机器学习PAI

PAI算法组件说明

PAI算法组件说明

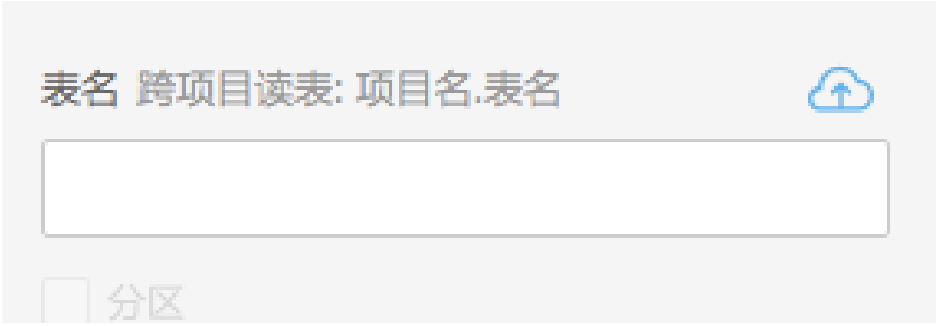
目录

读数据表

读取Maxcompute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。MaxCompute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MaxCompute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

读MaxCompute表的输入框



左上角为创建odps表的功能；

分区功能介绍

PAI的读数据组件包含读取分区表的功能，在日期定义上与大数据开发套件略有不同。PAI在读取分区表时需要指定dt=@@{yyyyMMdd},其中@@{yyyyMMdd}表示当前日期，@@{yyyyMMdd-1d}表示当前日期前一天。

写数据表

写入MaxCompute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

Mysql数据同步

功能说明

- 同步Mysql 数据到MaxCompute 项目

参数说明

参数名称	参数描述	取值范围	是否必选，默认值/行为
source	cdp 同步数据源标识，常量为 cdp_mysql	cdp_mysql	cdp_mysql
project_name	必填，ODPS项目的Project Name	-	-

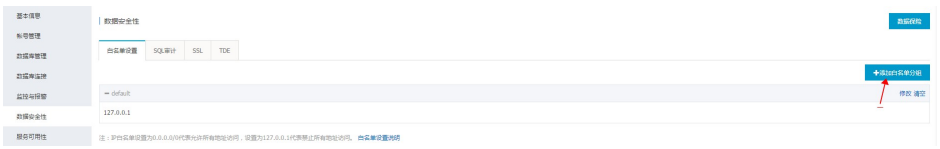
access_id	必填，ODPS项目的access_id	-	-
access_key	必填，ODPS项目的access_key	-	-
end_point	必填，ODPS项目的end_point	-	http://service.odps.aliyun.com/api
instanceName	必填，RDS的实例名称	-	
database	必填，RDS数据库	-	
username	必填，RDS该数据库的用户名	-	-
password	必填，RDS该数据库密码	-	-
table	必填，欲同步的数据表	-	-
column	选填，默认同步该数据库所有字段	-	
outputTable	必填，RDS该数据库密码	-	-
mbps	选填，数据同步带宽	单位MB/s	1
errorLimit	选填，数据错误数，默认0容忍数据错误	-	0
lifecycle	输出结果表的生命周期	-	7

ps:由于CDP服务是对外服务，不支持集团内部数据同步，集团内部数据同步请走数据同步中心或者datax

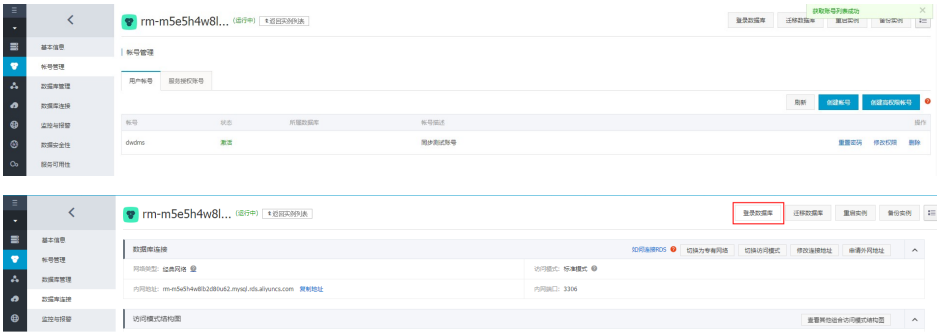
如何获取组件参数

1. 登录aliyun.com，使用主账号登录，切换到rds控制台，如下图所示，获取rds的accessKey 和 获取实例名称

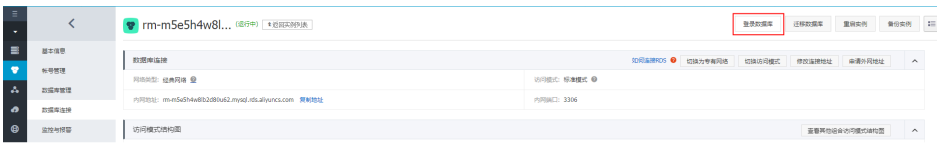
2. 添加白名单。 由于rds对访问的ip有限制，需要点击 数据安全性 ，添加白名单，其中0.0.0.0/0表示运行任意ip访问.



