中进行镜像构建

使用**ma-cli image build**命令基于指定的Dockerfile进行镜像构建，仅支持在ModelArts Notebook里使用该命令。

```
$ ma-cli image build -h
Usage: ma-cli image build [OPTIONS] FILE_PATH

  Build docker image in Notebook.

  Example:

  # Build a image and push to SWR
  ma-cli image build .ma/customize_from_ubuntu_18.04_to_modelarts/Dockerfile -swr my_organization/
my_image:0.0.1

  # Build a image and push to SWR, dockerfile context path is current dir
  ma-cli image build .ma/customize_from_ubuntu_18.04_to_modelarts/Dockerfile -swr my_organization/
my_image:0.0.1 -context .

  # Build a local image and save to local path and OBS
  ma-cli image build .ma/customize_from_ubuntu_18.04_to_modelarts/Dockerfile --target ./build.tar --
obs_path obs://bucket/object --swr-path my_organization/my_image:0.0.1

Options:
  -t, --target TEXT        Name and optionally a tag in the 'name:tag' format.
  -swr, --swr-path TEXT    SWR path without swr endpoint, eg:organization/image:tag.  [required]
  --context DIRECTORY      build context path.
  -arg, --build-arg TEXT   build arg for Dockerfile.
  -obs, --obs-path TEXT    OBS path to save local built image.
  -f, --force              Force to overwrite the existing swr image with the same name and tag.
  -C, --config-file PATH   Configure file path for authorization.
  -D, --debug              Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT       CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help           Show this message and exit.
```

表 10-7 参数说明

参数名	参数类型	是否必选	参数说明
FILE_PATH	String	是	Dockerfile文件所在的路径。
-t / --target	String	否	表示构建生成的tar包保存在本地的路径，默认是当前文件夹目录。
-swr / --swr-path	String	是	SWR镜像名称，遵循organization/image_name:tag格式，针对于构建保存tar包场景可以省略。
--context	String	否	Dockerfile构建时的上下文信息路径，主要用于数据复制。
-arg / --build-arg	String	否	指定构建参数，多个构建参数可以使用--build-arg VERSION=18.04 --build-arg ARCH=X86_64
-obs / --obs-path	String	否	将生成的tar包自动上传到OBS中。
-f / --force	Bool	否	是否强制覆盖已存在的SWR镜像，默认不覆盖。

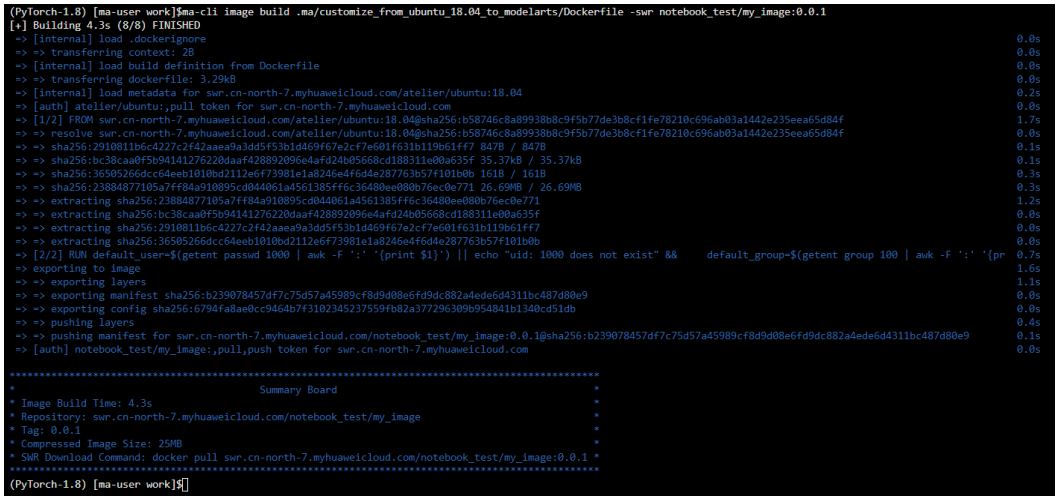
版权所有 © 华为云计算技术有限公司

示例

在ModelArts Notebook里进行镜像构建。

```
ma-cli image build .ma/customize_from_ubuntu_18.04_to_modelarts/Dockerfile -swr notebook_test/my_image:0.0.1
```

其中“.ma/customize_from_ubuntu_18.04_to_modelarts/Dockerfile”为Dockerfile文件所在路径，“notebook_test/my_image:0.0.1”为构建的新镜像的SWR路径。



10.5.6 在 ModelArts Notebook 中查询镜像构建缓存

使用**ma-cli image df**命令查询镜像构建缓存，仅支持在ModelArts Notebook里使用该命令。

```
$ ma-cli image df -h
Usage: ma-cli image df [OPTIONS]

  Query disk usage used by image-building in Notebook.

  Example:

  # Query image disk usage
  ma-cli image df

Options:
  -v, --verbose      Show detailed information on disk usage.
  -D, --debug        Debug Mode. Shows full stack trace when error occurs.

  -h, -H, --help     Show this message and exit.
```

表 10-8 参数说明

参数名	参数类型	是否必选	参数说明
-v / --verbose	Bool	否	显示详细的信息开关，默认关闭。

示例

- 在ModelArts Notebook里查看所有镜像缓存。
  ```
  ma-cli image df
  ```


版权所有 © 华为云计算技术有限公司

```
(PyTorch-1.8) [ma-user work]$ma-cli image df
ID                                                              RECLAIMABLE     SIZE            LAST ACCESSED
iwrrwsi9pdcjafe1ij6d0r918                                       true            98.50MB
cp52c4q81ud2abu2vp7sj5vyt                                       true            1.04MB
4jbo6v06r2wl575ddq3w8g12e                                       true            139.68kB
ojdjw5mok71s1nh2cauant051                                       true            86.86kB
k2jm6g061n5twmz7gmonmqjsh                                       true            16.55kB
efu5kwgig1ve44fe7smbrncnh*                                      true            8.19kB
uzikwqk5taxnslvajm14jrbje*                                      true            4.10kB
2g8p0qcb014g3qva7ucawkv87*                                      true            4.10kB
Reclaimable:    99.80MB
Total:          99.80MB
```

- 显示镜像的详细信息。
  ma-cli image df --verbose

```
(PyTorch-1.8) [ma-user work]$ma-cli image df --verbose
ID:             iwrrwsi9pdcjafe1ij6d0r918
Created at:     2023-03-28 12:23:28.353759532 +0000 UTC
Mutable:        false
Reclaimable:    true
Shared:         false
Size:           98.50MB
Description:    pulled from swr            .myhuaweicloud.com/atelier/ubuntu:18.04@sha256:b5874                                                                ea65d84f
Usage count:    1
Last used:      2023-03-28 12:23:28.37337776 +0000 UTC
Type:           regular

ID:             cp52c4q81ud2abu2vp7sj5vyt
Parents:        iwrrwsi9pdcjafe1ij6d0r918
Created at:     2023-03-28 12:23:28.366910223 +0000 UTC
Mutable:        false
Reclaimable:    true
Shared:         false
Size:           1.04MB
Description:    pulled from swr.           .myhuaweicloud.com/atelier/ubuntu:18.04@sha256:b5874                                                               2e235eea65d84f
Usage count:    1
Last used:      2023-03-28 12:23:28.38560437 +0000 UTC
Type:           regular

ID:             4jbo6v06r2wl575ddq3w8g12e
Parents:        k2jm6g061n5twmz7gmonmqjsh
Created at:     2023-03-28 12:23:30.681643727 +0000 UTC
Mutable:        false
Reclaimable:    true
Shared:         false
Size:           139.68kB
Description:    mount / from exec /bin/sh -c default_user=$(getent passwd 1000 | awk -F ':' '{print $1}') || echo "uid: 1000 does not exist" &&     default_group=$(getent
up 100 | awk -F ':' '{print $1}') || echo "gid: 100 does not exist" &&     if [ ! -z ${default_user} ] && [ ${default_user} != "ma-user" ]; then         userdel -r ${defau
user};     fi &&     if [ ! -z ${default_group} ] && [ ${default_group} != "ma-group" ]; then         groupdel -f ${default_group};     fi &&     groupadd -g 100 ma-group
useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user &&     chmod -R 750 /home/ma-user
Usage count:    2
Last used:      2023-03-28 12:25:39.149080471 +0000 UTC
Type:           regular
```

10.5.7 在 ModelArts Notebook 中清理镜像构建缓存

使用**ma-cli image prune**命令清理镜像构建缓存，仅支持在ModelArts Notebook里使用该命令。

```
$ ma-cli image prune -h
Usage: ma-cli image prune [OPTIONS]

  Prune image build cache by image-building in Notebook.

  Example:

  # Prune image build cache
  ma-cli image prune

Options:
  -ks, --keep-storage INTEGER  Amount of disk space to keep for cache below this limit (in MB) (default: 0).
  -kd, --keep-duration TEXT    Keep cache newer than this limit, support second(s), minute(m) and hour(h)
(default:   0s).
  -v, --verbose                Show more verbose output.
  -D, --debug                  Debug Mode. Shows full stack trace when error occurs.
  -h, -H, --help               Show this message and exit.
```

表 10-9 参数说明

参数名	参数类型	是否必选	参数说明
-ks / --keep-storage	Int	否	清理缓存时保留的缓存大小，单位是MB，默认是0，表示全部清理。

版权所有 © 华为云计算技术有限公司

参数名	参数类型	是否必选	参数说明
-kd / --keep-duration	String	否	清理缓存时保留较新的缓存，只清除历史缓存，单位为s（秒）、m（分钟）、h（小时），默认是0s，表示全部清理。
-v / --verbose	Bool	否	显示详细的信息开关，默认关闭。

