

阿里云大数据平台

机器学习



机器学习

产品简介

关于PAI

- 阿里PAI机器学习平台是构建在阿里云ODPS计算平台之上，集数据处理、建模、离线预测、在线预测为一体的机器学习平台。该平台为算法开发者提供了丰富的MPI、PS、BSP等编程框架和数据存取接口，同时为算法使用者提供了基于Web的IDE+可视化实验搭建控制台。平台目前整合了集团内最先进的算法，为集团内、外不同用户提供算法服务。

产品特点

- 一站式的算法与智能应用的开发、发布与分享的平台
- 支持处理亿万级大规模数据
- 无需编码，据简单的拖拽即可完成数据挖掘，数据分析等功能
- 孕育于阿里“数据-云-计算”生态系统，高效配置计算资源为数据赋能

名词解释

- **ODPS (Open Data Processing Service)** : 开放数据处理服务由阿里云自主研发，提供针对TB/PB级数据、实时性要求不高的分布式处理能力，应用于数据分析、挖掘、商业智能等领域。
- **项目(Project)**: 项目（也称项目空间）是ODPS最基本的组织对象。其他对象，例如表(Table)和实例(Instance)等都归属于某个项目。
- **实验 (Experiment)** : 实验是指PAI平台用户搭建的数据工作流程或者数据应用。用户需要先建立一个实验实例，然后在实验画布上搭建数据流程。
- **ODPS源表与ODPS目标表(Table)**: 表(Table)是ODPS中数据存储对象。与常见的关系型数据类似，ODPS中的表逻辑上也是二维结构。源表指一个算法节点的输入，目标表指算法节点的输出。
- **组件 (Nodes)** : 组件是用户可以在PAI平台上调用执行的最小操作单元，例如数据导入导出、数据处理、数据分析、模型训练或者预测。
- **模型 (Model)** : 模型是特指一个算法或者机器学习训练组件产生的结果数据。模型是一类特殊的组件。
- **分区(partition)** : ODPS表分区

应用场景

PAI的应用场景

PAI主要应用在数据挖掘场景下，即指从大量的数据中通过算法搜索隐藏于其中信息的过程。通过统计、在线分析处理、情报检索、机器学习、专家系统（依靠过去的经验法则）和模式识别等诸多方法来实现隐藏信息的探索和发现。其场景主要有：

分类：分类可以找出这些不同种类客户之间的特征，让用户了解不同行为类别客户的分布特征，从而进行商业决策和业务活动，如：在银行行业，可以通过PAI对客户进行分类，以便进行风险评估和防控；在销售领域，通过对客户的细分，进行潜客挖掘、客户提升和交叉销售、客户挽留等

聚类：通常“人以群分，物以类聚”，通过对数据对象划分为若干类，同一类的对象具有较高的相似度，不同类的对象相似度较低，以便我们度量对象间的相似性，发现相关性。如在安全领域，通过异常点的检测，可以发现异常的安全行为。通过人与人之间的相似性，实现团伙犯罪的发掘

预测：通过对历史事件的学习来积累经验，得出事物间的相似性和关联性，从而对事物的未来状况做出预测。比如：预测销售收入和利润，预测用户下一个阶段的消费行为等

关联：分析各个物品或者商品之间同时出现的机率，典型的场景如：购物篮分析。比如超市购物时，顾客购买记录常常隐含着很多关联规则，比如购买圆珠笔的顾客中有65%也购买了笔记本，利用这些规则，商场人员可以很好的规划商品摆放问题。在电商网站中，利用关联规则可以发现哪些用户更喜欢哪类的商品，当发现有类似的客户的时候，可以将其它客户购买的商品推荐给相类似的客户，以提高网站的收入。

快速入门

一个完整的建模步骤包括以下6个步骤：

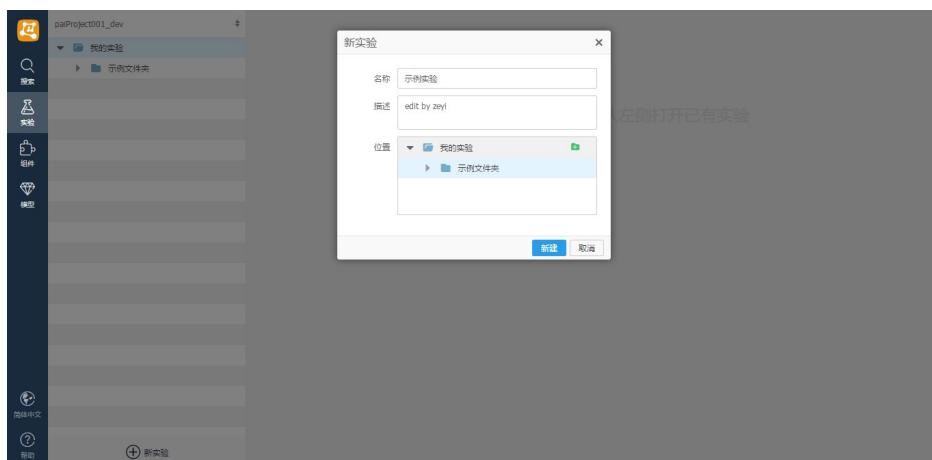
- step1：数据准备
- step2：数据预处理
- step3：数据可视化
- step4：算法建模
- step5：多模型评估
- step6：数据存储

下面在PAI上给大家做一个入门级的示例：

注意：以下步骤默认用户已拥有自己的project并将表数据传入对应的project中

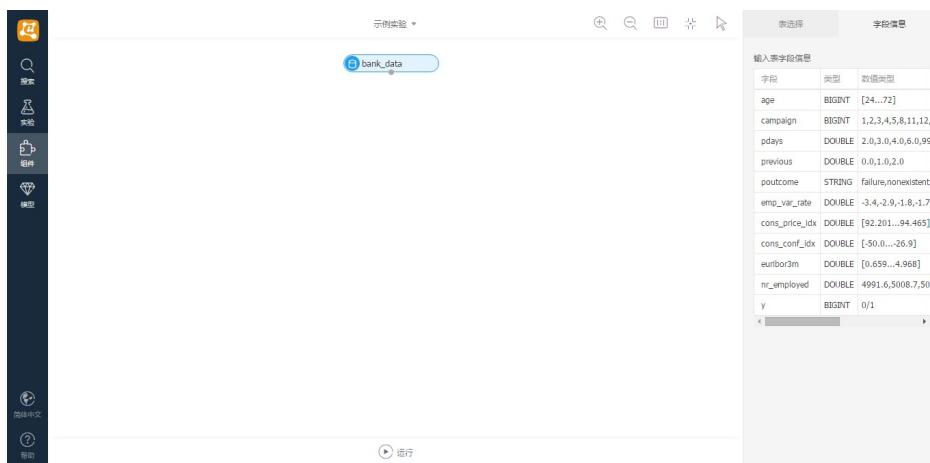
1、数据准备

1.1 右击新建实验，输入实验名和实验描述。



1.2 切换到组件栏，向画布中拖入odps源，点击odps源，在右侧表选择栏填入你的odps表。

1.3 切换到字段信息栏，可以查看输入表的字段名、数据类型和前100行数据的数值分布。



The screenshot shows the 'bank_data' dataset table schema. The columns and their types are:

字段	类型	数据类型
age	BIGINT	[24..72]
campaign	BIGINT	[1,2,3,4,5,8,11,12]
pdays	DOUBLE	2.0,3.0,4.0,6.0,9.9
previous	DOUBLE	0.0,1.0,2.0
poutcome	STRING	failure,noneexistent,
emp_var_rate	DOUBLE	-3.4,-2.9,-1.8,-1.7
cons_price_idx	DOUBLE	[92.201..94.465]
cons_conf_idx	DOUBLE	[-50.0...,-26.9]
euribor3m	DOUBLE	[0.659..4.968]
nr_employed	DOUBLE	4991.6,5008.7,50
y	BIGINT	0/1

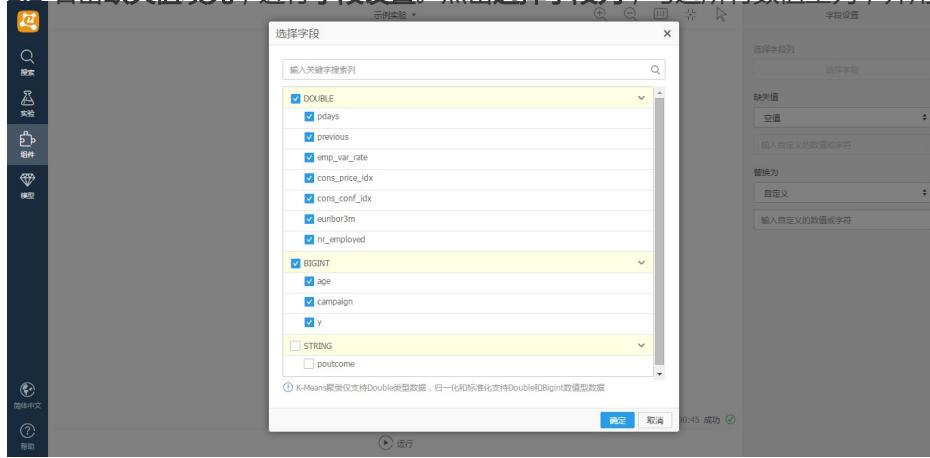
2、数据预处理

2.1 从处理栏拖入缺失值填充组件到画布

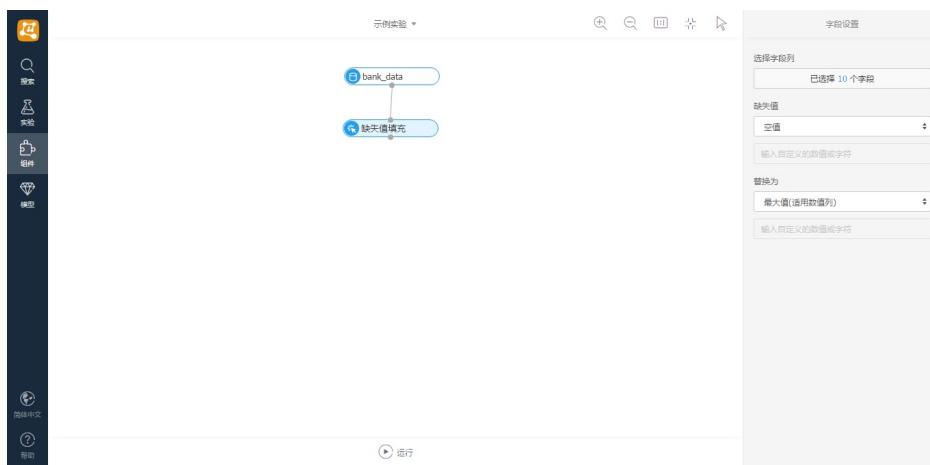


The screenshot shows the 'processing' component library with the 'Missing Value Imputation' component selected.

2.2 右击缺失值填充，进行字段设置。点击选择字段列，勾选所有数值型列，并用最大值进行填充

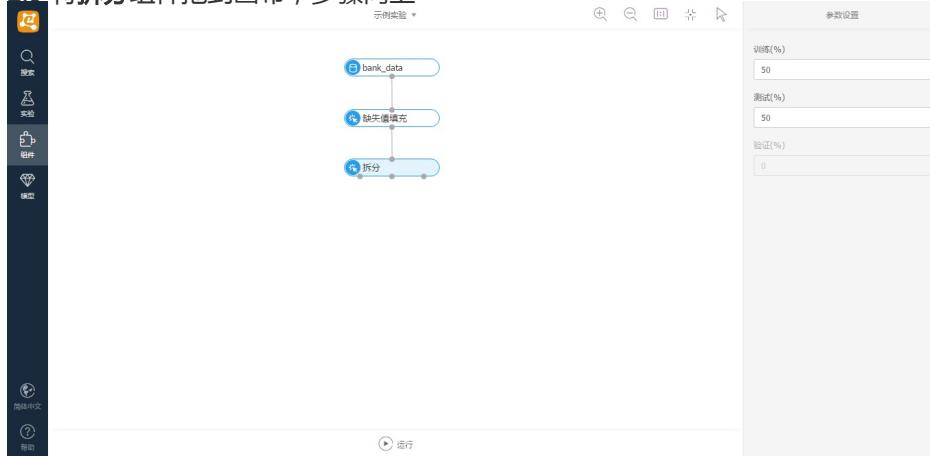


The screenshot shows the 'Missing Value Imputation' configuration dialog. Under '选择字段列' (Select Field Column), all numerical columns ('pdays', 'previous', 'emp_var_rate', 'cons_price_idx', 'cons_conf_idx', 'euribor3m', 'nr_employed') are selected. Under '替换为' (Replace With), '空值' (Null) is selected, and under '输入自定义的数据或字符串' (Input custom data or string), '最大' (Max) is selected.



- 将数字进行缺失值填充是模型训练前数据处理很重要的步骤之一

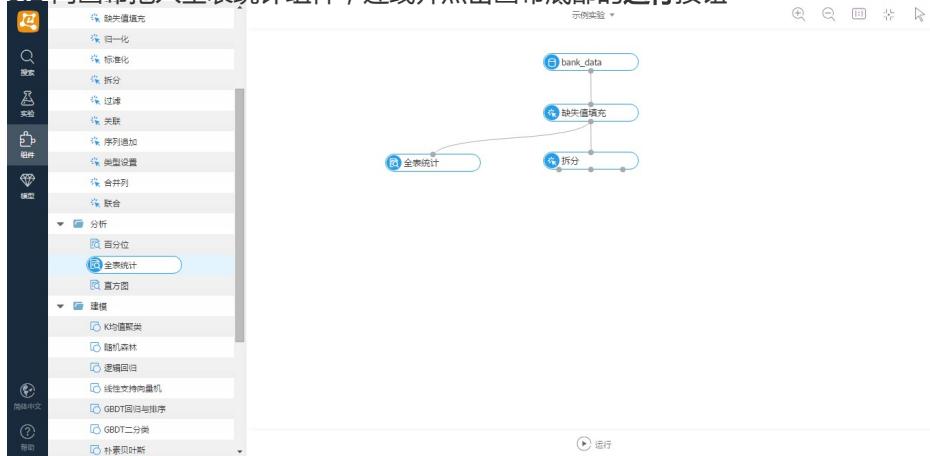
2.3 将拆分组件拖到画布，步骤同上



- 此步骤的目的是将数据拆分成两份，50%作为模型训练集，50%作为模型预测集

3、数据可视化

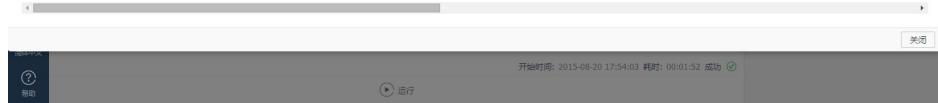
3.1 向画布拖入全表统计组件，连线并点击画布底部的运行按钮



3.2 待实验运行结束后，右击全表统计，点击查看分析报告，可看到数据的全表统计信息，如下：

全表简单分析

字段	Total	Missing	NaN	Positive Infinity	Negative Infinity	Min	Max	Mean	Variance	SD
euribor3m	41188	0	0	0	0	0.634	5.045	3.621290812858...	3.00830780019434...	1.734447404850...
previous	41188	0	0	0	0	0	7	0.172962998931...	0.244927078826...	0.494901079839...
cons_conf_idx	41188	0	0	0	0	-50.8	-26.9	-40.50260027191...	21.42021539621...	4.628197856208...
poutcome	41188	0	0	0	0	0	0	0.02406040594...	108.6024511651...	10.42124998093...
age	41188	0	0	0	0	17	98	40.02406040594...	186.9109073447...	10.42124998093...
pdays	41188	0	0	0	0	0	999	962.4754540157...	34935.68728443...	186.9109073447...
emp_var_rate	41188	0	0	0	0	-3.4	1.4	0.081885500631...	2.467914506326...	1.57995740517...
nr_employed	41188	0	0	0	0	4963.6	5228.1	5167.035910942...	5220.283274428...	72.25152783456...
cons_price_idx	41188	0	0	0	0	92.2009999999...	94.767	93.575646436827...	0.335055798570...	0.578840404575...
campaign	41188	0	0	0	0	1	56	2.567592502670...	7.672975027862...	2.770013542902...
Y	41188	0	0	0	0	0	1	0.112654171117...	0.099965635904...	0.316173426942...



4、算法建模

4.1、向画布拖入两个逻辑回归组件

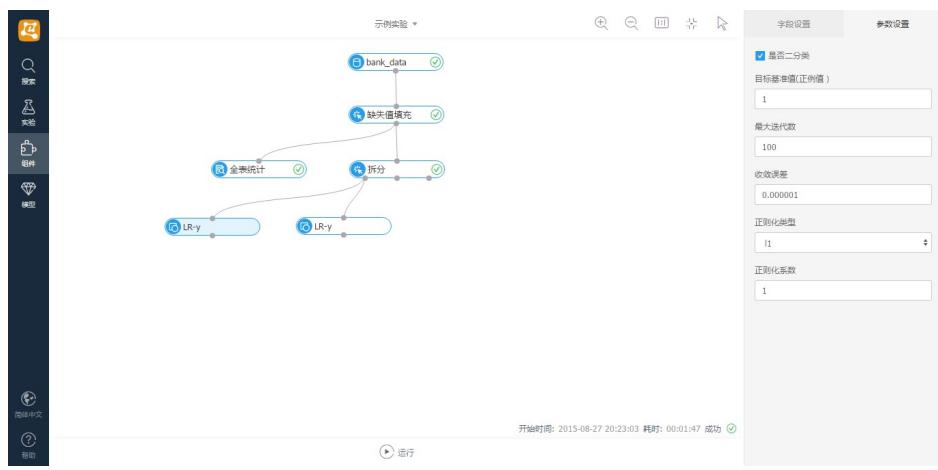
4.2、组件1选择输入列 选择9个feature列，参数设置采用系统默认参数

选择字段

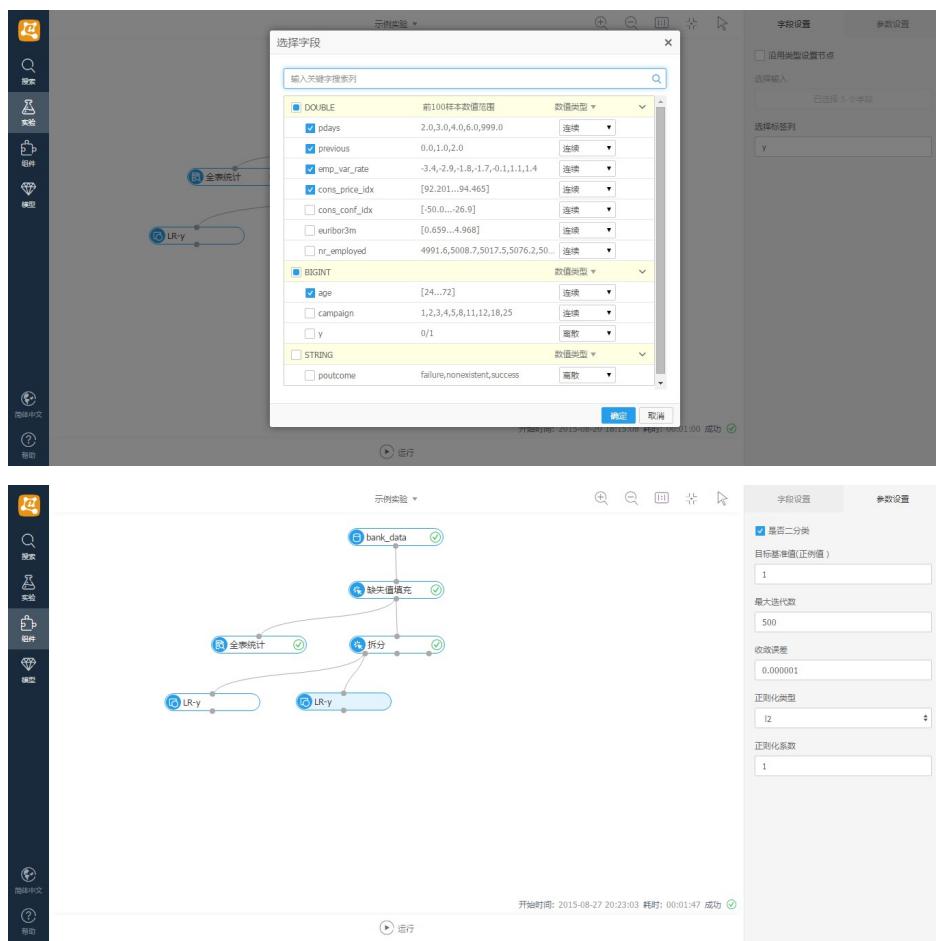
已选择 9 个字段

选择标签列

y

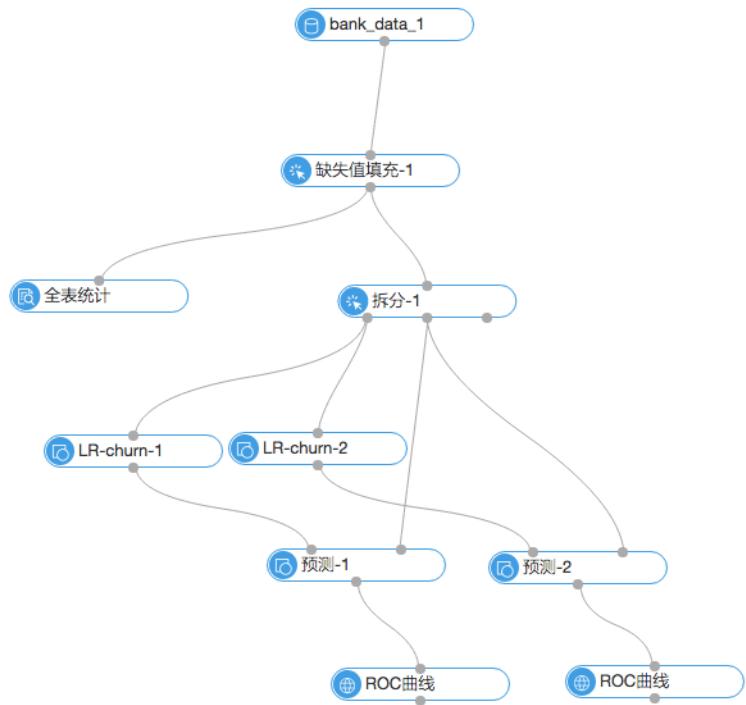


4.3、组件2选择输入列 选择5个feature列，参数设置稍作修改

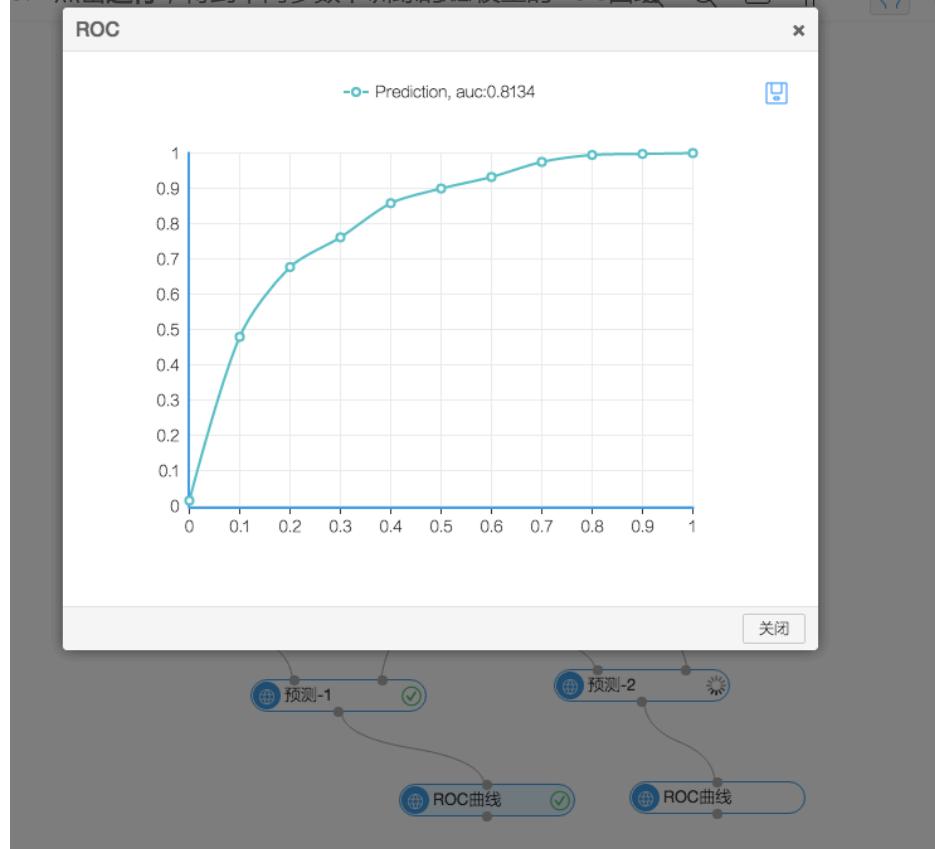


5、多模型评估

5.1 向画布拖入两个预测组件和两个ROC组件，分别连接对应的组件流和数据流，如下图。



5.2 点击运行，得到不同参数下训练的LR模型的ROC曲线



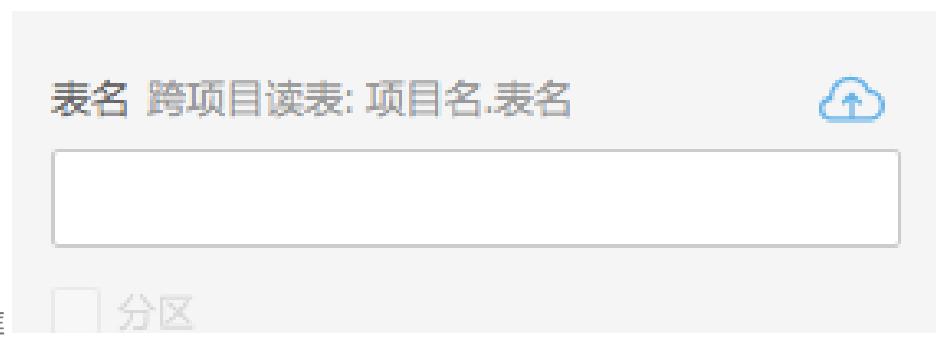
源/目标

读ODPS表

读取ODPS的表数据组件，默认读取本工程下的数据；若读取其他工程的表数据且拥有该project的操作权限），只需在表名前添加工程名，格式：工程名.表名，如：tianchi_project.weibo_data 当输入表后，会自动读取表的结构数据，可点击字段信息查看 本组件不支持视图

ODPS表字段修改后，如增加或删除某个字段，在算法平台中是无法感知的，需要用户重新设置一下ODPS源，reload一下这个表信息。

若输入表是分区表，后台会自动勾选分区框，用户可选择或输入分区参数，目前仅支持输入单个分区。不勾选分区框或勾选后不输入分区参数均默认为输入全表 若输入表是非分区表，分区框不可勾选



写ODPS表

写入ODPS表的数据组件，同样支持写入其他工程的表数据。写入表数据不支持分区操作

读Volume

目前只有GPU算法需要

读取Volume的信息

ODPS表转Volume

功能介绍

- 将ODPS表中的数据转换为pluto可以直接使用的数据格式，并写入指定volume partition中
pai命令示例

```
pai -name table2volume
-DtrainTableName=algo_test.mnist_train -DtestTableName=algo_test.mnist_test
-DfeatureColName=feature -DfeatureDimensions=784 -DlabelColName=label
-DtargetVolume=algo_test/volumes/xuege_test -DtargetVolumePartition=data
-DtrainDataFolderName=mnist_train -DtestDataFolderName=mnist_test
-Dchannels=1 -Dheight=28 -Dwidth=28;
```

算法参数

参数名称	参数描述	取值范围	是否必选，默认值/行为
trainTableName	存放训练数据的输入表的名称	表名，格式形如 : algo_test.mnist_train	必选
trainTablePartition	存放训练数据的输入表的partition	partition名称，格式 形如 : "pt=201602231500"	可选，默认值为空
testTableName	存放测试数据的输入表的名称	表名，格式形如 : algo_test.mnist_test	必选
testTablePartition	存放测试数据的输入表的partition	partition名称，格式 形如 : "pt=201602231500"	可选，默认值为空
featureColName	输入表中特征列列名 (trainTable和 testTable两个表相同的schema)	列名 (特征列只有一列)	必选，只能选定一个特征列，若选择多个特征列则报错
featureDimensions	特征列中特征的维数	正整数，大于等于1	必选参数
labelColName	输入表中label列列名 (trainTable和 testTable两个表相同的schema)	列名 (label列只有一列)	必选，只能选定一个特征列，若选择多个特征列则报错
labelDimensions	label列中label的维数	正整数，大于等于1	可选参数，默认值为1
delimiter	feature列和label列中 数值之间的分隔符	string类型	可选参数，默认值为 ";"
targetVolume	目标volume	数据写入的volume	必选参数
targetVolumePartition	目标volume的 partition	数据写入的partition	必选参数
trainDataFolderName	训练数据写入的文件夹 名	string类型	可选参数，默认值为 "train"
testDataFolderName	测试数据写入的文件夹 名	string类型	可选参数，默认值为