3. 点击实例链接，可以查看实例的详细信息，比如账号信息(如果没有账号，可以新建一个账号)，数据库信息



4. 在 数据库链接 一栏中点击 登录数据库



5. 登录后，可以查看数据库 database， 数据库下对应的table和schema

OSS数据同步

功能说明

- 同步OSS的文本到ODPS 数据源

ps: cdp服务不提供命令行执行语句

参数说明

参数名称	参数描述	取值范围	是否必选，默认值/行

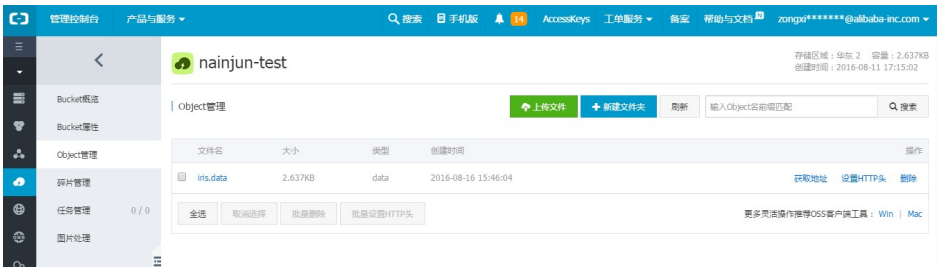
			为
source	cdp 同步数据源标识，常量为 cdp_mysql	cdp_mysql	cdp_mysql
project_name	必填，ODPS项目的Project Name	-	-
access_id	必填，ODPS项目的access_id	-	-
access_key	必填，ODPS项目的access_key	-	-
end_point	必填，ODPS项目的end_point	-	http://service.odps.aliyun.com/api
OSSendpoint	必填，OSS存储服务所在的endpoint	oss-cn-xxxx.aliyuncs.com	oss-cn-shanghai.aliyuncs.com
OSSaccessId	必填，OSS服务的accessId	-	-
OSSaccessKey	必填，OSS服务的accessKey	-	-
bucket	必填，OSS服务的bucket	-	-
object	必填，欲同步的OSS object	-	-
OSScolumn	必填，同步的字段映射.格式是index:name，表示OSS第index列同步到ODPS字段名为name的字段中，字段类型默认string，比如0:label,1:s_width,2:s_length,3:v_width,4:v_length	-	-
fieldDilimeter	必填，OSS object的文本分隔符(列分隔符)	逗号	,
encoding	选填，OSS文本的编码	utf-8	utf-8
compress	选填，OSS文本压缩格式，默认无	gzip,zip,bzip2	
mbps	选填，数据同步带宽	单位MB/s	1
errorLimit	选填，数据错误数，默认0容忍数据错误	-	0
lifecycle	输出结果表的生命周期	-	7

ps:由于CDP服务是对外服务，不支持集团内部数据同步，集团内部数据同步请走数据同步中心或者datax

如何获取组件参数

1 使用主账号登录 aliyun.com，切换到OSS 控制台，点击界面右上角的accessKey，获取accessId和accessKey

2 在OSS控制台，可以看到用户拥有的 bucket，比如下图bucket名为nianjun-test(没有可以创建)， 点击bucket，进入bucket的详情，左边栏有Bucket属性，Object管理等. 从中可以获取bucket，object等信息.



3 点击Bucket概览，可以获取该OSS bucket所在的endpoint

目录

加权采样

随机采样

过滤与映射

分层采样

join

合并列

UNION

增加序号列

拆分

缺失值填充

归一化

标准化

类型转换

KV2Table

Table2KV

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

- 可手动输入采样个数（或者采样比例）
- 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
- 下拉框选择加权列，加权列支持double型和bigint型
- 可配置随机数种子，默认系统自动分配

PAI 命令

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

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	-	-
sampleSize	可选，采样个数	正整数	默认为空
sampleRatio	可选，采样比例	浮点数，范围(0,1)	默认为空
probCol	必选，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	-	-
replace	可选，是否放回，boolean类型	true / false	false(默认不放回)
randomSeed	可选，随机数种子	正整数	默认系统自动生成
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，计算的核心数目	正整数	系统自动分配
memSizePerCore	可选，每个核心的内存（单位是兆）	正整数，(1, 65536)	系统自动分配

注：当sampleSize 与 sampleRatio都为空时，报错； 当sampleSize 与 sampleRatio都不为空时，以sampleSize为准。

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。

参数设置

- 可手动输入采样个数（或者采样比例）

- 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
- 可配置随机数种子，默认系统自动分配
- 可以自己配置并发计算核心数目与内存，默认系统自动分配

PAI 命令

```
pai -name RandomSample -project algo_public \
-DinputTableName=wbpc \
-DoutputTableName=wpbc_sample \
-DsampleSize=100 \
-Dreplace=false \
-DrandomSeed=1007;
```

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	-	-
sampleSize	可选，采样个数	-	默认为空
sampleRatio	可选，采样比例，范围(0,1)	-	默认为空
replace	可选，是否放回，boolean类型	true / false	false(默认不放回)
randomSeed	可选，随机数种子	正整数	默认系统自动生成
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，计算的核心数目	正整数	系统自动分配
memSizePerCore	可选，每个核心的内存（单位是兆）	正整数，(1, 65536)	系统自动分配

注：当sampleSize 与 sampleRatio都为空时，报错； 当sampleSize 与 sampleRatio都不为空时，以sampleSize为准。

过滤与映射

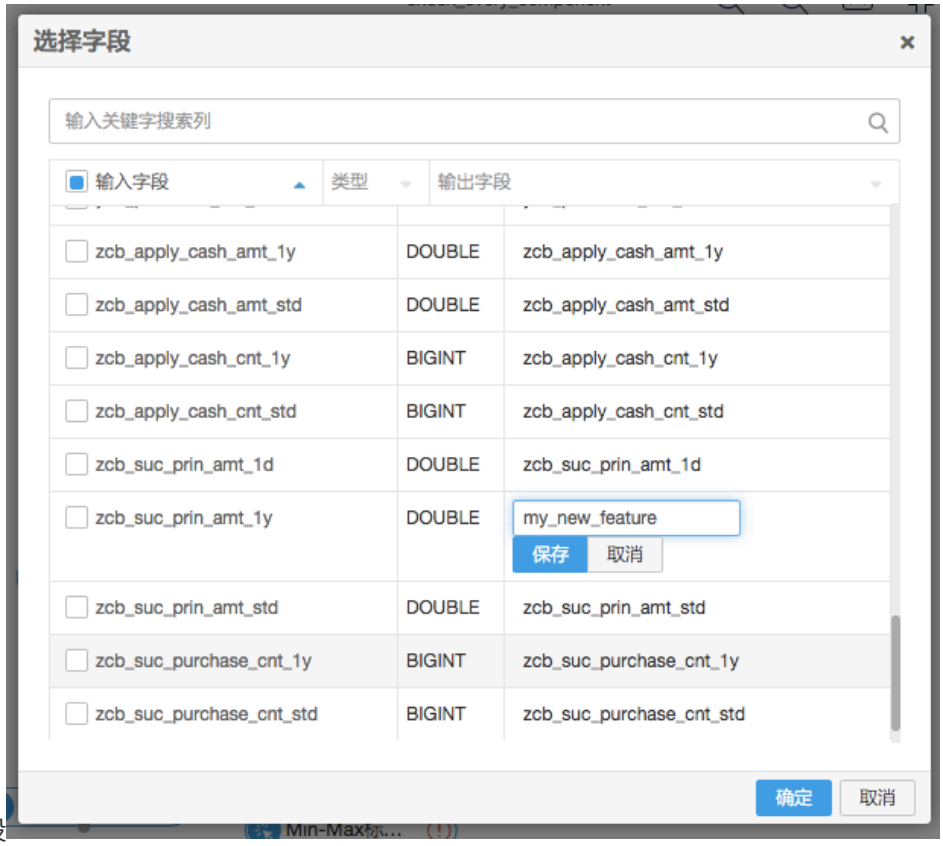
对数据按照过滤表达式进行筛选，可以重命名字段名。

参数设置

1、通过where条件实现数据过滤，与SQL类似，如：

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

2、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“rlike”