示例

清理保留1MB镜像缓存。

```
ma-cli image prune -ks 1
```

```
(PyTorch-1.8) [ma-user work]$ma-cli image prune -ks 1
ID                                                                RECLAIMABLE     SIZE    LAST ACCESSED
uzikwqk5taxnslvajm14jrbje*                                        true            4.10kB
4jbo6v06r2wl575ddq3w8g12e                                         true            139.68kB
k2jm6g061n5twmz7gmonmqjsh                                         true            16.55kB
ojdjw5mok71s1nh2cauant051                                         true            86.86kB
cp52c4q81ud2abu2vp7sj5vyt                                         true            1.04MB
iwrrwsi9pdcjafe1ij6d0r918                                         true            98.50MB
Total:  99.79MB
```

10.5.8 注册 SWR 镜像到 ModelArts 镜像管理

调试完成后，使用**ma-cli image register**命令将新镜像注册到ModelArts镜像管理服务中，进而在能够在ModelArts中使用该镜像。

```
$ma-cli image register -h
Usage: ma-cli image register [OPTIONS]

  Register image to ModelArts.

  Example:

  # Register image into ModelArts service
  ma-cli image register --swr-path=xx

  # Share SWR image to DLI service
  ma-cli image register -swr xx -td

  # Register image into ModelArts service and specify architecture to be 'AARCH64'
  ma-cli image register --swr-path=xx --arch AARCH64

Options:
  -swr, --swr-path TEXT           SWR path without swr endpoint, eg:organization/image:tag.  [required]
  -a, --arch [X86_64|AARCH64]     Image architecture (default: X86_64).
  -s, --service [NOTEBOOK|MODELBOX]
                                  Services supported by this image(default NOTEBOOK).
  -rs, --resource-category [CPU|GPU|ASCEND]
                                  The resource category supported by this image (default: CPU and GPU).
  -wi, --workspace-id TEXT        The workspace to register this image (default: "0").
  -v, --visibility [PUBLIC|PRIVATE]
                                  PUBLIC: every user can use this image. PRIVATE: only image owner can use this
image (Default: PRIVATE).
  -td, --to-dli                   Register swr image to DLI, which will share SWR image to DLI service.
  -d, --description TEXT          Image description (default: "").
  -C, --config-file PATH          Configure file path for authorization.
  -D, --debug                     Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT              CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help                  Show this message and exit.
```


版权所有 © 华为云计算技术有限公司

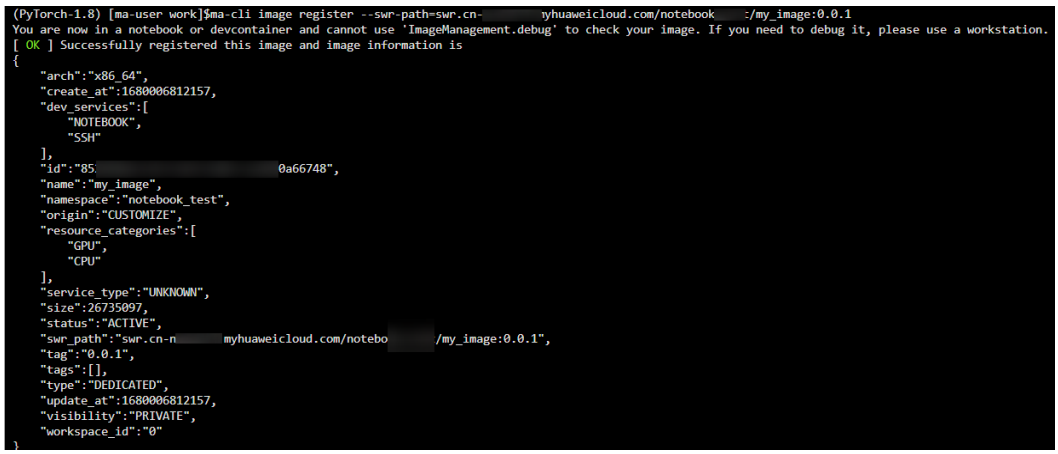
表 10-10 参数说明

参数名	参数类型	是否必选	参数说明
-swr / --swr-path	String	是	需要注册的镜像的SWR路径。
-a / --arch	Sring	否	注册镜像的架构，X86_64或者AARCH64，默认是X86_64。
-s / --service	String	否	注册镜像的服务类型，NOTEBOOK或者MODELBOX，默认是NOTEBOOK。 可以输入多个值，如-s NOTEBOOK -s MODELBOX。
-rs / --resource-category	String	否	注册镜像能够使用的资源类型，默认是CPU和GPU。
-wi / --workspace-id	String	否	注册镜像到指定的工作空间，workspace ID默认是0。
-v / --visibility	Bool	否	注册的镜像可见性，PRIVATE（仅自己可见）或者PUBLIC（所有用户可见），默认是PRIVATE。
-td / --to-dli	Bool	否	注册镜像到DLI服务。
-d/ --description	String	否	填写镜像描述，默认为空。

示例

注册SWR镜像到ModelArts。

```
ma-cli image register --swr-path=xx
```



10.5.9 取消注册 ModelArts 镜像管理中的已注册镜像

使用**ma-cli image unregister**命令将注册的镜像从ModelArts中删除。

版权所有 © 华为云计算技术有限公司

```
$ ma-cli image unregister -h
Usage: ma-cli image unregister [OPTIONS]

  Unregister image from ModelArts.

  Example:

  # Unregister image
  ma-cli image unregister --image-id=xx

  # Unregister image and delete it from swr
  ma-cli image unregister --image-id=xx -d

Options:
  -i, --image-id TEXT     Unregister image details by image id.  [required]
  -d, --delete-swr-image  Delete the image from swr.
  -C, --config-file PATH  Configure file path for authorization.
  -D, --debug             Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT      CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help          Show this message and exit.
```

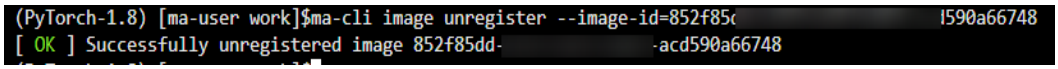
表 10-11 参数说明

参数名	参数类型	是否必选	参数说明
-i / -image-id	String	是	需要取消注册的镜像ID。
-d / --delete-swr-image	Bool	否	取消注册后同步删除SWR镜像开关，默认关闭。

示例

取消ModelArts镜像管理中已注册镜像。

```
ma-cli image unregister --image-id=xx
```



10.5.10 在 ECS 上调试 SWR 镜像是否能在 ModelArts Notebook 中使用

ma-cli支持在ECS上调试SWR镜像是否可以在ModelArts开发环境中运行，发现镜像中可能存在的问题。

版权所有 © 华为云计算技术有限公司

```
ma-cli image debug -h
Usage: ma-cli image debug [OPTIONS]

  Debug SWR image as a Notebook in ECS.

  Example:

  # Debug cpu notebook image
  ma-cli image debug --swr-path=xx --service=NOTEBOOK --region=

  # Debug gpu notebook image
  ma-cli image debug --swr-path=xx --service=NOTEBOOK --region=  --gpu

Options:
  -swr, --swr-path TEXT           SWR path without SWR endpoint, eg:organization/image:tag.  [required]
  -r, --region TEXT               Region name.  [required]
  -s, --service [NOTEBOOK|MODELBOX]
                                  Services supported by this image(default NOTEBOOK).
  -a, --arch [X86_64|AARCH64]     Image architecture(default X86_64).
  -g, --gpu                       Use all gpus to debug.
  -D, --debug                     Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT              CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help                  Show this message and exit.
```

表 10-12 参数说明

参数名	参数类型	是否必选	参数说明
-swr / --swr-path	String	是	需要调试的镜像的SWR路径。
-r / --region	String	是	需要调试的镜像所在的区域。
-s / --service	String	否	调试镜像的服务类型，NOTEBOOK或者MODELBOX，默认是NOTEBOOK。
-a / --arch	Sring	否	调试镜像的架构，X86_64或者AARCH64，默认是X86_64。
-g / --gpu	Bool	否	使用GPU进行调试开关，默认关闭。

10.6 使用 ma-cli ma-job 命令提交 ModelArts 训练作业

10.6.1 ma-cli ma-job 命令概述

使用**ma-cli ma-job**命令可以提交训练作业，查询训练作业日志、事件、使用的AI引擎、资源规格及停止训练作业等。

```
$ ma-cli ma-job -h
Usage: ma-cli ma-job [OPTIONS] COMMAND [ARGS]...

  ModelArts job submission and query jod details.
```


版权所有 © 华为云计算技术有限公司

```
Options:
  -h, -H, --help  Show this message and exit.

Commands:
  delete       Delete training job by job id.
  get-engine   Get job engines.
  get-event    Get job running event.
  get-flavor   Get job flavors.
  get-job      Get job details.
  get-log      Get job log details.
  get-pool     Get job engines.
  stop         Stop training job by job id.
  submit       Submit training job.
```

表 10-13 训练作业支持的命令

命令	命令详情
get-job	查询ModelArts训练作业列表及详情。
get-log	查询ModelArts训练作业运行日志。
get-engine	查询ModelArts训练AI引擎。
get-event	查询ModelArts训练作业事件。
get-flavor	查询ModelArts训练资源规格。
get-pool	查询ModelArts训练专属池。
stop	停止ModelArts训练作业。
submit	提交ModelArts训练作业。
delete	删除指定作业id的训练作业。