	名		"test"
channels	图像通道数目，当数据为图像时需要制定	非负整数	可选参数，默认值为0
height	图像的height，当数据为图像时需要制定	非负整数	可选参数，默认值为0
width	图像的width，当数据为图像时需要制定	非负整数	可选参数，默认值为0

输入输出说明

- 输入的两个表trainTable和testTable使用相同的schema，第一列存放label，第二列存放feature
- 输入表的列中，写入的是讲数值和delimiter一起串接起来组成的字符串，如| 7 | 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,...|
- 当为多label任务时，将多个label使用delimiter链接起来
- 该方法运行结束之后，会生成一个file_list.txt存放数据文件在pluto workspace中的相对路径，直接作为pluto binary data layer的输入
- 当表中的数据为图像数据时，需要制定channels、height和width三个参数

Volume转ODPS

功能介绍

- 将pluto生成在volume中的预测结果导入到ODPS表中

pai命令示例

```
pai -name volume2table
-DsourceVolume=algo_test/volumes/xuege_test -DsourceVolumePartition=output
-DsourceVolumeFile=predict_result.txt -DttargetTable=predict_result_table
```

算法参数

参数名称	参数描述	取值范围	是否必选，默认值/行为
sourceVolume	工作volume名称	volume名，格式形如：algo_test_stgctl/volumes/xuege_test	必选
sourceVolumePartition	存放输入的预测结果文件的partition	partition名	必选
sourceVolumeFile	存放预测结果的文件的名称	文件名	必选
targetTable	输出表的名称	ODPS表名	必选
targetTablePartition	预测结果写入的输出表的partition	partition名称，形式如："pt=20160223150	可选，默认值为""

		0"	
--	--	----	--

输入输出说明

- pluto的预测结果如：9 9 0.99275 {"9":0.99275,"7":0.006447,"4":0.000503,"8":0.000294,"3":0.000003,"2":0.000003,"5":0.000001,"0":0.000000,"1":0.000000,"6":0.000000,"others":0.000000}
 - 预测结果第一个数为真实label，第二个为预测的label，第三个为预测结果的得分，第四个json串为预测结果的细节
 - 本方法会自动生成一个表来存放最终转换结果，用户只需要指定表的名称和partition即可
 - 最终生成表的schema为：(true_label int, predict_label int, prediction_score double, prediction_detail)
-

数据预处理

采样与过滤

加权采样

以加权方式生成采样数据；权重列必须为double或int类型，按照该列的value大小采样；如col的值是1.2和1.0；则value=1.2所属样本的被采样的概率就大一些。

参数设置

参数框



- 可以选择放回采样或者不放回采样，默认为不放回，勾选后变为放回
- 可手动输入采样个数，默认是10000个点。注意：不放回时采样个数不能多于数据条数
- 下拉框选择加权列，加权列支持double型和bigint型

PAI 命令

```
PAI -name WeightedSample -project algo_public -DprobCol="previous" -DsampSize="500" \
-DoutputTableName="test2" -DinputPartitions="pt=20150501" -DinputTableName="bank_data_partition";
```

- name: 组件名字
- project: project名字，用于指定算法所在空间。系统默认是algo_public，用户自己更改后系统会报错
- replace: 是否放回。其中true为放回，false为不放回
- probCol : 选择的要加权的列，每个值代表所在record出现的权重，不需要归一化
- sampleSize : 采样个数。注意：不放回时采样个数不能多于数据条数
- outputTableNames: 输出表的名字，多张表用逗号分隔
- inputPartitions : (可选) 训练输入表分区。输入表对应的输入分区，选中全表则为None
- inputTableName: 输入表的名字
- replace: (可选) 是否放回。其中true为放回，不放回时无此参数

随机采样

以随机方式生成采样数据，每次采样是各自独立的。

PAI 命令

```
pai -name sample -project algo_public
-DinputTableName=wbpc
-DoutputTableName=wpbc_sample
-Dratio=0.3;
```

算法参数

参数名称	参数描述	参数值可选项	默认值
inputTableName	必选，输入表的表名	-	-
inputTablePartitions	可选，输入表中指定哪些分区参与训练，格式为: partition_name=value。如果是多级，格式为 name1=value1/name2=value2；如果指定多个分区，中间用','分 开	-	输入表的所有partition
ratio	必选，指定采样比例	(0,1)	-
outputTableName	必选，输出结果表	-	-
outputTablePartition	可选，输出结果表partition	-	输出表为非partition表
lifecycle	可选，指定输出表生命周期	正整数，[1,3650]	输出表没有生命周期

过滤与映射

对数据按照过滤表达式进行筛选；可以重命名字段名；

参数设置

1、通过where条件实现数据过滤，与SQL类似，如：

字段设置

条件

```
age>40
```

- 筛选条件：目前操作符支持"=" , "!=" , ">" , "<" , ">="和"<=" , like , rlike

选择字段

输入字段	类型	输出字段
<input type="checkbox"/> zcb_apply_cash_amt_1y	DOUBLE	zcb_apply_cash_amt_1y
<input type="checkbox"/> zcb_apply_cash_amt_std	DOUBLE	zcb_apply_cash_amt_std
<input type="checkbox"/> zcb_apply_cash_cnt_1y	BIGINT	zcb_apply_cash_cnt_1y
<input type="checkbox"/> zcb_apply_cash_cnt_std	BIGINT	zcb_apply_cash_cnt_std
<input type="checkbox"/> zcb_suc_prin_amt_1d	DOUBLE	zcb_suc_prin_amt_1d
<input type="checkbox"/> zcb_suc_prin_amt_1y	DOUBLE	my_new_feature
<input type="checkbox"/> zcb_suc_prin_amt_std	DOUBLE	zcb_suc_prin_amt_std
<input type="checkbox"/> zcb_suc_purchase_cnt_1y	BIGINT	zcb_suc_purchase_cnt_1y
<input type="checkbox"/> zcb_suc_purchase_cnt_std	BIGINT	zcb_suc_purchase_cnt_std

重命名字段

PAI命令

```
PAI -name Filter -project algo_public -DoutTableName="test_9" -DinputPartitions="pt=20150501\" -DinputTableName="bank_data_partition" -Dfilter="age>=40";
```

- name: 组件名字

- project: project名字，用于指定算法所在空间。系统默认是algo_public，用户自己更改后系统会报错
- outTableName: 输出表的名字
- inputPartitions : (可选) 训练输入表分区。输入表对应的输入分区，选中全表则为None
- inputTableName: 输入表的名字
- filter: where筛选条件，目前操作符支持"="，"!="，">"，"<"，">="，"<=","like"和"rlke"

分层采样

先将总体的单位按某种特征分为若干次级总体(层)，然后再从每一层内进行单纯随机抽样，组成一个样本的统计学计算方法。

参数设置

1.字段设置

分组列-必选项，按此列划分层次

2.参数设置

采样比例/样本数-必选项，小于1表示每层采样的比例，大于1表示每层采样的个数

特殊采样配置-可选项，提供不同层采集不同数量的功能 随机种子数-可选项，1234567

PAI 命令

```
pai -name sample -project algo_public
-DinputTableName=wbpc
-DoutputTableName=wpbc_sample
-DstrataColName="label"
-DsampleSize="A:200,B:300,C:500"
-DrandomSeed=1007
-Dlifecycle=30
```

算法参数

参数名称	参数描述	参数值可选项	默认值
inputTableName	必选，输入表的表名	-	-
inputTablePartitions	可选，输入表中指定哪些分区参与训练，格式为: partition_name=value。如果是多级，格式为 name1=value1/name2=value2；如果指定多个分区，中间用','分 开	-	输入表的所有partition
strataColName	必选，指定分层列	-	-
outputTableName	必选，输出结果表	-	-

sampleSize	可选，整数时表示每个 stratum的采样个数。 字符串时，格式为 strata0:n0,strata1:n1...表示每个stratum分别配置采样个数	-	
sampleRatio	可选，数字时：范围 [0,1]表示每个 stratum的采样比例。 字符串时：格式为 strata0 : r0 , strata1 : r1...表示每个 stratum分别配置采样比例	-	-
randomSeed	可选，随机种子数	-	0
lifecycle	可选，指定输出表生命周期	正整数，[1,3650]	输出表没有生命周期
coreNum	可选，核心数，默认自动分配	-	-
memSizePerCore	可选，每个核心的内存，默认自动分配	-	-

数据合并

join

两张表通过关联信息，合成一张表，并决定输出的字段;与SQL的join语句功能类似

参数设置

填写参数

字段设置

连接类型

左连接

关联条件

(+) +

age	=	age	✖
campaign	=	campaign	✖
pdays	=	pdays	✖

选择左表输出字段列

已选择 11 个字段

选择右表输出字段列

已选择 1 个字段

- 连接类型支持：左连接，内连接，右连接，全连接
- 关联条件目前只支持等式
- 可手动添加或删除关联条件

PAI 命令

不提供pai命令

合并列

将两张表的数据按列合并，需要表的行数保持一致

参数设置

如图：



左表选择输入列

选择字段

输入字段	类型	输出字段
<input type="checkbox"/> category	STRING	category
<input type="checkbox"/> dp	STRING	dp
<input type="checkbox"/> dt	STRING	dt
<input checked="" type="checkbox"/> petal_length	DOUBLE	petal_length
<input checked="" type="checkbox"/> petal_width	DOUBLE	petal_width
<input type="checkbox"/> sepal_length	DOUBLE	sepal_length
<input type="checkbox"/> sepal_width	DOUBLE	sepal_width

确定 取消

右表选择输入列

选择字段

输入字段	类型	输出字段
<input type="checkbox"/> category	STRING	category
<input type="checkbox"/> dp	STRING	dp
<input type="checkbox"/> dt	STRING	dt
<input checked="" type="checkbox"/> petal_length	DOUBLE	petal_length2
<input checked="" type="checkbox"/> petal_width	DOUBLE	petal_width2
<input type="checkbox"/> sepal_length	DOUBLE	sepal_length
<input type="checkbox"/> sepal_width	DOUBLE	sepal_width

确定 取消

- 选择的两张表行数需保持一致
- 左表与右表选择的输出列名不能重复
- 选择输出字段列时可手动更改输出字段名
- 若左表右表均不选择输出列，则默认输出全表。此时勾选'是否自动重命名输出列'，会将重复列重命名后输出

PAI 命令

```
PAI -name AppendColumns -project algo_public -
DoutputTableColNames="petal_length,petal_width,petal_length2,petal_width2" \
-DautoRenameCol="false" -DoutputTableName="pai_temp_770_6840_1" -
DinputTableNames="iris_twopartition,iris_twopartition" \
-DinputPartitionsInfoList="dt=20150125/dp=20150124;dt=20150124/dp=20150123" \
-DselectedColNamesList="petal_length,petal_width;sepal_length,sepal_width";
```

- name: 组件名字
 - project: project名字，用于指定算法所在空间。系统默认是algo_public，用户自己更改后系统会报错
 - outputTableColNames: 新表中各列的列名。逗号分隔，如果autoRenameCol为true，则此参数无效
 - autoRenameCol: (可选) 输出表是否自动重命名列，true为重命名，false不进行重命名，默认false
 - outputTableName: 输出的表名
 - inputTableNames: 输入的表名，如果有多个，逗号分隔
 - inputPartitionsInfoList: (可选) 输入表对应选择的partition列表，同一个表的各partition按逗号分隔，不同表的partition分号分隔
 - selectedColNamesList: 选择输入的列名，同张表之间列名用逗号分隔，不同表之间用分号分隔
-

UNION

将两张表的数据按行合并，左表及右表选择输出的字段个数以及类型应保持一致。整合了union和union all的功能。

参数设置 调整参数，如下：

字段设置

选择左表联合列

已选择 11 个字段

输入左表的where条件

age<30

选择右表联合列

已选择 11 个字段

输入右表的where条件

age>50

去重

- 进行联合操作时，左右表选择的列数需相同，对应列的类型需保证一致
- 可根据实际情况在条件框中手动输入已选字段的过滤条件，默认情况下全表，目前操作符支持`=`、`!=`、`>`、`<`、`>=`、`<=`、`like` 和 `rlike`
- 系统默认勾选去重。勾选后将会对生成的数据表的重复行进行去重操作（`distinct`）左表联合列

选择字段

输入关键字搜索列 ×

<input checked="" type="checkbox"/> 输入字段	类型	输出字段
<input checked="" type="checkbox"/> age	BIGINT	age
<input checked="" type="checkbox"/> campaign	BIGINT	campaign
<input checked="" type="checkbox"/> cons_conf_idx	DOUBLE	cons_conf_idx
<input checked="" type="checkbox"/> cons_price_idx	DOUBLE	cons_price_idx
<input checked="" type="checkbox"/> emp_var_rate	DOUBLE	emp_var_rate
<input checked="" type="checkbox"/> euribor3m	DOUBLE	euribor3m
<input checked="" type="checkbox"/> nr_employed	DOUBLE	nr_employed
<input checked="" type="checkbox"/> pdays	DOUBLE	pdays
<input checked="" type="checkbox"/> poutcome	STRING	poutcome
<input checked="" type="checkbox"/> previous	DOUBLE	previous

确定 取消

右表联合列

选择字段

输入关键字搜索列 ×

<input type="checkbox"/> 输入字段	类型	输出字段
<input checked="" type="checkbox"/> age	BIGINT	age
<input checked="" type="checkbox"/> campaign	BIGINT	campaign
<input checked="" type="checkbox"/> cons_conf_idx	DOUBLE	cons_conf_idx
<input checked="" type="checkbox"/> cons_price_idx	DOUBLE	cons_price_idx
<input checked="" type="checkbox"/> emp_var_rate	DOUBLE	emp_var_rate
<input checked="" type="checkbox"/> euribor3m	DOUBLE	euribor3m
<input type="checkbox"/> label	STRING	label
<input checked="" type="checkbox"/> nr_employed	DOUBLE	nr_employed
<input checked="" type="checkbox"/> pdays	DOUBLE	pdays
<input checked="" type="checkbox"/> poutcome	STRING	poutcome

确定 取消

PAI 命令

不提供pai命令

其它

增加序号列

在数据表第一列追加ID列，并将append id后的內容存为新表

参数设置 略

PAI命令

```
PAI -name AppendId -project algo_public -DIDColName="append_id" -DoutputTableName="test_11" -  
DinputTableName="bank_data" \  
-  
DselectedColNames="age,campaign,cons_conf_idx,cons_price_idx,emp_var_rate,euribor3m,nr_employed,pdays,pout  
come,previous,y";
```

- name: 组件名字
- project: project名字，用于指定算法所在空间。系统默认是algo_public，用户自己更改后系统会报错
- IDColName : 追加的ID列列名，从0开始编号，依次为0,1,2,3...
- outputTableNames: 输出表的名字

inputTableName: 输入表的名字

selectedColNames: 要保留的字段名，多个用逗号分隔

缺失值填充

通过给定一个缺失值的配置列表，来实现将输入表的缺失值用指定的值来填充

参数设置

字段设置

选择字段列

已选择 1 个字段

缺失值

空值和空字符

输入自定义的数值或字符

替换为

自定义

testing

- 待填充的缺失值可以选择空值或空字符，也可以自定义
- 缺失值若选择空字符，则填充的目标列应是string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

PAI 命令

```
PAI -name fillMissValue -project algo_public -Dconfigs="poutcome,null-empty,testing" \
-DoutputTableName="test_3" -DinputPartitions="pt=20150501" -DinputTableName="bank_data_partition";
```

归一化

对一个表的某一列或多列，进行归一化处理，产生的数据存入新表中。

目前支持的是线性函数转换，表达式如下： $y = (x - \text{MinValue}) / (\text{MaxValue} - \text{MinValue})$
MaxValue、MinValue分别为样本的最大值和最小值。

参数设置 参数设置：

Table Select

New table name

Partition

user-defined

- 可以选择是否保留原始列，勾选后原始列会被保留，处理过的列重命名
- 点击选择字段按钮可以选择想要归一化的列，目前支持double类型与bigint类型

PAI 命令

```
PAI -name normalize_wf -project algo_public -DkeepOriginal="true" -DoutputTableName="test_4" -
-DinputPartitions="pt=20150501" \
-DinputTableName="bank_data_partition" -DselectedColNames="emp_var_rate,euribor3m";
```

算法参数

参数名称	参数描述	参数值可选项	默认值
inputTableName	必选，输入表	-	-
selectedColNames	可选，输入表选择列	-	默认选中全部列
inputTablePartitions	可选，训练输入表分区 。	-	默认选中全表
outputTableName	必选，输出表	-	-
outputParaTableName	必选，配置输出表	-	-
inputParaTableName	可选，配置输入表	-	默认不输入
outputPMMLTableName	必选，PMML输出表	-	-
keepOriginal	可选，是否保留原始列 (keepOriginal =true时，处理过的列 重命名 ("normalized_"前缀)，原始列保留	-	默认为false

	, keepOriginal=false时，全部列保留且不重命名)		
lifecycle	可选，输出表生命周期	-	默认不设置
coreNum	可选，核心个数	-	默认不设置，系统自动分配
memSizePerCore	可选，单个核心使用的内存数	-	默认不设置，系统自动分配

标准化

给定输入数据的一列或多列，进行归一化处理存入新表。目前支持的线性函数转换 $y=(x-\text{MinValue})/(\text{MaxValue}-\text{MinValue})$

参数设置

选择需要标准化的列。

PAI 命令

```
pai -name sample -project algo_public
-DinputTableName=wbpc
-DoutputTableName=wpbc_sample
-selectedColNames=a1,a2,a3
```

算法参数

参数名称	参数描述	参数值可选项	默认值
inputTableName	必选，输入表的表名	-	-
inputTablePartitions	可选，输入表中指定哪些分区参与训练，格式为: partition_name=value。如果是多级，格式为 name1=value1/name2=value2；如果指定多个分区，中间用','分开	-	输入表的所有partition
outputTableName	必选，输出结果表	-	-
outputPartition	可选，输出表的partition	-	-
selectedColNames	必选，要归一化的列名，逗号分隔	-	所有bigint类型和double类型列
keepOriginal	可选，保留原始列，原数列被重命名后保留	true/false	false

	, 重命名规则是加 "_orig"后缀		
--	------------------------	--	--

特征工程

特征变换

主成分分析PCA

- PCA 利用主成分分析方法，实现降维和降维的功能. PCA算法原理见wiki
- 目前支持稠密数据格式

PAI 命令

```
PAI -name PrinCompAnalysis
-project algo_public
-DinputTableName=bank_data
-DeigOutputTableName=pai_temp_2032_17900_2
-DprincompOutputTableName=pai_temp_2032_17900_1
-DselectedColNames=pdays,previous,emp_var_rate,cons_price_idx,cons_conf_idx,euribor3m,nr_employed
-DtransType=Simple
-DcalcuType=CORR
-DcontriRate=0.9;
```

算法参数说明

参数名称	参数描述	参数值可选项	默认值
inputTableName	必选，进行主成分分析的输入表	-	-
eigOutputTableName	必选，特征向量与特征值的输出表	-	-
princompOutputTableName	必选，进行主成分降维降噪后的结果输出表	-	-
selectedColNames	必选，参与主成分分析运算的特征列	-	-
transType	可选，原表转换为主成分表的方式,支持 Simple, Sub-Mean, Normalization	-	Simple
calcuType	可选，对原表进行特征分解的方式,支持 CORR/COVAR_SAMP/COVAR_POP	-	CORR
contriRate	可选，降维后数据信息	-	0.9

	保留的百分比		
remainColumns	可选，降维表保留原表的字段	-	-

PCA输出示例

数据探查 - pai_temp_2032_17900_1 - (仅显示前一百条)

prin0 ▲	prin1 ▲	prin2 ▲	prin3 ▲
95.78807909...	-42.95729950747...	-87.75761249391427	-78.22763106620326
94.55356656...	-42.09090844438...	-68.61978267691727	-78.13608278651054
89.91510009...	-47.88349039874...	-70.2131838386143	-79.3464187998396
92.45302559...	-41.19278330986...	-69.35650482673981	-78.92704651042823
89.76236928...	-46.44677428041...	-66.95927201734735	-77.10906366265652
95.94066194...	-42.65895343179...	-69.30780740061341	-78.59136424888288
92.33542049...	-41.16439652633...	-69.07389275424413	-78.73867394929054
92.33143893...	-41.16317033059...	-69.07404676926707	-78.74043328209397
90.14859611...	-46.10294878666...	-69.21610483502704	-77.68690680501652
Q1 Q3 Q5 Q7 Q9	-42.95729950747...	-68.61978267691727	-78.22763106620326

降维后的数据表

特征值和特征向量表

数据探查 - pai_temp_2032_17900_2 - (仅显示前一百条)

prin_name ▲	pdays ▲	previous ▲	emp_var_rate ▲	cons_price_idx ▲	cons_conf_idx ▲	euribor3m ▲	nr_employed ▲	eigenvalue ▲	contributionrate ▲	sumcontributionrate ▲
prin0	0.2289...	-0.307801...	0.49043713976...	0.3676221023847...	0.103704168633...	0.49327195...	0.4723413876...	3.864397508...	0.5520567869804135	0.5520567869804135
prin1	0.6651...	-0.511030...	-0.1750362148...	-0.306944720053...	-0.38529019595...	-0.1519122...	0.0083997057...	1.333469023...	0.19049574717606...	0.7425623616980203
prin2	0.1344...	-0.248552...	-0.1027813425...	-0.355514661117...	0.882898436451...	0.01908078...	-0.057503451...	0.953306884...	0.136186697819829...	0.8787390595178498
prin3	-0.216...	0.2034146...	0.07125284421...	-0.732942136387...	-0.16599040564...	0.21796215...	0.5426739826...	0.426458636...	0.060922662369309...	0.9396617218871589

特征选择

偏好计算

功能介绍

- 给定用户的明细行为特征数据，自动计算用户对特征值的偏好得分。
- 输入表包含用户id和用户明细行为特征输入，假设在口碑到店场景，某用户2088xxx1在3个月内吃了两次川菜，一次西式快餐，那么输入表形式如下：

user_id	cate
2088xxx1	川菜
2088xxx1	川菜
2088xxx1	西式快餐

- 输出表为用户对川菜和西式快餐的偏好得分,形式如下：

user_id	cate
2088xxx1	川菜:0.0544694,西式快餐:0.0272347

PAI命令示例

```
pai -name=preference
    -project=algo_public
    -DInputTableName=preference_input_table
    -DIdColumnName=user_id
    -DFeatureColNames=cate
    -DOOutputTableName=preference_output_table
    -DmapInstanceNum=2
    -DreduceInstanceNum=1;
```

算法参数

参数key名称	参数描述	必/选填	默认值
InputTableName	输入表名	必填	-
IdColumnName	用户id列	必填	-
FeatureColNames	用户特征列	必填	-
OutputTableName	输出表名	必填	-
OutputTablePartitions	输出表分区	选填	-
mapInstanceNum	mapper数量	选填	2
reduceInstanceNum	reducer数量	选填	1

实例

测试数据

新建数据SQL

```
drop table if exists preference_input_table;
create table preference_input_table as
select
  *
from
(
  select '2088xxx1' as user_id, '川菜' as cate from alipaydw.dual
  union all
  select '2088xxx1' as user_id, '川菜' as cate from alipaydw.dual
  union all
  select '2088xxx1' as user_id, '西式快餐' cate from alipaydw.dual
  union all
  select '2088xxx3' as user_id, '川菜' as cate from alipaydw.dual
  union all
  select '2088xxx3' as user_id, '川菜' as cate from alipaydw.dual
```

```
union all
select '2088xxx3' as user_id, '西式快餐' as cate from alipaydw.dual
) tmp;
```

运行结果

```
+-----+-----+
| user_id | cate    |
+-----+-----+
| 2088xxx1 | 川菜:0.0544694,西式快餐:0.0272347 |
| 2088xxx3 | 川菜:0.0544694,西式快餐:0.0272347 |
+-----+-----+
```

随机森林特征重要性

- 计算随机森林模型中特征重要性

PAI 命令

```
pai -name feature_importance -project algo_public
-DinputTableName=input
-DoutputTableName=output
-Dlabel=label
-DmodelName=model
```

算法参数

参数名称	参数描述	参数值可选项	默认值
inputTableName	必选，输入表	-	-
outputTableName	必选，输出表	-	-
labelColName	必选，label所在的列	-	-
modelName	必选，输入的模型名	-	-
featureColNames	可选，输入表选择的特征	-	默认除label外的其他列
inputTablePartitions	可选，输入表选择的分区	-	默认选择全表
lifecycle	可选，输出表的生命周期	-	默认不设置
coreNum	可选，核心数	-	默认自动计算
memSizePerCore	可选，内存数	-	默认自动计算

特征选择

- 功能：根据用户选择的特征，特征筛选方法，过滤出TopN的特征数据，同时保持所有特征重要性权

重

PAI 命令

```

pai -name fe_select_runner -project algo_public
-DinputTable=pai_dense
-DoutputTable=pai_temp_select
-DfeatImportanceTable=pai_tmp_feature_weight
-DselectedCols=age,workclass,fwlgth,edu,edu_num,married,family,race,sex,gail,loss,work_year,country
-DlabelCol=income
-DcategoryCols=age
-DmaxBins=100
-DselectMethod=iv
-DtopN=2
-DisSparse=false
-Dlifecycle=7;

```

算法参数

参数名称	参数描述	参数值可选项	默认值
inputTable	必选，输入表的表名	-	-
inputTablePartitions	可选，输入表中指定哪些分区参与训练，格式为: partition_name=value。如果是多级，格式为 name1=value1/name2=value2；如果指定多个分区，中间用','分开	-	输入表的所有 partition
outputTable	必选，过滤后的特征结果表	-	-
featImportanceTable	必选，存放所有输入特征的重要性权重值	-	-
selectedCols	必选，特征列	-	-
labelCol	必选，标签列/目标列	-	-
categoryCols	存在在Int或者Double字符的枚举特征，可选		""
maxBins	连续类特征划分最大区间数，可选		100
selectMethod	特征选择方法，可选，默认iv 目前支持:iv, Gini增益, 信息增益		iv
topN	挑选的TopN个特征，如果topN答应输入特征数，则输出所有特		10

	征 , 可选 , 默认10		
isSparse	是否是k:v的稀疏特征 , 可选 , 默认稠密数据		100
itemSplitter	稀疏特征item分隔符 , 可选 , 默认逗号		" , "
kvSplitter	稀疏特征item分隔符 , 可选 , 默认冒号		" : "
lifecycle	结果表生命周期 , 可选 , 默认7		7

特征生成

窗口变量统计

功能介绍

- 给定时间窗口 , 计算相应用户在相应时间窗内的行为次数和金额。如时间窗口为'1,7,30,90,180' , 则计算用户相应天数内的行为次数和金额。

PAI命令示例

```
pai -name=rfm
-project=algo_public
-DinputTableName=window_input_table
-DuserName=user_id
-DdtName=dt
-DcntName=cnt
-DamtName=amt
-Dwindow=1,7,30,90
-DoutputTableName=window_output_table
-DmapInstanceNum=2
-DreduceInstanceNum=2;
```

算法参数

参数key名称	参数描述	必/选填	默认值
inputTableName	输入表名	必填	-
userName	用户id列	必填	-
dtName	时间列 (格式 '20160101')	必填	-
cntName	次数列	必填	-
amtName	金额列	必填	-
window	时间窗口(格式为 1,7,30,90...)	必填	-

outputTableName	输出表名	必填	-
outputTablePartitions	输出表分区	选填	-
mapInstanceNum	mapper数量	选填	2
reduceInstanceNum	reducer数量	选填	2

实例

测试数据

新建数据SQL

```

drop table if exists window_input_table;
create table window_input_table as
select
*
from
(
  select 'a' as user_id, '20151201' as dt, 2 as cnt, 32.0 as amt from dual
  union all
  select 'a' as user_id, '20160201' as dt, 3 as cnt, 37.0 as amt from dual
  union all
  select 'a' as user_id, '20160223' as dt, 1 as cnt, 22.0 as amt from dual
  union all
  select 'b' as user_id, '20151212' as dt, 1 as cnt, 12.0 as amt from dual
  union all
  select 'b' as user_id, '20160110' as dt, 2 as cnt, 30.0 as amt from dual
  union all
  select 'c' as user_id, '20151001' as dt, 3 as cnt, 60.0 as amt from dual
  union all
  select 'c' as user_id, '20151201' as dt, 2 as cnt, 39.0 as amt from dual
) tmp;

```

运行结果

```

+-----+-----+-----+-----+-----+-----+-----+
+
| user_id | cnt_1d_sum | amt_1d_sum | cnt_7d_sum | amt_7d_sum | cnt_30d_sum | amt_30d_sum | cnt_90d_sum |
| amt_90d_sum |
+-----+-----+-----+-----+-----+-----+-----+
+
| a      | 1       | 22.0    | 1       | 22.0    | 4       | 59.0    | 6       | 91.0    |
| c      | 0       | 0.0     | 0       | 0.0     | 0       | 0.0     | 2       | 39.0    |
| b      | 0       | 0.0     | 0       | 0.0     | 0       | 0.0     | 3       | 42.0    |
+-----+-----+-----+-----+-----+-----+-----+
+

```

统计分析

百分位

对一个存在的表，单列数据计算百分位

参数设置

选择需要分析的字段，仅支持double类型和bigint类型

运行结果，如下

key(%)	value▼
Min	0.634
Max	5.045
Median	4.857
Lower Quartile(Q1)	1.344
Upper Quartile(Q3)	4.961
0	0.634
1	0.655
2	0.714
3	0.72
4	0.74
5	0.797
6	0.854
7	0.879
8	0.884
9	0.904

PAI 命令

```
PAI -name Percentile -project algo_public -DoutputTableName="pai_temp_666_6014_1" \
-DcolName="euribor3m" -DinputTableName="bank_data";
```

- name: 组件名字
- project: project名字，用于指定算法所在空间。系统默认是algo_public，用户自己更改后系统会报错
- outputTableName: 系统执行百分位运算后自动分配的结果表
- colName：要计算百分位的列，仅支持数字型
- inputTableName: 输入表的名字

全表统计

对一个存在的表，进行全表基本统计，或者仅对选中的列做统计

参数设置

调整参数，如：

参数设置

选择输入列 (默认全部列)

已选择 2 个字段

条件 (默认全表分析)

age>40

- 选择输入列框中可选择需要进行统计的列，默认情况下统计全部列
- 条件框中可手动输入已选字段的过滤条件，默认情况下全表统计，目前操作符支持
"=" , "!=" , ">" , "<" , ">=" , "<=" , "like" 和 "rlike"

运行结果，如下

字段	Total	Missing	NaN	Positive Infinity	Negative Infinity	Min	Max	Mean	Variance	SD
pdays	17420	0	0	0	0	0	999	961.9393800229...	35438.5590438776...	188.2513188370...
euribor3m	17420	0	0	0	0	0.634	5.045	3.748874856487...	2.869936781053...	1.694088776024...