分层采样

数据集分层抽取一定比例或者一定数据的随机样本

参数设置

参数框


字段设置
参数设置
分组列 必选 分层按此列划分

- 下拉框选择分组列（最大支持100个分组）
- 可手动输入分组的采样个数（或者采样比例）
- 可配置随机数种子，默认1234567
- 可以自己配置并发计算核心数目与内存，默认系统自动分配

pai 命令

```
PAI -name StratifiedSample -project algo_public
-DinputTableName="test_input"
-DoutputTableName="test_output"
-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	-	-
inputTablePartitions	可选，训练输入表分区。	-	默认选中全表
outputTableName	必选，输出表	-	-
strataColName	必选，层次列,分层就是按照此列作为key分层的	-	-
sampleSize	可选，采样大小，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，采样比例，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	-	默认123456
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

注：当sampleSize 与 sampleRatio都为空时，报错； 当sampleSize 与 sampleRatio都不为空时，以sampleSize为准。

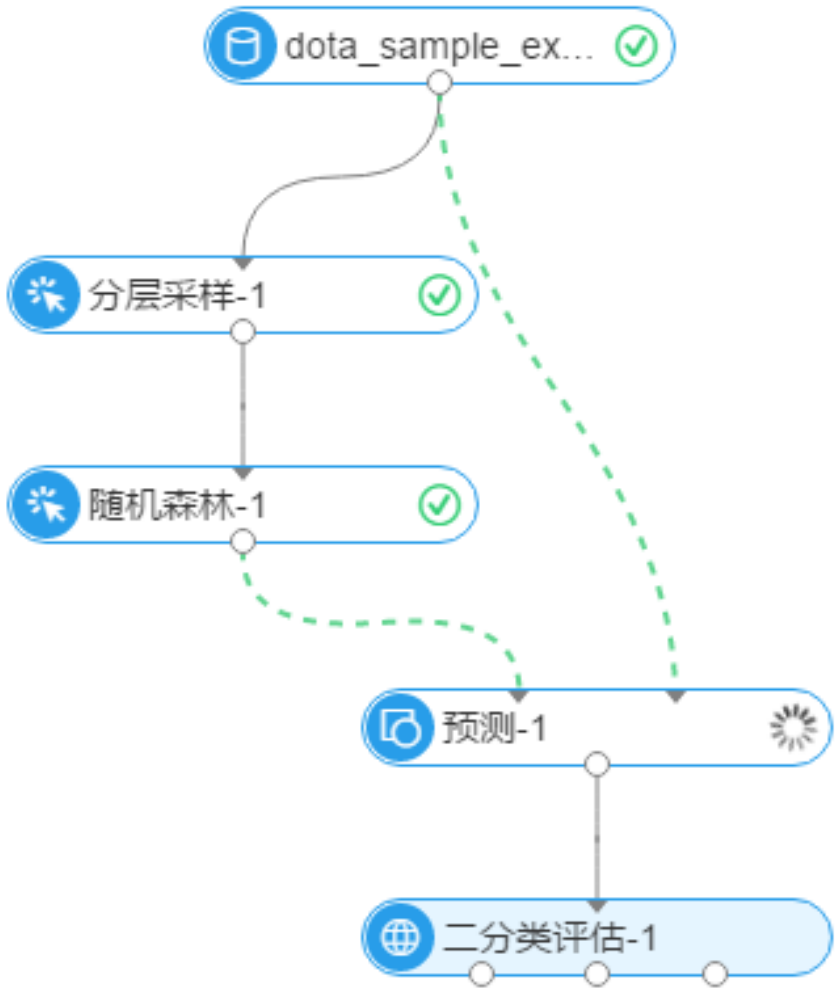
示例

源数据

id	outlook	temperature	humidity	windy	play
0	Sunny	85	85	false	No
1	Sunny	80	90	true	No

2	Overcast	83	86	false	Yes
3	Rainy	70	96	false	Yes
4	Rainy	68	80	false	Yes
5	Rainy	65	70	true	No
6	Overcast	64	65	true	Yes
7	Sunny	72	95	false	No
8	Sunny	69	70	false	Yes
9	Rainy	75	80	false	Yes
10	Sunny	75	70	true	Yes
11	Overcast	72	90	true	Yes