10.6.2 查询 ModelArts 训练作业

使用**ma-cli ma-job get-job**命令可以查看训练作业列表或某个作业详情。

```
$ ma-cli ma-job get-job -h
Usage: ma-cli ma-job get-job [OPTIONS]

  Get job details.

  Example:

  # Get train job details by job name
  ma-cli ma-job get-job -n ${job_name}

  # Get train job details by job id
  ma-cli ma-job get-job -i ${job_id}

  # Get train job list
  ma-cli ma-job get-job --page-size 5 --page-num 1

Options:

  -i, --job-id TEXT               Get training job details by job id.
  -n, --job-name TEXT             Get training job details by job name.
  -pn, --page-num INTEGER         Specify which page to query.  [x>=1]
  -ps, --page-size INTEGER RANGE  The maximum number of results for this query.  [1<=x<=50]
```


版权所有 © 华为云计算技术有限公司

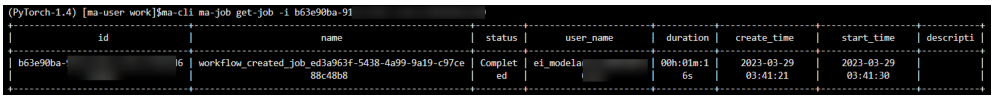
```
-v, --verbose          Show detailed information about training job details.
-C, --config-file TEXT    Configure file path for authorization.
-D, --debug            Debug Mode. Shows full stack trace when error occurs.
-P, --profile TEXT     CLI connection profile to use. The default profile is "DEFAULT".
-h, -H, --help         Show this message and exit.
```

表 10-14 参数说明

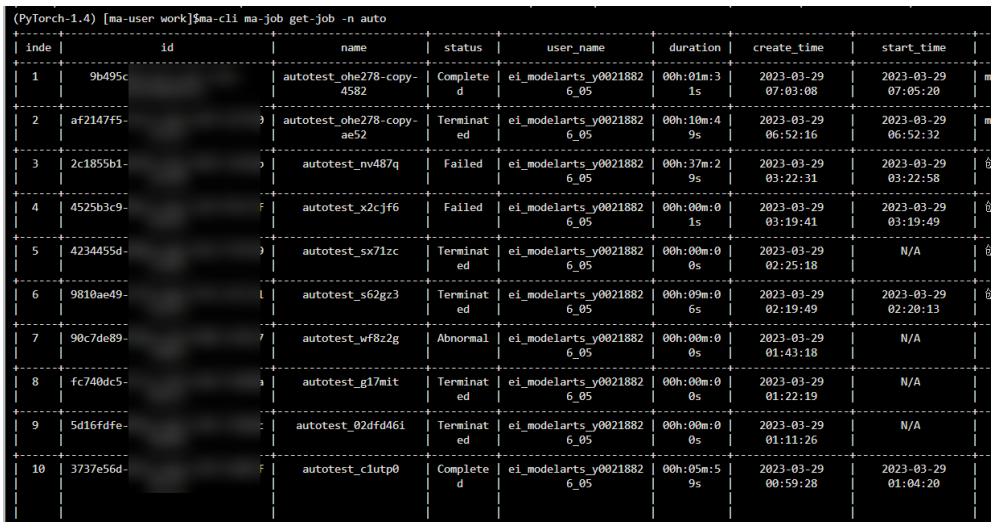
参数名	参数类型	是否必选	参数说明
-i / --job-id	String	否	查询指定训练任务ID的任务详情。
-n / --job-name	String	否	查询指定任务名称的训练任务或根据任务名称关键字过滤训练任务。
-pn / --page-num	Int	否	页面索引，默认是第1页。
-ps / --page-size	Int	否	每页显示的训练作业数量，默认是10。
-v / --verbose	Bool	否	显示详细的信息开关，默认关闭。

示例

- 查询指定任务ID的训练任务。
  ma-cli ma-job get-job -i *b63e90xxx*



- 根据任务名称关键字“auto”过滤训练任务。
  ma-cli ma-job get-job -n *auto*



10.6.3 提交 ModelArts 训练作业

执行**ma-cli ma-job submit**命令提交ModelArts训练作业。

版权所有 © 华为云计算技术有限公司

ma-cli ma-job submit命令需要指定一个位置参数YAML_FILE表示作业的配置文件路径，如果不指定该参数，则表示配置文件为空。配置文件是一个YAML格式的文件，里面的参数就是命令的option参数。此外，如果用户在命令行中同时指定YAML_FILE配置文件和option参数，命令行中指定的option参数的值将会覆盖配置文件相同的值。

```
$ma-cli ma-job submit -h
Usage: ma-cli ma-job submit [OPTIONS] [YAML_FILE]...

  Submit training job.

  Example:

  ma-cli ma-job submit --code-dir obs://your_bucket/code/
                       --boot-file main.py
                       --framework-type PyTorch
                       --working-dir /home/ma-user/modelarts/user-job-dir/code
                       --framework-version pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64
                       --data-url obs://your_bucket/dataset/
                       --log-url obs://your_bucket/logs/
                       --train-instance-type modelarts.vm.cpu.8u
                       --train-instance-count 1

Options:
  --name TEXT                   Job name.
  --description TEXT            Job description.
  --image-url TEXT              Full swr custom image path.
  --uid TEXT                   Uid for custom image (default: 1000).
  --working-dir TEXT            ModelArts training job working directory.
  --local-code-dir TEXT         ModelArts training job local code directory.
  --user-command TEXT           Execution command for custom image.
  --pool-id TEXT               Dedicated pool id.
  --train-instance-type TEXT    Train worker specification.
  --train-instance-count INTEGER  Number of workers.
  --data-url TEXT               OBS path for training data.
  --log-url TEXT               OBS path for training log.
  --code-dir TEXT               OBS path for source code.
  --output TEXT                Training output parameter with OBS path.
  --input TEXT                 Training input parameter with OBS path.
  --env-variables TEXT          Env variables for training job.
  --parameters TEXT             Training job parameters (only keyword parameters are supported).
  --boot-file TEXT             Training job boot file path behinds `code_dir`.
  --framework-type TEXT         Training job framework type.
  --framework-version TEXT      Training job framework version.
  --workspace-id TEXT           The workspace where you submit training job(default "0")
  --policy [regular|economic|turbo|auto]
                          Training job policy, default is regular.
  --volumes TEXT                Information about the volumes attached to the training job.
  -q, --quiet                  Exit without waiting after submit successfully.
  -C, --config-file PATH        Configure file path for authorization.
  -D, --debug                  Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT            CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help               Show this message and exit.
```

表 10-15 参数说明

参数名	参数类型	是否必选	参数说明
YAML_FILE	String	否	表示训练作业的配置文件，如果不传则表示配置文件为空。

版权所有 © 华为云计算技术有限公司

参数名	参数类型	是否必选	参数说明
--code-dir	String	是	训练源代码的OBS路径。
--data-url	String	是	训练数据的OBS路径。
--log-url	String	是	存放训练生成日志的OBS路径。
--train-instance-count	String	是	训练作业计算节点个数，默认是1，表示单节点。
--boot-file	String	否	当使用自定义镜像或自定义命令时可以省略，当使用预置命令提交训练作业时需要指定该参数。
--name	String	否	训练任务名称。
--description	String	否	训练任务描述信息。
--image-url	String	否	自定义镜像SWR地址，遵循organization/image_name:tag
--uid	String	否	自定义镜像运行的UID，默认值1000。
--working-dir	String	否	运行算法时所在的工作目录。
--local-code-dir	String	否	算法的代码目录下载到训练容器内的本地路径。
--user-command	String	否	自定义镜像执行命令。需为/home下的目录。 当code-dir以file://为前缀时，当前字段不生效。
--pool-id	String	否	训练作业选择的资源池ID。可在ModelArts管理控制台，单击左侧“专属资源池”，在专属资源池列表中查看资源池ID。
--train-instance-type	String	否	训练作业选择的资源规格。
--output	String	否	训练的输出信息，指定后，训练任务将会把训练脚本中指定输出参数对应训练容器的输出目录上传到指定的OBS路径。如果需要指定多个参数，可以使用--output output1=obs://bucket/output1 --output output2=obs://bucket/output2

版权所有 © 华为云计算技术有限公司

参数名	参数类型	是否必选	参数说明
--input	String	否	训练的输入信息，指定后，训练任务将会把对应OBS上的数据下载到训练容器，并将数据存储路径通过指定的参数传递给训练脚本。如果需要指定多个参数，可以使用--input data_path1=obs://bucket/data1 --input data_path2=obs://bucket/data2
--env-variables	String	否	训练时传入的环境变量，如果需要指定多个参数，可以使用--env-variables ENV1=env1 --env-variables ENV2=env2
--parameters	String	否	训练入参，可以通过--parameters "--epoch 0 --pretrained"指定多个参数。
--framework-type	String	否	训练作业选择的引擎规格。
--framework-version	String	否	训练作业选择的引擎版本。
-q / --quiet	Bool	否	提交训练任务成功后直接退出，不再同步打印作业状态。
--workspace-id	String	否	作业所处的工作空间，默认值为“0”。
--policy	String	否	训练资源规格模式，可选值regular、economic、turbo、auto。
--volumes	String	否	挂载EFS，如果需要指定多个参数，可以使用--volumes。 "local_path=/xx/yy/zz;read_only=false;nfs_server_path=xxx.xxx.xxx.xxx:/" -volumes "local_path=/xxx/yyy/zzz;read_only=false;nfs_server_path=xxx.xxx.xxx.xxx:/"