PAI 命令

```
PAI -name SimpleSummary -project algo_public -DsummaryColNames="euribor3m,pdays"\n-DoutputTableNames="pai_temp_667_6017_1" -DinputTableName="bank_data" -Dfilter="age>40";
```

- name: 组件名字
- project: project名字，用于指定算法所在空间。系统默认是algo_public，用户自己更改后系统会报错
- summaryColNames：需要进行统计的列，多个列用逗号分隔

- outputTableNames: 系统执行全表统计运算后自动分配的结果表
- inputTableName: 输入表的名字
- filter: 过滤条件，目前操作符支持"="，"!="，">"，"<"，">="，"<="，"like" 和 "rlike"

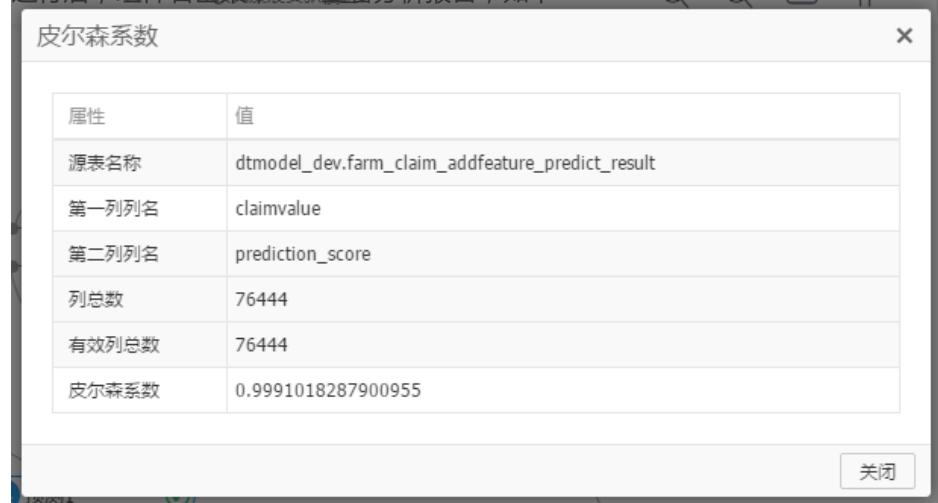
皮尔森系数

对输入表或分区的2列（必须为数值列），计算其pearson相关系数，结果存入输出表。

使用说明

组件的仅两个参数：输入列1、输入列2；将需要计算相关系数的两列的列名填入即可；

运行后，组件右击菜单--> 查看分析报告，如下



最后一列皮尔森系

数值

pai命令示例

```
pai -name pearson
-project algo_test
-DinputTableName=wpbc
-Dcol1Name=f1
-Dcol2Name=f2
-DoutputTableName=wpbc_pear;
```

算法参数

参数key名称	参数描述	取值范围	是否必选，默认值/行为
inputTableName	输入表的表名	表名	必选
inputTablePartitions	输入表中指定哪些分区参与计算	格式为: partition_name=valu	输入表的所有partition

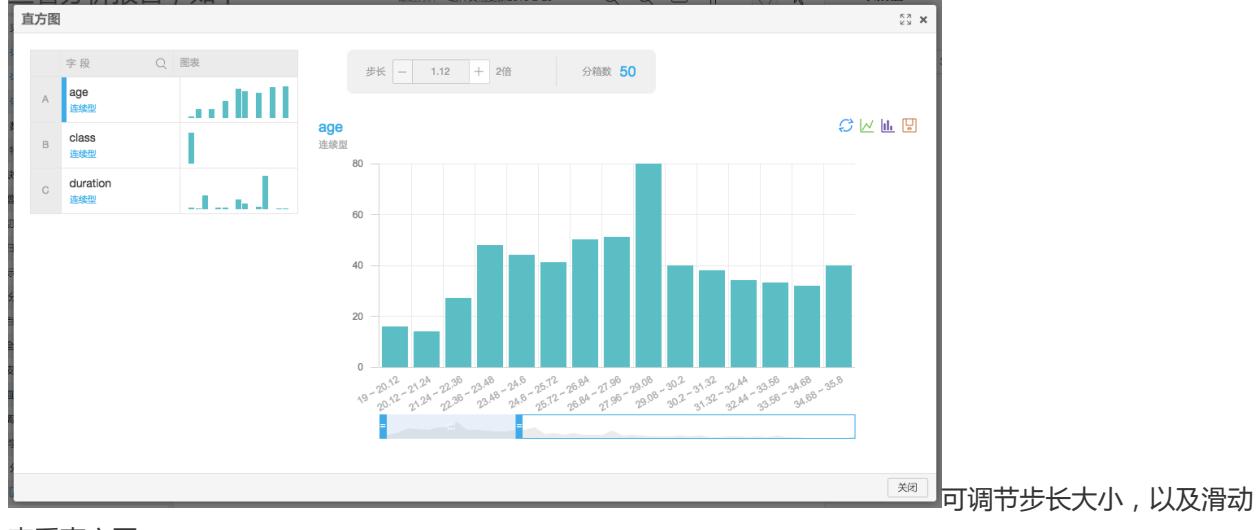
		e。如果是多级格式为 name1=value1/nam e2=value2 ; 如果是指定多个分区，中间用 ',' 分开	
col1Name	输入列1	列名	必选
col2Name	输入列2	列名	必选
outputTableName	输出结果表	表名	必选

直方图

对一个存在的表，单列数据计算直方图

参数设置 选择需要分析字段，支持double类型和bigint类型

查看分析报告，如下



查看直方图

离散值特征分析

离散值特征分析统计离散特征的分布，gini，entropy，gini gain，info gain等指标

pai命令示例

```
PAI
-name enum_feature_selection
-project algo_public
-DinputTableName=enumfeautreselection_input
-DlabelColName=label
-DfeatureColNames=col0,col1
```

```
-DenableSparse=false
-DoutputCntTableName=enumfeautreselection_output_cntTable
-DoutputValueTableName=enumfeautreselection_output_valuetable
-DoutputEnumValueTableName=enumfeautreselection_output_enumvaluetable;
```

算法参数

参数key名称	参数描述	取值范围	默认值
inputTableName	必选，输入表名	-	-
inputTablePartitions	可选，输入表选择的分区	-	默认选择全表
featureColNames	可选，输入表选择的列名	-	默认选择除label外的其他列，如果输入表为KV格式，则默认选择所有的string类型的列
labelColName	必选，label所在的列	-	-
enableSparse	可选，输入表是否是KV格式	-	默认为表
kvFeatureColNames	可选，KV格式的特征	-	默认选择全表
kvDelimiter	可选，KV之间的分隔符	-	默认为:
itemDelimiter	可选，K和V的分隔符	-	默认为,
outputCntTableName	必选，输出离散特征的枚举值分布数表	-	-
outputValueTableName	必选，输出离散特征的gini, entropy表	-	-
outputEnumValueTableName	必选，输出离散特征枚举值gini, entropy表	-	-
lifecycle	可选，输入表的声明周期	-	默认不设置声明周期
coreNum	可选，总得core个数	-	默认自动设置
memSizePerCore	可选，单个core对应的内存数量，单位为M	-	默认为自动设置

T检验

单样本T检验是检验某个变量的总体均值和某指定值之间是否存在显著差异。T检验的前提是样本总体服从正态分布。

pai命令示例

```
pai -name t_test -project algo_public
-DxTableName=pai_t_test_all_type
-DxColName=col1_double
-DoutputTableName=pai_t_test_out
-DxTablePartitions=ds=2010/dt=1
-Dalternative=less
-Dmu=47
-DconfidenceLevel=0.95
```

算法参数

参数名称	参数描述	取值范围	是否必选，默认值/行为
xTableName	输入表x	表名	必选
xColName	需要做t检验的列	列名,只能是double或者bigint	必选
outputTableName	输出表	不存在的表名	必选
xTablePartitions	输入表x的分区列表	分区列表	可选, 默认值:""
alternative	对立假设	two.sided, less, greater"	可选, 默认值: two.sided
mu	假设的均值	double	可选 , 默认值:0
confidenceLevel	置信度	0.8,0.9,0.95,0.99,0.995,0.999	可选 , 默认值:0.95

输出说明

输出是一个表，只有一行一列，是json格式。

```
{
  "AlternativeHypothesis": "mean not equals to 0",
  "ConfidenceInterval": "(44.72234194006504, 46.27765805993496)",
  "ConfidenceLevel": 0.95,
  "alpha": 0.05,
  "df": 99,
  "mean": 45.5,
  "p": 0,
  "stdDeviation": 3.919647479510927,
  "t": 116.081867662439
}
```

残差直方图

残差直方图

pai命令示例

```
pai -name residual_histogram -project algo_public
-DinputTableName=input_table
-DyColName=y_col
-DpredictionColName=prediction_col
-DoutputTableName=output_table;
```

算法参数

参数名称	参数描述	取值范围	是否必选，默认值/行为
inputTableName	必选，输入表的表名	-	-
outputTableName	必选，输出表表名	-	-
ycolName	必选，输入表原始因变量列名，数值类型	-	-
predictionColName	必选，预测结果因变量列名，数值类型	-	-
yColName	y表的列	列名,只能是double或者bigint	必选
intervalNum	可选，区间隔数	-	100
lifecycle	可选，指定输出表的生命周期	-	不设置
coreNum	可选，指定instance个数	-	不设置
memSizePerCore	可选，每个核的内存大小	-	不设置

机器学习

二分类

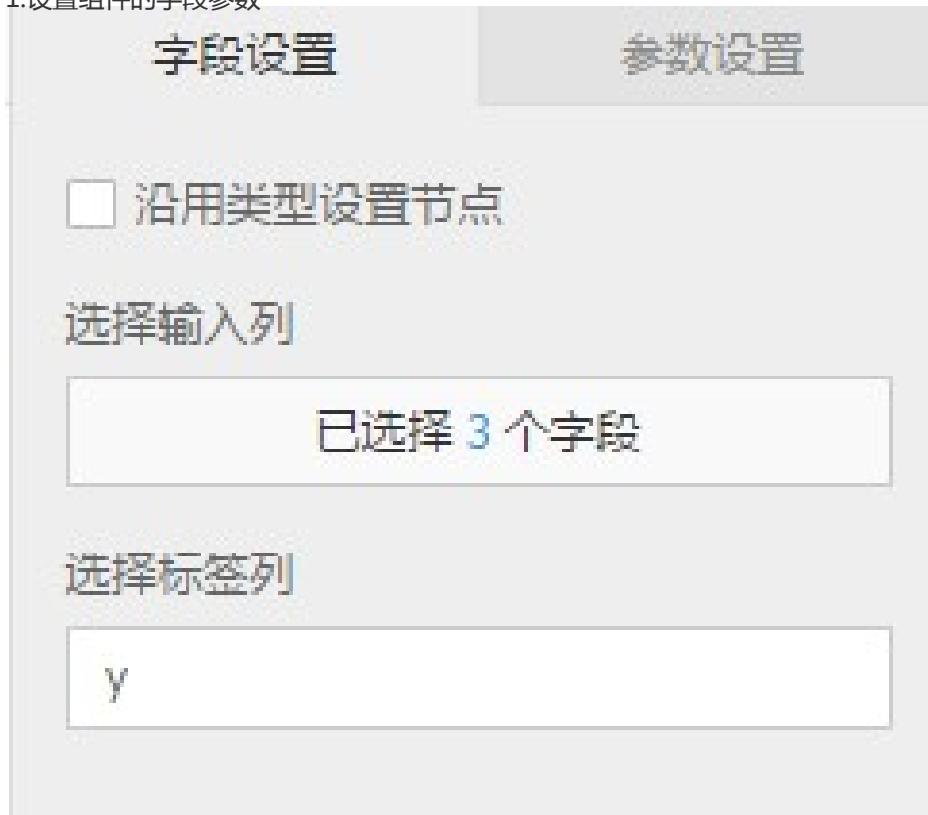
线性支持向量机

支持向量机(SVM)是建立在VC维理论和结构风险最小原理基础上的一种模式识别方法。本版线性支持向量机不是采用核函数方式实现的，具体实现理论详见：

<http://www.csie.ntu.edu.tw/~cjlin/papers/logistic.pdf> 中的6. Trust Region Method for L2-SVM；本算法仅支持二分类

参数设置

1. 设置组件的字段参数



- 输入列：选择输入列只支持bigint 与 double类型
- 标签列：支持bigint类型、double类型和string类型; 本组件仅支持二分类问题

1. 设置算法参数

字段设置 参数设置

惩罚因子

设置正负例因子权重

目标基准值(正例值, 可选)

正例权重值

负例权重值

收敛系数

- 惩罚因子：默认为1
 - 目标基准值：（可选）正例的值，不指定则随机选一个。建议正负例样本差异大时指定
 - 正例权重值：（可选）正例惩罚因子，默认1.0，范围(0, ~)
 - 负例权重值：（可选）负例惩罚因子，默认1.0，范围(0, ~)
 - 收敛系数：（可选）收敛误差，默认0.001，范围(0, 1)
- 注意：当不指定目标基准值时，正例权重值和负例权重值必须相同

3.运行结果（见随机森林组件说明）

PAI 命令 (未沿用类型设置节点)

```
PAI -name LinearSVM -project algo_public -DnegativeCost="1.0" -DmodelName="xlab_m_LinearSVM_6143" \
-DpositiveCost="1.0" -Depsiilon="0.001" -DlabelColName="y" - \
DfeatureColNames="pdays,emp_var_rate,cons_conf_idx" \
-DinputTableName="bank_data" -DpositiveLabel="0";
```

- name: 组件名字
- project: project名字。用于指定算法所在空间，系统默认是algo_public，用户自己更改后系统会报错
- negativeCost : (可选)负例权重值。即负例惩罚因子，默认1.0，范围(0,~)
- modelName: 输出的模型名
- positiveCost : (可选)正例权重值。即正例惩罚因子，默认1.0，范围(0, ~)
- epsilon : (可选)收敛系数。默认值为0.001，范围(0, 1)
- labelColName : 标签列列名
- featureColNames : 输入表中选择的用于训练的特征列名
- inputTableName: 训练输入表的表名
- positiveLabel : (可选) 正例值。不指定则随机选一个

逻辑回归二分类

经典逻辑回归是一个二分类算法，算法平台的逻辑回归可以支持多分类。逻辑回归组件支持稀疏、稠密两种数据格式

参数设置

逻辑回归组件的参数

是否二分类 输入是否为稀疏矩阵

最大迭代数

收敛误差

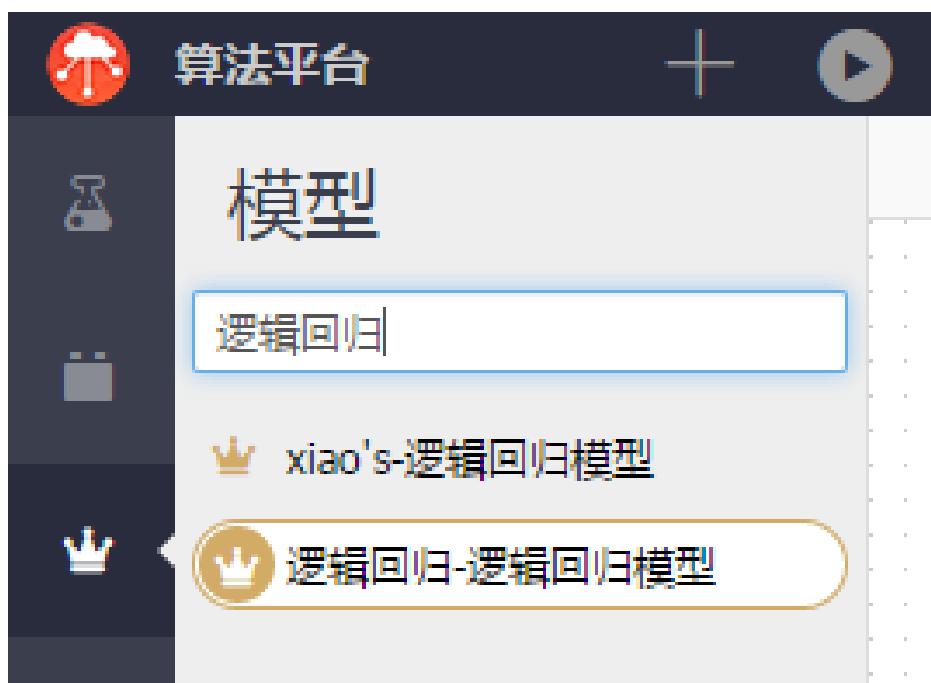
正则化类型



正则化系数

- 是否二分类：(可选)默认多分类，勾选后表示选择二分类
- 是否支持稀疏矩阵：组件可支持稀疏矩阵的格式
- 目标基准值：(可选)二分类时，指定训练系数针对的label值；如果填空，会随机选择一个
- 最大迭代数：(可选)L-BFGS的最大迭代次数，默认是100
- 收敛误差：(可选)L-BFGS的终止条件，即两次迭代之间log-likelihood的差，默认为 $1.0e-06$
- 正则化类型：(可选)正则化类型，可以选择'l1'、'l2'、'None'，默认为'l1'
- 正则化系数：(可选)正则项系数，默认为1.0；当 regularizedType 为 None 时，该项会被忽略

逻辑回归组件运行结果是模型，可在模型列表中查找：



模型名称的命名规则：实验名 + "—" + 算法组件名称 + "model"

PAI 命令（未沿用类型设置节点）

```
PAI -name LogisticRegression -project algo_public -DmodelName="xlab_m_logistic_regression_6096" \
-DregularizedLevel="1" -DmaxIter="100" -DregularizedType="l1" -Depslon="0.000001" -DlabelColName="y" \
-DfeatureColNames="pdays,emp_var_rate" -DgoodValue="1" -DinputTableName="bank_data";
```

- name: 组件名字
- project: project名字。用于指定算法所在空间，系统默认是algo_public，用户自己更改后系统会报错
- modelName: 输出的模型名
- regularizedLevel : (可选) 正则化系数。默认为 1.0；当 regularizedType 为 None 时，该项会被忽略
- maxIter : (可选) 最大迭代数。指定L-BFGS的最大迭代次数，默认是100
- regularizedType : (可选) 正则化类型，可以选择'l1'、'l2'、'None'，默认为'l1'
- epsilon : (可选) 收敛误差。L-BFGS的终止条件，即两次迭代之间log-likelihood的差，默认为1.0e-06
- labelColName : 输入表标签列列名
- featureColNames : 输入表中选择的用于训练的特征列名
- goodValue : (可选) 目标基准值。二分类时，指定训练系数针对的 label 值；如果填空，会随机选择一个
- inputTableName : 训练输入表的表名

GBDT二分类

在GBDT回归与排序的基础上，用于二分类问题，既设定阈值：大于阈值为正例，反之为负例。

组件介绍

向画布拖入GBDT二分类组件训练，调整参数，如：



字段设置 参数设置

沿用类型设置节点

选择输入列
已选择 4 个字段
y

选择目标列
nr_employed

选择的输入列可调整字段类型，如：

- 沿用类型设置节点必须在输入表进行类型设置处理后才能勾选（参照随机森林训练和生成模型部分的步骤2,3）
- 输入列：支持double类型与bigint类型，最多支持800列输入
- 标签列：只能选择非输入列的其它列，value只能是二分类，否则会报错，支持bigint类型
- 是否选择分组的列，默认是全表，支持double类型与bigint类型

选择的输入列可调整字段类型，如：

选择字段

		前100采集	测量
<input checked="" type="checkbox"/>	DOUBLE		连续
<input checked="" type="checkbox"/>	euribor3m	[0.659...4.968]	连续
<input checked="" type="checkbox"/>	previous	0.0,1.0,2.0	连续
<input checked="" type="checkbox"/>	cons_conf_idx	[-50.0...-26.9]	连续
<input type="checkbox"/>	pdays	2.0,3.0,4.0,6.0,999.0	连续
<input type="checkbox"/>	emp_var_rate	-3.4,-2.9,-1.8,-1.7,-0.1,1.1,1.4	连续
<input type="checkbox"/>	nr_employed	4991.6,5008.7,5017.5,5076.2...	连续
<input type="checkbox"/>	cons_price_idx	[92.201...94.465]	连续
<input checked="" type="checkbox"/>	BIGINT		连续
<input type="checkbox"/>	campaign	1,2,3,4,5,8,11,12,18,25	连续
<input type="checkbox"/>	y	0/1	离散
<input checked="" type="checkbox"/>	age	[24...72]	连续
<input type="checkbox"/>	STRING		无类型
<input type="checkbox"/>	pt	20150501	无类型

确定 取消

- 系统会对字段类型进行初步判断（离散，连续，无类型）
- GBDT二分类输入列只支持连续型，选择连续型和离散型处理方式一致

调整参数设置，如：

字段设置 参数设置

metric类型

AUC

树的数目

500

学习速率

0.5

最大叶子数

32

树最大深度

11

叶节点最少样本数

500

训练采集样本比例

0.6



- metric类型 : 0(NDCG)- : normalized discounted cumulative gain ; 1(DCG) : discounted cumulative gain ; 2 (AUC) 只适应0/1 label (默认值)
- 树的数目 : 范围[1,10000] , 默认500
- 学习速率 : 范围(0,1] , 默认0.05
- 最大叶子数: 必须为整数 , 范围[2,1000] , 默认32
- 树最大深度 : 必须为整数 , 范围[1,11] , 默认为11
- 叶节点最少样本数 : 必须为整数 , 范围[100,1000] , 默认500
- 训练采集样本比例: 范围(0,1] , 默认0.6
- 训练采集特征比例 : 范围 (0,1] , 默认0.6
- 测试数据比例 : 范围[0,1) , 默认0.0
- 随机数种子 : 必须为整数 , 范围[0,10] , 默认0
- 特征分裂的最大数量 : 范围[1,1000] , 默认500

3.运行结果（见随机森林组件说明）

1. 注意:

- GBDT与GBDT_LR默认损失函数类型不一致 : GBDT 默认认为regression loss:mean squared error loss , GBDT_LR 默认认为logistic regression loss。其中GBDT_LR不需要用户设置损失函数类型 , 系统直接写入默认损失函数。
- GBDT二分类的标签列只能选择二分类列 , 且不支持string型
- 连接ROC曲线时预测组件应该选择自定义并选择目标基准值

PAI 命令 (未沿用类型设置节点)

```
PAI -name GBDT_LR -project algo_public -DfeatureSplitValueMaxSize="500" -DrandSeed="0" \
-Dshrinkage="0.5" -DmaxLeafCount="32" -DlabelColName="y" -DinputTableName="bank_data_partition" \
-DminLeafSampleCount="500" -DgroupIDColName="nr_employed" -DsampRatio="0.6" -DmaxDepth="11" \
-DmodelName="xlab_m_GBDT_LR_21208" -DmetricType="2" -DfeatureRatio="0.6" - \
DinputTablePartitions="pt=20150501" \
-DtestRatio="0.0" -DfeatureColNames="age,previous,cons_conf_idx,euribor3m" -DtressCount="500";
```

- name: 组件名字
- project: (可选)默认project是algo_public。如果选择其它的Project，需要指定algo_public下的算法包，否则报错
- featureSplitValueMaxSize : (可选)一个特征分裂的最大数量，范围[1,1000]，默认500
- randSeed: (可选)随机数种子，必须为整数，范围：[0,10]，默认0
- shrinkage:(可选)学习速率。范围(0-1]，默认0.05
- maxLeafCount : (可选)最大叶子数，必须为整数，范围：[2,1000]，默认32
- labelColName : 输入表中选择的标签列列名
- inputTableName : 训练输入表的表名
- minLeafSampleCount : (可选)叶子节点容纳的最少样本数，必须为整数，范围：[100,1000]，默认500
- groupIDColName: (可选)数据分组列，默认将整个表作为一个组
- sampleRatio : (可选)训练采集样本比例，范围：(0,1]，默认0.6
- maxDepth : (可选)一棵树的最大深度，必须为整数，范围：[1,11]，默认为11
- modelName : 输出的模型名
- metricType : metric类型。 (可选) 0(NDCG)- : normalized discounted cumulative gain ; 1(DCG) : discounted cumulative gain ; 2 (AUC) 只适应0/1 label (默认值)
- featureRatio : (可选)训练中采集的特征比例，范围：(0,1] ，默认0.6
- inputTablePartitions : (可选)预测输入表分区。输入表对应的输入分区，选中全表则为None
- testRatio : (可选)测试样本数比例，范围：[0-1]，默认0.0
- featureColNames : 输入表中用于训练的特征列
- treeCount : (可选)树数量，范围 [1,10000]，默认500

XGBOOST分类

- xgboost是一种高效率boosting算法，适用回归和二分类问题，详见
<https://github.com/dmlc/xgboost>。
- PAI平台目前只开放树形模型 (gbtree)，暂不开放线性模型 (gblinear)，其他参数见下表；原始参数说明 <https://github.com/dmlc/xgboost/blob/master/doc/parameter.md>。
- PAI平台支持稠密和稀疏两种数据格式作为输入。

组件介绍

字段设置（略，参考随机森林）

输入特征和目标特征都必须是int和double类型

算法参数设置

参数名称	参数描述	参数value可选项	默认值
输入是否为稀疏矩阵	是否稀疏数据	true , false	false
更新步长	为了防止过拟合，更新过程中用到的收缩步长	[0-1]	0.3

	。在每次提升计算之后，算法会直接获得新特征的权重。eta通过缩减特征的权重使提升计算过程更加保守		
最小损失衰减	最小损失衰减	[0,无穷]	0
树最大深度	树的最大深度	[1,无穷]	6
模型最小样本数	孩子节点中最小的样本权重和。如果一个叶子节点的样本权重和小于模型最小样本数则拆分过程结束。在线性回归模型中，这个参数是指建立每个模型所需的小样本数	[0,无穷]	1
最大Delta步进	每个树所允许的最大delta步进	[0,无穷]	0
特征采样比例	在建立树时对特征采样的比例	(0,1]	1
求解目标	定义学习任务及相应的学习目标，可选的目标函数	线性回归；逻辑回归	线性回归
初识预测值	初始的predict score	-	0.5
随机数种子	随机数的种子	[0,无穷]	0

pai命令示例

```
pai -name xgboost
-DinputTableName=wpbc
-DfeatureColNames=f1,f2,f3
-DlabelColName=label
-Dobjective=binary:logistic
-DmodelName=algo_adult_binary_model;
```