12	Overcast	81	75	false	Yes
13	Rainy	71	91	true	No

1. 创建实验


dota_sample_ex...
分层采样-1
随机森林-1
预测-1
二分类评估-1

2. 选取分组列


字段设置
参数设置
分组列 必选 分层按此列划分
play

- 选择play列，做为分组列

3. 配置采样个数

- play=Yes的分组采4条；play=No有分组采3条

4. 采样结果

id ▲	outlook ▲	temperature ▲	humidity ▲	windy ▲	play ▲
1	Sunny	80	90	true	No
5	Rainy	65	70	true	No
9	Rainy	75	80	false	Yes
10	Sunny	75	70	true	Yes
11	Overcast	72	90	true	Yes
12	Overcast	81	75	false	Yes
13	Rainy	71	91	true	No

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

- 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

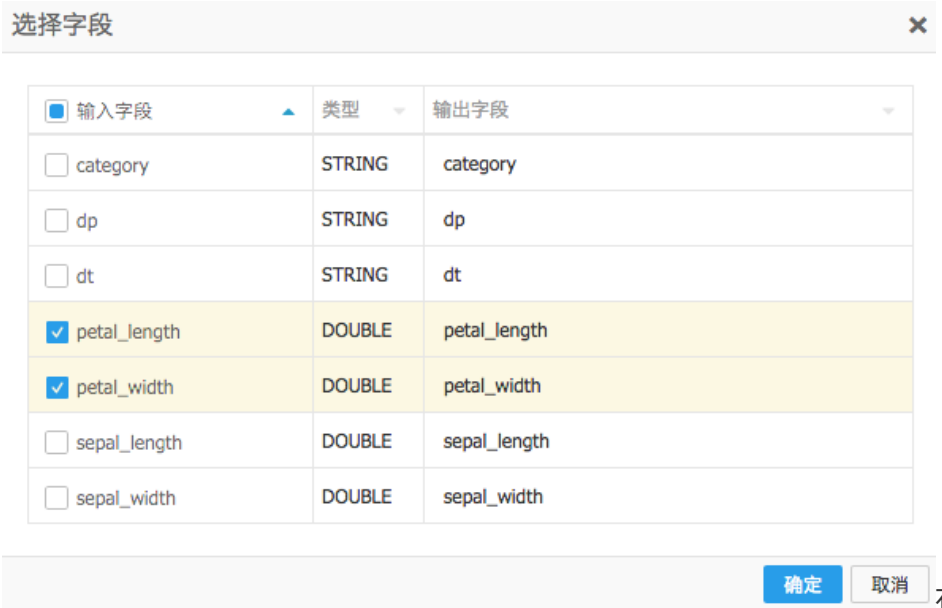
不提供pai命令

合并列

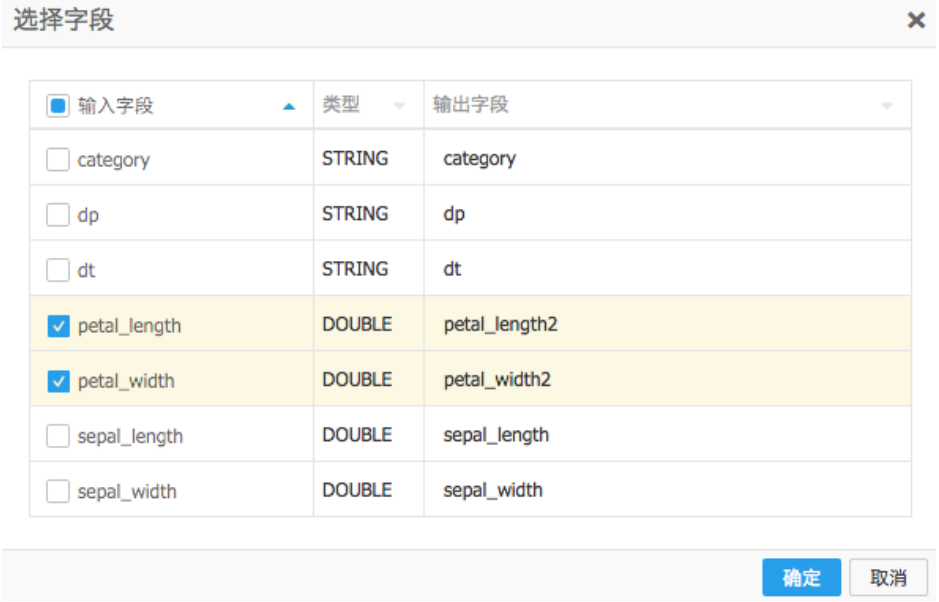
将两张表的数据按列合并，需要表的行数保持一致

参数设置

如图：左表选择输入列



右表选择输入列



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

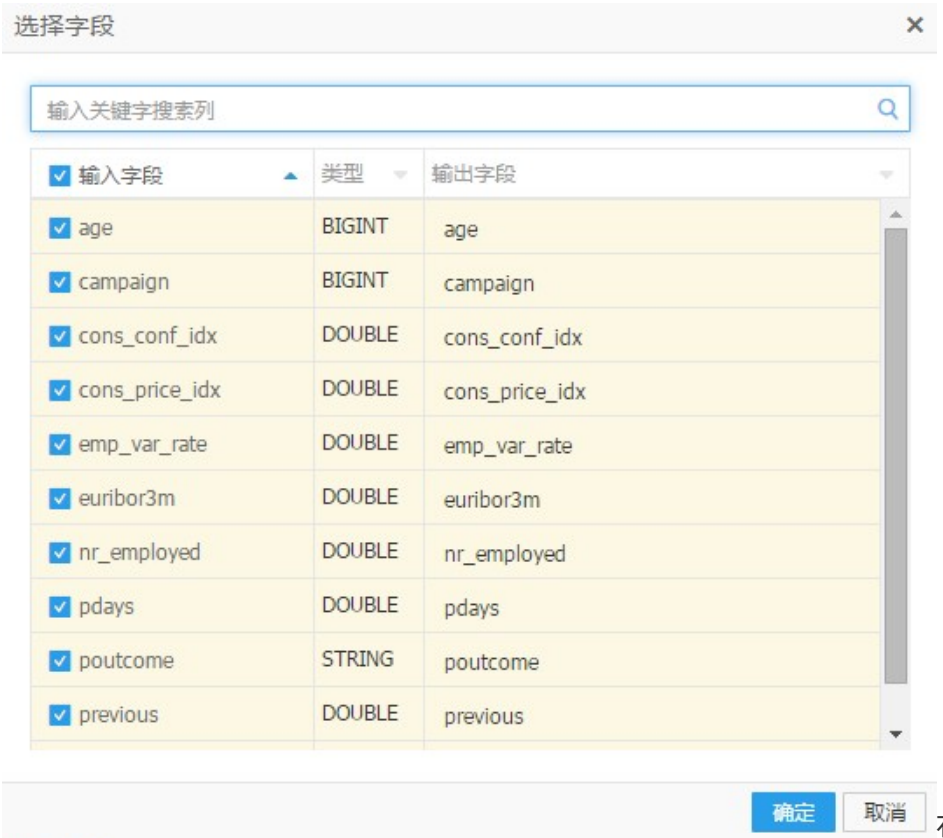
UNION

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。

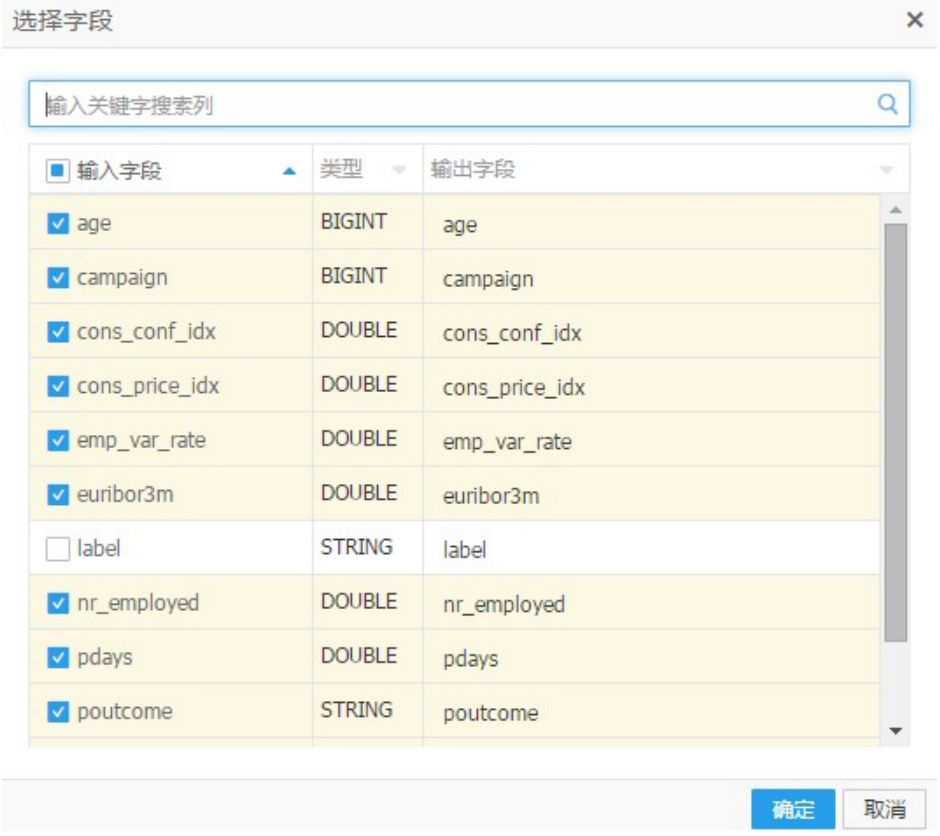
参数设置

调整参数，如下：

- 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



右表联合列



PAI 命令

不提供pai命令

增加序号列

在数据表第一列追加ID列，并将append id后的内容存为新表，增加的ID列为BigInt类型。

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";
```