基于 ModelArts 预置镜像提交训练作业

指定命令行options参数提交训练作业

```
ma-cli ma-job submit --code-dir obs://your-bucket/mnist/code/ \
                --boot-file main.py \
                --framework-type PyTorch \
                --working-dir /home/ma-user/modelarts/user-job-dir/code \
                --framework-version pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64 \
                --data-url obs://your-bucket/mnist/dataset/MNIST/ \
                --log-url obs://your-bucket/mnist/logs/ \
                --train-instance-type modelarts.vm.cpu.8u \
```


版权所有 © 华为云计算技术有限公司

```
                --train-instance-count 1  \
                -q
```

使用预置镜像的train.yaml样例：

```
# .ma/train.yaml样例（预置镜像）
# pool_id: pool_xxxx
train-instance-type: modelarts.vm.cpu.8u
train-instance-count: 1
data-url: obs://your-bucket/mnist/dataset/MNIST/
code-dir: obs://your-bucket/mnist/code/
working-dir: /home/ma-user/modelarts/user-job-dir/code
framework-type: PyTorch
framework-version: pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64
boot-file: main.py
log-url: obs://your-bucket/mnist/logs/

##[Optional] Uncomment to set uid when use custom image mode
uid: 1000

##[Optional] Uncomment to upload output file/dir to OBS from training platform
output:
   - name: output_dir
     obs_path: obs://your-bucket/mnist/output1/

##[Optional] Uncomment to download input file/dir from OBS to training platform
input:
   - name: data_url
     obs_path: obs://your-bucket/mnist/dataset/MNIST/

##[Optional] Uncomment pass hyperparameters
parameters:
   - epoch: 10
   - learning_rate: 0.01
   - pretrained:

##[Optional] Uncomment to use dedicated pool
pool_id: pool_xxxx

##[Optional] Uncomment to use volumes attached to the training job
volumes:
  - efs:
      local_path: /xx/yy/zz
      read_only: false
      nfs_server_path: xxx.xxx.xxx.xxx:/
```

基于自定义镜像创建训练作业

指定命令行options参数提交训练作业

```
ma-cli ma-job submit --image-url atelier/pytorch_1_8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-
x86_64-20220926104358-041ba2e \
                --code-dir obs://your-bucket/mnist/code/ \
                --user-command "export LD_LIBRARY_PATH=/usr/local/cuda/compat:$LD_LIBRARY_PATH &&
cd /home/ma-user/modelarts/user-job-dir/code && /home/ma-user/anaconda3/envs/PyTorch-1.8/bin/
python main.py" \
                --data-url obs://your-bucket/mnist/dataset/MNIST/ \
                --log-url obs://your-bucket/mnist/logs/ \
                --train-instance-type modelarts.vm.cpu.8u \
                --train-instance-count 1  \
                -q
```

使用自定义镜像的train.yaml样例：

```
# .ma/train.yaml样例（自定义镜像）
image-url: atelier/pytorch_1_8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-
x86_64-20220926104358-041ba2e
```


版权所有 © 华为云计算技术有限公司

```
user-command: export LD_LIBRARY_PATH=/usr/local/cuda/compat:$LD_LIBRARY_PATH && cd /home/ma-user/modelarts/user-job-dir/code && /home/ma-user/anaconda3/envs/PyTorch-1.8/bin/python main.py
train-instance-type: modelarts.vm.cpu.8u
train-instance-count: 1
data-url: obs://your-bucket/mnist/dataset/MNIST/
code-dir: obs://your-bucket/mnist/code/
log-url: obs://your-bucket/mnist/logs/

##[Optional] Uncomment to set uid when use custom image mode
uid: 1000

##[Optional] Uncomment to upload output file/dir to OBS from training platform
output:
  - name: output_dir
    obs_path: obs://your-bucket/mnist/output1/

##[Optional] Uncomment to download input file/dir from OBS to training platform
input:
  - name: data_url
    obs_path: obs://your-bucket/mnist/dataset/MNIST/

##[Optional] Uncomment pass hyperparameters
parameters:
  - epoch: 10
  - learning_rate: 0.01
  - pretrained:

##[Optional] Uncomment to use dedicated pool
pool_id: pool_xxxx

##[Optional] Uncomment to use volumes attached to the training job
volumes:
 - efs:
    local_path: /xx/yy/zz
    read_only: false
    nfs_server_path: xxx.xxx.xxx.xxx:/
```

示例

- 基于yaml文件提交训练作业

  ```
  ma-cli ma-job submit ./train-job.yaml
  ```

  ```
  (PyTorch-1.4) [ma-user work]$ma-cli ma-job submit ./train_job.yaml
  [ OK ] Current training job id is: 4d7c8584-b213-4f88-9833-d3e6a82f9e42
  [ OK ] Creating
  [ OK ] Running
  ```

- 基于命令行和预置镜像pytorch1.8-cuda10.2-cudnn7-ubuntu18.04提交训练作业。

  ```
  ma-cli ma-job submit --code-dir obs://automation-use-only/Original/TrainJob/TrainJob-v2/pytorch1.8.0_cuda10.2/code/ \
                       --boot-file test-pytorch.py \
                       --framework-type PyTorch \
                       --working-dir /home/ma-user/modelarts/user-job-dir/code \
                       --framework-version pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64 \
                       --data-url obs://automation-use-only/Original/TrainJob/TrainJob-v2/pytorch1.8.0_cuda10.2/data/ \
                       --log-url obs://automation-use-only/Original/TrainJob/TrainJob-v2/pytorch1.8.0_cuda10.2/data/logs/ \
                       --train-instance-type modelarts.vm.cpu.8u \
                       --train-instance-count 1 \
  ```


版权所有 © 华为云计算技术有限公司

10.6.4 查询 ModelArts 训练作业日志

执行**ma-cli ma-job get-log**命令查询ModelArts训练作业日志。

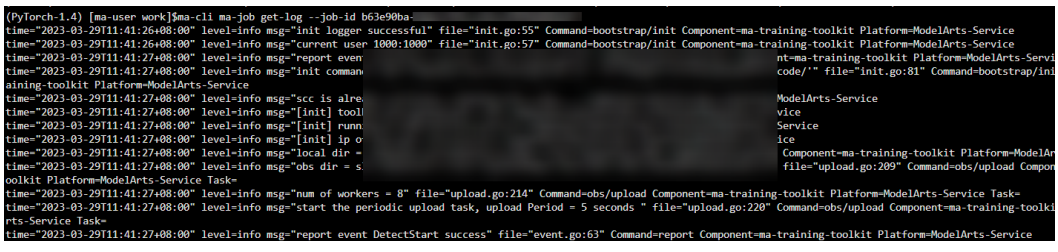
```
$ ma-cli ma-job get-log -h
Usage: ma-cli ma-job get-log [OPTIONS]

  Get job log details.

  Example:

  # Get job log by job id
  ma-cli ma-job get-log --job-id ${job_id}

Options:
  -i, --job-id TEXT       Get training job details by job id.  [required]
  -t, --task-id TEXT      Get training job details by task id (default "worker-0").
  -C, --config-file TEXT  Configure file path for authorization.
  -D, --debug             Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT      CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help          Show this message and exit.
```

参数名	参数类型	是否必选	参数说明
-i / --job-id	String	是	查询指定训练任务ID的任务日志。
-t / --task-id	String	否	查询指定task的日志，默认是work-0。

示例

查询指定训练任务ID的作业日志。

```
ma-cli ma-job get-log --job-id b63e90baxxx
```



10.6.5 查询 ModelArts 训练作业事件

执行**ma-cli ma-job get-event**命令查看ModelArts训练作业事件。

```
$ ma-cli ma-job get-event -h
Usage: ma-cli ma-job get-event [OPTIONS]

  Get job running event.
```


版权所有 © 华为云计算技术有限公司

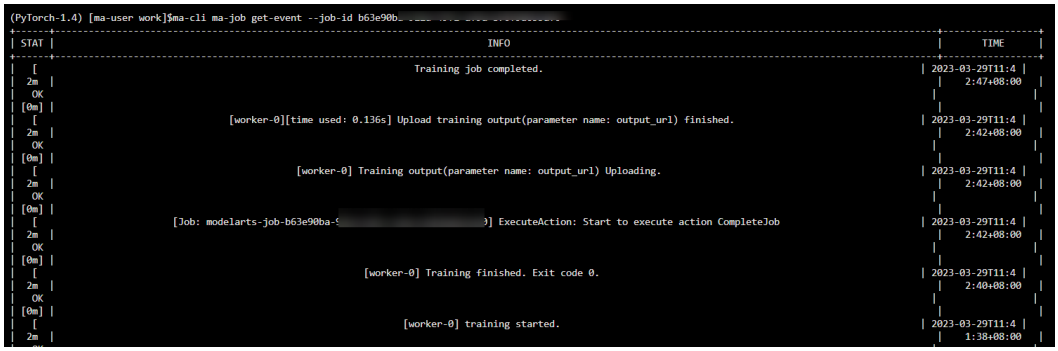
```
Example:

# Get training job running event
ma-cli ma-job get-event --job-id ${job_id}

Options:
  -i, --job-id TEXT       Get training job event by job id.  [required]
  -C, --config-file TEXT  Configure file path for authorization.
  -D, --debug             Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT      CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help          Show this message and exit.
```