算法参数

参数key名称	参数描述	参数value可选项	默认值
inputTableName	输入表的表名	-	-
featureColNames	输入表中用于训练的特征的列名	-	-
labelColName	输入表中标签列的列名	-	-
modelName	输出的模型名	-	-
inputPartitions	输入表中指定哪些分区参与训练，格式为：partition_name=value。如果是多级格式为	-	输入表的所有partition

	name1=value1/nam e2=value2 ; 如果是指 定多个分区，中间用 '分隔		
enableSparse	是否稀疏数据	true , false	false
itemDelimiter	item(key value对)之 间的分隔符	冒号、空格、逗号	空格
kvDelimiter	表中每个item的key和 value之间的分隔符	冒号、空格、逗号	冒号
eta	为了防止过拟合，更新 过程中用到的收缩步长 。在每次提升计算之后 ，算法会直接获得新特 征的权重。 eta通过缩 减特征的权重使提升计 算过程更加保守	[0-1]	0.3
gamma	最小损失衰减	[0,无穷]	0
max_depth	树的最大深度	[1,无穷]	6
min_child_weight	孩子节点中最小的样本 权重和。如果一个叶子 节点的样本权重和小于 min_child_weight则 拆分过程结束。在线性 回归模型中，这个参数 是指建立每个模型所需 的最小样本数	[0,无穷]	1
max_delta_step	每个树所允许的最大 delta步进	[0,无穷]	0
subsample	用于训练模型的字样本 占整个样本集合的比例	(0,1]	1
colsample_bytree	在建立树时对特征采样 的比例	(0,1]	1
objective	定义学习任务及相应 的学习目标，可选的目 标函数	reg:linear reg:logistic binary:logistic	reg:linear
base_score	初始的predict score	-	0.5
seed	随机数的种子	[0,无穷]	0

多分类

k近邻

组件介绍

- 对于预测表的每一行，从训练表中选出距离该行最近的K条记录，K条记录中类别数最多的一类作为该行的类别。
- 该算法解决分类问题。
- 该算法只支持稠密数据格式

PAI命令

```
pai -name knn
-DtrainTableName=train_table
-DtrainFeatureColNames=col1,col2,col3
-DtrainLabelColName=label_name
-DpredictTableName=predict_table
-DpredictFeatureColNames=col4,col5,col6
-DoutputTableName=output_table
-Dk=4;
```

算法参数&说明

参数名称	参数描述	参数值可选项	参数默认值
trainTableName	必选，训练表的表名	-	-
trainFeatureColNames	必选，训练表中的特征列名	-	-
trainLabelColName	必选，训练表中标签列的列名	-	-
trainTablePartitions	可选，训练表中指定哪些分区参与训练	-	所有partitions
predictTableName	必选，预测表的表名	-	-
outputTableName	必选，输出表的表名	-	-
predictFeatureColNames	可选，预测表中特征列名	-	默认与trainFeatureColNames相同
predictTablePartitions	可选，预测表中指定哪些分区参与预测	-	所有partitions
appendColNames	可选，输出表中附加预测表的列名	-	默认与predictFeatureColNames相同
outputTablePartition	可选，输出表分区	-	输出表不分区
k	可选，最近邻个数	正整数，[1,1000]	100
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期

逻辑回归

经典逻辑回归是一个二分类算法，算法平台的逻辑回归可以支持多分类。逻辑回归组件支持稀疏、稠密两种数据格式

参数设置

逻辑回归组件的参数

是否二分类

输入是否为稀疏矩阵

最大迭代数

100

收敛误差

0.000001

正则化类型

'l2'

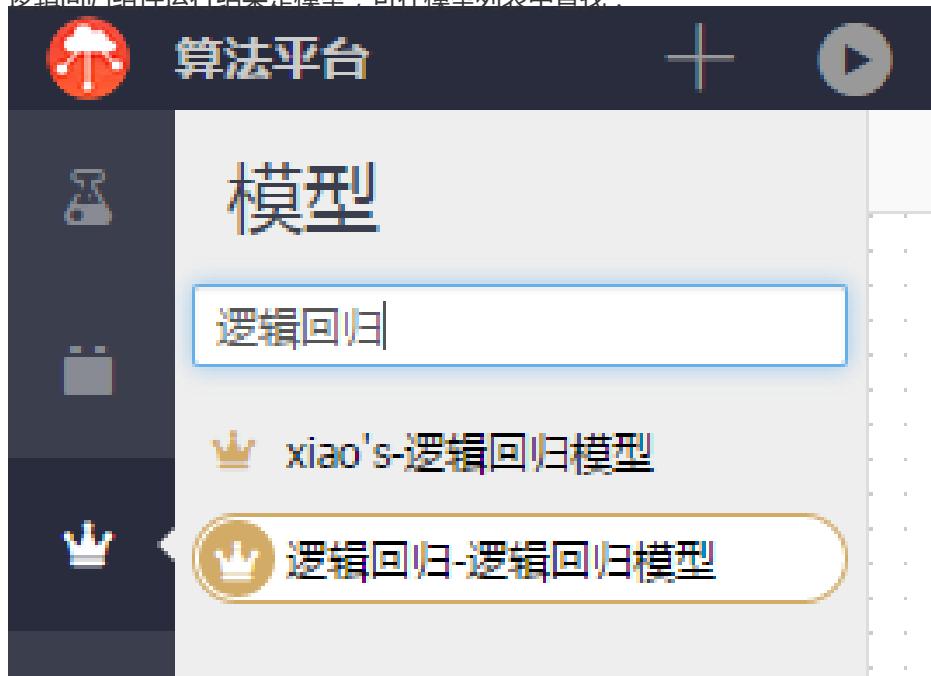


正则化系数

1

- 是否二分类：(可选)默认多分类，勾选后表示选择二分类
- 是否支持稀疏矩阵：组件可支持稀疏矩阵的格式
- 目标基准值：(可选)二分类时，指定训练系数针对的label值；如果填空，会随机选择一个
- 最大迭代数：(可选)L-BFGS的最大迭代次数，默认是100
- 收敛误差：(可选)L-BFGS的终止条件，即两次迭代之间log-likelihood的差，默认为 $1.0e-06$
- 正则化类型：(可选)正则化类型，可以选择'l1'、'l2'、'None'，默认为'l1'
- 正则化系数：(可选)正则项系数，默认为1.0；当 regularizedType 为 None 时，该项会被忽略

逻辑回归组件运行结果是模型，可在模型列表中查找：



模型名称的命名规则：实验名 + " - " + 算法组件名称 + "model"

PAI 命令（未沿用类型设置节点）

```
PAI -name LogisticRegression -project algo_public -DmodelName="xlab_m_logistic_regression_6096" \
-DregularizedLevel="1" -DmaxIter="100" -DregularizedType="l1" -Depsi=on="0.000001" -DlabelColName="y" \
-DfeatureColNames="pdays,emp_var_rate" -DgoodValue="1" -DinputTableName="bank_data";
```

- name: 组件名字
- project: project名字。用于指定算法所在空间，系统默认是algo_public，用户自己更改后系统会报错
- modelName: 输出的模型名
- regularizedLevel : (可选) 正则化系数。默认为 1.0；当 regularizedType 为 None 时，该项会被忽略
- maxIter : (可选) 最大迭代数。指定L-BFGS的最大迭代次数，默认是100
- regularizedType : (可选) 正则化类型，可以选择'l1'、'l2'、'None'，默认为'l1'
- epsilon : (可选) 收敛误差。L-BFGS的终止条件，即两次迭代之间log-likelihood的差，默认为1.0e-06
- labelColName : 输入表标签列列名
- featureColNames : 输入表中选择的用于训练的特征列名
- goodValue : (可选) 目标基准值。二分类时，指定训练系数针对的 label 值；如果填空，会随机选择一个
- inputTableName : 训练输入表的表名

随机森林

随机森林是一个包含多个决策树的分类器，并且其输出的类别是由个别树输出的类别的众数而定。

参数设置

向画布拖入随机森林组件训练，设置字段

字段设置

参数设置

沿用类型设置节点

选择输入列

已选择 8 个字段

选择标签列

campaign

选择权重列

y

或者



字段设置 参数设置

沿用类型设置节点

选择输入列

选择字段

选择标签列

选择权重列

- 沿用类型设置节点必须在输入表进行类型设置处理后才能勾选
- 输入列：支持string类型、double类型与bigint类型
- 标签列：只能选择非输入列的其它列，支持string类型、double类型和bigint类型
- 权重列：(可选) 只能选择非输入列和非标签列，支持double类型和bigint类型

选择的输入列可调整字段类型，如：

选择字段

		前100样本数值范围	数值类型
<input checked="" type="checkbox"/>	DOUBLE		连续
<input checked="" type="checkbox"/>	pdays	2.0,3.0,4.0,6.0,999.0	连续
<input checked="" type="checkbox"/>	previous	0.0,1.0,2.0	连续
<input checked="" type="checkbox"/>	emp_var_rate	-3.4,-2.9,-1.8,-1.7,-0.1,1.1,1.4	连续
<input checked="" type="checkbox"/>	cons_price_idx	[92.201...94.465]	连续
<input checked="" type="checkbox"/>	cons_conf_idx	[-50.0...-26.9]	连续
<input checked="" type="checkbox"/>	euribor3m	[0.659...4.968]	连续
<input checked="" type="checkbox"/>	nr_employed	4991.6,5008.7,5017.5,5076.2,...	连续
<input type="checkbox"/>	BIGINT		数值类型
<input checked="" type="checkbox"/>	age	[24...72]	连续
<input type="checkbox"/>	campaign	1,2,3,4,5,8,11,12,18,25	连续
<input type="checkbox"/>	y	0/1	离散
<input type="checkbox"/>	STRING		数值类型
<input type="checkbox"/>	poutcome	failure,nonexistent,success	离散

确定 取消

- 系统会对字段类型进行初步判断（离散，连续，无类型）
- 随机森林输入列支持连续型和离散型输入，选择的数据类型不一样，计算方式也不一样

设置步骤2拖入的随机森林组件的参数

树的数目 正整数, (0, 1000]

10

单棵树随机特征数 [1, 特征数], -1表示log2N

-1

树最大深度 [0, 无穷), -1 表示无穷

10

叶子节点最少记录数 正整数

2

叶子节点最少记录百分比(%)

默认-1,范围: [0,100]

每棵树最大记录数 (1000, 1000000]

100000

单棵树算法在森林中分布 具体参考帮助文档

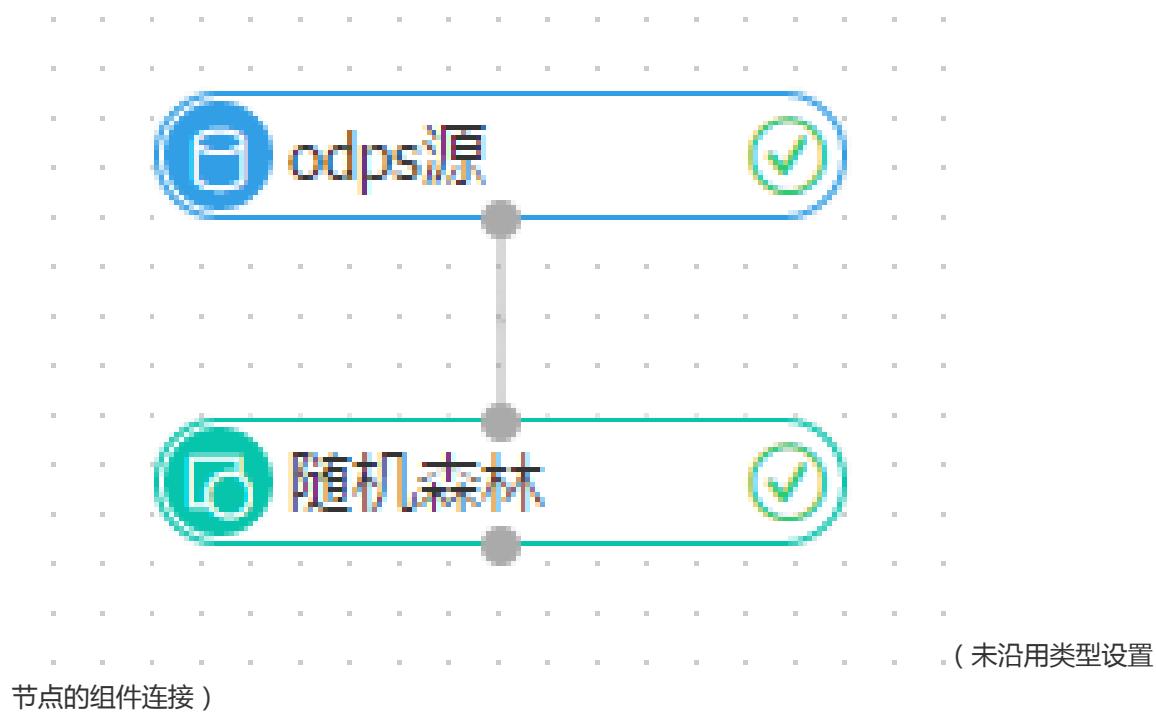
格式为两个递增整数, 例如: 2,4

- 树的数目 : 森林中树的个数, 范围(0, 1000]

- 随机属性类型 : (可选) 单颗树在生成时 , 随机选择的特征个数。可供选择的类型有 logN , N/3 , sqrtN , N四种类型

- 树最大深度：(可选) 单颗树的最大深度。-1表示完全生长。范围[1, ∞)
- 叶子节点最少记录数：(可选) 叶节点数据的最小个数。最小个数为2
- 叶子节点最少记录百分比：(可选) 叶节点数据个数占父节点的最小比例，-1表示无这一限制。默认-1，范围[0,100]
- 每棵树最大记录数：(可选) 森林中单颗树输入的随机数据的个数。范围为(1000, 1000000)
- 单棵树的算法类型：(可选)。设置森林中的每棵树的算法类型，有三种算法类型：id3,c4.5, cart。设置格式必须为[a,b]，a、b为数字；比如有n棵树，algorithmTypes=[a,b]，则[0,a]是id3，[a,b]是cart，[b,n]是c4.5。例如：在一个拥有5棵树的森林中，[2, 4]表示0，1为id3算法，2, 3为cart算法，4为c4.5算法。如果输入为None，则算法在森林中均分。

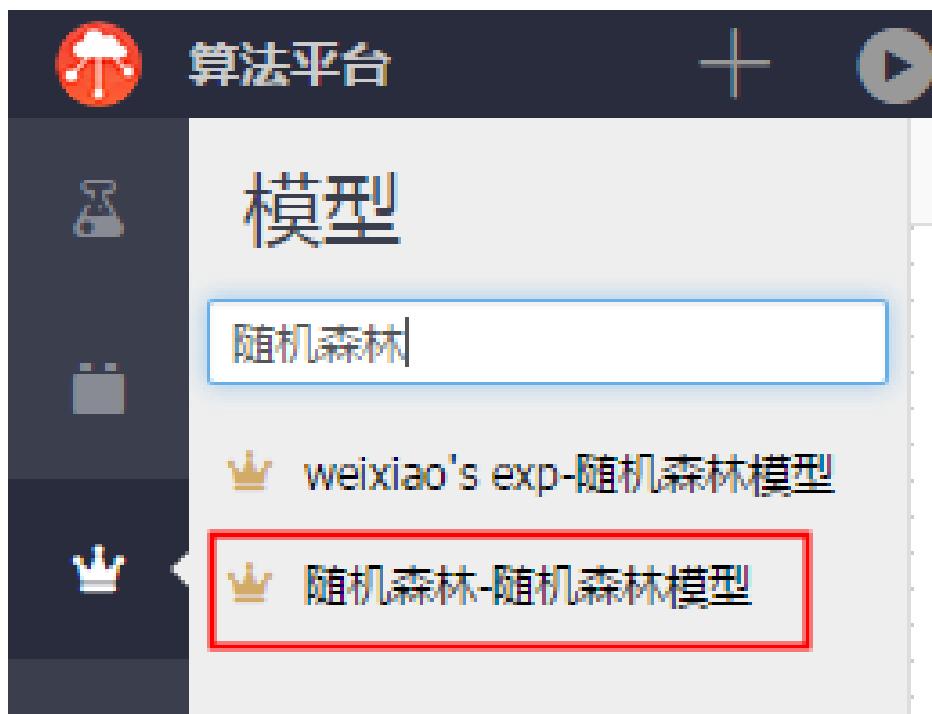
链接所有组件后，点击执行





(沿用类型设置节点的组件连接)

执行结束后可在模型栏看到生成的随机森林模型



模型名称的命名规则：实验名 + "—" + 算法组件名称 + "model"

1. 注意事项：

- 随机森林通过对bagging方法的改进，在大数据集上构建一个单颗树不相关的森林。在很多问题上，随机森林跟boosting方法类似，有类似的训练过程。
- 本方法对于单颗树的增长，可选的有id3，cart或c4.5。通过参数treeNum来指定森林中树的个数，树个数的范围为[1, 1000]。通过编辑好的模版，可以控制单棵树的结构。可以通过其他参数控制单颗树叶结点上数据的最小个数，叶结点上数据占父节点的最小比例数，树的最大深度等。
- 权重列的每一行和样本对应，表示这条样本在训练时占的比重。比如选择了age列作为权重列，那page列中权重高的value所在的条样本在训练时比重会更高
- 当出现input table is empty!错误时，可能是以下情况：抽样比率设置太小，也就是maxRecordSize太小；输入数据为空表。

PAI 命令（未沿用类型设置节点）

```
PAI -name RandomForests -project algo_public -DmodelName="xlab_m_random_forests_6036" \
-DrandomColNum="1.0" -DlabelColName="campaign" -DmaxTreeDeep="10" -DmaxRecordSize="100000" \
-DfeatureColNames="age,pdays,previous,emp_var_rate,cons_price_idx,cons_conf_idx,euribor3m,nr_employed" \
-DisFeatureContinuous="1,1,1,1,1,1,1" -DminNumPer="-1" -DminNumObj="2" - \
DinputTableName="bank_data" \
-DweightColName="y" -DtreeNum="10";
```

- name: 组件名字
- project: project名字。用于指定算法所在空间，系统默认是algo_public，用户自己更改后系统会报错
- modelName: 输出的模型名

- randomColNum: (可选) 随机属性类型。单颗树在生成时，每次随机选择的特征个数。-1表示 $\log_2 N$ (N为特征的总个数)，有效值为[1,N]，其中N为feature 个数
- labelColName: 目标列。输入表中选择的标签列列名
- maxTreeDeep: (可选) 树最大深度。-1表示完全生长。范围[1, ∞)
- maxRecordSize:(可选) 每棵树最大记录数。森林中单颗树输入的随机数据的个数。范围为(1000, 1000000]。-1表示100000
- featureColNames : 特征列。输入表中选择的用于训练的特征列名
- isFeatureContinuous: 特征列类型，表示对应特征是否为连续值。1为连续，0为离散。1,0,0表示三列特征中第一列为连续，第二和第三列为离散，值个数对应features 长度。
- minNumPer:(可选) 叶子节点最少记录百分比，即叶节点数据个数占父节点的最小比例。-1表示无这一限制。范围[0.0,1.0]
- minNumObj:(可选) 叶子节点最少记录数。即叶节点数据的最小个数，最小个数为
- inputTableName: 训练输入表的表名
- weightColName:(可选) 权重列。输入表中权重列的列名，无权重列则为None , weightColName>0.
- treeNum: 树的数目。即森林中树的个数，范围(0, 1000]

朴素贝叶斯

朴素贝叶斯分类是一种应用基于独立假设的贝叶斯定理的简单概率分类算法,更精确的描述这种潜在的概率模型为独立特征模型。 算法详见：[Naive Bayes classifier 组件介绍](#) 组件设置参数（略，参照随机森林组件）

PAI命令

```
PAI -name NaiveBayes -project algo_public -DmodelName="xlab_m_NaiveBayes_23772" \
-DinputTablePartitions="pt=20150501" -DlabelColName="poutcome" \
-DfeatureColNames="age,previous,cons_conf_idx,euribor3m" -DisFeatureContinuous="1,1,1,1" \
-DinputTableName="bank_data_partition";
```

- name: 组件名字
- project: (可选)默认project是algo_public。如果选择其它的Project，需要指定algo_public下的算法包，否则报错
- modelName : 训练生成的模型名
- inputTablePartitions : (可选) 预测输入表分区。输入表对应的输入分区，选中全表则为None
- labelColName : 输入表中选择的标签列列名
- featureColNames : 输入表中用于训练的特征列
- isFeatureContinuous:特征对应的类型，表示对应特征是否为连续值。1为连续，0为离散。1,0,0表示三列特征中第一列为连续，第二和第三列为离散，值个数对应features 长度。
- inputTableName : 输入的表名

聚类

k均值聚类

K均值聚类是一种得到最广泛使用的聚类算法，把n个对象分为k个簇，使簇内具有较高的相似度。相似度的计算根据一个簇中对象的平均值来进行。它与处理混合正态分布的最大期望算法很相似，因为他们都试图找到数据中自然聚类的中心。

算法首先随机地选择k个对象，每个对象初始地代表了一个簇的平均值或中心。对剩余的每个对象根据其与各个簇中心的距离，将它赋给最近的簇，然后重新计算每个簇的平均值。这个过程不断重复，直到准则函数收敛。

它假设对象属性来自于空间向量，并且目标是使各个群组内部的均方误差总和最小。KMeans的更多详细介绍请参考 [相关文档链接](#)

参数设置

[设置参数](#)

字段设置 参数设置

距离计算方式

euclidean

聚类数

10

初始质心位置选择

sample

最大迭代次数

100

最小迭代精度

0

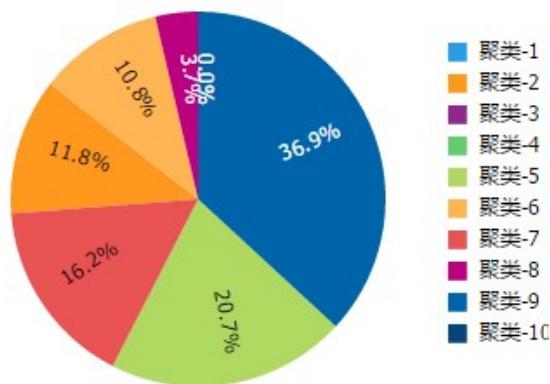
- 距离计算方式：支持欧式距离(euclidean，默认)，绝对误差和(cityblock)，夹角余弦(cosine)
- 聚类数：单位整型，默认是10
- 初始质心位置选择：支持sample(随机选取，默认)，topk(前K行)，uniform(分布范围均匀随机生成)，matrix(需指定表作为初始质心位置)，kmpp (kmeans++初始化)。
- 最大迭代次数：计算最大迭代次数，默认为100
- 最小迭代精度：计算最小迭代精度，默认为0.0

运行后，查看输出，可以查看饼状图和聚类质心表 (centerTable)

K-means

x

饼图 表格



最小聚类大小	0(0.00%)
最大聚类大小	15183(36.86%)
大小比率(最大聚类比最小聚类)	Infinity

关闭

K-means

x

饼图 表格

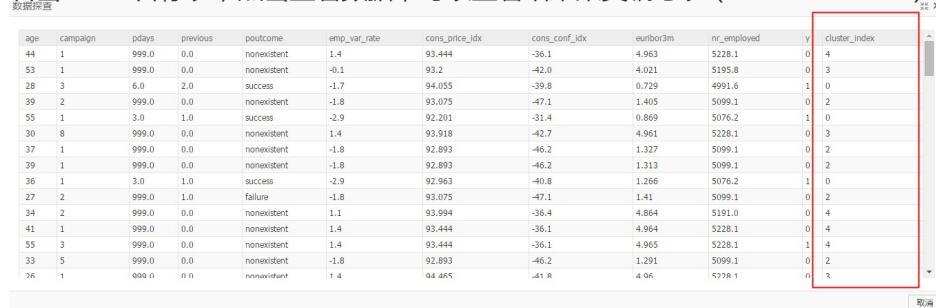
序号	cons_conf_idx	emp_var_rate	euribor3m	pdays	previo
1	-42.32996845...	1.400000000000...	4.96744164037...	999	
2	-41.532412940...	-0.6796826705...	3.31051535132...	999	0.241
3	-46.817128383...	-1.7640335602...	1.32882839955...	980.466360614...	0.060
4	-41.709602313...	0.27042660882...	4.04805813449...	956.225885755...	0.216
5	-46.669089841...	-1.7811861421...	1.33412166764...	999	0.304
6	-41.786612576...	1.37870182555...	4.93101487491...	999	
7	-42.700000000...	1.39999999999...	4.96158683620...	999	
8	-38.341386138...	-2.0962376237...	0.98594917491...	6.01452145214...	1.660
9	-35.588032668...	0.62816966343...	4.29341467430...	999	0.055
10	-35.486934111...	0.45705148109...	4.09411445072...	951.145314537...	0.138

< >

关闭

- 聚类质心表：列数等于输入表列数(有选中则等于选中列总数)，行数等于聚类数，每行表示一个聚类质心位置

右击ODPS目标表，点击查看数据，可以查看结果聚类编号表 (idxTablename)



age	campaign	pdays	previous	poutcome	emp_var_rate	cons_price_idx	cons_conf_idx	euribor3m	nr_employed	y	cluster_index
44	1	999.0	0.0	nonexistent	1.4	93.444	-36.1	4.963	5228.1	0	4
53	1	999.0	0.0	nonexistent	-0.1	93.2	-42.0	4.021	5195.8	0	3
28	3	6.0	2.0	success	-1.7	94.055	-39.8	0.729	4991.6	1	0
39	2	999.0	0.0	nonexistent	-1.8	93.075	-47.1	1.405	5099.1	0	2
55	1	3.0	1.0	success	-2.9	92.201	-31.4	0.869	5076.2	1	0
30	8	999.0	0.0	nonexistent	1.4	93.918	-42.7	4.961	5228.1	0	3
37	1	999.0	0.0	nonexistent	-1.8	92.893	-46.2	1.327	5099.1	0	2
39	1	999.0	0.0	nonexistent	-1.8	92.893	-46.2	1.313	5099.1	0	2
36	1	3.0	1.0	success	-2.9	92.963	-40.8	1.266	5076.2	1	0
27	2	999.0	1.0	failure	-1.8	93.075	-47.1	1.41	5099.1	0	2
34	2	999.0	0.0	nonexistent	1.1	93.994	-36.4	4.864	5191.0	0	4
41	1	999.0	0.0	nonexistent	1.4	93.444	-36.1	4.964	5228.1	0	4
55	3	999.0	0.0	nonexistent	1.4	93.444	-36.1	4.965	5228.1	1	4
33	5	999.0	0.0	nonexistent	-1.8	92.893	-46.2	1.291	5099.1	0	2
76	1	999.0	0.0	nonexistent	1.4	94.465	-41.8	4.06	5228.1	0	3

- 聚类编号表：行数等于输入表总行数，每行的值表示输入表对应行表示的点的聚类编号
- 会显示所有列名，并在最后追加一个分类标示字段
- 其中显示的0,1,2,3为分类标识
- 聚类质心表、聚类编号表及聚类个数表还可以通过pai命令生成的表名在IDE里查看

注意事项

- 若初始质心位置选择matrix，初始质心表需要用户自己定义，定义的新表列和原始表相同，行数是聚类数。做表的时候要定好k个质心，可以用sql，mr 采样，或者根据自己需要选其他的模式

PAI命令

```
PAI -name KMeans -project algo_public -DcenterCount="10" -DidxTableName="bank_data_index" \
-DdistanceType="euclidean" -DappendColsIndex="0,1,2,3,4,5,6,7,8,9,10" \
-DcenterTableName="pai_temp_3300_27298_3" -Dloop="100" - \
DclusterCountTableName="pai_temp_3300_27298_2"\ \
-DinitCentersMethod="sample" -Daccuracy="0.0" -DinputTableName="bank_data" \
-DselectedColNames="cons_conf_idx,emp_var_rate,euribor3m,pdays,previous";
```

- name: 组件名字
- project: (可选) 默认project是algo_public。如果选择其它的Project，需要指定algo_public下的算法包，否则报错
- centerCount: 聚类数，单位整型。默认是10
- idxTableName: 输出聚类编号表。行数等于输入表总行数，每行的值表示输入表对应行表示的点的聚类编号
- distanceType: (可选) 距离测度方法。支持欧式距离(euclidean ,)，绝对误差和(cityblock)，夹角余弦(cosine)。默认欧式距离(euclidean ,)
- appendColsIndex : (可选) 输出表附加的id列名，默认输出表不附加id列
- centerTableName: 输出聚类质心表，列数等于输入表(有选中则等于选中列总数)，行数等于聚类数，每行表示一个聚类质心位置
- loop: (可选) 计算最大迭代次数，整型，默认为100
- clusterCountTableName: 输出聚类点总数表名，行数等于聚类数，每行聚类质心的类中所包含的聚类点的总数
- initCentersMethod: (可选) 初始质心位置选择，支持sample(随机选取)，topk(前K行)，uniform(分布范围均匀随机生成)，matrix(指定表作为初始质心位置)，kmpp (kmeans++ 初始化)

)。默认sample

- accuracy : 计算最小迭代精度，默认为0.0
- inputTableName: 输入表的名字
- selectedColNames : 输入表选择列，逗号分隔。只支持double型输入
- initCenterTableName : (可选) 输入起始center值的表格，当initCentersMethod为matrix时候必须输入，否则不用填。注意：若初始质心位置选择matrix，初始质心表需要用户自己定义，定义的新表列和原始表相同，行数是聚类数。做表的时候要定好k个质心，可以用sql，mr 采样，或者根据自己需要选其他的模式

回归

线性回归

功能介绍

- 将输入表的数据，选择某些列作为feature，某一列作为label，进行线性回归训练，产生线性回归模型。
- 该算法解决回归问题。
- 该算法支持稠密和稀疏两种数据格式作为输入。

pai命令示例

```
pai -name linearregression
-DinputTableName=wpbc
-DfeatureColNames=f1,f2,f3
-DlabelColName=label
-DmodelName=linear_model;
```

算法参数

参数名称	参数描述	参数值可选项	默认值
inputTableName	必选，输入表的表名	-	-
featureColNames	必选，输入表中用于训练的特征的列名	-	-
labelColName	必选，输入表中标签列的列名	-	-
modelName	必选，输出的模型名	-	-
inputTablePartitions	可选，输入表中指定哪些分区参与训练，格式为: Partition_name=value。如果是多级格式为 name1=value1/name2=value2；如果是指定多个分区，中间用 ' ' 分开	-	输入表的所有partition

maxIter	可选，最大迭代次数	-	100
epsilon	可选，终止条件，即两次迭代之间log-likelihood的差	-	1.0e-06
regularizedType	可选，正则化类型	I1 , I2 , None	None
regularizedLevel	可选，正则项系数,当regularizedType 为None 时，该参数会被忽略	(0, ∞)	1.0
enableSparse	可选，输入表数据是否为稀疏格式	true , false	false
itemDelimiter	可选，当输入表数据为稀疏格式时，kv间的分割符	-	空格
kvDelimiter	可选，当输入表数据为稀疏格式时，key和value的分割符	-	冒号

GBDT回归

GBDT--梯度渐进回归树，是一种迭代的决策树算法，该算法由多棵决策树组成，所有树的结论累加起来做最终答案。GBDT几乎可用于所有回归问题（线性/非线性），相对逻辑回归仅能用于线性回归，GBDT的适用面非常广。参考论文: (a) A Regression Framework for Learning Ranking Functions Using Relative Relevance Judgments , (b) From RankNet to LambdaRank to LambdaMART: An Overview

组件介绍

1.字段设置：

- 输入列：仅支持double类型与bigint类型,最多支持800列输入
- 标签列：仅支持double类型与bigint类型
- 是否指定分组列为可选项，勾选后可选择非输入列和非标签列的其它列作为数据分组列。不勾选时默认将整个表作为一个组，支持double类型与bigint类型

调整参数设置，如：

字段设置 参数设置

损失函数类型

GBRANK LOSS

metric类型

AUC

树的数目

500

学习速率

0.05

最大叶子数

32

树最大深度

11

叶节点最少样本数

500

训练采集特征比例

0.6

测试数据比例

0

Tau参数(Gbrank)

0.6

指数底数(p, regression loss与gbrank l...