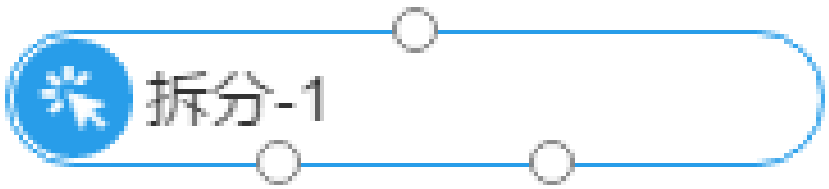
- name: 组件名字
- project: project名字，用于指定算法所在空间。系统默认是algo_public，用户自己更改后系统会报错
- IDColName：追加的ID列列名，从0开始编号，依次为0,1,2,3...
- outputTableNames: 输出表的名字
- inputTableNmae: 输入表的名字
- selectedColNames: 要保留的字段名，多个用逗号分隔

拆分

背景

对输入表或分区进行按比例拆分，分别写入两张输出表。

算法组件



- 拆分组件，对应两个输出桩
- 如图所示配置0.6，则左边的输出桩对应60%的数据，右边对应40%的数据

PAI命令

```
pai -name split -project algo_public \
-DinputTableName=wbpc \
-Doutput1TableName=wpbc_split1 \
-Doutput2TableName=wpbc_split2 \
-Dfraction=0.25;
```

参数设置

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
output1TableName	必选，输出结果表1	-	-
output1TablePartition	可选，输出结果表1分区名	-	输出表1为非分区表
output2TableName	必选，输出结果表2	-	-
output2TablePartition	可选，输出结果表2分区名	-	输出表2为非分区表
fraction	必选，切分至输出表1的数据比例	(0,1)	-
lifecycle	可选，指定输出表生命	正整数，[1,3650]	输出表没有生命周期

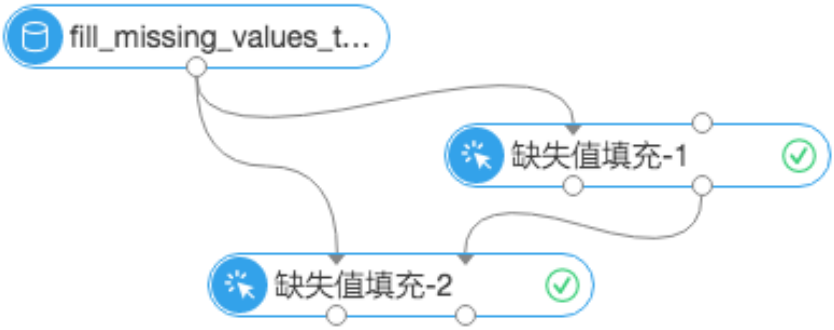
	周期		

缺失值填充

缺失值填充用来将空值或者一个指定的值替换为最大值，最小值，均值或者一个自定义的值。可以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。

- 可以将数值型的空值替换为最大值，最小值，均值或者一个自定义的值
- 可以将字符型的空值，空字符串，空值和空字符串，指定值替换为一个自定义的值
- 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
- 数值型替换可以自定义，也可以直接选择替换成数值最大值，最小值或者均值

缺失值填充界面



两个输入桩依次对应参数为：

inputTableName 输入表，即要填充的表

inputParaTableName 配置输入表，即缺失值填充节点生成的参数列表，通过此参数，可以将一张表的配置参数应用到一张新的表

两个输出桩依次对应参数为：

outputTableName 输出表，即填充完成的表

outputParaTableName 输出配置表，用于应用到其他的数据集上

缺失值填充参数界面

- 填充的字段，原值，替换为三个部分组成了config参数，分别对应config参数的三个部分：列名，原值，替换值

PAI 命令

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

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	-	-
inputTablePartitions	可选，训练输入表分区	-	默认选中全表

	。		
outputTableName	必选，输出表	-	-
configs	必选，缺失值填充的配置。格式如 “col1, null, 3.14; col2, empty, hello; col3, empty-null, world” ，其中null表示空值，empty表示空字符。缺失值若选择空字符，则填充的目标列应是string型。若采用最大值、最小值、均值，可以采用变量，其命名规范形如：min, max, mean。若用户自定义替换值，则使用user-defined，格式如”col4,user-defined,str,str123”	-	-
outputParaTableName	必选，配置输出表	-	-
inputParaTableName	可选，配置输入表	-	默认不输入
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

实例

测试数据

新建数据SQL

```
drop table if exists fill_missing_values_test_input;
create table fill_missing_values_test_input(
col_string string,
col_bigint bigint,
col_double double,
col_boolean boolean,
col_datetime datetime);
insert overwrite table fill_missing_values_test_input
select
*
from
(
select
```

```
'01' as col_string,
10 as col_bigint,
10.1 as col_double,
True as col_boolean,
cast('2016-07-01 10:00:00' as datetime) as col_datetime
from dual
union all
select
cast(null as string) as col_string,
11 as col_bigint,
10.2 as col_double,
False as col_boolean,
cast('2016-07-02 10:00:00' as datetime) as col_datetime
from dual
union all
select
'02' as col_string,
cast(null as bigint) as col_bigint,
10.3 as col_double,
True as col_boolean,
cast('2016-07-03 10:00:00' as datetime) as col_datetime
from dual
union all
select
'03' as col_string,
12 as col_bigint,
cast(null as double) as col_double,
False as col_boolean,
cast('2016-07-04 10:00:00' as datetime) as col_datetime
from dual
union all
select
'04' as col_string,
13 as col_bigint,
10.4 as col_double,
cast(null as boolean) as col_boolean,
cast('2016-07-05 10:00:00' as datetime) as col_datetime
from dual
union all
select
'05' as col_string,
14 as col_bigint,
10.5 as col_double,
True as col_boolean,
cast(null as datetime) as col_datetime
from dual
) tmp;
```