参数名	参数类型	是否必选	参数说明
-i / --job-id	String	是	查询指定训练任务ID的事件。

示例

查看指定ID的训练作业的事件详情等。

```
ma-cli ma-job get-event --job-id b63e90baxxx
```



10.6.6 查询 ModelArts 训练 AI 引擎

执行**ma-cli ma-job get-engine**命令查询ModelArts训练使用的AI引擎。

```
$ ma-cli ma-job get-engine -h
Usage: ma-cli ma-job get-engine [OPTIONS]

  Get job engine info.

  Example:

  # Get training job engines
  ma-cli ma-job get-engine

Options:
  -v, --verbose           Show detailed information about training engines.
  -C, --config-file TEXT  Configure file path for authorization.
  -D, --debug             Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT      CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help          Show this message and exit.
```


版权所有 © 华为云计算技术有限公司

表 10-16 参数说明

参数名	参数类型	是否必选	参数说明
-v / --verbose	Bool	否	显示详细的信息开关，默认关闭。

示例

查看训练作业的AI引擎。

```
ma-cli ma-job get-engine
```

```
(PyTorch-1.4) [ma-user work]$ma-cli ma-job get-engine
+-------+-----------------------------------------------------+-----------------------+----------+
| index |                      engine id                      |      engine name      | run user |
+-------+-----------------------------------------------------+-----------------------+----------+
|   1   |                caffe-1.0.0-python2.7                |         Caffe         |          |
+-------+-----------------------------------------------------+-----------------------+----------+
|   2   |                horovod-cp36-tf-1.16.2               |        Horovod        |          |
+-------+-----------------------------------------------------+-----------------------+----------+
|   3   | horovod_0.20.0-pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64 |        Horovod        |   1102   |
+-------+-----------------------------------------------------+-----------------------+----------+
|   4   | horovod_0.20.0-tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64 |        Horovod        |   1102   |
+-------+-----------------------------------------------------+-----------------------+----------+
|   5   |              kungfu-0.2.2-tf-1.13.1-python3.6       |         KungFu        |          |
+-------+-----------------------------------------------------+-----------------------+----------+
|   6   |       mindspore_1.3.0-cuda_10.1-py_3.7-ubuntu_1804-x86_64 |          MPI          |   1102   |
+-------+-----------------------------------------------------+-----------------------+----------+
|   7   |      mindspore_1.7.0-cann_5.1.0-py_3.7-euler_2.8.3-aarch64 | Ascend-Powered-Engine |   1000   |
+-------+-----------------------------------------------------+-----------------------+----------+
|   8   |      mindspore_1.8.0-cann_5.1.2-py_3.7-euler_2.8.3-aarch64 | Ascend-Powered-Engine |   1000   |
+-------+-----------------------------------------------------+-----------------------+----------+
|   9   |      mindspore_1.9.0-cann_6.0.0-py_3.7-euler_2.8.3-aarch64 | Ascend-Powered-Engine |   1000   |
+-------+-----------------------------------------------------+-----------------------+----------+
|   10  |                mxnet-1.2.1-python3.6                |         MXNet         |          |
+-------+-----------------------------------------------------+-----------------------+----------+
|   11  |   optverse_0.2.0-pygrassland_1.1.0-py_3.7-ubuntu_18.04-x86_64 |           OR          |   1000   |
+-------+-----------------------------------------------------+-----------------------+----------+
|   12  |                pytorch-cp36-1.0.0                   |        PyTorch        |          |
+-------+-----------------------------------------------------+-----------------------+----------+
|   13  |                pytorch-cp36-1.3.0                   |        PyTorch        |          |
+-------+-----------------------------------------------------+-----------------------+----------+
|   14  |                pytorch-cp36-1.4.0                   |        PyTorch        |          |
+-------+-----------------------------------------------------+-----------------------+----------+
|   15  |      pytorch_1.8.0-cann_5.1.0-py_3.7-euler_2.8.3-aarch64 | Ascend-Powered-Engine |   1000   |
+-------+-----------------------------------------------------+-----------------------+----------+
|   16  |       pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64 |        PyTorch        |   1000   |
+-------+-----------------------------------------------------+-----------------------+----------+
|   17  |      pytorch_1.8.1-cann_5.1.2-py_3.7-euler_2.8.3-aarch64 | Ascend-Powered-Engine |   1000   |
+-------+-----------------------------------------------------+-----------------------+----------+
|   18  |      pytorch_1.8.1-cann_6.0.0-py_3.7-euler_2.8.3-aarch64 | Ascend-Powered-Engine |   1000   |
+-------+-----------------------------------------------------+-----------------------+----------+
|   19  |       pytorch_1.8.1-cuda_11.1-py_3.7-ubuntu_18.04-x86_64 |        PyTorch        |   1000   |
+-------+-----------------------------------------------------+-----------------------+----------+
|   20  |       pytorch_1.8.2-cuda_10.2-py_3.7-ubuntu_18.04-x86_64 |        PyTorch        |   1000   |
+-------+-----------------------------------------------------+-----------------------+----------+
|   21  |       pytorch_1.9.1-cuda_11.1-py_3.7-ubuntu_18.04-x86_64 |        PyTorch        |   1000   |
+-------+-----------------------------------------------------+-----------------------+----------+
|   22  |                ray-cp36-0.7.4                       |          Ray          |          |
+-------+-----------------------------------------------------+-----------------------+----------+
```

10.6.7 查询 ModelArts 训练资源规格

执行**ma-cli ma-job get-flavor**命令查询ModelArts训练的资源规格。

```
$ ma-cli ma-job get-flavor -h
Usage: ma-cli ma-job get-flavor [OPTIONS]

  Get job flavor info.

  Example:

  # Get training job flavors
  ma-cli ma-job get-flavor
```


版权所有 © 华为云计算技术有限公司

```
Options:
  -t, --flavor-type [CPU|GPU|Ascend]
                                  Type of training job flavor.
  -v, --verbose                   Show detailed information about training flavors.
  -C, --config-file TEXT          Configure file path for authorization.
  -D, --debug                     Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT              CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help                  Show this message and exit.
```

表 10-17 参数说明

参数名	参数类型	是否必选	参数说明
-t / --flavor-type	String	否	资源规格类型，如果不指定默认返回所有的资源规格。
-v / --verbose	Bool	否	显示详细的信息开关，默认关闭。

示例

查看训练作业的资源规格及类型。

```
ma-cli ma-job get-flavor
```

```
(PyTorch-1.4) [ma-user work]$ma-cli ma-job get-flavor
+-------+-----------------------------+-----------------------------------+-------------+
| index |          flavor id          |           flavor name             | flavor type |
+-------+-----------------------------+-----------------------------------+-------------+
|   1   |    modelarts.kat1.8xlarge   | Computing NPU(8*Ascend) instance  |    Ascend   |
+-------+-----------------------------+-----------------------------------+-------------+
|   2   |    modelarts.kat1.xlarge    |  Computing NPU(Ascend) instance   |    Ascend   |
+-------+-----------------------------+-----------------------------------+-------------+
|   3   |     modelarts.vm.cpu.2u     |    Computing CPU(2U) instance     |     CPU     |
+-------+-----------------------------+-----------------------------------+-------------+
|   4   |     modelarts.vm.cpu.8u     |    Computing CPU(8U) instance     |     CPU     |
+-------+-----------------------------+-----------------------------------+-------------+
|   5   | modelarts.vm.cpu.8u16g.119  |    Computing CPU(8U) instance     |     CPU     |
+-------+-----------------------------+-----------------------------------+-------------+
|   6   |   modelarts.vm.v100.large   |   Computing GPU(V100) instance    |     GPU     |
+-------+-----------------------------+-----------------------------------+-------------+
|   7   | modelarts.vm.v100.large.free |   Computing GPU(V100) instance    |     GPU     |
+-------+-----------------------------+-----------------------------------+-------------+
```

10.6.8 停止 ModelArts 训练作业

执行**ma-cli ma-job stop**命令，可停止指定作业id的训练作业。

```
$ ma-cli ma-job stop -h
Usage: ma-cli ma-job stop [OPTIONS]
```


版权所有 © 华为云计算技术有限公司

```
  Stop training job by job id.

  Example:

  Stop training job by job id
  ma-cli ma-job stop --job-id ${job_id}

Options:
  -i, --job-id TEXT       Get training job event by job id.  [required]
  -y, --yes               Confirm stop operation.
  -C, --config-file TEXT  Configure file path for authorization.
  -D, --debug             Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT      CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help          Show this message and exit.
```

表 10-18 参数说明

参数名	参数类型	是否必选	参数说明
-i / --job-id	String	是	ModelArts训练作业ID。
-y / --yes	Bool	否	强制关闭指定训练作业。

示例

停止运行中的训练作业。

```
ma-cli ma-job stop --job-id efd3e2f8xxx
```

10.7 使用 ma-cli dli-job 命令提交 DLI Spark 作业

10.7.1 命令总览

```
$ma-cli dli-job -h
Usage: ma-cli dli-job [OPTIONS] COMMAND [ARGS]...

  DLI spark job submission and query jod details.

Options:
  -h, -H, --help  Show this message and exit.

Commands:
  get-job       Get DLI spark job details.
  get-log       Get DLI spark log details.
  get-queue     Get DLI spark queues info.
  get-resource  Get DLI resources info.
  stop          Stop DLI spark job by job id.
  submit        Submit dli spark batch job.
  upload        Upload local file or OBS object to DLI resources.
```


版权所有 © 华为云计算技术有限公司

表 10-19 提交 DLI Spark 作业命令总览

命令	命令详情
get-job	查询DLI Spark作业列表及详情。
get-log	查询DLI Spark运行日志。
get-queue	查询DLI 队列。
get-resource	查询DLI 分组资源。
stop	停止DLI Spark作业。
submit	提交DLI Spark作业。
upload	上传本地文件或OBS文件到DLI分组资源。