1

随机数种子

0

使用newton方法来学习

否



特征分裂的最大数量

500

- 损失函数类型：默认是 regression loss, 选择类型：GBRANK LOSS论文，DCG LOSS论文，NDCG LOSS论文，REGRESSION LOSS : mean squared error loss。若字段设置时没有指定分组列，则损失函数类型默认regression loss。
- metric类型：0(NDCG)- : normalized discounted cumulative gain ; 1(DCG) : discounted cumulative gain ; 2 (AUC) 只适应0/1 label (默认值)
- 树的数目：范围[1,10000]，默认500
- 学习速率：范围(0 , 1],默认0.05
- 最大叶子数: 必须为整数，范围[2,1000]，默认32
- 树最大深度：必须为整数，范围[1,11]，,默认为11
- 叶节点最少样本数：必须为整数，范围[100,1000]，默认500
- 训练采集样本比例: 范围(0,1]，默认0.6
- 训练采集特征比例：范围 (0,1] , 默认0.6
- 测试数据比例：范围[0,1]，默认0.0
- Tau参数 (Gbrank) : gbrank loss中的Tau参数，范围[0,1]，默认0.6
- 指数底数(p,regression loss与gbrank loss) : gbrank与regression loss中得指数底数，默认1。如果p>1，将样本的label映射为p&circ label；当p<=1时，不做映射，保持原来的label。范围[1,5]
- 随机数种子：必须为整数，范围[0,10]，默认0
- 使用newton方法来学习：范围 0 (不用) , 1 (使用)。默认0
- 特征分裂的最大数量：范围[1,1000]，默认500

注意事项

- 模型训练时，对于同样的训练集和特征集，树的数目不同或随机数种子数不同均会产生不同的采样训练集和feature集。
- 在生成每棵树的时候，会重新set种子
- 分组列标识每条样本所属的分组，分组列是固定的，与选用feature无关。另外，选择数据分组列时，建议用户用已知分组信息的列作为分组列，不建议选择不熟悉的列。选择分组时选用gbrank排序算法逻辑，理论上会使结果更好。
- 只有选择分组列后，损失函数类型才可供用户选择，默认情况下损失函数类型为 : regression loss:mean squared error loss
- GBDT与GBDT_LR默认损失函数类型不一致：GBDT 默认为regression loss:mean squared error loss , GBDT_LR 默认为logistic regression loss。其中GBDT_LR不需要用户设置损失函数类型，系统直接写入默认损失函数。
- 叶节点最小样本数应小于数据条数
- GBDT回归与排序模型预测时不支持自定义分类类型，且不支持做ROC曲线

PAI 命令（未沿用类型设置节点）

```
PAI -name GBDT -project algo_public -DfeatureSplitValueMaxSize="500" \
-DlossType="0" -DrandSeed="0" -DnewtonStep="0" -Dshrinkage="0.05" -DmaxLeafCount="32" \
-DlabelColName="campaign" -DinputTableName="bank_data_partition" -DminLeafSampleCount="500" \
-DsampleRatio="0.6" -DgroupIDColName="age" -DmaxDepth="11" -DmodelName="xlab_m_GBDT_83602" \
-DmetricType="2" -DfeatureRatio="0.6" -DinputTablePartitions="pt=20150501" -Dttau="0.6" -Dp="1" \
-DtestRatio="0.0" -DfeatureColNames="previous,cons_conf_idx,euribor3m" -DtreeCount="500";
```

- name: 组件名字
- project: (可选)默认project是algo_public。如果选择其它的Project，需要指定algo_public下的算法包，否则报错
- featureSplitValueMaxSize : (可选)一个特征分裂的最大数量，范围[1,1000]，默认500
- lossType : (可选)损失函数类型：默认是 0-regression loss, 若在字段设置里没有设置分组列，则只能选择regression loss。选择类型：0-GBRANK LOSS论文，1-DCG LOSS论文，2-NDCG LOSS论文，3-REGRESSION LOSS : mean squared error loss
 - groupIDColname没有指定时，则只能选择regression loss。
- randSeed: (可选)随机数种子，必须为整数，范围：[0,10]，默认0
- newtonStep : 是否使用newton方法来学习，默认0。范围：0-不用，1-使用
- shrinkage: (可选)学习速率。范围(0,1]，默认0.05
- maxLeafCount : (可选)最大叶子数，必须为整数，范围：[2,1000]，默认32
- labelColName : 输入表中选择的目标标签列列名
- inputTableName : 训练输入表的表名
- minLeafSampleCount : (可选)叶子节点最少样本数，必须为整数，范围：[100,1000]，默认500
- sampleRatio : (可选)训练采集样本比例，范围：(0,1]，默认0.6
- groupIDColName: (可选)选择分组列，默认将整个表作为一个组
- maxDepth : (可选)一棵树的最大深度，必须为整数，范围：[1,11]，默认为11
- modelName : 输出的模型名
- metricType : metric类型。 (可选) 0(NDCG)- : normalized discounted cumulative gain ; 1(DCG) : discounted cumulative gain ; 2(AUC) 只适应0/1 label (默认值)
- featureRatio : (可选)训练中采集的特征比例，范围：(0,1]，默认0.6
- inputTablePartitions : (可选)预测输入表分区。输入表对应的输入分区，选中全表则为None
- tau : (可选) Tau参数。gbrank loss中的Tau参数，默认0.6，范围：[0,1]
- p : (可选)指数底数。gbrank与regression loss中的指数底数，默认1如果p > 1,将样本的label映身为p^label；当p <= 1时，不做映射，保持原来的label.范围：[1,5]
- testRatio : (可选)测试数据比例，范围：[0-1]，默认0.0
- featureColNames : 输入表中用于训练的特征列
- treeCount : (可选)树数量，范围 [1,10000]，默认500

xgboost回归

组件介绍 (略, 详见XGBoost分类)

关联推荐

协同过滤etrec

功能介绍

- 将用户-物品表或物品表作为输入，计算物品间的相似度。

dai命令示例

```
pai -name etrec
-DinputTableName=user_item
-DoutputTableName=item_similarity;
```

算法参数

参数名称	参数描述	参数值可选项	默认值
inputTableName	必选，输入表的表名	-	-
selectedColNames	可选，输入表中用于训练的列名	-	输入表的所有列
inputTablePartitions	可选，输入表中指定哪些分区参与训练，格式为: partition_name=value。如果是多级格式为 name1=value1/name2=value2；如果是指定多个分区，中间用';'分开	-	输入表的所有partition
outputTableName	必选，输出的表名	-	-
inputTableFormat	可选，输入表的格式，支持user-item-payload格式的表，也支持把某一个user的所有item合并到一行的表	user-item,items	user-item
similarityType	可选，相似度类型	wbcosine,asymcosine,jaccard	wbcosine
topN	可选，输出结果中最多保留多少个相似物品	正整数，[1,10000]	2000
minUserBehavior	可选，当用户的物品数小于此值时，忽略该用户的行为	正整数，[2,∞)	2
maxUserBehavior	可选，当用户的物品数大于此值时，忽略该用户的行为	正整数，[2,100000]	500
itemDelimiter	可选，输出表不同物品间的分割符，当输入表格式为items时，也是输入表物品间的分隔符	-	空格
kvDelimiter	可选，输出表物品id与相似度间的分割符，当输入表格式为items时，也是输入表物品与payload的分隔符	-	冒号
alpha	可选，当similarityType为	double值	0.5

	asymcosine类型时，平滑因子的值		
weight	可选，当similarityType为asymcosine类型时，权重指数	double值	1.0
operator	可选，当同一user的某个物品出现多次时，payload的计算行为	add,mul,min,max	add
lifecycle	可选，输出表的生命周期	正整数	没有生命周期

输入表格式

输入表支持两种格式，一种是表的每一行对应一个用户-物品-payload，对应的inputTableFormat为user-item，如下表所示：

user	item	payload
user1	item1	1.0
user1	item2	2.0
user1	item3	3.0
user2	item1	1.0
user2	item3	2.0
user2	item4	3.0
user3	item2	1.0
user3	item3	2.0
user3	item4	3.0

另一种是表的每一行包含了某一用户的所有物品与payload，对应的inputTableFormat为items，如下表所示：

item_payload
item1:1 item2:2 item3:3
item1:1 item3:2 item4:3
item2:1 item3:2 item4:3

输出表格式

输出表有两列，第一列为物品id，第二列为与之相似的物品及相似度，如下表所示：

itemid	similarity
item2	item3:0.816497 item1:0.5 item4:0.5

item1	item3:0.816497 item2:0.5 item4:0.5
item4	item3:0.816497 item2:0.5 item1:0.5
item3	item1:0.816497 item2:0.816497 item4:0.816497

评估

混淆矩阵

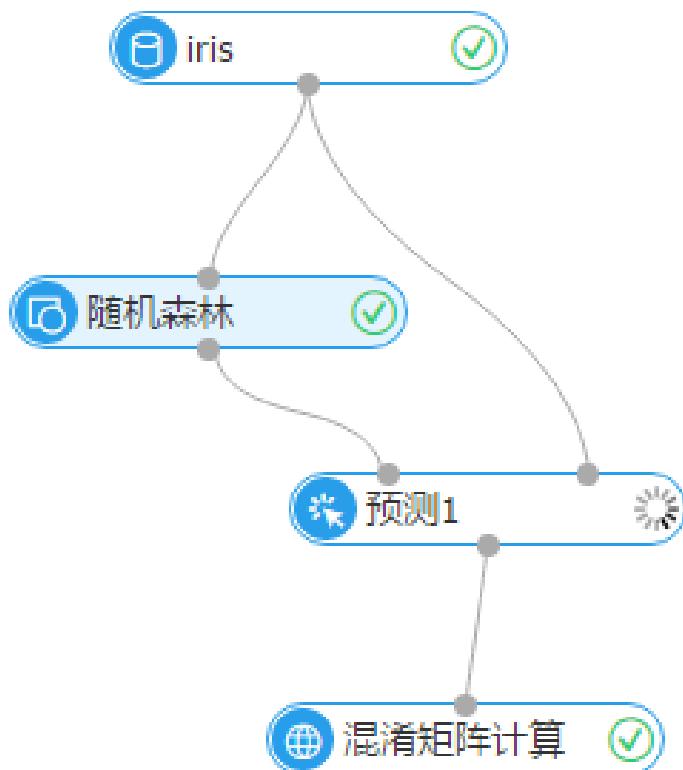
混淆矩阵（confusion matrix）是可视化工具，特别用于监督学习，在无监督学习一般叫做匹配矩阵，主要用于比较分类结果和实际测得值，可以把分类结果的精度显示在一个混淆矩阵里面。

混淆矩阵参数如下图：



- 一般情况下采用默认沿用设置，用户也可自行选择目标列和预测概率列，预测概率列指预测组件生成预测目标列名

混淆矩阵组件上层节点只能为预测组件；只能分类模型才可进行混淆矩阵分析



1. 右击混淆矩阵组件-->查看评估报告，如下



A screenshot of a modal window titled '混淆矩阵' (Confusion Matrix). The window displays a table for the '预测1' component, comparing predicted classes against actual classes for three Iris species. The table includes columns for Model, Correct Count, Error Count, Total, Accuracy, Recall, and F1 Score.

模型	正确个数	错误个数	总计	正确率	召回率	F1指标
Iris-versicolor	50	2	52	96.154%	100.000%	98.039%
Iris-virginica	48	0	48	100.000%	96.000%	97.959%
Iris-setosa	50	0	50	100.000%	100.000%	100.000%

PAI命令

```
PAI -name confusionmatrix -project algo_public -DoutputTableName="pai_temp_2954_24178_1" \
-DlabelColName="age" -DpredictionColName="prediction_result" - \
DinputTableName="pai_temp_2954_24176_1";
```

- name: 组件名字
- project: (可选)默认project是algo_public。如果选择其它的Project，需要指定algo_public下的算法包，否则报错
- outputTableName : 混淆矩阵结果表
- labelColName : 输入表的标签列列名

- predictionColName : 预测结果列
- inputTableName : 输入预测结果表名

回归模型评估

- 基于预测结果和原始结果，评价回归算法模型的优劣，指标包括SST, SSE, SSR, R2, R, MSE, RMSE, MAE, MAD, MAPE, count, yMean和predictMean

PAI 命令

```
pai -name regression_evaluation -project algo_public
-DinputTableName=input_table
-DyColName=y_col
-DpredictionColName=prediction_col
-DoutputTableName=output_table;
```

参数名称	参数描述	参数值可选项	默认值
inputTableName	必选，输入表的表名	-	-
inputTablePartitions	可选，输入表中指定参与计算的分区	-	输入表的所有 partitions
yColName	必选，输入表原始因变量列名，数值类型	-	-
predictionColName	必选，预测结果因变量列名，数值类型	-	-
outputTableName	必选，输出表表名	-	-
inputTablePartitions	可选，输入表选择的分区	-	-
lifecycle	可选，输出表的生命周期	-	默认不设置

输出结果

输出结果是一个json，json个字段描述

字段名称	描述
SST	总平方和
SSE	误差平方和
SSR	回归平方和
R2	判定系数
R	多重相关系数
MSE	均方误差
RMSE	均方根误差

MAE	平均绝对误差
MAD	平均误差
MAPE	平均绝对百分误差
count	行数
yMean	原始因变量的均值
predictionMean	预测结果的均值

多分类评估

算法参数

参数名称	参数描述	参数可选项	参数默认值
inputTableName	必选，输入表的表名	-	-
inputTablePartitions	可选，输入表中指定参与计算的分区	-	输入表的所有 partitions
labelColName	必选，输入表原始 label列名，数值类型	-	-
predictionColName	必选，预测结果 label列名，数值类型	-	-
outputTableName	必选，输出表表名	-	-
predictionColName	可选，预测结果的概率列，格式如 : {"A":0.2,"B":0.3}	-	-
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期

二分类评估

- 基于预测结果和原始结果，评价回归算法模型的优劣，指标包括MSE、MAE、MAPE

pai命令示例

```
pai -name evaluation
-DinputTableName=input_table
-DlabelColName=label_name
-DpredictionColName=prediction_score
-DoutputTableName=output_table;
```

算法参数

参数名称	参数描述	参数可选项	参数默认值
inputTableName	必选，输入表的表名	-	-

inputTablePartitions	可选，输入表中指定参与计算的分区	-	输入表的所有 partitions
labelColName	必选，输入表原始 label列名，数值类型	-	-
predictionColName	必选，预测结果 label列名，数值类型	-	-
outputTableName	必选，输出表表名	-	-
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期

输出表

字段名称	描述
MSE	均方误差，衡量平均误差，可以评价数据的变化程度，MSE越小，说明预测模型描述实验数据具有更好的精确度
MAE	平均绝对误差，衡量预测值与真实值之间平均相差多大
MAPE	平均绝对百分误差，衡量预测精度，值为百分比。举例：若MAPE值为15，则表示平均绝对百分误差为15%

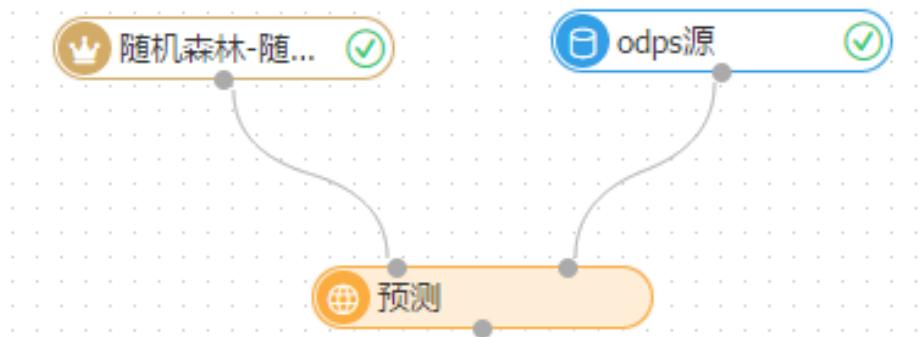
其它

预测

预测组件是专门用于模型预测的组件，两个输入：训练模型和预测数据；输出为预测结果；传统的数据挖掘算法一般都采用该组件进行预测操作

组件介绍

链接所有组件,如下图：



参数说明

特征列：用于预测的数据输入特征，默认为全选；

原样输出列：在输出结果中，本参数所选择的字段将一同输出；

输出结果列名、输出分数列名、输出详细列名均取默认值即可；

若数据是稀疏格式，需要选择稀疏矩阵的特征列，稀疏的格式见“格式说明”，如下图，

特征列 默认全选

原样输出列 推荐添加label列,方便评估

输出结果列名

`prediction_result`

输出分数列名

`prediction_score`

输出详细列名

`prediction_detail`

稀疏矩阵 格式: k1:v1, k2:v2

执行结束后，右键点击预测组件 --> 查看评估报告，如下

数据探查

age	campaign	pdays	previous	poutcome	emp_var_rate	cons_price_idx	cons_conf_idx	euribor3m	nr_employed	y	prediction_result	prediction_score	prediction_detail
44	1	999.0	0.0	nonexistent	1.4	93.444	-36.1	4.963	5228.1	0	1	1.0	{"1": 1}
53	1	999.0	0.0	nonexistent	-0.1	93.2	-42.0	4.021	5195.8	0	1	1.0	{"1": 1}
28	3	6.0	2.0	success	-1.7	94.055	-39.8	0.729	4991.6	1	1	0.9	{"1": 0.9, "2": 0.1}
39	2	999.0	0.0	nonexistent	-1.8	93.075	-47.1	1.405	5099.1	0	1	1.0	{"1": 1}
55	1	3.0	1.0	success	-2.9	92.201	-31.4	0.869	5076.2	1	1	0.6	{"1": 0.6, "2": 0.4}
30	8	999.0	0.0	nonexistent	1.4	93.918	-42.7	4.961	5228.1	0	1	1.0	{"1": 1}
37	1	999.0	0.0	nonexistent	-1.8	92.893	-46.2	1.327	5099.1	0	1	1.0	{"1": 1}
39	1	999.0	0.0	nonexistent	-1.8	92.893	-46.2	1.313	5099.1	0	1	1.0	{"1": 1}
36	1	3.0	1.0	success	-2.9	92.963	-40.8	1.266	5076.2	1	1	0.6	{"1": 0.6, "2": 0.4}
27	2	999.0	1.0	failure	-1.8	93.075	-47.1	1.41	5099.1	0	1	1.0	{"1": 1}
34	2	999.0	0.0	nonexistent	1.1	93.994	-36.4	4.864	5191.0	0	1	1.0	{"1": 1}
41	1	999.0	0.0	nonexistent	1.4	93.444	-36.1	4.964	5228.1	0	1	1.0	{"1": 1}
55	3	999.0	0.0	nonexistent	1.4	93.444	-36.1	4.965	5228.1	1	1	1.0	{"1": 1}
33	5	999.0	0.0	nonexistent	-1.8	92.893	-46.2	1.291	5099.1	0	1	1.0	{"1": 1}
26	1	999.0	0.0	nonexistent	1.4	94.465	-41.8	4.96	5228.1	0	1	1.0	{"1": 1}

预测的数据中，会新增三列：

- predict_result: 预测结果列；
- predict_score : 预测结果概率得分；仅模型为二分类时有效
- predict_detail : 每个类别的预测结果；仅模型为二分类时有效

PAI 命令

```
PAI -name Prediction -project algo_public -DdetailColName="prediction_detail" -DsplitCharacteristic="2" \
-DappendColNames="age,campaign,pdays,previous,poutcome,emp_var_rate,cons_price_idx,cons_conf_idx,euribor3m,nr_employed,y" \
-DmodelName="xlab_m_random_forests_6036" -DresultColName="prediction_result" - \
DoutputTableName="pai_temp_675_6048_1" \
-DscoreColName="prediction_score" -DinputTableName="bank_data";
```

- name: 组件名字
- project: project名字。用于指定算法所在空间，系统默认是algo_public，用户自己更改后系统会报错
- detailColName : (可选) 详细信息列。输出表中detail列名，默认prediction_detail
- splitCharacteristic : (可选) 是否二分类。模型对应的分类类型，1代表二分类，2代表多分类。
- appendColNames : (可选) 向输出表追加列。输出表中需要增加的预测输入表的列
- modelName : 随机森林模型
- resultColName : (可选) 结果列名。输出表中Result列名, 默认 prediction_result
- outputTableName : 预测输出表的表名
- scoreColName : (可选) 评分列名。输出表中Score列名，默认prediction_score
- inputTableName : 预测输入表的表名

文本分析

阿里分词

- 基于AliWS(Alibaba Word Segmenter的简称)词法分析系统，对指定列对应的文章内容进行分词，分词后的各个词语间以空格作为分隔符，若用户指定了词性标注或语义标注相关参数，则会将分词

结果、词性标注结果和语义标注结果一同输出，其中词性标注分隔符为"/"，语义标注分隔符为"|"。目前仅支持中文淘宝分词和互联网分词。

功能介绍

字段设置（略）

参数设置

分词算法：CRF,UNIGRAM

识别选项：分词中，是否识别特殊意义的名词；

合并选项：将具有特殊领域的名词作为整体，不进行切分操作

字符串切分长度：>=0,数字串按指定长度进行截断作为检索单元；默认为0,不对数字串进行长度切分

使用词频纠错：是否使用纠错词典；

标注词性：输出结果中，标注词性

实例介绍

输入包含两列的表，第一列是文档id，第二列是文档内容text,如下：

数据探查	
id	text
1	基原田鼠“胎畜”竟交配治癒？可千万别播什么灵魂伴侣了！雄性基原田鼠在交配之前压得儿就不知道它们要和谁生孩子，因为它们根本分不清谁是谁。不过研究发现，当它们交配之后，基原田鼠辨别异性的能力会大大增强，这种技能帮助它们更容易识别配偶。那么成为更浪漫的骗子。
2	学术讲座：知识问答的問題与挑战 主讲人：中科院自动化所 刘德利研究员 时间：2015年9月11日（周五）上午10:00-11:30 地点：北部教三611室 本报告将主要介绍知识问答的主流方法，同时介绍大规模开放知识库问答所遇到的问题、挑战与对策。欢迎大家参加！
3	寒江清：有些学者为了突出论文，多方收集资料，这属于地道方面，下—篇可归语言学方面，属于“游击战”，没有主要研究方向。记得首次开学术会议时俺们听完某少壮派学者的报告后，好多人都评论道“真搞不懂他这研究在于啥。”
4	【轨道交通4号线初步设计获批】根据交委发委批复，轨道交通4号线工程起于慈城站，终于东钱湖站，全长约35.95公里，共设25座车站，其中地下车站18座，高架车站7座，其中含换乘站5座。在车辆设置上，4号线工程采用6辆编组的B型快线车，与目前1、2号线所采用的钢轮钢轨一致。
5	地铁1号线二期工程穿跨鄞州东部和北仑城区，全长约23.4公里，设车站9座，分别是邱隘站、五乡站、宝幢站、郭隘站、大碶站、松花江路站、中河路站、长江西路站、霞浦站。1号线二期开通后，将和一期贯通运营，届时可从鄞州西到北仑霞浦。与即将开通的2号线一期构成十字网。
6	【路透】麾巴油拒绝中企提出的收购其铁路业务要约】路透看到的文件显示，加拿大麾巴拒绝了北京市基础设施投资有限公司提出的收购其铁路业务的要约。路透看到的日期为8月14日的信函显示，英控提出收购麾巴逾60-100%股权。

关闭

输出结果如下：

数据探查	
id	text
1	基原田鼠“胎畜”竟交配治癒？可千万别播什么灵魂伴侣了！雄性基原田鼠在交配之前压得儿就不知道它们要和谁生孩子，因为它们根本分不清谁是谁。不过研究发现，当它们交配之后，基原田鼠辨别异性的能力会大大增强，这种技能帮助它们更容易识别配偶。那么成为更浪漫的骗子。
2	学术讲座：知识问答的問題与挑战 主讲人：中科院自动化所 刘德利研究员 时间：2015年9月11日（周五）上午10:00-11:30 地点：北部教三611室 本报告将主要介绍知识问答的主流方法，同时介绍大规模开放知识库问答所遇到的问题、挑战与对策。欢迎大家参加！
3	寒江清：有些学者为了突出论文，多方收集资料，这属于地道方面，下—篇可归语言学方面，属于“游击战”，没有主要研究方向。记得首次开学术会议时俺们听完某少壮派学者的报告后，好多人都评论道“真搞不懂他这研究在于啥。”
4	【轨道交通4号线初步设计获批】根据交委发委批复，轨道交通4号线工程起于慈城站，终于东钱湖站，全长约35.95公里，共设25座车站，其中地下车站18座，高架车站7座，其中含换乘站5座。在车辆设置上，4号线工程采用6辆编组的B型快线车，与目前1、2号线所采用的钢轮钢轨一致。
5	地铁1号线二期工程穿跨鄞州东部和北仑城区，全长约23.4公里，设车站9座，分别是邱隘站、五乡站、宝幢站、郭隘站、大碶站、松花江路站、中河路站、长江西路站、霞浦站。1号线二期开通后，将和一期贯通运营，届时可从鄞州西到北仑霞浦。与即将开通的2号线一期构成十字网。
6	【路透】麾巴油拒绝中企提出的收购其铁路业务要约】路透看到的文件显示，加拿大麾巴拒绝了北京市基础设施投资有限公司提出的收购其铁路业务的要约。路透看到的日期为8月14日的信函显示，英控提出收购麾巴逾60-100%股权。