输入数据说明

```
+------------+------------+------------+-------------+--------------+
| col_string | col_bigint | col_double | col_boolean | col_datetime |
+------------+------------+------------+-------------+--------------+
| 04 | 13 | 10.4 | NULL | 2016-07-05 10:00:00 |
| 02 | NULL | 10.3 | true | 2016-07-03 10:00:00 |
```

```
| 03 | 12 | NULL | false | 2016-07-04 10:00:00 |
| NULL | 11 | 10.2 | false | 2016-07-02 10:00:00 |
| 01 | 10 | 10.1 | true | 2016-07-01 10:00:00 |
| 05 | 14 | 10.5 | true | NULL |
+------------+------------+------------+-------------+--------------+
```

运行命令

```
drop table if exists fill_missing_values_test_input_output;
drop table if exists fill_missing_values_test_input_model_output;
PAI -name FillMissingValues
-project algo_public
-Dconfigs="col_double,null,mean;col_string,null-
empty,str_type_empty;col_bigint,null,max;col_boolean,null,true;col_datetime,null,2016-07-06 10:00:00"
-DoutputParaTableName="fill_missing_values_test_input_model_output"
-Dlifecycle="28"
-DoutputTableName="fill_missing_values_test_input_output"
-DinputTableName="fill_missing_values_test_input";

drop table if exists fill_missing_values_test_input_output_using_model;
drop table if exists fill_missing_values_test_input_output_using_model_model_output;
PAI -name FillMissingValues
-project algo_public
-DoutputParaTableName="fill_missing_values_test_input_output_using_model_model_output"
-DinputParaTableName="fill_missing_values_test_input_model_output"
-Dlifecycle="28"
-DoutputTableName="fill_missing_values_test_input_output_using_model"
-DinputTableName="fill_missing_values_test_input";
```

运行结果

fill_missing_values_test_input_output

```
+------------+------------+------------+-------------+--------------+
| col_string | col_bigint | col_double | col_boolean | col_datetime |
+------------+------------+------------+-------------+--------------+
| 04 | 13 | 10.4 | true | 2016-07-05 10:00:00 |
| 02 | 14 | 10.3 | true | 2016-07-03 10:00:00 |
| 03 | 12 | 10.3 | false | 2016-07-04 10:00:00 |
| str_type_empty | 11 | 10.2 | false | 2016-07-02 10:00:00 |
| 01 | 10 | 10.1 | true | 2016-07-01 10:00:00 |
| 05 | 14 | 10.5 | true | 2016-07-06 10:00:00 |
+------------+------------+------------+-------------+--------------+
```

fill_missing_values_test_input_model_output

```
+------------+------------+
| feature | json |
+------------+------------+
| col_string | {"name": "fillMissingValues", "type": "string", "paras":{"missing_value_type": "null-empty",
"replaced_value": "str_type_empty"}} |
| col_bigint | {"name": "fillMissingValues", "type": "bigint", "paras":{"missing_value_type": "null", "replaced_value":
```

```
14}} |
| col_double | {"name": "fillMissingValues", "type": "double", "paras":{"missing_value_type": "null", "replaced_value":
10.3}} |
| col_boolean | {"name": "fillMissingValues", "type": "boolean", "paras":{"missing_value_type": "null",
"replaced_value": 1}} |
| col_datetime | {"name": "fillMissingValues", "type": "datetime", "paras":{"missing_value_type": "null",
"replaced_value": 1467770400000}} |
+------------+------------+
```

fill_missing_values_test_input_output_using_model

```
+------------+------------+------------+-------------+--------------+
| col_string | col_bigint | col_double | col_boolean | col_datetime |
+------------+------------+------------+-------------+--------------+
| 04 | 13 | 10.4 | true | 2016-07-05 10:00:00 |
| 02 | 14 | 10.3 | true | 2016-07-03 10:00:00 |
| 03 | 12 | 10.3 | false | 2016-07-04 10:00:00 |
| str_type_empty | 11 | 10.2 | false | 2016-07-02 10:00:00 |
| 01 | 10 | 10.1 | true | 2016-07-01 10:00:00 |
| 05 | 14 | 10.5 | true | 2016-07-06 10:00:00 |
+------------+------------+------------+-------------+--------------+
```

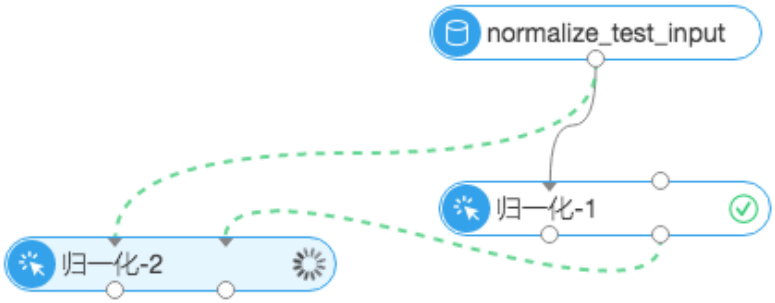
fill_missing_values_test_input_output_using_model_model_output

```
+------------+------------+
| feature | json |
+------------+------------+
| col_string | {"name": "fillMissingValues", "type": "string", "paras":{"missing_value_type": "null-empty",
"replaced_value": "str_type_empty"}} |
| col_bigint | {"name": "fillMissingValues", "type": "bigint", "paras":{"missing_value_type": "null", "replaced_value":
14}} |
| col_double | {"name": "fillMissingValues", "type": "double", "paras":{"missing_value_type": "null", "replaced_value":
10.3}} |
| col_boolean | {"name": "fillMissingValues", "type": "boolean", "paras":{"missing_value_type": "null",
"replaced_value": 1}} |
| col_datetime | {"name": "fillMissingValues", "type": "datetime", "paras":{"missing_value_type": "null",
"replaced_value": 1467770400000}} |
+------------+------------+
```

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-MinValue)/(MaxValue-MinValue)，MaxValue、MinValue分别为样本的最大值和最小值。
- 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。

归一化界面



两个输入桩依次对应参数为：

inputTableName 输入表，即要归一化的表

inputParaTableName 配置输入表，即归一化节点生成的参数列表，通过此参数，可以将一张表的配置参数应用到一张新的表

两个输出桩依次对应参数为：

outputTableName 输出表，即归一化完成的表

outputParaTableName 输出配置表，用于应用到其他的数据集上

归一化参数界面

- 保留原始列对应参数keepOriginal

PAI 命令

```
PAI -name Normalize -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
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，keepOriginal=false时，全部列保留且不重命名)	-	默认为false
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

实例

测试数据

新建数据SQL