10.7.2 查询 DLI Spark 作业

执行**ma-cli dli-job get-job**查询DLI Spark作业列表或单个作业详情。

```
ma-cli dli-job get-job -h
Usage: ma-cli dli-job get-job [OPTIONS]

  Get DLI Spark details.

  Example:

  # Get DLI Spark job details by job name
  ma-cli dli-job get-job -n ${job_name}

  # Get DLI Spark job details by job id
  ma-cli dli-job get-job -i ${job_id}

  # Get DLI Spark job list
  ma-cli dli-job get-job --page-size 5 --page-num 1

Options:
  -i, --job-id TEXT              Get DLI Spark job details by job id.
  -n, --job-name TEXT            Get DLI Spark job details by job name.
  -pn, --page-num INTEGER RANGE  Specify which page to query.  [x>=1]
  -ps, --page-size INTEGER RANGE  The maximum number of results for this query.  [x>=1]
  -v, --verbose                  Show detailed information about DLI Spark job details.
  -C, --config-file PATH         Configure file path for authorization.
  -D, --debug                    Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT             CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help                 Show this message and exit.
```

表 10-20 参数说明

参数名	参数类型	是否必选	参数说明
-i / --job-id	String	否	查询指定DLI Spark作业ID的任务详情。
-n / --job-name	String	否	查询指定作业名称的DLI Spark作业或根据作业名称关键字过滤DLI Spark作业。
-pn / --page-num	Int	否	作业索引页，默认是第1页。

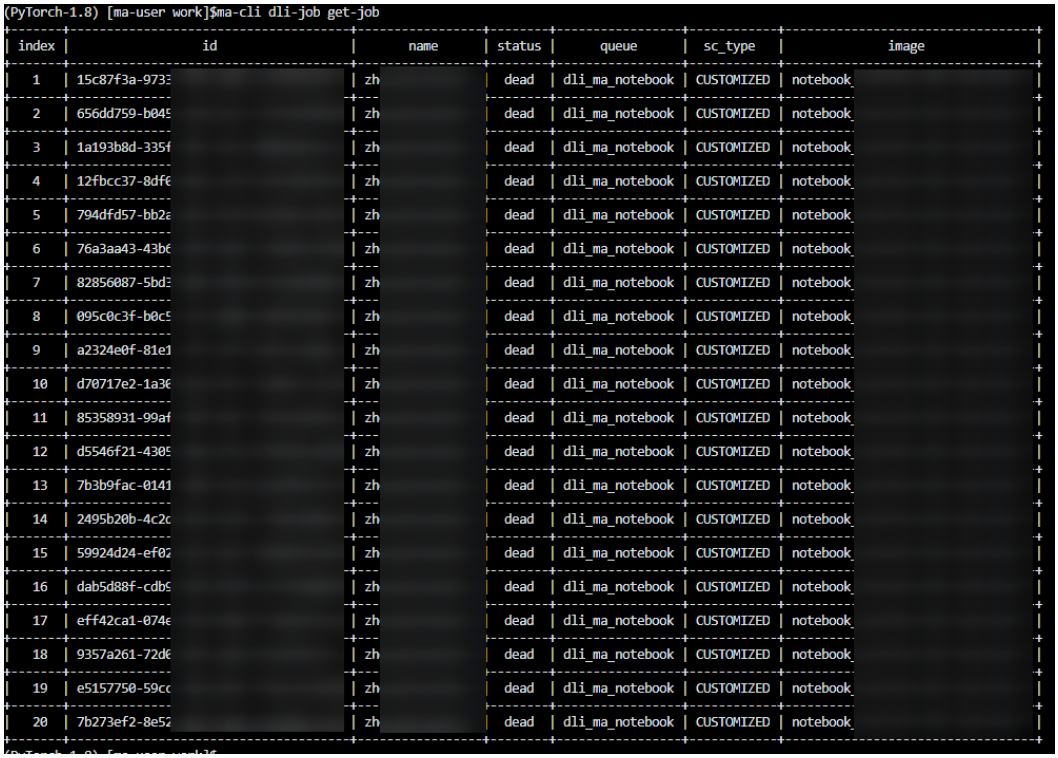
版权所有 © 华为云计算技术有限公司

参数名	参数类型	是否必选	参数说明
-ps / --page-size	Int	否	每页显示的作业数量，默认是20。
-v / --verbose	Bool	否	显示详细的信息开关，默认关闭。

示例

查询DLI Spark所有作业。

```
ma-cli dli-job get-job
```



10.7.3 提交 DLI Spark 作业

执行**ma-cli dli-job submit**命令提交DLI Spark作业。

ma-cli dli-job submit命令需要指定一个位置参数YAML_FILE表示作业的配置文件路径，如果不指定该参数，则表示配置文件为空。配置文件是一个YAML格式的文件，里面的参数就是命令的option参数。此外，如果用户在命令行中同时指定YAML_FILE配置文件和option参数，命令行中指定的option参数的值将会覆盖配置文件相同的值。

命令参数预览

```
ma-cli dli-job submit -h
Usage: ma-cli dli-job submit [OPTIONS] [YAML_FILE]...

  Submit DLI Spark job.

  Example:
```


版权所有 © 华为云计算技术有限公司

```
ma-cli dli-job submit  --name test-spark-from-sdk
                       --file test/sub_dli_task.py
                       --obs-bucket dli-bucket
                       --queue dli_test
                       --spark-version 2.4.5
                       --driver-cores 1
                       --driver-memory 1G
                       --executor-cores 1
                       --executor-memory 1G
                       --num-executors 1

Options:
  --file TEXT                  Python file or app jar.
  -cn, --class-name TEXT       Your application's main class (for Java / Scala apps).
  --name TEXT                  Job name.
  --image TEXT                 Full swr custom image path.
  --queue TEXT                 Execute queue name.
  -obs, --obs-bucket TEXT      DLI obs bucket to save logs.
  -sv, --spark-version TEXT    Spark version.
  -st, --sc-type [A|B|C]       Compute resource type.
  --feature [basic|custom|ai]  Type of the Spark image used by a job (default: basic).
  -ec, --executor-cores INTEGER  Executor cores.
  -em, --executor-memory TEXT  Executor memory (eg. 2G/2048MB).
  -ne, --num-executors INTEGER  Executor number.
  -dc, --driver-cores INTEGER  Driver cores.
  -dm, --driver-memory TEXT    Driver memory (eg. 2G/2048MB).
  --conf TEXT                  Arbitrary Spark configuration property (eg. <PROP=VALUE>).
  --resources TEXT             Resources package path.
  --files TEXT                 Files to be placed in the working directory of each executor.
  --jars TEXT                  Jars to include on the driver and executor class paths.
  -pf, --py-files TEXT         Python files to place on the PYTHONPATH for Python apps.
  --groups TEXT                User group resources.
  --args TEXT                  Spark batch job parameter args.
  -q, --quiet                  Exit without waiting after submit successfully.
  -C, --config-file PATH       Configure file path for authorization.
  -D, --debug                  Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT           CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help               Show this message and exit.
```

yaml文件预览

```
# dli-demo.yaml
name: test-spark-from-sdk
file: test/sub_dli_task.py
obs-bucket: ${your_bucket}
queue: dli_notebook
spark-version: 2.4.5
driver-cores: 1
driver-memory: 1G
executor-cores: 1
executor-memory: 1G
num-executors: 1

## [Optional]
jars:
  - ./test.jar
  - obs://your-bucket/jars/test.jar
  - your_group/test.jar

## [Optional]
files:
  - ./test.csv
  - obs://your-bucket/files/test.csv
  - your_group/test.csv

## [Optional]
python-files:
  - ./test.py
  - obs://your-bucket/files/test.py
```


版权所有 © 华为云计算技术有限公司

```
  - your_group/test.py

## [Optional]
resources:
  - name: your_group/test.py
    type: pyFile
  - name: your_group/test.csv
    type: file
  - name: your_group/test.jar
    type: jar
  - name: ./test.py
    type: pyFile
  - name: obs://your-bucket/files/test.py
    type: pyFile

## [Optional]
groups:
  - group1
  - group2
```

指定options参数提交DLI Spark作业示例：

```
$ ma-cli dli-job submit --name test-spark-from-sdk \
                        --file test/sub_dli_task.py \
                        --obs-bucket ${your_bucket} \
                        --queue dli_test \
                        --spark-version 2.4.5 \
                        --driver-cores 1 \
                        --driver-memory 1G \
                        --executor-cores 1 \
                        --executor-memory 1G \
                        --num-executors 1
```

表 10-21 参数说明

参数名	参数类型	是否必选	参数说明
YAML_FILE	String，本地文件路径	否	DLI Spark作业的配置文件，如果不传则表示配置文件为空。
--file	String	是	程序运行入口文件，支持本地文件路径、OBS路径或者用户已上传到DLI资源管理系统的类型为jar或pyFile的程序包名。
-cn / --class_name	String	是	批处理作业的Java/Spark主类。
--name	String	否	创建时用户指定的作业名称，不能超过128个字符。
--image	String	否	自定义镜像路径,格式为：组织名/镜像名:镜像版本。当用户设置“**feature**”为“custom”时，该参数生效。用户可通过与“**feature**”参数配合使用，指定作业运行使用自定义的Spark镜像。