关闭

pai命令示例

```
pai -name split_word
-project algo_public
-DinputTableName=doc_test
-DselectedColNames=content1,content2
-DoutputTableName=doc_test_split_word
-DinputTablePartitions="region=cctv_news"
-DoutputTablePartition="region=news"
-Dtokenizer=TAOBAO_CHN
-DenableDfa=true
-DenablePersonNameTagger=false
-DenableOrgnizationTagger=false
-DenablePosTagger=false
```

-DenableTelephoneRetrievalUnit=true
 -DenableTimeRetrievalUnit=true
 -DenableDateRetrievalUnit=true
 -DenableNumberLetterRetrievalUnit=true
 -DenableChnNumMerge=false
 -DenableNumMerge=true
 -DenableChnTimeMerge=false
 -DenableChnDateMerge=false
 -DenableSemanticTagger=true

算法参数

参数key名称	参数描述	参数value可选项	默认值
inputTableName	输入表名	-	-
selectedColNames	输入表中用于分词的列名	可指定多列，列名间用逗号(,)间隔	-
outputTableName	输出表名	-	-
inputTablePartitions	输入表中指定参与分词的分区名, 格式为: partition_name=value。如果是多级格式为 name1=value1/name2=value2；如果是指定多个分区，中间用','分开	-	输入表的所有partition
outputTablePartition	指定输出表的分区	-	输出表不进行分区
tokenizer	分类器类型	TAOBAO_CHN, INTERNET_CHN	默认为 TAOBAO_CHN , 淘宝 中文分词 ; INTERNET_CHN , 互联网中文分词
enableDfa	简单实体识别	true,false	true
enablePersonNameTagger	人名识别	true,false	false
enableOrganizationTagger	机构名识别	true,false	false
enablePosTagger	是否词性标注	true,false	false
enableTelephoneRetrievalUnit	检索单元配置 - 电话号码识别	true,false	true
enableTimeRetrievalUnit	检索单元配置 - 时间号码识别	true,false	true
enableDateRetrievalUnit	检索单元配置 - 日期号码识别	true,false	true
enableNumberLetterRetrievalUnit	检索单元配置 - 数字字母识别	true,false	true
enableChnNumMer	中文数字合并为一个检	true,false	false

ge	索单元		
enableNumMerge	普通数字合并为一个检索单元	true,false	true
enableChnTimeMerge	中文时间合并为一个语意单元	true,false	false
enableChnDateMerge	中文日期合并为一个语意单元	true,false	false
enableSemanticTagger	是否语义标准	true,false	false

词频统计

功能介绍

- 在对文章进行分词的基础上，按行保序输出对应文章ID列(docId)对应文章的词，统计指定文章ID列(docId)对应文章内容(docContent)的词频。

参数设置

输入参数：经过分词组件生成两列--文档ID列和分词后的文档内容列

两个输出参数：

1. 第一个输出端：输出表包含三个字段--id,word,count，如下图：

数据探查

id	word	count
1	"	1
1	"	1
1	。	2
1	不	1
1	不知道	1
1	不过	1
1	之前	1
1	之后	1
1	了	1
1	交配	3
1	什么	1
1	会	1
1	伴侣	2
1	分	1

关闭

应word词汇出现的次数

count--统计每个文档中，对

2. 第二个输出端：输出包含两个字段--id,word,如下图：



id	word
1	草原
1	田鼠
1	"
1	脸
1	盲
1	症
1	"
1	靠
1	交配
1	治愈
1	？
1	可
1	千万
1	别提

本端口输出表按词语在文章

中出现的顺序依次输出，没有统计词语的出现次数，因此同一文档中某个词汇可能出现多条记录。包输出表格式主要用于兼容Word2Vec组件使用。

实例

采用阿里分词实例数据中，将分别将输出表的两个列作为词频统计的输入参数：

选择文档ID列 -- id；选择文档内容列 -- text 经过词频统计运算后，生成的结果 见本组件中第一个输出参数展示图。

pai命令示例

```
pai -name doc_word_stat
-project algo_public
-DinputTableName=doc_test_split_word
-DdocId=id
-DdocContent=content
-DoutputTableNameMulti=doc_test_stat_multi
-DoutputTableNameTriple=doc_test_stat_triple
-DinputTablePartitions="region=cctv_news"
```

算法参数

参数key名称	参数描述	参数value可选项	默认值
inputTableName	输入表名	-	-
docId	标识文章id的列名	仅可指定一列	-

docContent	标识文章内容的列名	仅可指定一列	-
outputTableNameMulti	输出保序词语表名	-	-
outputTableNameTriple	输出词频统计表名	-	-
inputTablePartitions	输入表中指定参与分词的分区名, 格式为: partition_name=value。如果是多级格式为 name1=value1/nam e2=value2 ; 如果是指定多个分区 , 中间用 '; 分开	-	输入表的所有 partition

备注 : 其中参数outputTableNameMulti指定的表是docId列及docId列对应的文章内容(docContent)完成分词后, 按各个词语在文章中出现的顺序依次输出。参数outputTableNameTriple指定的表输出docId列及
~~docId列对应的文章内容(docContent)完成分词后, 统计得到的各个词语及其在文章中出现的次数。~~

TF-IDF

- TF-IDF (term frequency-inverse document frequency) 是一种用于资讯检索与文本挖掘的常用加权技术。TF-IDF是一种统计方法, 用以评估一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。词的重要性随着它在文件中出现的次数成正比增加, 但同时会随着它在语料库中出现的频率成反比下降。TF-IDF加权的各种形式常被搜索引擎应用, 作为文件与用户查询之间相关程度的度量或评级。
- 详细介绍, 请参考 : [维基百科tf-idf]
- 本组件是词频统计输出的基础上, 计算各个word对于各个文章的tfidf值

参数设置 (略)

实例

以词频统计组件实例中的输出表作为TF-IDF组件的输入表, 对应的参数设置如下:

选择文档ID列 : id

选择单词列 : word

选择单词计数列 : count

输出表有9列 : docid , word , word_count (当前word在当前doc中出现次数), total_word_count (当前 doc中总word数) , doc_count (包含当前word的总doc数) , total_doc_count (全部doc数) , tf, idf, tfidf
结果如下 :

数据探查										
id	word	count	total_word_count	doc_count	total_doc_count	tf	idf	tfidf		
1	"	1	82	2	6	0.012195121951219513	1.0986122886681098	0.013397710837415974		
1	"	1	82	2	6	0.012195121951219513	1.0986122886681098	0.013397710837415974	0.0	
1	.	2	82	6	6	0.024390243902439025	0.0			
1	不	1	82	2	6	0.012195121951219513	1.0986122886681098	0.013397710837415974		
1	不知道	1	82	1	6	0.012195121951219513	1.791759469228055	0.021850725234488475		
1	不过	1	82	1	6	0.012195121951219513	1.791759469228055	0.021850725234488475		
1	之前	1	82	1	6	0.012195121951219513	1.791759469228055	0.021850725234488475		
1	之后	1	82	1	6	0.012195121951219513	1.791759469228055	0.021850725234488475		
1	了	1	82	2	6	0.012195121951219513	1.0986122886681098	0.013397710837415974		
1	卖萌	3	82	1	6	0.036585365853658534	1.791759469228055	0.08555217570346542		
1	什么	1	82	1	6	0.012195121951219513	1.791759469228055	0.021850725234488475		
1	会	1	82	1	6	0.012195121951219513	1.791759469228055	0.021850725234488475		
1	伸展	2	82	1	6	0.024390243902439025	1.791759469228055	0.04370145046807695		
1	分	1	82	1	6	0.012195121951219513	1.791759469228055	0.021850725234488475		

关闭

pai命令示例

```
pai -name tfidf
-project algo_public
-DinputTableName=rgdoc_split_triple_out
-DdocIdCol=id
-DwordCol=word
-DcountCol=count
-DoutputTableName=rg_tfidf_out;
```

算法参数

参数key名称	参数描述	取值范围	是否必选，默认值/行为
inputTableName	输入表名	表名	必选
inputTablePartitions	输入表中指定参与分词的分区名	格式为: partition_name=value。如果是多级格式为 name1=value1/name2=value2；如果是指定多个分区，中间用','分开	非必选，默认行为：输入表的所有partition
docIdCol	标识文章id的列名	仅可指定一列	必选
wordCol	word列名	仅可指定一列	必选
countCol	count列名	仅可指定一列	必选
outputTableName	输出表名	表名	必选
outputTablePartition	输出表partition	分区名	非必选，默认行为：输出表为非partition表

PLDA

- 主题模型，返回文档对应的主题
- LDA(Latent Dirichlet allocation)，是一种主题模型，它可以将文档集中每篇文档的主题按照概率分布的形式给出。同时它是一种无监督学习算法，在训练时不需要手工标注的训练集，需要的仅仅是文档集以及指定主题的数量k即可。LDA首先由David M. Blei、Andrew Y. Ng和Michael I. Jordan于

2003年提出，目前在文本挖掘领域包括文本主题识别、文本分类以及文本相似度计算方面都有应用。

参数设置

主题个数: 设置LDA的输出的主题个数

Alpha:P(z/d)的先验狄利克雷分布的参数

beta: P(w/z)的先验狄利克雷分布的参数

burn In:burn in 迭代次数,必须小于总迭代次数，默认值：100

总迭代次数: 正整数 | 非必选，默认值：150

注 : z是主题, w是词, d是文档

输入输出设置

输入：数据必须为稀疏矩阵的格式（格式见数据格式说明章节）。目前需要用户自己写一个MR或者R脚本，实现数据的转换

输入格式如下：

```
+-----+-----+
| id   | features |
+-----+-----+
| 2    | 38:3.0,39:1.0,40:3.0,41:1.0,42:1.0,43:2.0,44:1.0,45:1.0,46:1.0,47:1.0,48:1.0,49:2.0,50:1.0,51:1.0,52:1.0,53:1.0,54:1.0,55:1.0,56:1.0,57:1.0,58:1.0,59:1.0,60:1.0,61:1.0,62:1.0,63:1.0,64:1.0,65:1.0,66:1.0,67:1.0,68:1.0,69:1.0,70:1.0,71:1.0,72:1.0,73:1.0,74:1.0,75:1.0,76:1.0,77:2.0 |
| 1    | 0:1.0,1:2.0,3:1.0,4:1.0,5:1.0,6:1.0,7:1.0,8:1.0,9:1.0,10:1.0,11:1.0,12:1.0,13:1.0,14:2.0,15:1.0,16:1.0,17:1.0,18:1.0,19:1.0,20:1.0,21:1.0,22:1.0,23:1.0,24:1.0,25:1.0,26:1.0,27:1.0,28:1.0,29:1.0,30:1.0,31:1.0,32:1.0,33:1.0,34:1.0,35:1.0,36:2.0,39:2.0,77:3.0 |
+-----+
```

第一列：docid；第二列：单词及词频的kv数据

输出依次为：·topic-word频率贡献表 ·单词|主题输出表 ·主题|单词输出表 ·文档|主题输出表 ·主题|文档输出表 ·主题输出表

topic-word频率贡献表的输出格式如下：

wordid	topic_0	topic_1
0	1	0
1	2	0
2	0	0
3	1	0
4	1	0
5	1	0
6	1	0
7	1	0
8	0	1
9	1	0
10	1	0
11	1	0
12	1	0

pai命令示例

```
pai -name PLDA
-project algo_public
-DinputTableName=lda_input
-DtopicNum=10
-topicWordTableName=lda_output;
```

算法参数

参数key名称	参数描述	取值范围	是否必选，默认值/行为
inputTableName	输入表名	表名	必选
inputTablePartitions	输入表中指定参与分词的分区名	格式为: partition_name=value。如果是多级格式为 name1=value1/name2=value2；如果是指定多个分区，中间用','分开	非必选，默认值：输入表的所有partition
selectedColNames	输入表中用于LDA的列名	列名，逗号分隔	非必选，默认值：输入表中所有的列名
topicNum	topic的数量	[2, 500]	必选
kvDelimiter	key和value间的分分隔符	空格、逗号、冒号	非必选，默认值：冒号
itemDelimiter	key和key间的分隔符	空格、逗号、冒号	非必选，默认值：空格
alpha	P(z/d)的先验狄利克雷分布的参数	(0, ∞)	非必选，默认值：0.1
beta	P(w/z)的先验狄利克雷分布的参数	(0, ∞)	非必选，默认值：0.01
topicWordTableName	topic-word频率贡献表	表名	必选
pzwTableName	P(w/z)输出表	表名	非必选，默认行为：不输出P(w/z)表
pzwTableName	P(z/w)输出表	表名	非必选，默认行为：不输出P(z/w)表
pdzTableName	P(d/z)输出表	表名	非必选，默认行为：不输出P(d/z)表
pzdTableName	P(z/d)输出表	表名	非必选，默认行为：不输出P(z/d)表
pzTableName	P(z)输出表	表名	非必选，默认行为：不输出P(z)表
burnInIterations	burn in 迭代次数	正整数	非必选，必须小于totalIterations，默认值：100
totalIterations	迭代次数	正整数	非必选，默认值：150

注：z是主题，w是词，d是文档

Word2Vec

功能介绍

- Word2Vec是Google在2013年开源的一个将词表转为向量的算法，其利用神经网络，可以通过训练，将词映射到K维度空间向量，甚至对于表示词的向量进行操作还能和语义相对应，由于其简单和高效引起了很多人的关注。
- Google Word2Vec的工具包相关链接：<https://code.google.com/p/word2vec/>

参数设置

算法参数：

单词的特征维度：建议 0-1000
向下采样阈值 建议值为1e-3-1e-5

输入：单词列和词汇表

输出：输出词向量表和词汇表

pai命令示例

```
pai -name Word2Vec
    -project algo_public
    -DinputTableName=w2v_input
    -DwordColName=word
    -DoutputTableName=w2v_output;
```

算法参数

参数key名称	参数描述	取值范围	是否必选，默认值/行为
inputTableName	输入表名	表名	必选
inputTablePartitions	输入表中指定参与分词的分区名	格式为: partition_name=value。如果是多级格式为 name1=value1/name2=value2；如果是指定多个分区，中间用 '；'分开	非必选，默认值：输入表的所有partition
wordColName	单词列名，单词列中每行为一个单词，语料中换行符用</s>表示	列名	必选
inVocabularyTableName	输入词表，该表为inputTableName的wordcount输出	表名	非必选，默认行为：程序内部会对输出表做wordcount
inVocabularyPartitions	输入词表分区	分区名	非必选，默认值： : inVocabularyTableName对应表的所有分区
layerSize	单词的特征维度	0-1000	非必选，默认值：100

cbow	语言模型	值为1：表示cbow模型，值为0：skip-gram模型	非必选，默认值：0
window	单词窗口大小	正整数	非必选，默认值：5
minCount	截断的最小词频	正整数	非必选：默认值：5
hs	是否采用 HIERARCHICAL SOFTMAX	值为1：表示采用，值为0：不采用	非必选，默认值：1
negative	NEGATIVE SAMPLING	值为0不可用，建议值5-10	非必选，默认值：0
sample	向下采样阈值	值为小于等于0：不采用，建议值为1e-3-1e-5	非必选，默认值：0
alpha	开始学习速率	大于0	非必选，默认值：0.025
iterTrain	训练的迭代次数	大于等于1	非必选，默认值：1
randomWindow	window是否随机	值为1，表示大小在1~5间随机；值为0，表示不随机，其值由window参数指定	非必选，默认值：1
outVocabularyTableName	输出词表	表名	非必选，默认行为：不输出'输出词表'
outVocabularyPartition	输出词表分区	分区名	非必选，默认行为：输出词表为非分区表
outputTableName	输出表	表名	必选
outputPartition	输出表分区信息	分区名	非必选，默认行为：输出表为非分区表

网络分析

k-Core

功能介绍

- 一个图的KCore是指反复去除度小于或等于k的节点后，所剩余的子图。若一个节点存在于KCore，而在(K+1)CORE中被移去，那么此节点的核数（coreness）为k。因此所有度为1的节点的核数必然为0，节点核数的最大值被称为图的核数。

参数设置

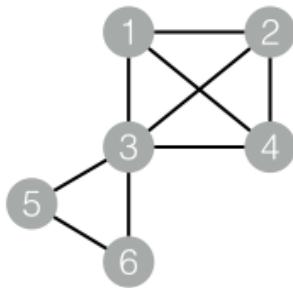
k：核数的值，必填，默认3

实例 测试数据

新建数据SQL

```
drop table if exists KCore_func_test_edge;
create table KCore_func_test_edge as
select * from
(
  select '1' as flow_out_id,'2' as flow_in_id from dual
  union all
  select '1' as flow_out_id,'3' as flow_in_id from dual
  union all
  select '1' as flow_out_id,'4' as flow_in_id from dual
  union all
  select '2' as flow_out_id,'3' as flow_in_id from dual
  union all
  select '2' as flow_out_id,'4' as flow_in_id from dual
  union all
  select '3' as flow_out_id,'4' as flow_in_id from dual
  union all
  select '3' as flow_out_id,'5' as flow_in_id from dual
  union all
  select '3' as flow_out_id,'6' as flow_in_id from dual
  union all
  select '5' as flow_out_id,'6' as flow_in_id from dual
)tmp;
```

数据对应的graph结构如下图：



运行结果 设定k = 2： 运行结果： 结果如下：

node1	node2
1	2
1	3
1	4
2	1
2	3
2	4
3	1

```

| 3 | 2 |
| 3 | 4 |
| 4 | 1 |
| 4 | 2 |
| 4 | 3 |
+-----+-----+

```

pai命令示例

```

pai -name KCore
-project algo_public
-DinputEdgeTableName=KCore_func_test_edge
-DfromVertexCol=flow_out_id
-DtoVertexCol=flow_in_id
-DoutputTableName=KCore_func_test_result
-Dk=2;

```

算法参数

参数key名称	参数描述	必/选填	默认值
inputEdgeTableName	输入边表名	必填	-
inputEdgeTablePartitions	输入边表的分区	选填	全表读入
fromVertexCol	边表中起点所在列	必填	-
toVertexCol	边表中终点所在列	必填	-
outputTableName	输出表名	必填	-
outputTablePartitions	输出表的分区	选填	-
lifecycle	输出表申明周期	选填	-
workerNum	进程数量	选填	未设置
workerMem	进程内存	选填	4096
splitSize	数据切分大小	选填	64
k	核数	必填	3

单源最短路径

功能介绍

- 单源最短路径参考Dijkstra算法，本算法中当给定起点，则输出该点和其他所有节点的最短路径。

参数设置

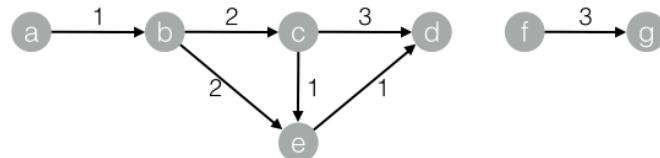
起始节点id：用于计算最短路径的起始节点，必填

实例

测试数据

新建数据的SQL语句：

```
drop table if exists SSSP_func_test_edge;
create table SSSP_func_test_edge as
select
    flow_out_id,flow_in_id,edge_weight
from
(
    select "a" as flow_out_id,"b" as flow_in_id,1.0 as edge_weight from dual
    union all
    select "b" as flow_out_id,"c" as flow_in_id,2.0 as edge_weight from dual
    union all
    select "c" as flow_out_id,"d" as flow_in_id,1.0 as edge_weight from dual
    union all
    select "b" as flow_out_id,"e" as flow_in_id,2.0 as edge_weight from dual
    union all
    select "e" as flow_out_id,"d" as flow_in_id,1.0 as edge_weight from dual
    union all
    select "c" as flow_out_id,"e" as flow_in_id,1.0 as edge_weight from dual
    union all
    select "f" as flow_out_id,"g" as flow_in_id,3.0 as edge_weight from dual
    union all
    select "a" as flow_out_id,"d" as flow_in_id,4.0 as edge_weight from dual
) tmp
;
```



数据对应的graph结构：

运行结果

结果如下：

start_node	dest_node	distance	distance_cnt
a	b	1.0	1
a	c	3.0	1
a	d	4.0	3
a	a	0.0	0
a	e	3.0	1

pai命令示例

```
pai -name SSSP
    -project algo_public
    -DinputEdgeTableName=SSSP_func_test_edge
    -DfromVertexCol=flow_out_id
    -DtoVertexCol=flow_in_id
    -DoutputTableName=SSSP_func_test_result
    -DhasEdgeWeight=true
    -DedgeWeightCol=edge_weight
    -DstartVertex=a;
```

算法参数

参数key名称	参数描述	必/选填	默认值
inputEdgeTableName	输入边表名	必填	-
inputEdgeTablePartitions	输入边表的分区	选填	全表读入
fromVertexCol	输入边表的起点所在列	必填	-
toVertexCol	输入边表的终点所在列	必填	-
outputTableName	输出表名	必填	-
outputTablePartitions	输出表的分区	选填	-
lifecycle	输出表申明周期	选填	-
workerNum	进程数量	选填	未设置
workerMem	进程内存	选填	4096
splitSize	数据切分大小	选填	64
startVertex	起始节点ID	必填	-
hasEdgeWeight	输入边表的边是否有权重	选填	false
edgeWeightCol	输入边表边的权重所在列	选填	-

PageRank

功能介绍

- PageRank起于网页的搜索排序，google利用网页的链接结构计算每个网页的等级排名，其基本思路是：如果一个网页被其他多个网页指向，这说明该网页比较重要或者质量较高。除考虑网页的链接数量，还考虑网页本身的权重级别，以及该网页有多少条出链到其它网页。对于用户构成的人际网络，除了用户本身的影响力之外，边的权重也是重要因素之一。例如：新浪微博的某个用户，会更容易

影响粉丝中关系比较亲密的家人、同学、同事等，而对陌生的弱关系粉丝影响较小。在人际网络中，边的权重等价为用户-用户的关系强弱指数。带连接权重的PageRank公式为：

$$W(A) = (1 - d) + d * \left(\sum_i W(i) * C(Ai) \right)$$

其中， $w(i)$ 为节点*i*的权重， $c(A,i)$ 为链接权重， d 为阻尼系数，算法迭代稳定后的节点权重*W*即为每个用户的影响力指数。

参数设置

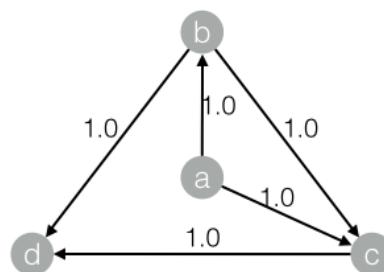
最大迭代次数：算法自身会收敛并停止迭代，选填，默认30

实例

测试数据

新建数据的SQL语句：

```
drop table if exists PageRankWithWeight_func_test_edge;
create table PageRankWithWeight_func_test_edge as
select * from
(
  select 'a' as flow_out_id,'b' as flow_in_id,1.0 as weight from dual
  union all
  select 'a' as flow_out_id,'c' as flow_in_id,1.0 as weight from dual
  union all
  select 'b' as flow_out_id,'c' as flow_in_id,1.0 as weight from dual
  union all
  select 'b' as flow_out_id,'d' as flow_in_id,1.0 as weight from dual
  union all
  select 'c' as flow_out_id,'d' as flow_in_id,1.0 as weight from dual
)tmp
;
```



对应的graph结构：

运行结果

结果如下：

node	weight

```
+----+-----+
| a | 0.0375 |
| b | 0.06938 |
| c | 0.12834 |
| d | 0.20556 |
+----+-----+
```

pai命令示例

```
pai -name PageRankWithWeight
-project algo_public
-DinputEdgeTableName=PageRankWithWeight_func_test_edge
-DfromVertexCol=flow_out_id
-DtoVertexCol=flow_in_id
-DoutputTableName=PageRankWithWeight_func_test_result
-DhasEdgeWeight=true
-DedgeWeightCol=weight
-DmaxIter 100;
```

算法参数

参数key名称	参数描述	必/选填	默认值
inputEdgeTableName	输入边表名	必填	-
inputEdgeTablePartitions	输入边表的分区	选填	全表读入
fromVertexCol	输入边表的起点所在列	必填	-
toVertexCol	输入边表的终点所在列	必填	-
outputTableName	输出表名	必填	-
outputTablePartitions	输出表的分区	选填	-
lifecycle	输出表申明周期	选填	-
workerNum	进程数量	选填	未设置
workerMem	进程内存	选填	4096
splitSize	数据切分大小	选填	64
hasEdgeWeight	输入边表的边是否有权重	选填	false
edgeWeightCol	输入边表边的权重所在列	选填	-
maxIter	最大迭代次数	选填	30

标签传播分类

功能介绍

该算法为半监督的分类算法，原理为用已标记节点的标签信息去预测未标记节点的标签信息。

在算法执行过程中，每个节点的标签按相似度传播给相邻节点，在节点传播的每一步，每个节点根据相邻节点的标签来更新自己的标签，与该节点相似度越大，其相邻节点对其标注的影响权值越大，相似节点的标签越趋于一致，其标签就越容易传播。在标签传播过程中，保持已标注数据的标签不变，使其像一个源头把标签传向未标注数据。

最终，当迭代过程结束时，相似节点的概率分布也趋于相似，可以划分到同一个类别中，从而完成标签传播过程

参数设置

阻尼系数:默认0.8 收敛系数:默认0.000001

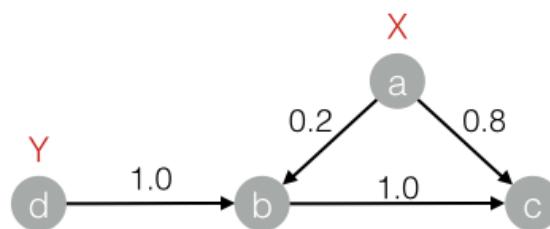
实例

测试数据

生成数据的SQL:

```
drop table if exists LabelPropagationClassification_func_test_edge;
create table LabelPropagationClassification_func_test_edge as
select * from
(
  select 'a' as flow_out_id, 'b' as flow_in_id, 0.2 as edge_weight from dual
  union all
  select 'a' as flow_out_id, 'c' as flow_in_id, 0.8 as edge_weight from dual
  union all
  select 'b' as flow_out_id, 'c' as flow_in_id, 1.0 as edge_weight from dual
  union all
  select 'd' as flow_out_id, 'b' as flow_in_id, 1.0 as edge_weight from dual
)tmp
;

drop table if exists LabelPropagationClassification_func_test_node;
create table LabelPropagationClassification_func_test_node as
select * from
(
  select 'a' as node,'X' as label, 1.0 as label_weight from dual
  union all
  select 'd' as node,'Y' as label, 1.0 as label_weight from dual
)tmp
;
```



对应的图结构：

运行结果

结果如下：

```
+----+---+-----+
| node | tag | weight   |
+----+---+-----+
| a   | X   | 1.0      |
| b   | X   | 0.16667  |
| b   | Y   | 0.83333  |
| c   | X   | 0.53704  |
| c   | Y   | 0.46296  |
| d   | Y   | 1.0      |
+----+---+-----+
```

pai命令示例

```
pai -name LabelPropagationClassification
  -project algo_public
  -DinputEdgeTableName=LabelPropagationClassification_func_test_edge
  -DfromVertexCol=flow_out_id
  -DtoVertexCol=flow_in_id
  -DinputVertexTableName=LabelPropagationClassification_func_test_node
  -DvertexCol=node
  -DvertexLabelCol=label
  -DoutputTableName=LabelPropagationClassification_func_test_result
  -DhasEdgeWeight=true
  -DedgeWeightCol=edge_weight
  -DhasVertexWeight=true
  -DvertexWeightCol=label_weight
  -Dalpha=0.8
  -Depslon=0.000001;
```

算法参数

参数key名称	参数描述	必/选填	默认值
inputEdgeTableName	输入边表名	必填	-
inputEdgeTablePartitions	输入边表的分区	选填	全表读入
fromVertexCol	输入边表的起点所在列	必填	-
toVertexCol	输入边表的终点所在列	必填	-
inputVertexTableName	输入点表名称	必填	-
inputVertexTablePartitions	输入点表的分区	选填	全表读入
vertexCol	输入点表的点所在列	必填	-
vertexLabelCol	输入点表的点的标签	必填	-
outputTableName	输出表名	必填	-
outputTablePartition	输出表的分区	选填	-

s			
lifecycle	输出表申明周期	选填	-
workerNum	进程数量	选填	未设置
workerMem	进程内存	选填	4096
splitSize	数据切分大小	选填	64
hasEdgeWeight	输入边表的边是否有权重	选填	false
edgeWeightCol	输入边表边的权重所在列	选填	-
hasVertexWeight	输入点表的点是否有权重	选填	false
vertexWeightCol	输入点表的点的权重所在列	选填	-
alpha	阻尼系数	选填	0.8
epsilon	收敛系数	选填	0.000001
maxIter	最大迭代次数	选填	30