```
drop table if exists normalize_test_input;
create table normalize_test_input(
col_string string,
col_bigint bigint,
col_double double,
```

```
col_boolean boolean,
col_datetime datetime);
insert overwrite table normalize_test_input
select
*
from
(
select
'01' as col_string,
10 as col_bigint,
10.1 as col_double,
True as col_boolean,
cast('2016-07-01 10:00:00' as datetime) as col_datetime
from dual
union all
select
cast(null as string) as col_string,
11 as col_bigint,
10.2 as col_double,
False as col_boolean,
cast('2016-07-02 10:00:00' as datetime) as col_datetime
from dual
union all
select
'02' as col_string,
cast(null as bigint) as col_bigint,
10.3 as col_double,
True as col_boolean,
cast('2016-07-03 10:00:00' as datetime) as col_datetime
from dual
union all
select
'03' as col_string,
12 as col_bigint,
cast(null as double) as col_double,
False as col_boolean,
cast('2016-07-04 10:00:00' as datetime) as col_datetime
from dual
union all
select
'04' as col_string,
13 as col_bigint,
10.4 as col_double,
cast(null as boolean) as col_boolean,
cast('2016-07-05 10:00:00' as datetime) as col_datetime
from dual
union all
select
'05' as col_string,
14 as col_bigint,
10.5 as col_double,
True as col_boolean,
cast(null as datetime) as col_datetime
from dual
) tmp;
```

输入数据说明

```
+------------+------------+------------+-------------+--------------+
| col_string | col_bigint | col_double | col_boolean | col_datetime |
+------------+------------+------------+-------------+--------------+
| 01 | 10 | 10.1 | true | 2016-07-01 10:00:00 |
| NULL | 11 | 10.2 | false | 2016-07-02 10:00:00 |
| 02 | NULL | 10.3 | true | 2016-07-03 10:00:00 |
| 03 | 12 | NULL | false | 2016-07-04 10:00:00 |
| 04 | 13 | 10.4 | NULL | 2016-07-05 10:00:00 |
| 05 | 14 | 10.5 | true | NULL |
+------------+------------+------------+-------------+--------------+
```

运行命令

```
drop table if exists normalize_test_input_output;
drop table if exists normalize_test_input_model_output;
PAI -name Normalize
-project algo_public
-DoutputParaTableName="normalize_test_input_model_output"
-Dlifecycle="28"
-DoutputTableName="normalize_test_input_output"
-DinputTableName="normalize_test_input"
-DselectedColNames="col_double,col_bigint"
-DkeepOriginal="true";

drop table if exists normalize_test_input_output_using_model;
drop table if exists normalize_test_input_output_using_model_model_output;
PAI -name Normalize
-project algo_public
-DoutputParaTableName="normalize_test_input_output_using_model_model_output"
-DinputParaTableName="normalize_test_input_model_output"
-Dlifecycle="28"
-DoutputTableName="normalize_test_input_output_using_model"
-DinputTableName="normalize_test_input";
```

运行结果

normalize_test_input_output

```
+------------+------------+------------+-------------+--------------+----------------------+----------------------+
| col_string | col_bigint | col_double | col_boolean | col_datetime | normalized_col_bigint | normalized_col_double |
+------------+------------+------------+-------------+--------------+----------------------+----------------------+
| 01 | 10 | 10.1 | true | 2016-07-01 10:00:00 | 0.0 | 0.0 |
| NULL | 11 | 10.2 | false | 2016-07-02 10:00:00 | 0.25 | 0.2499999999999989 |
| 02 | NULL | 10.3 | true | 2016-07-03 10:00:00 | NULL | 0.5000000000000022 |
| 03 | 12 | NULL | false | 2016-07-04 10:00:00 | 0.5 | NULL |
| 04 | 13 | 10.4 | NULL | 2016-07-05 10:00:00 | 0.75 | 0.7500000000000011 |
| 05 | 14 | 10.5 | true | NULL | 1.0 | 1.0 |
+------------+------------+------------+-------------+--------------+----------------------+----------------------+
```

normalize_test_input_model_output

```
+------------+------------+
| feature | json |
+------------+------------+
| col_bigint | {"name": "normalize", "type":"bigint", "paras":{"min":10, "max": 14}} |
| col_double | {"name": "normalize", "type":"double", "paras":{"min":10.1, "max": 10.5}} |
+------------+------------+
```

normalize_test_input_output_using_model

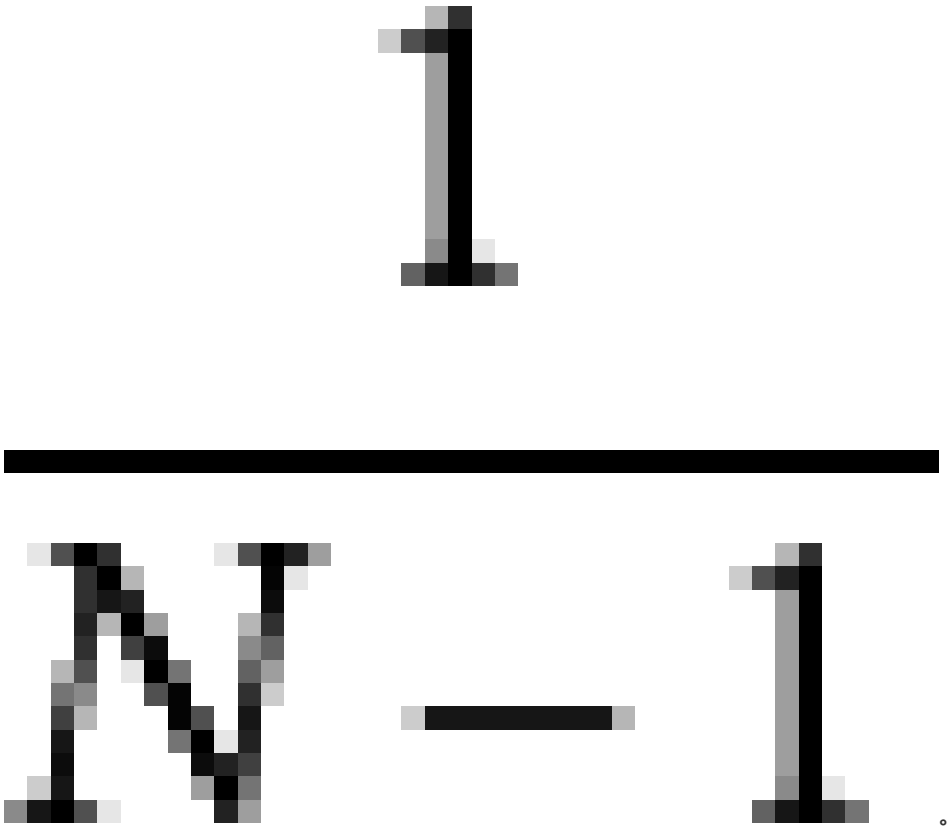
```
+------------+------------+------------+-------------+--------------+
| col_string | col_bigint | col_double | col_boolean | col_datetime |
+------------+------------+------------+-------------+--------------+
| 01 | 0.0 | 0.0 | true | 2016-07-01 10:00:00 |
| NULL | 0.25 | 0.2499999999999989 | false | 2016-07-02 10:00:00 |
| 02 | NULL | 0.5000000000000022 | true | 2016-07-03 10:00:00 |
| 03 | 0.5 | NULL | false | 2016-07-04 10:00:00 |
| 04 | 0.75 | 0.7500000000000011 | NULL | 2016-07-05 10:00:00 |
| 05 | 1.0 | 1.0 | true | NULL |
+------------+------------+------------+-------------+--------------+
```

normalize_test_input_output_using_model_model_output