版权所有 © 华为云计算技术有限公司

参数名	参数类型	是否必选	参数说明
-obs / --obs-bucket	String	否	保存Spark作业的obs桶，需要保存作业时配置该参数。同时也可作为提交本地文件到resource的中转站。
-sv/ --spark-version	String	否	作业使用Spark组件的版本号。
-st / `--sc-type	String	否	如果当前Spark组件版本为2.3.2，则不填写该参数。 如果当前Spark组件版本为2.3.3，则在“**feature**”为“basic”或“ai”时填写。如果不填写，则使用默认的Spark组件版本号2.3.2。
--feature	String	否	作业特性。表示用户作业使用的Spark镜像类型，默认值为basic。 ● basic：表示使用DLI提供的基础Spark镜像。 ● custom：表示使用用户自定义的Spark镜像。 ● ai：表示使用DLI提供的AI镜像。
--queue	String	否	用于指定队列，填写已创建DLI的队列名。必须为通用类型的队列。队列名称的获取请参考**表10-23**。
-ec / --executor-cores	String	否	Spark应用每个Executor的CPU核数。该配置项会替换sc_type中对应的默认参数。
-em / --executor-memory	String	否	Spark应用的Executor内存，参数配置例如2G, 2048M。该配置项会替换“**sc_type**”中对应的默认参数，使用时必需带单位，否则会启动失败。
-ne / --num-executors	String	否	Spark应用Executor的个数。该配置项会替换sc_type中对应的默认参数。
-dc / --driver-cores	String	否	Spark应用Driver的CPU核数。该配置项会替换sc_type中对应的默认参数。
-dm / --driver-memory	String	否	Spark应用的Driver内存，参数配置例如2G, 2048M。该配置项会替换“**sc_type**”中对应的默认参数，使用时必需带单位，否则会启动失败。
--conf	Array of String	否	batch配置项，参考**Spark Configuration**。如果需要指定多个参数，可以使用--conf conf1 --conf conf2。
--resources	Array of String	否	资源包名称。支持本地文件，OBS路径及用户已上传到DLI资源管理系统的文件。如果需要指定多个参数，可以使用--resources resource1 --resources resource2。

版权所有 © 华为云计算技术有限公司

参数名	参数类型	是否必选	参数说明
--files	Array of String	否	用户已上传到DLI资源管理系统的类型为file的资源包名。也支持指定OBS路径，例如：obs://桶名/包名。同时也支持本地文件。如果需要指定多个参数，可以使用--files file1 --files file2。
--jars	Array of String	否	用户已上传到DLI资源管理系统的类型为jar的程序包名。也支持指定OBS路径，例如：obs://桶名/包名。也支持本地文件。如果需要指定多个参数，可以使用--jars jar1 --jars jar2。
-pf /--python-files	Array of String	否	用户已上传到DLI资源管理系统的类型为pyFile的资源包名。也支持指定OBS路径，例如：obs://桶名/包名。也支持本地文件。如果需要指定多个参数，可以使用--python-files py1 --python-files py2。
--groups	Array of String	否	资源分组名称，如果需要指定多个参数，可以使用--groups group1 --groups group2。
--args	Array of String	否	传入主类的参数，即应用程序参数。如果需要指定多个参数，可以使用--args arg1 --args arg2。
-q / --quiet	Bool	否	提交DLI Spark作业成功后直接退出，不再同步打印任务状态。

示例

- 通过YAML_FILE文件提交DLI Spark作业。

```
$ma-cli dli-job submit dli_job.yaml
```

```
(PyTorch-1.4) [ma-user work]$ma-cli dli-job submit ./dli-job.yaml
[ OK ] Current DLI job id is: 01b698b8-9fd6-4a8e-bc3c-6821c6405b14
[ OK ] starting
[ OK ] running
[ OK ] success
[ OK ] Successfully submit DLI spark job [ 01b698b8-9fd6-4a8e-bc3c-6821c6405b14 ].
```

- 指定命令行options参数提交DLI Spark作业。

```
$ma-cli dli-job submit --name test-spark-from-sdk \
>                      --file test/jumpstart-trainingjob-gallery-pytorch-sample.ipynb \
>                      --queue dli_ma_notebook \
>                      --spark-version 2.4.5 \
>                      --driver-cores 1 \
>                      --driver-memory 1G \
>                      --executor-cores 1 \
>                      --executor-memory 1G \
>                      --num-executors 1
```


版权所有 © 华为云计算技术有限公司

```
(PyTorch-1.4) [ma-user work]$ma-cli dli-job submit --name test-spark-from-sdk \
>                          --file test/jumpstart-trainingjob-gallery-pytorch-sample.ipynb \
>                          --queue dli_ma_notebook \
>                          --spark-version 2.4.5 \
>                          --driver-cores 1 \
>                          --driver-memory 1G \
>                          --executor-cores 1 \
>                          --executor-memory 1G \
>                          --num-executors 1
[ OK ] Current DLI job id is: ae856c20-e9ae-49ca-8409-7a02652297b8
[ OK ] starting
```

10.7.4 查询 DLI Spark 运行日志

执行**ma-cli dli-job get-log**命令查询DLI Spark作业后台的日志。

```
$ ma-cli dli-job get-log -h
Usage: ma-cli dli-job get-log [OPTIONS]

  Get DLI spark job log details.

  Example:

  # Get job log by job id
  ma-cli dli-job get-log --job-id ${job_id}

Options:
  -i, --job-id TEXT       Get DLI spark job details by job id.  [required]
  -C, --config-file TEXT  Configure file path for authorization.
  -D, --debug             Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT      CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help          Show this message and exit.
```

表 10-22 参数说明

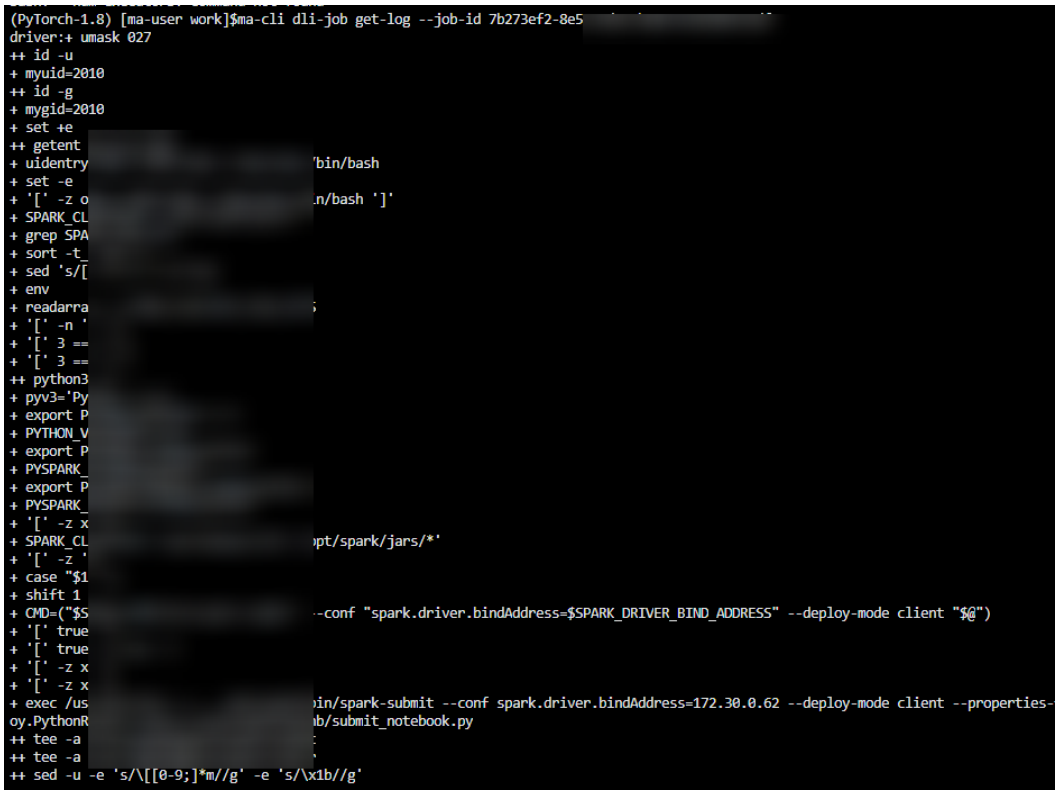
参数名	参数类型	是否必选	参数说明
-i / --job-id	String	是	查询指定DLI Spark作业ID的任务日志。

示例

查询指定作业ID的DLI Spark作业运行日志。

```
ma-cli dli-job get-log --job-id ${your_job_id}
```


版权所有 © 华为云计算技术有限公司



10.7.5 查询 DLI 队列

执行**ma-cli dli-job get-queue**命令查询DLI对列。

```
ma-cli dli-job get-queue -h
Usage: ma-cli dli-job get-queue [OPTIONS]

  Get DLI queues info.

  Example:

  # Get DLI queue details by queue name
  ma-cli dli-job get-queue --queue-name $queue_name}

Options:
  -pn, --page-num INTEGER RANGE   Specify which page to query.  [x>=1]
  -ps, --page-size INTEGER RANGE  The maximum number of results for this query.  [x>=1]
  -n, --queue-name TEXT           Get DLI queue details by queue name.
  -t, --queue-type [sql|general|all]
                                  DLI queue type (default "all").
  -tags, --tags TEXT              Get DLI queues by tags.
  -C, --config-file PATH          Configure file path for authorization.
  -D, --debug                     Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT              CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help                  Show this message and exit.
```