Modularity

功能介绍

- Modularity是一种评估社区网络结构的指标，来评估网络结构中划分出来社区的紧密程度，往往0.3以上是比较明显的社区结构。

实例

测试数据

略

运行结果

结果如下：

```
+-----+
| val    |
+-----+
| 0.4230769 |
+-----+
```

pai命令示例

```
pai -name Modularity
-project algo_public
```

```
-DinputEdgeTableName=Modularity_func_test_edge
-DfromVertexCol=flow_out_id
-DfromGroupCol=group_out_id
-DtoVertexCol=flow_in_id
-DtoGroupCol=group_in_id
-DoutputTableName=Modularity_func_test_result;
```

算法参数

参数key名称	参数描述	必/选填	默认值
inputEdgeTableName	输入边表名	必填	-
inputEdgeTablePartitions	输入边表的分区	选填	全表读入
fromVertexCol	输入边表的起点所在列	必填	-
fromGroupCol	输入边表起点的群组	必填	-
toVertexCol	输入边表的终点所在列	必填	-
toGroupCol	输入边表终点的群组	必填	-
outputTableName	输出表名	必填	-
outputTablePartitions	输出表的分区	选填	-
lifecycle	输出表申明周期	选填	-
workerNum	进程数量	选填	未设置
workerMem	进程内存	选填	4096
splitSize	数据切分大小	选填	64

最大联通子图

功能介绍

在无向图G中，若从顶点A到顶点B有路径相连，则称A和B是连通的；在图G中存在若干子图，其中每个子图中所有顶点之间都是连通的，但在不同子图间不存在顶点连通，那么称图G的这些子图为最大连通子图。

参数设置

无

实例

测试数据

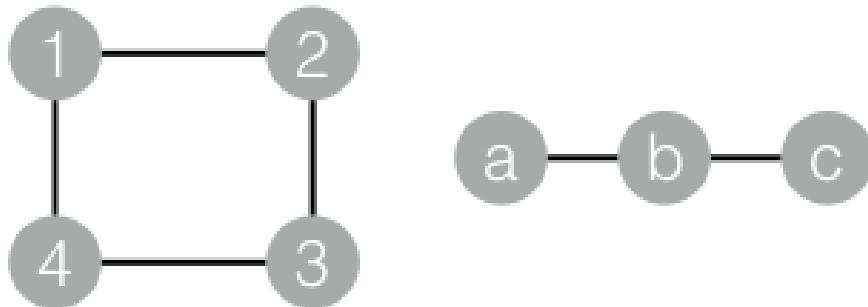
生成数据的SQL:

```

drop table if exists MaximalConnectedComponent_func_test_edge;
create table MaximalConnectedComponent_func_test_edge as
select * from
(
  select '1' as flow_out_id,'2' as flow_in_id from dual
  union all
  select '2' as flow_out_id,'3' as flow_in_id from dual
  union all
  select '3' as flow_out_id,'4' as flow_in_id from dual
  union all
  select '1' as flow_out_id,'4' as flow_in_id from dual
  union all
  select 'a' as flow_out_id,'b' as flow_in_id from dual
  union all
  select 'b' as flow_out_id,'c' as flow_in_id from dual
)tmp;

drop table if exists MaximalConnectedComponent_func_test_result;
create table MaximalConnectedComponent_func_test_result
(
  node string,
  grp_id string
);

```



对应的图结构：

运行结果

```

结果如下：
+-----+-----+
| node | grp_id|
+-----+-----+
| 1   | 4   |
| 2   | 4   |
| 3   | 4   |
| 4   | 4   |
| a   | c   |
| b   | c   |
| c   | c   |
+-----+-----+

```

pai命令示例

```
pai -name MaximalConnectedComponent
-project algo_public
-DinputEdgeTableName=MaximalConnectedComponent_func_test_edge
-DfromVertexCol=flow_out_id
-DtoVertexCol=flow_in_id
-DoutputTableName=MaximalConnectedComponent_func_test_result;
```

算法参数

参数key名称	参数描述	必/选填	默认值
inputEdgeTableName	输入边表名	必填	-
inputEdgeTablePartitions	输入边表的分区	选填	全表读入
fromVertexCol	输入边表的起点所在列	必填	-
toVertexCol	输入边表的终点所在列	必填	-
outputTableName	输出表名	必填	-
outputTablePartitions	输出表的分区	选填	-
lifecycle	输出表申明周期	选填	-
workerNum	进程数量	选填	未设置
workerMem	进程内存	选填	4096
splitSize	数据切分大小	选填	64

点聚类系数

功能介绍

在无向图G中，计算每一个节点周围的稠密度，星状网络稠密度为0，全联通网络稠密度为1。

参数设置

maxEdgeCnt：若节点度大于该值，则进行抽样，默认500，选填。

实例

测试数据

生成数据的SQL：

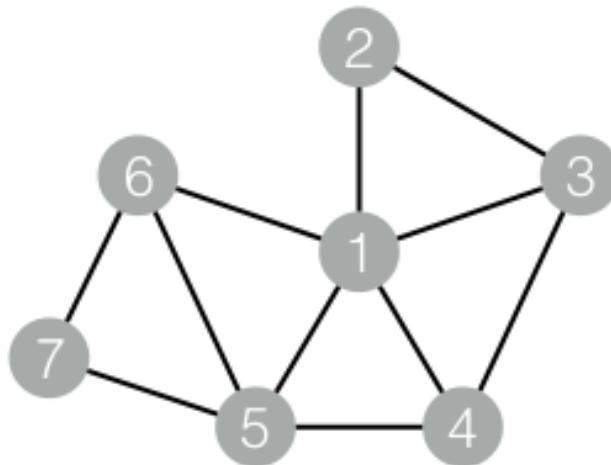
```
drop table if exists NodeDensity_func_test_edge;
create table NodeDensity_func_test_edge as
select * from
```

```

(
  select '1' as flow_out_id, '2' as flow_in_id from dual
  union all
  select '1' as flow_out_id, '3' as flow_in_id from dual
  union all
  select '1' as flow_out_id, '4' as flow_in_id from dual
  union all
  select '1' as flow_out_id, '5' as flow_in_id from dual
  union all
  select '1' as flow_out_id, '6' as flow_in_id from dual
  union all
  select '2' as flow_out_id, '3' as flow_in_id from dual
  union all
  select '3' as flow_out_id, '4' as flow_in_id from dual
  union all
  select '4' as flow_out_id, '5' as flow_in_id from dual
  union all
  select '5' as flow_out_id, '6' as flow_in_id from dual
  union all
  select '5' as flow_out_id, '7' as flow_in_id from dual
  union all
  select '6' as flow_out_id, '7' as flow_in_id from dual
)tmp;

drop table if exists NodeDensity_func_test_result;
create table NodeDensity_func_test_result
(
  node string,
  node_cnt bigint,
  edge_cnt bigint,
  density double,
  log_density double
);

```



对应的图结构：

运行结果

结果如下：
1,5,4,0.4,1.45657

```
2,2,1,1.0,1.24696
3,3,2,0.66667,1.35204
4,3,2,0.66667,1.35204
5,4,3,0.5,1.41189
6,3,2,0.66667,1.35204
7,2,1,1.0,1.24696
```

pai命令示例

```
pai -name NodeDensity
    -project algo_public
    -DinputEdgeTableName=NodeDensity_func_test_edge
    -DfromVertexCol=flow_out_id
    -DtoVertexCol=flow_in_id
    -DoutputTableName=NodeDensity_func_test_result
    -DmaxEdgeCnt=500;
```

算法参数

参数key名称	参数描述	必/选填	默认值
inputEdgeTableName	输入边表名	必填	-
inputEdgeTablePartitions	输入边表的分区	选填	全表读入
fromVertexCol	输入边表的起点所在列	必填	-
toVertexCol	输入边表的终点所在列	必填	-
outputTableName	输出表名	必填	-
outputTablePartitions	输出表的分区	选填	-
lifecycle	输出表申明周期	选填	-
maxEdgeCnt	若节点度大于该值，则进行抽样。	选填	500
workerNum	进程数量	选填	未设置
workerMem	进程内存	选填	4096
splitSize	数据切分大小	选填	64

边聚类系数

功能介绍

在无向图G中，计算每一条边周围的稠密度。

参数设置

无

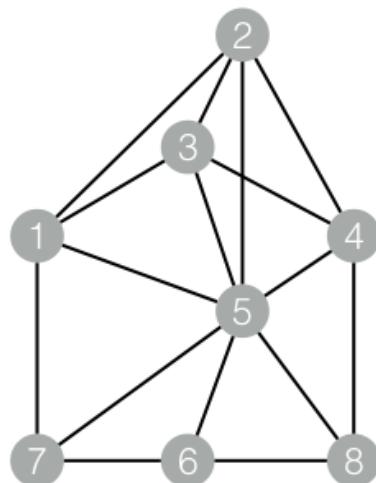
实例

测试数据

生成数据的SQL:

```
drop table if exists EdgeDensity_func_test_edge;
create table EdgeDensity_func_test_edge as
select * from
(
    select '1' as flow_out_id,'2' as flow_in_id from dual
    union all
    select '1' as flow_out_id,'3' as flow_in_id from dual
    union all
    select '1' as flow_out_id,'5' as flow_in_id from dual
    union all
    select '1' as flow_out_id,'7' as flow_in_id from dual
    union all
    select '2' as flow_out_id,'5' as flow_in_id from dual
    union all
    select '2' as flow_out_id,'4' as flow_in_id from dual
    union all
    select '2' as flow_out_id,'3' as flow_in_id from dual
    union all
    select '3' as flow_out_id,'5' as flow_in_id from dual
    union all
    select '3' as flow_out_id,'4' as flow_in_id from dual
    union all
    select '4' as flow_out_id,'5' as flow_in_id from dual
    union all
    select '4' as flow_out_id,'8' as flow_in_id from dual
    union all
    select '5' as flow_out_id,'6' as flow_in_id from dual
    union all
    select '5' as flow_out_id,'7' as flow_in_id from dual
    union all
    select '5' as flow_out_id,'8' as flow_in_id from dual
    union all
    select '7' as flow_out_id,'6' as flow_in_id from dual
    union all
    select '6' as flow_out_id,'8' as flow_in_id from dual
)tmp;

drop table if exists EdgeDensity_func_test_result;
create table EdgeDensity_func_test_result
(
    node1 string,
    node2 string,
    node1_edge_cnt bigint,
    node2_edge_cnt bigint,
    triangle_cnt bigint,
    density double
);
```



对应的图结构：

运行结果

```
结果如下：
1,2,4,4,2,0.5
2,3,4,4,3,0.75
2,5,4,7,3,0.75
3,1,4,4,2,0.5
3,4,4,4,2,0.5
4,2,4,4,2,0.5
4,5,4,7,3,0.75
5,1,7,4,3,0.75
5,3,7,4,3,0.75
5,6,7,3,2,0.666667
5,8,7,3,2,0.666667
6,7,3,3,1,0.333333
7,1,3,4,1,0.333333
7,5,3,7,2,0.666667
8,4,3,4,1,0.333333
8,6,3,3,1,0.333333
```

pai命令示例

```
pai -name EdgeDensity
    -project algo_public
    -DinputEdgeTableName=EdgeDensity_func_test_edge
    -DfromVertexCol=flow_out_id
    -DtoVertexCol=flow_in_id
    -DoutputTableName=EdgeDensity_func_test_result;
```

算法参数

参数key名称	参数描述	必/选填	默认值
inputEdgeTableName	输入边表名	必填	-
inputEdgeTablePartitions	输入边表的分区	选填	全表读入

fromVertexCol	输入边表的起点所在列	必填	-
toVertexCol	输入边表的终点所在列	必填	-
outputTableName	输出表名	必填	-
outputTablePartitions	输出表的分区	选填	-
lifecycle	输出表申明周期	选填	-
workerNum	进程数量	选填	未设置
workerMem	进程内存	选填	4096
splitSize	数据切分大小	选填	64

计数三角形

功能介绍

在无向图G中，输出所有三角形。

参数设置

maxEdgeCnt : 若节点度大于该值，则进行抽样，默认500，选填。

实例

测试数据

生成数据的SQL：

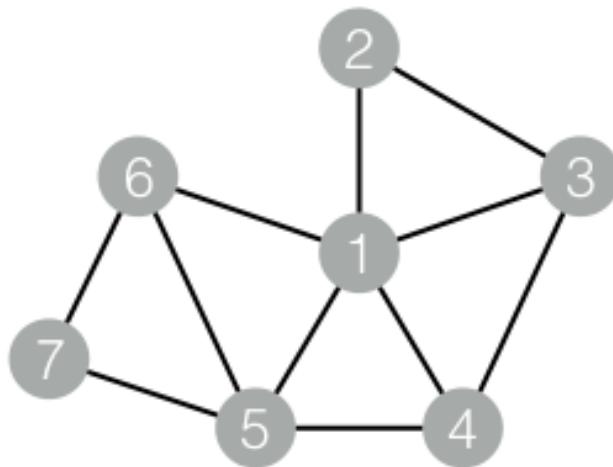
```
drop table if exists TriangleCount_func_test_edge;
create table TriangleCount_func_test_edge as
select * from
(
  select '1' as flow_out_id,'2' as flow_in_id from dual
  union all
  select '1' as flow_out_id,'3' as flow_in_id from dual
  union all
  select '1' as flow_out_id,'4' as flow_in_id from dual
  union all
  select '1' as flow_out_id,'5' as flow_in_id from dual
  union all
  select '1' as flow_out_id,'6' as flow_in_id from dual
  union all
  select '2' as flow_out_id,'3' as flow_in_id from dual
  union all
  select '3' as flow_out_id,'4' as flow_in_id from dual
  union all
  select '4' as flow_out_id,'5' as flow_in_id from dual
  union all
  select '5' as flow_out_id,'6' as flow_in_id from dual
)
```

```

union all
select '5' as flow_out_id,'7' as flow_in_id from dual
union all
select '6' as flow_out_id,'7' as flow_in_id from dual
)tmp;

drop table if exists TriangleCount_func_test_result;
create table TriangleCount_func_test_result
(
  node1 string,
  node2 string,
  node3 string
);

```



对应的图结构：

运行结果

结果如下：

1,2,3
1,3,4
1,4,5
1,5,6
5,6,7

pai命令示例

```

pai -name TriangleCount
  -project algo_public
  -DinputEdgeTableName=TriangleCount_func_test_edge
  -DfromVertexCol=flow_out_id
  -DtoVertexCol=flow_in_id
  -DoutputTableName=TriangleCount_func_test_result;

```

算法参数

参数key名称	参数描述	必/选填	默认值
inputEdgeTableName	输入边表名	必填	-

e			
inputEdgeTablePartitions	输入边表的分区	选填	全表读入
fromVertexCol	输入边表的起点所在列	必填	-
toVertexCol	输入边表的终点所在列	必填	-
outputTableName	输出表名	必填	-
outputTablePartitions	输出表的分区	选填	-
lifecycle	输出表申明周期	选填	-
maxEdgeCnt	若节点度大于该值，则进行抽样。	选填	500
workerNum	进程数量	选填	未设置
workerMem	进程内存	选填	4096
splitSize	数据切分大小	选填	64

工具

SQL脚本

用户可通过sql脚本编辑器编写sql语句。 [ODPS sql介绍链接](#)

Demo

向画布拖入odps源组件，右侧栏填入具体表名,如：



- sql脚本支持1-2个输入
- 若输入表是分区表，后台会自动勾选分区框，用户可选择或输入分区参数，目前仅支持输入单个分区。不勾选分区框或勾选后不输入分区参数均默认为输入全表
- 若输入表是非分区表，分区框不可勾选

向画布拖入sql脚本组件，连接输入表和sql脚本组件后，点击sql脚本，出现如下信息框：

参数设置

输入源

```
t1 fromName"bank_data"  
t2
```

脚本编辑

```
1 /*  
2 输入数据已自动映射成t1~t2，用  
3 户可直接使用；  
4 本节点仅支持单条SQL语句  
5 例如：select * from ${t1}  
6      输出端口1的数据为${t1}；  
7 */
```

在相应位置写入想要实现的功能的sql脚本，具体介绍如下图：

参数设置

输入源

t1 fromName"bank_data"
t2

脚本编辑

```
1 /*
2  输入数据已自动映射成t1~t2，用户可直接使用;
3  本节点仅支持单条SQL语句
4  例如: select * from ${t1}
5  输出端口1的数据为${t1};
6 */
7 select count(*) from ${t1}
```

- sql脚本支持1-2个输入，1个输出
- 用户只能写入单条sql语句，sql语法介绍
- 输入数据已自动映射成t1~t2，用户可直接使用。使用时直接调用\${t1},\${t2}，不用再写入原表名。
- 示例的sql脚本用于实现对输入表进行行数统计

向画布拖入一个odps目标组件，输入新表名。如果需要分区表，需填入具体分区信息，会直接创建新表或新分区表，如：

表选择

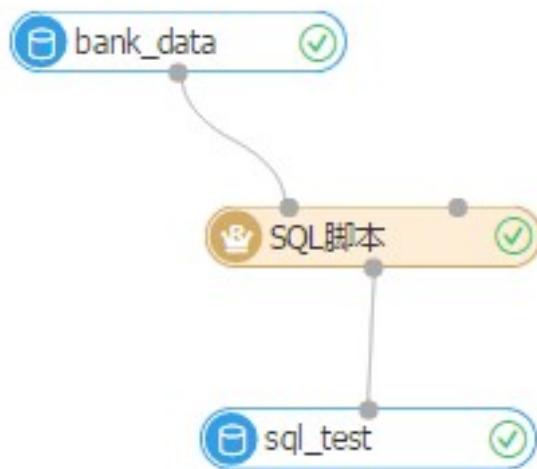
输入新表名

sql_test

分区

- 目前不支持具体分区

链接所有组件后，点击执行



右键点击odps目标组件，可以选择查看数据
数据探查



PAI 命令

不提供pai命令

ODPS XLib

在PAI中调用XLib，具体方法见XLib

FAQ

Q1：算法平台是否支持单节点运行或部分组件运行功能？

答：支持，点击右键，选择“单点运行”或“从此处运行”，“运行到此处”

Q2：随机采样报错：sample count is larger or equal with row num，为什么？

答：不放回采样时随机采样的样本数不得多于数据条数。解决方案：参数设置->采样个数，调整参数即可

Q3：归一化报错 symbol nan not found in column-name-array, expr: (poutcome-NaN)/NaN，为什么？

详细的报错信息

答：归一化的组件，支持double与bigint类型，不支持string类型

Q4：模型生成后连ODPS目标为什么连不上？

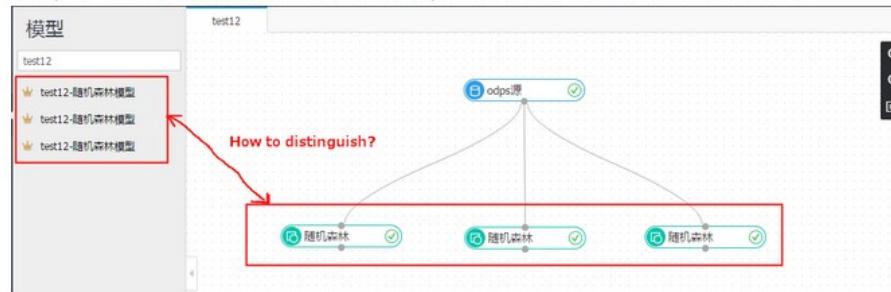
如图：



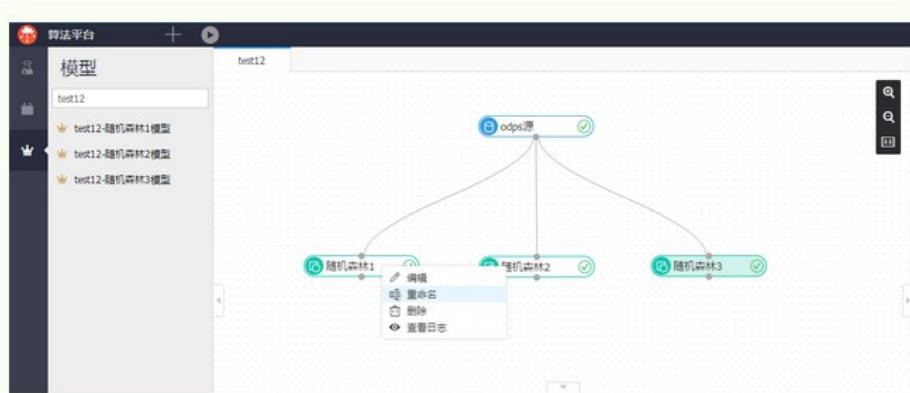
答：ODPS目标是存储表数据，算法节点的输出是模型，仅能连接预测组件，生成的模型可以在模型栏看到，不需要再接一个ODPS目标。所有的模型皆是如此

Q5：如何区分不同参数下同时生成的多个随机森林？

如图，当一个实验中同时生成多个随机森林模型时，如何区分不同参数生成的对应随机森林模型



答：在组件运行前对各个随机森林组件重命名进行区分，运行结束后能较清楚的看到生成的对应模型，如下图：



Q6 : GDBT回归与排序，报错labelCol:poutcome is not numeric，为什么？

详细的错误信息

```
[2] FAILED: Failed Task checkParameters:[string "function split(s, spliter)..."]:103: labelCol:poutcome is not numeric
```

答：GBDT的输入列与标签列只支持Bigint 与 double类型，不支持string类型

Q7 : GBDT报错：User Error-gbdt only support numeric feature，为什么？

详细错误信息

```
[2] FAILED: Failed 20150429160015f8aa9826_9914_41eb_8a7f_ae12700ae4cf:ODPS-1202005:Algo Job Failed-UnretryableException: User Error-gbdt only support numeric feature
```

答：GBDT的输入列与标签列只支持bigInt 与 double 类型，不支持string类型

Q8 : 如何删除已生成的模型？

答：有两种方案：

方案1：直接在命令端执行 `DROP OFFLINEMODEL [IF EXISTS] <offline_model_name>;`

- `offline_model_name`是要删除的模型名

方案2：通过算法平台UI端来进行删除。进入模型管理，点击同步（见下图），可以同步本Project下所有的模型，选择你要删除的模型，删除即可。多选后支持批量删除。



算法平台

模型

逻辑回归

- xiao's-逻辑回归模型
- 逻辑回归-逻辑回归模型
- ROC曲线-逻辑回归模型
- test status-逻辑回归模型
- Jmeter-逻辑回归模型
- Jmeter01-逻辑回归模型
- 逻辑回归-逻辑回归模型
- ROC曲线-逻辑回归模型

同步

删除

Q9：如何将pai命令生成的模型同步到算法平台？

答：参考Q8方案2的示例图片，在算法平台上点击同步按钮，可以实现。但此方法同步的模型在算法平台上不支持模型查看。

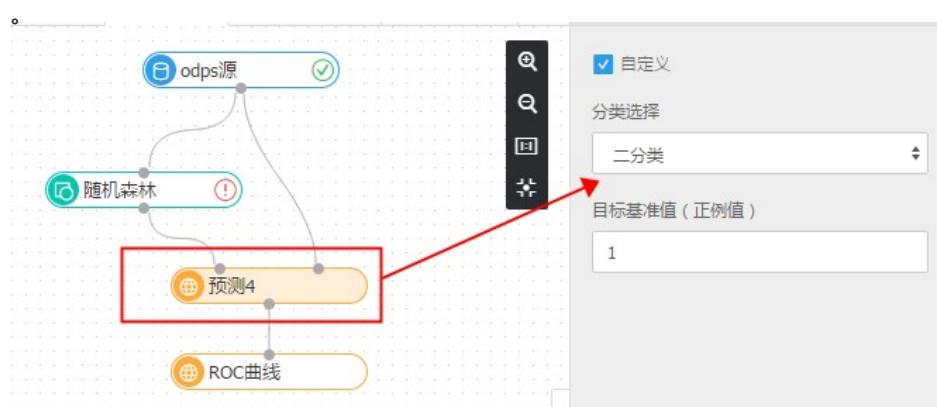
- 注意：算法平台上生成的模型暂时只支持查看随机森林和逻辑回归。

Q10：加权采样如何使正样本采样概率大一点？

答：建议先transfrom，造出权重列，然后再用appendcolumns，把权重列和原表拼成新表后进行

Q11：做ROC曲线时，标签列是二分类，为什么日志里会报错误：param goodvalue not found？

答：ROC曲线只支持标签列为二分类，必须要接收到一个目标基准值（也就是goodvalue）后，才能对生成的模型进行评估。用户在做预测时参数应该设置成自定义--二分类，并指定目标基准值(如图)。否则会报以上错误



- 目标基准值需指定二分类标签列中的value
- GBDT回归与排序不支持ROC曲线

Q12：运行随机森林时报错如下，为什么？

```
[2] FAILED: Failed 20150505112438af853b4a_61d9_43fa_95cf_f6c75c6599c9:ODPS-1202005:Algo Job Failed-UnretryableException: User Error-attribute user_id group is more than 70000
```

答：随机森林不支持分类类别超过70000，当超过时可考虑回归

Q13：运行逻辑回归时报错如下，为什么？

```
FAILED: Failed 20150505113107aa1069e_ae82_493a_82eb_16daefc65f07:ODPS-1202005:Algo Job Failed-UnretryableException: User Error-Invalid table schema
```

答：逻辑回归的输入列和标签列不支持string型，换成数字型即可。

- 注意：模型的输入和标签列要谨慎选择，输入支持类型可参考具体组件文档

Q14：组件运行时日志项出现 No task resources left in the project，为什么？

答：用户的每个project只能跑3个task，每个task里能跑800个instance，多了就会报No task resources的错误，可在数据开发台用如下命令解决，步骤如下：

- step1：show p--看所有的task；

- step2 : status instance--看instance状态；
- step3 : kill instance--关掉在执行的某个暂时不用的instance；
- 注意：若要查看几天前的instance，可以采用show p from date1 to date2 number，eg：show p from 2015-5-20 to 2015-5-22 2000，再一次重复step2和step3

Q15:如何查看或者查杀正在运行的任务？

答：参考Q14

Q16：浏览器登录超时后数据开发台的日志无法查看，怎么办？

答：可在数据开发台输入如下示例指令：

```
desc instance 20150624121855110grormr06
wait 20150624121855110grormr06
```

- 注意：示例指令中20150624121855110grormr06是用户的某个instance id，使用时根据需求自行更改。

Q17：模型预测组件报错：DataValue has no double value，为什么？

答：请检查模型训练输入表与模型预测输入表的字段信息是否一致，若不致，会报上述错误。具体检查方法，在数据开发台用'desc 表名'语句查看训练表和预测表的字段类型。

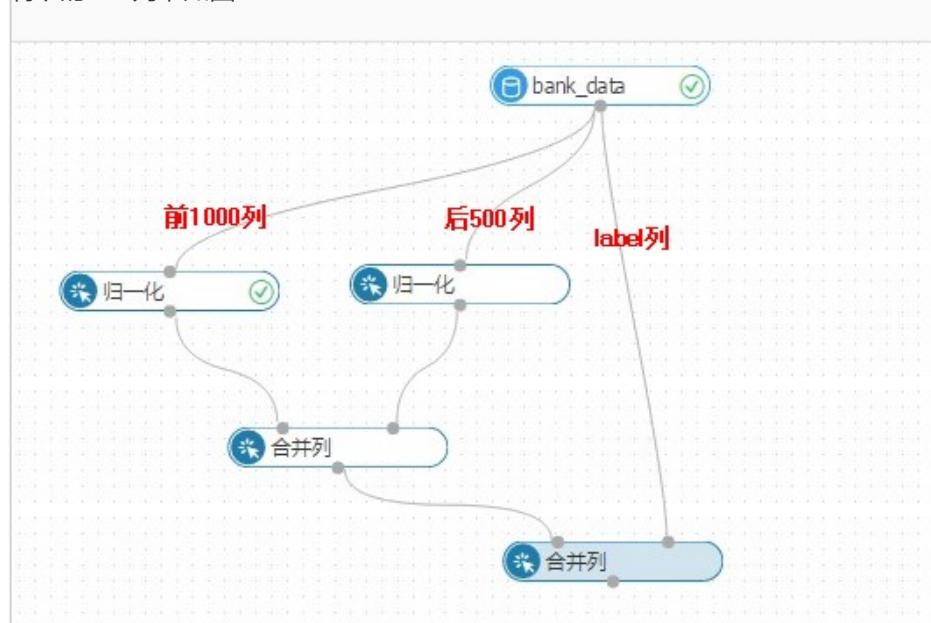
Q18：随机森林中我设置的树的最大深度是10，为什么最后生成的树的深度不到10？

答：随机森林生成的模型中，树的深度和数据本身、数据量以及feature都有关系，只能限制最大深度，无法确定每一次实验生成的树的具体深度

Q19：算法平台输入列最多只可以输入1024列，超过后会报错：java.lang.Exception: java.lang.Exception: input selectedColNum shoud be less than 1024! 为什么？

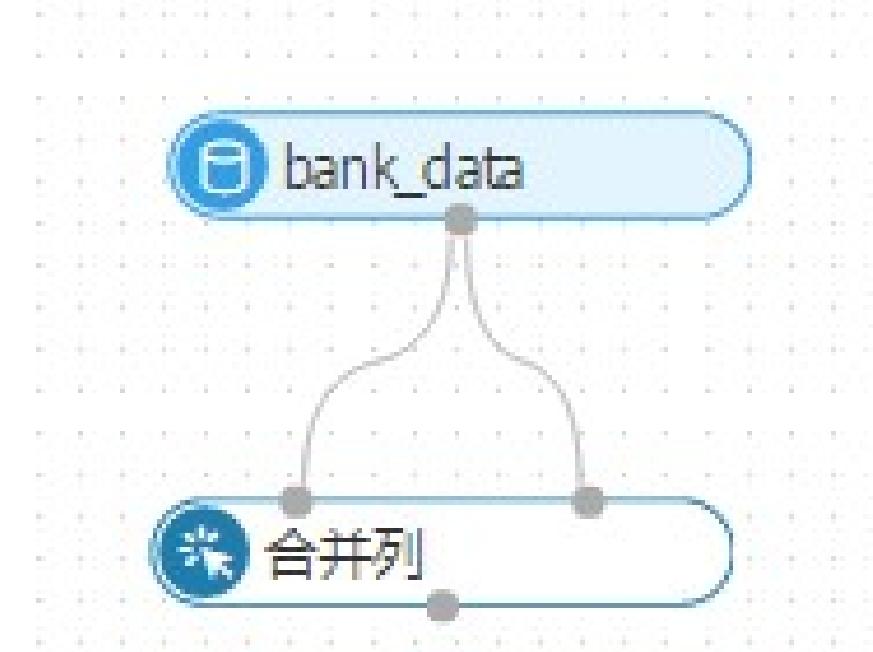
答：ODPS底层只支持最多输入1024列进行处理。

如果用户输入列超过1024列，比如用1500列做归一化处理，可采用两个归一化组件，先处理1000列，再处理剩下的500列，如图：



Q20：算法平台合并列功能，如何确定的那两行的列合并在一起？

答：若是一张表，要合并其中的列a，列b，可选择如下图方法，左边输出列选择列a，右表输出列选择列b即可。多张表合并其中的某些列思路也是如此。



Q21：算法平台中模型如何处理空值null？是忽略这一列的所有数据，还是给空值赋一个默认值？

答：若feature有null值，训练算法不识别，可能会报错。如果null有特殊意义，请使用缺省值填充组件，将其转成算法认识的值。

Q22：pai命令实现GBDT回归与预测模型，做预测时，DsplitCharacteristic参数如何设置？

答：DsplitCharacteristic参数是对二分类和多分类的一个定义，二分类时，DsplitCharacteristic=1，多分类时，DsplitCharacteristic=2。GBDT回归与排序是一个回归模型，不涉及到分类问题，跑pai命令时默认写成DsplitCharacteristic=2，后台处理时会自动过滤掉。另外，若是直接在web端连接组件跑任务，对应预测组件参数应保持“默认”，不需要勾选“自定义”。

Q23：GBDT回归与排序组件为什么做混淆矩阵及ROC曲线计算时会报错？

答：“GBDT回归与排序模型属于回归模型，不能做混淆矩阵和ROC曲线计算。混淆矩阵及ROC曲线是针对分类模型计算的。”

Q24：逻辑回归模型报错FAILED: Failed Task getTargetValues:[string "function split(s, splitter)..."]:149: Too many classes and logistic regression can not support.为什么？

答：模型做多分类预测时，分类最多支持100维，若分类的label列value超过100个，则会报上述错误。

案例

[玩转数据]心脏病预测案例

一、背景

心脏病是人类健康的头号杀手。全世界1 / 3的人口死亡是因心脏病引起的，而我国，每年有几十万人死于心脏病。所以，如果可以通过提取人体相关的体侧指标，通过数据挖掘的方式来分析不同特征对于心脏病的影响，对于预测和预防心脏病将起到至关重要的作用。本文将会通过真实的数据，通过阿里云机器学习平台搭建心脏病预测案例。

二、数据集介绍

数据源：UCI开源数据集heart_disease针对美国某区域的心脏病检查患者的体测数据，共303条数据。具体字段如下表：

字段名	含义	类型	描述
age	年龄	string	对象的年龄，数字表示
sex	性别	string	对象的性别，female和male
cp	胸部疼痛类型	string	痛感由重到无 typical、atypical、non-anginal、asymptomatic
trestbps	血压	string	血压数值
chol	胆固醇	string	胆固醇数值
fbs	空腹血糖	string	血糖含量大于120mg/dl为true，否则为false
restecg	心电图结果	string	是否有T波，由轻到重为norm、hyp
thalach	最大心跳数	string	最大心跳数
exang	运动时是否心绞痛	string	是否有心绞痛，true为是，false为否
oldpeak	运动相对于休息的ST depression	string	st段压数值
slop	心电图ST segment的倾斜度	string	ST segment的slope，程度分为down、flat、up
ca	透视检查看到的血管数	string	透视检查看到的血管数
thal	缺陷种类	string	并发种类，由轻到重 norm、fix、rev
status	是否患病	string	是否患病，buff是健康

--	--	--

 、sick是患病

三、数据探索流程

数据挖掘流程如下：



整体实验流程：



1.数据预处理

数据预处理也叫作数据清洗，主要在数据进入算法流程前对数据进行去噪、填充缺失值、类型变换等操作。本次实验的输入数据包括14个特征和1个目标队列。需要解决的场景是根据用户的体检指标预测是否会患有心脏病，每个样本只有患病或不患病两种，是分类问题。因为本次分类实验选用的是线性模型逻辑回归，要求输入的特征都是double型的数据。输入数据展示：

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slop	ca	thal	status	style
63.0	male	ang...	145.0	233.0	true	hyp	150.0	fa1	2.3	down	0.0	fix	buff	H
67.0	male	asy...	160.0	286.0	fa1	hyp	108.0	true	1.5	flat	3.0	norm	sick	S2
67.0	male	asy...	120.0	229.0	fa1	hyp	129.0	true	2.6	flat	2.0	rev	sick	S1
37.0	male	not...	130.0	250.0	fa1	norm	187.0	fa1	3.5	down	0.0	norm	buff	H
41.0	fem	abn...	130.0	204.0	fa1	hyp	172.0	fa1	1.4	up	0.0	norm	buff	H
56.0	male	abn...	120.0	236.0	fa1	norm	178.0	fa1	0.8	up	0.0	norm	buff	H
62.0	fem	asy...	140.0	268.0	fa1	hyp	160.0	fa1	3.6	down	2.0	norm	sick	S3
57.0	fem	asy...	120.0	354.0	fa1	norm	163.0	true	0.6	up	0.0	norm	buff	H
63.0	male	asy...	130.0	254.0	fa1	hyp	147.0	fa1	1.4	flat	1.0	rev	sick	S2
53.0	male	asy...	140.0	203.0	true	h1	155.0	true	3.1	down	0.0	rev	sick	S1

我们看到有很多数据是文字

描述的，在数据预处理的过程中我们需要根据每个字段的含义将字符型转为数值。

1) 二值类的数据二值类的比较容易转换，如sex字段有两种表现形式female和male，我们可以将female表示成0，把male表示成1。

2) 多值类的数据比如cp字段，表示胸部的疼痛感，我们可以通过疼痛的由轻到重映射成0~3的数值。

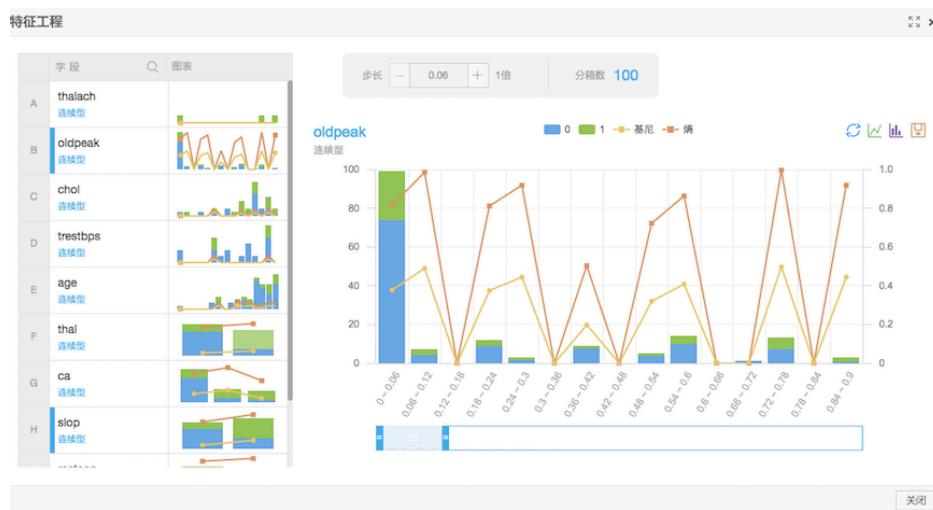
数据的预处理通过sql脚本来实现，具体请参考SQL脚本-1组件，

```
select age,
(case sex when 'male' then 1 else 0 end) as sex,
(case cp when 'angina' then 0 when 'notang' then 1 else 2 end) as cp,
trestbps,
chol,
(case fbs when 'true' then 1 else 0 end) as fbs,
(case restecg when 'norm' then 0 when 'abn' then 1 else 2 end) as restecg,
thalach,
(case exang when 'true' then 1 else 0 end) as exang,
oldpeak,
(case slop when 'up' then 0 when 'flat' then 1 else 2 end) as slop,
ca,
(case thal when 'norm' then 0 when 'fix' then 1 else 2 end) as thal,
(case status when 'sick' then 1 else 0 end) as ifHealth
from ${t1};
```

2.特征工程

特征工程主要是包括特征的衍生、尺度变化等。本例中有两个组件负责特征工程的部分。

1) 过滤式特征选择主要是通过这个组件判断每个特征对于结果的影响，通过信息熵和基尼系数来表示，可以通过查看评估报告来显示最终的结果。



2) 归一化因为本次实验选择的是通过逻辑回归二分类来进行模型训练，需要每个特征去除量纲的影响。归一化的作用是将每个特征的数值范围变为0到1之间。归一化的公式为 $result = (val - min) / (max - min)$ 。归一化结果：

数据探查 - pai_temp_2954_36756_1 - (仅显示前一百条)

sex ▲	cp ▲	fbs ▲	restecg ▲	exang ▲	slop ▲	thal ▲	ifhealth ▲	age ▲	trestbps ▲	chol ▲	thalach ▲	oldpeak ▲
1	0	1	1	0	1	0.5	0	0.70...	0.4811320...	0.244...	0.603053...	0.370967...
1	1	0	1	1	0.5	0	1	0.79...	0.6226415...	0.365...	0.282442...	0.241935...
1	1	0	1	1	0.5	1	1	0.79...	0.2452830...	0.235...	0.442748...	0.419354...
1	0.5	0	0	0	1	0	0	0.16...	0.3396226...	0.283...	0.885496...	0.564516...
0	1	0	1	0	0	0	0	0.25	0.3396226...	0.178...	0.770992...	0.225806...
1	1	0	0	0	0	0	0	0.5625	0.2452830...	0.251...	0.816793...	0.129032...
0	1	0	1	0	1	0	1	0.6875	0.4339622...	0.324...	0.679389...	0.580645...
0	1	0	0	1	0	0	0	0.58...	0.2452830...	0.520...	0.702290...	0.096774...
1	1	0	1	0	0.5	1	1	0.70...	0.3396226...	0.292...	0.580152...	0.225806...
1	1	1	1	1	1	1	1	0.5	0.4339622...	0.175...	0.641221...	0.5
1	1	0	0	0	0.5	0	0	0.58...	0.4339622...	0.150...	0.587786...	0.064516...

3. 模型训练和预测

本次实验是监督学习，因为我们已经知道每个样本是否患有心脏病，所谓监督学习就是已知结果来训练模型。解决的问题是预测一组用户是否患有心脏病。

1) 拆分首先通过拆分组件将数据分为两部分，本次实验按照训练集和预测集7：3的比例拆分。训练集数据流入逻辑回归二分类组件用来训练模型，预测集数据进入预测组件。

2) 逻辑回归二分类逻辑回归是一个线性模型，在这里通过计算结果的阈值实现分类。具体的算法详情推荐大家在网上或者书籍中自行了解。逻辑回归训练好的模型可以在模型页签中查看。

逻辑回归输出

字段名 ▲	权重	
	1 ▲	0 ▲
sex	1.473569994686197	-
cp	2.730064736238172	-
fbs	-0.6007338270729394	-
restecg	0.8990240712157691	-
exang	0.9026382341453308	-
slop	1.041821068646534	-
thal	1.562393603912368	-
age	-0.4278050593226199	-

1、PAI平台提供的逻辑回归可用于多分类的，采取的策略是OneVsAll，因此在多分类的情况下，会出现多个方程，每个方程针对目标特征的某个value值，即权重（weight）下方对应的列名；

2、逻辑回归的完整公式为： $\sigma(z) = 1 / (1 + \exp(-z))$ ； $z = w_0 + w_1 * x_1 + w_2 * x_2 + \dots + w_m * x_m$ 。（其中 x_1, x_2, \dots, x_m 是某样本数据的各个特征， w_1, w_2, \dots 是特征的权重值）



3)预测预测组件的两个输入分别是模型和预测集。预测结果展示的是预测数据、真实数据、每组数据不同结果的概率。

4.评估

通过混淆矩阵组件可以评估模型的准确率等参数，

混淆矩阵							
混淆矩阵		比例矩阵		统计信息			
模型 ▲	正确数 ▲	错误数 ▲	总计 ▲	正确率 ▲	准确率 ▲	召回率 ▲	F1指标 ▲
0	40	8	48	84.146%	83.333%	88.889%	86.022%
1	29	5	34	84.146%	85.294%	78.378%	81.69%

通过此组件可以方便的通过预测的准确性来评估模型。

四.总结

通过以上数据探索的流程我们可以得到以下的结论。

1) 特征权重我们可以通过过滤式特征选择得到每个特征对于结果的权重。

featname ▲	weight ▲
thalach	0.16569171224597157
oldpeak	0.14640697618779352
thal	0.13769166559906015
ca	0.11467097546217575
chol	0.10267709576600859
age	0.07876430484527841
trestbps	0.0772599125640569
slop	0.07702762609078306
restecg	0.015246832497405105
cp	0.0037507283721422424
exang	0
fbs	0
sex	0

-可以看出thalach(心跳数)对于是否发生心脏病影响最大。

-性别对于心脏病没有影响

2) 模型效果通过上文提供的14个特征，可以达到百分之八十多的心脏病预测准确率。模型可以用来做预测，辅助医生预防和治疗心脏病。

五、其它

参与讨论：云栖社区公众号

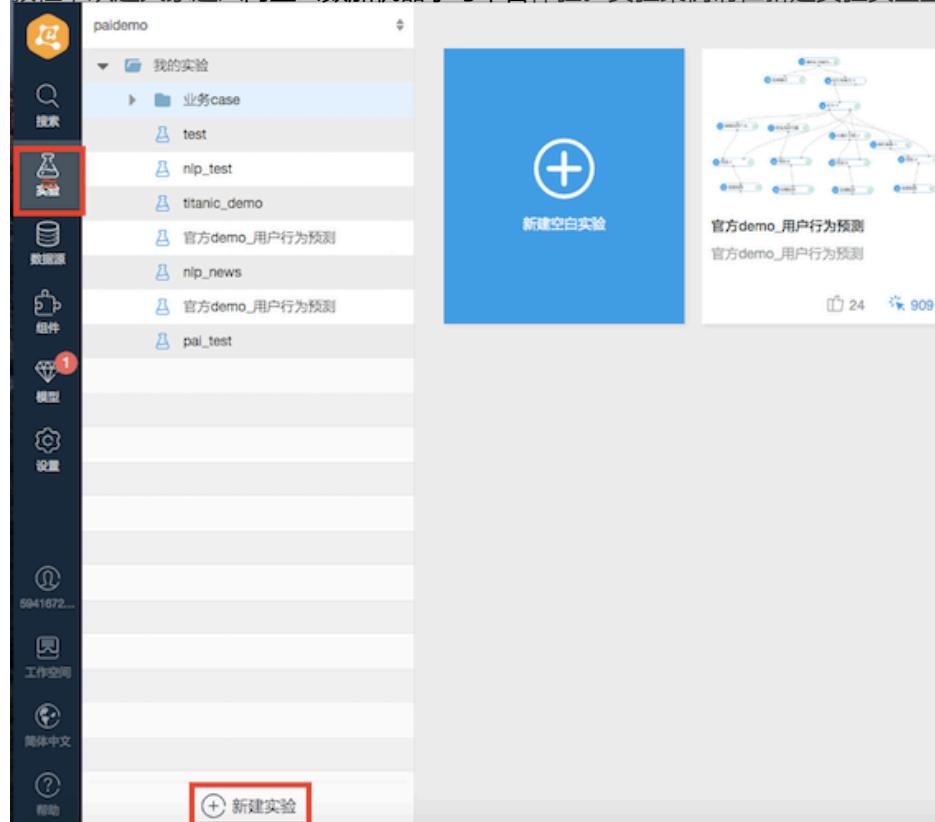
免费体验：阿里云数加机器学习平台

下期预告：机器学习图算法最短路径的计算

【玩转数据】人口普查统计案例

一、背景

感谢大家关注玩转数据系列文章，我们希望通过在阿里云机器学习平台上提供demo数据并搭建相关的实验流程的方式来帮助大家学习如何通过算法来挖掘数据中的价值。本系列文章包含详细的实验流程以及相关的文档教程，欢迎大家进入阿里云数加机器学习平台体验。实验案例请在新建实验页签查看，如下图。



本章作为玩转数据系列的开篇，先提供一个简单的案例给大家热身。通过截取一份人口普查的数据，对学历和收入进行统计和分析。主要目的是帮助大家学习阿里云机器学习实验的搭建流程和组件的使用方式。任何关于阿里云机器学习方面的交流欢迎访问我们的云栖社区公众号。

二、数据集介绍

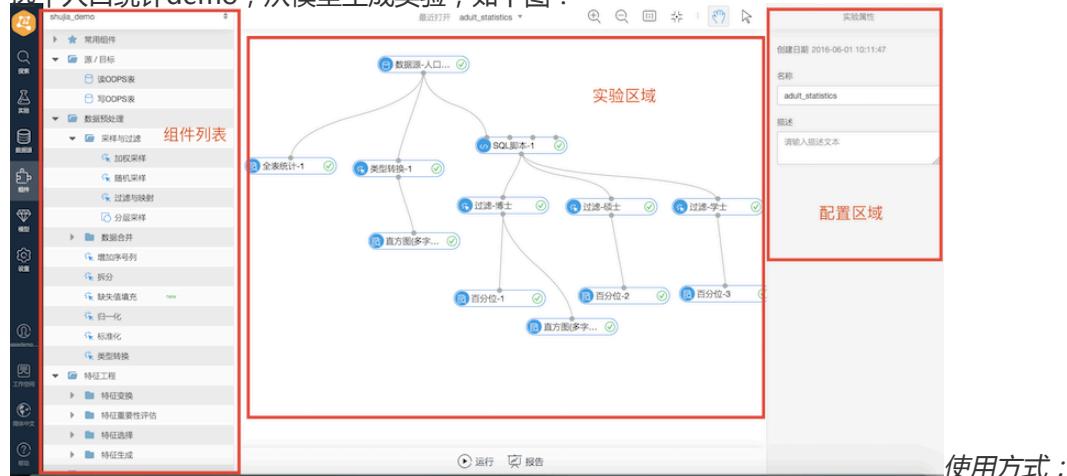
数据源：UCI开源数据集Adult针对美国某区域的一次人口普查结果，共32561条数据。具体字段如下表：

字段名	含义	类型
age	年龄	double
workclass	工作类型	string

fnlwgt	序号	string
education	教育程度	string
education_num	受教育时间	double
marital_status	婚姻状况	string
occupation	职业	string
relationship	关系	string
race	种族	string
sex	性别	string
capital_gain	资本收益	string
capital_loss	资本损失	string
hours_per_week	每周工作小时数	double
native_country	原籍	string
income	收入	string

三、数据探索流程

选中人口统计demo，从模型生成实验，如下图：



-用户通过从左边列表拖拽组件到试验区域搭建实验流程

-在配置区域对每个组件的参数进行设置

1. 数据导入

机器学习平台的底层计算式阿里云分布式计算系统MaxCompute（原名ODPS），所以实验数据需要先导入到ODPS表里，用户可以通过读ODPS表（图中的数据源-人口统计）组件导入数据。上传成功后，右键组件可以

数据探查 - adult_statistics_demo - (仅显示前一百条)

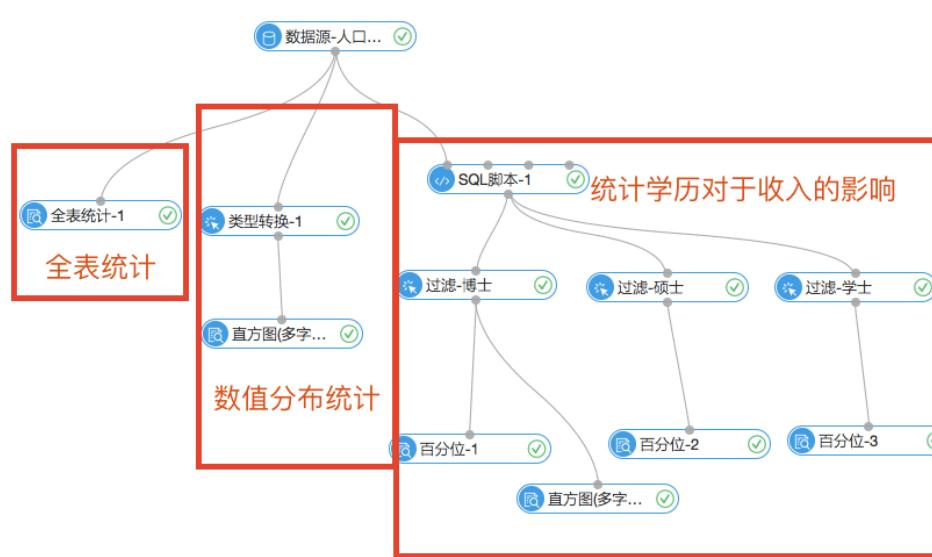
age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_lo
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0
50	Self-emp-n...	83311	Bachelors	13	Married-civ-spouse	Exec-manag...	Husband	White	Male	0	0
38	Private	215646	HS-grad	9	Divorced	Handlers-cle...	Not-in-family	White	Male	0	0
53	Private	234721	11th	7	Married-civ-spouse	Handlers-cle...	Husband	Black	Male	0	0
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fem...	0	0
37	Private	284582	Masters	14	Married-civ-spouse	Exec-manag...	Wife	White	Fem...	0	0
49	Private	160187	9th	5	Married-spouse-a...	Other-service	Not-in-family	Black	Fem...	0	0
52	Self-emp-n...	209642	HS-grad	9	Married-civ-spouse	Exec-manag...	Husband	White	Male	0	0
31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Fem...	14084	0
42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-manag...	Husband	White	Male	5178	0
37	Private	280464	Some-college	10	Married-civ-spouse	Exec-manag...	Husband	Black	Male	0	0
30	State-gov	141297	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	Asian...	Male	0	0
23	Private	122272	Bachelors	13	Never-married	Adm-clerical	Own-child	White	Fem...	0	0
32	Private	205019	Assoc-acdm	12	Never-married	Sales	Not-in-family	Black	Male	0	0
40	Private	121772	Assoc-acdm	11	Married-civ-spouse	Craft-repair	Husband	Asian...	Male	0	0
34	Private	245487	Assoc-acdm	4	Married-civ-spouse	Transport-mo...	Husband	Amer...	Male	0	0
95	Self-emp-n...	178756	WC-handicap	0	Never-married	Ex-farm	Family-child	White	Male	n	n

查看数据，如下图：

关闭

2.理解数据

数据导入后就可以对数据进行分析了，整个实现从纵向看分为三个部分。



其中全表统计和数值分布统

计是帮助用户更好的理解一份数据，理解一份数据是符合泊松分布或是高斯分布,连续或是离散的对之后的算法的选择会有一定帮助（具体的对照关系在之后的文章会详细介绍）。阿里云机器学习的每个套件都提供了可视化显示结果的功能，下图是数值统计的直方图组件结果，可以清楚地看到每个输入数值的分布情况。

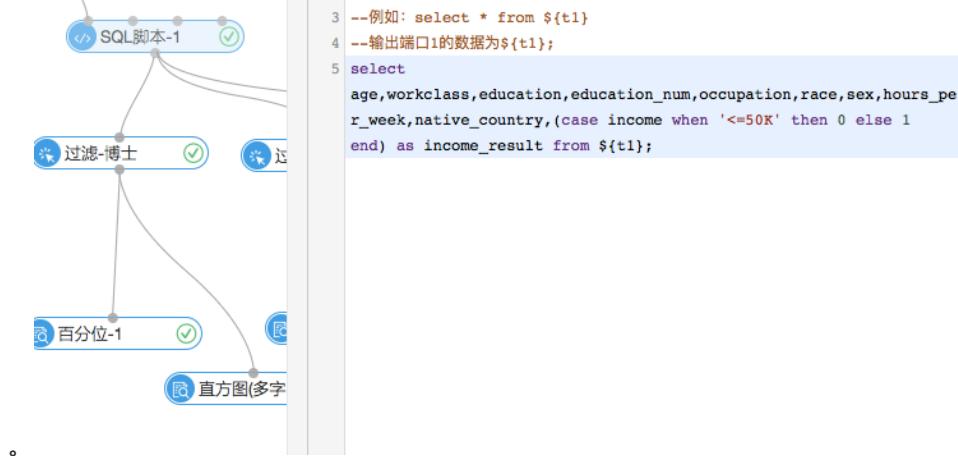


3.统计不同学历的人员的收入情况

每个人都想增加收入，都想知道哪些因素对收入的影响最大。这些问题都可以通过提取特征，利用机器学习算法训练来得到。本文主要目的是简单介绍一下机器学习平台的使用方法，这里简单的针对不同学历的人员的收入做一下统计。

(1)数据的预处理

我们看到在收入统计的这条线上，数据流入的第一个组件是SQL脚本（如下图），机器学习平台提供SQL脚本对于数据进行处理。这里是将string型的income字段转换成二值型的0和1的形式。0表示年收入在50K以下，1表示年收入在50K以上。这种将文本数据数值化是机器学习特征处理的常用方式，以后会经常用到这种方式



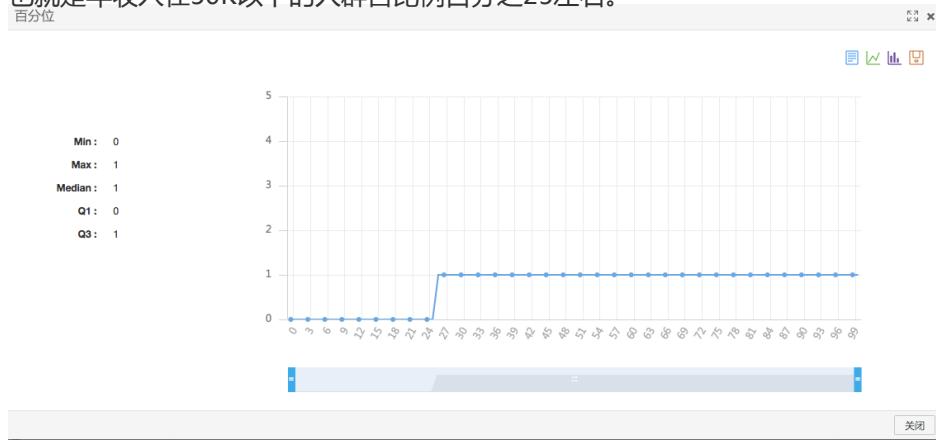
(2)过滤与映射

这一步主要是通过过滤与映射组件将数据按照学历分为三部分，分别是博士、硕士和学士。过滤与映射底层是SQL语法，支持where过滤条件，用户通过在右边的配置栏填写过滤条件即可。



(3)统计结果

通过每个百分位组件就可以方便的得到每个分类下的收入比例。下图是调成折线图的展示效果,结果中为0的点也就是年收入在50K以下的人群占比例百分之25左右。



结合三个百分位组件就可以得到如下图结果。

学历	年收入>50K比例
博士	75%
硕士	57%
学士	42%

四、其它

参与讨论：云栖社区公众号

免费体验：阿里云数加机器学习平台

下期预告：利用机器学习算法预测患者是否患有心脏病