表 10-23 参数说明

参数名	参数类型	是否必选	参数说明
-n / --queue-name	String	否	指定需要查询的DLI队列名称。

版权所有 © 华为云计算技术有限公司

参数名	参数类型	是否必选	参数说明
-t / --queue-type	String	否	指定查询的DLI队列类型，支持sql、general和all，默认是all。
-tags / --tags	String	否	指定查询的DLI队列tags。
-pn / --page-num	Int	否	DLI队列页索引，默认是第1页。
-ps / --page-size	Int	否	每页显示的DLI队列数量，默认是20。

示例

查询队列名为“dli_ma_notebook”的队列信息。

```
ma-cli dli-job get-queue --queue-name dli_ma_notebook
```

```
(PyTorch-1.8) [ma-user work]$ ma-cli dli-job get-queue --queue-name dli_ma_notebook
{'chargingMode': 1,
 'create_time': 1668585417422,
 'cuCount': 16,
 'cu_spec': 16,
 'description': '',
 'enterprise_project_id': '0',
 'is_success': True,
 'message': '',
 'owner': '                          ',
 'queueName': 'dli_ma_notebook',
 'queueType': 'general',
 'queue_id': 8[illegible],
 'resource_id': '242e7af8-c9d1                          e',
 'resource_mode': 1,
 'resource_type': 'container',
 'support_spark_versions': ['2.3.2', '2.4.5', '3.1.1']}
```

10.7.6 查询 DLI 分组资源

执行**ma-cli dli-job get-resource**命令获取DLI资源详细信息，如资源名称，资源类型等。

```
$ ma-cli dli-job get-resource -h
Usage: ma-cli dli-job get-resource [OPTIONS]

  Get DLI resource info.

  Example:

  # Get DLI resource details by resource name
  ma-cli dli-job get-resource --resource-name ${resource_name}

Options:
  -n, --resource-name TEXT        Get DLI resource details by resource name.
  -k, --kind [jar|pyFile|file|modelFile]
                                  DLI resources type.
  -g, --group TEXT                Get DLI resources by group.
  -tags, --tags TEXT              Get DLI resources by tags.
  -C, --config-file TEXT          Configure file path for authorization.
  -D, --debug                     Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT              CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help                  Show this message and exit.
```


版权所有 © 华为云计算技术有限公司

表 10-24 参数说明

参数名	参数类型	是否必选	参数说明
-n / --resource-name	String	否	按DLI分组资源名称查询DLI资源详细信息。
-k / --kind	String	否	按DLI分组资源类型查询DLI资源详细信息，支持jar、pyFile、file和modelFile。
-g / --group	String	否	按DLI分组资源组名查询DLI资源组详细信息。
-tags / --tags	String	否	通过DLI分组资源tags获取DLI资源详细信息。

示例

查询所有DLI分组资源信息。

```
ma-cli dli-job get-resource
```


版权所有 © 华为云计算技术有限公司

10.7.7 上传本地文件或 OBS 文件到 DLI 分组资源

ma-cli dli-job upload命令支持将本地文件或OBS文件上传到DLI资源组。

```
$ ma-cli dli-job upload -h
Usage: ma-cli dli-job upload [OPTIONS] PATHS...

  Upload DLI resource.

  Tips: --obs-path is need when upload local file.

  Example:

  # Upload an OBS path to DLI resource
  ma-cli dli-job upload obs://your-bucket/test.py -g test-group --kind pyFile

  # Upload a local path to DLI resource
  ma-cli dli-job upload ./test.py -g test-group -obs ${your-bucket} --kind pyFile

  # Upload local path and OBS path to DLI resource
  ma-cli dli-job upload ./test.py obs://your-bucket/test.py -g test-group -obs ${your-bucket}
```


版权所有 © 华为云计算技术有限公司

```
Options:
  -k, --kind [jar|pyFile|file]  DLI resources type.
  -g, --group TEXT              DLI resources group.
  -tags, --tags TEXT            DLI resources tags, follow --tags `key1`=`value1`.
  -obs, --obs-bucket TEXT       OBS bucket for upload local file.
  -async, --is-async            whether to upload resource packages in asynchronous mode. The default value is
False.
  -C, --config-file TEXT        Configure file path for authorization.
  -D, --debug                   Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT            CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help                Show this message and exit.
```

表 10-25 参数说明

参数名	参数类型	是否必选	参数说明
PATHS	String	是	需要上传到DLI分组资源的本地文件路径或者obs路径，支持同时传入多个路径。
-k / --kind	String	否	上传文件的类型，支持jar、pyFile和file。
-g / --group	String	否	上传文件的DLI分组名。
-tags / --tags	String	否	上传文件的tag。
-obs / --obs-bucket	String	否	如果上传文件包含本地路径，则需要指定一个OBS桶作为中转。
-async / --is-async	Bool	否	异步上传文件，推荐使用。

示例

- 上传本地文件到DLI分组资源
  ma-cli dli-job upload ./test.py -obs ${your-bucket} --kind pyFile

  ```
  (PyTorch-1.8) [ma-user work]$ma-cli dli-job upload ./test.py -obs obs://[redacted] --kind pyFile
  [ OK ] Upload ['test.py']  successfully.
  ```

- 上传OBS文件到DLI分组资源
  ma-cli dli-job upload obs://your-bucket/test.py --kind pyFile

  ```
  (PyTorch-1.8) [ma-user work]$ma-cli dli-job upload obs://[redacted]/test.py --kind pyFile
  [ OK ] Upload ['test.py']  successfully.
  ```

10.7.8 停止 DLI Spark 作业

执行**ma-cli dli-job stop**命令停止DLI Spark作业。

```
$ ma-cli dli-job stop -h
Usage: ma-cli dli-job stop [OPTIONS]

  Stop DLI spark job by job id.

  Example:

  Stop training job by job id
  ma-cli dli-job stop --job-id ${job_id}

Options:
  -i, --job-id TEXT       Get DLI spark job event by job id.  [required]
```


版权所有 © 华为云计算技术有限公司

```
-y, --yes               Confirm stop operation.
-C, --config-file TEXT  Configure file path for authorization.
-D, --debug             Debug Mode. Shows full stack trace when error occurs.
-P, --profile TEXT      CLI connection profile to use. The default profile is "DEFAULT".
-H, -h, --help          Show this message and exit.
```

表 10-26 参数说明

参数名	参数类型	是否必选	参数说明
-i / --job-id	String	是	DLI Spark作业ID。
-y / --yes	Bool	否	强制关闭指定DLI Spark作业。

示例

```
ma-cli dli-job stop -i ${your_job_id}
```

```
(PyTorch-1.8) [ma-user work]$ma-cli dli-job stop -i 4b[redacted]861
[ WARN ] Spark job 4b[redacted]1 will be stopped, do you want to continue (y for confirm)? [y/N]: y
[ OK ] Successfully stop spark batch job [ 4b[redacted]1 ].
```

10.8 使用 ma-cli 复制 OBS 数据

使用**ma-cli obs-copy [SRC] [DST]**可以实现本地和OBS文件或文件夹的相互复制。

```
$ma-cli obs-copy -h
Usage: ma-cli obs-copy [OPTIONS     ] SRC DST

  Copy file or directory to between OBS and local path. Example:

  # Upload local file to OBS path
  ma-cli obs-copy ./test.zip obs://your-bucket/copy-data/

  # Upload local directory to OBS path
  ma-cli obs-copy ./test/ obs://your-bucket/copy-data/

  # Download OBS file to local path
  ma-cli obs-copy obs://your-bucket/copy-data/test.zip ./test.zip

  # Download OBS directory to local path
  ma-cli obs-copy obs://your-bucket/copy-data/ ./test/

Options:
  -d, --drop-last-dir     Whether to drop last directory when copy folder. if True, the last directory of the
source folder will not copy to the destination folder.  [default: False]
  -C, --config-file PATH  Configure file path for authorization.
  -D, --debug             Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT      CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help          Show this message and exit.
```

表 10-27 参数说明

参数名	参数类型	是否必选	参数说明
-d / --drop-last-dir	Bool	否	如果指定，在复制文件夹时不会将源文件夹最后一级目录复制至目的文件夹下，仅对文件夹复制有效。

版权所有 © 华为云计算技术有限公司

命令示例

上传文件到OBS中

```
$ ma-cli obs-copy ./test.csv obs://${your_bucket}/test-copy/
[ OK ] local src path: [ /home/ma-user/work/test.csv ]
[ OK ] obs dst path: [ obs://${your_bucket}/test-copy/ ]
```

上传文件夹到OBS中，对应上传到OBS的目录为obs://${your_bucket}/test-copy/data/

```
$ ma-cli obs-copy /home/ma-user/work/data/ obs://${your_bucket}/test-copy/
[ OK ] local src path: [ /home/ma-user/work/data/ ]
[ OK ] obs dst path: [ obs://${your_bucket}/test-copy/ ]
```

上传文件夹到OBS中，并指定--drop-last-dir，对应上传到OBS的目录为obs://${your_bucket}/test-copy/

```
$ ma-cli obs-copy /home/ma-user/work/data/ obs://${your_bucket}/test-copy/ --drop-last-dir
[ OK ] local src path: [ /home/ma-user/work/data ]
[ OK ] obs dst path: [ obs://${your_bucket}/test-copy/ ]
```

从OBS下载文件夹到本地磁盘中

```
$ ma-cli obs-copy obs://${your_bucket}/test-copy/ ~/work/test-data/
[ OK ] obs src path: [ obs://${your_bucket}/test-copy/ ]
[ OK ] local dst path: [ /home/ma-user/work/test-data/ ]
```


版权所有 © 华为云计算技术有限公司