```
+------------+------------+
| feature | json |
+------------+------------+
| col_bigint | {"name": "normalize", "type":"bigint", "paras":{"min":10, "max": 14}} |
| col_double | {"name": "normalize", "type":"double", "paras":{"min":10.1, "max": 10.5}} |
+------------+------------+
```

标准化

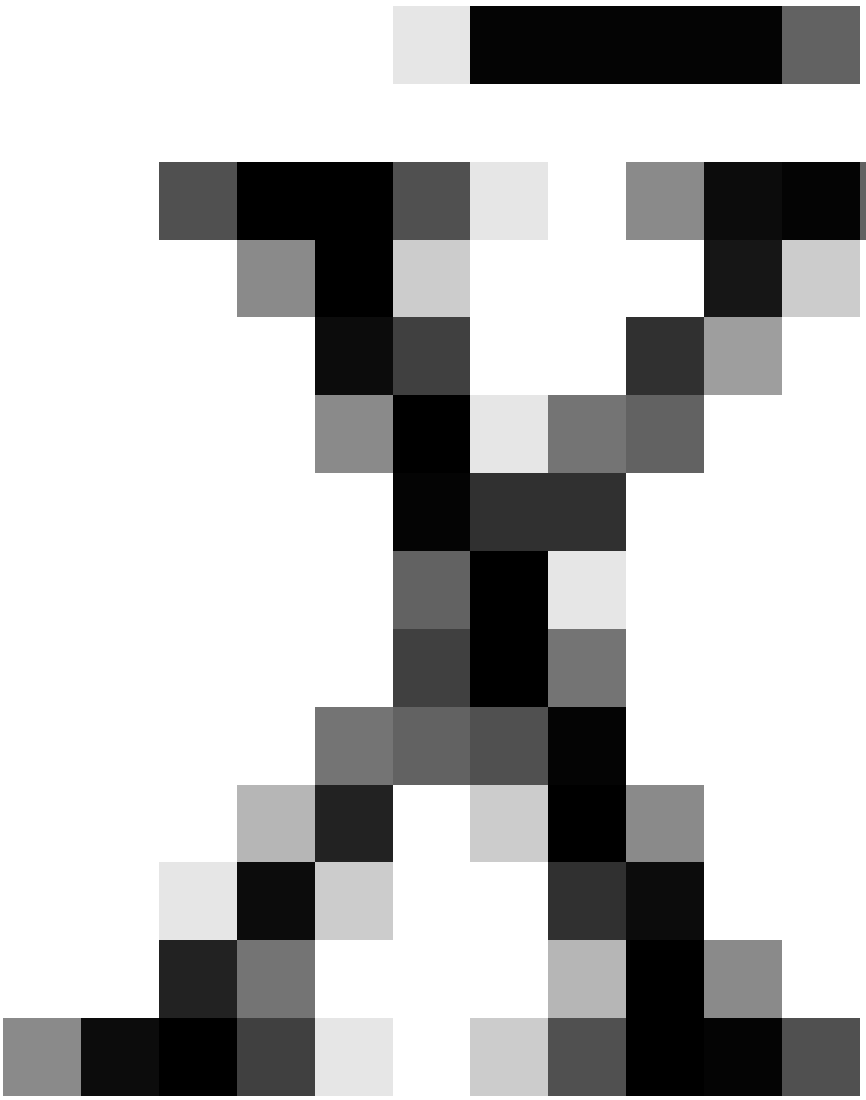
- 对一个表的某一列或多列，进行标准化处理，产生的数据存入新表中。
- 标准化所使用的公式：(X - Mean) / (standard deviation)。
- Mean：样本平均值。
- standard deviation：样本标准偏差，针对从总体抽样，利用样本来计算总体偏差，为了使算出的值与总体水平更接近，就必须将算出的标准偏差的值适度放大，即，



。

- 样本标准偏差公式：

$$S = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(X_i - \bar{X})^2}$$

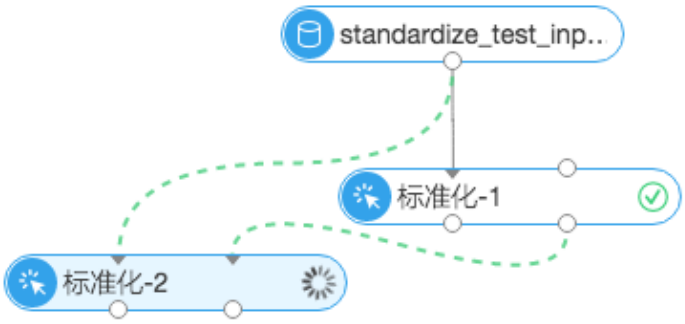


代表所采用的样本

X1，X2，...，Xn的均值。

- 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标准化的列，目前支持double类型与bigint类型

标准化界面



两个输入桩依次对应参数为：

inputTableName 输入表，即要标准化的表

inputParaTableName 配置输入表，即标准化节点生成的参数列表，通过此参数，可以将一张表的配置参数应用到一张新的表

两个输出桩依次对应参数为：

outputTableName 输出表，即标准化完成的表

outputParaTableName 输出配置表，用于应用到其他的数据集上

标准化参数界面

- 保留原始列对应参数keepOriginal

PAI 命令

```
PAI -name Standardize -project algo_public -DkeepOriginal="false" -DoutputTableName="test_5" \
-DinputPartitions="pt=20150501" -DinputTableName="bank_data_partition" -
DselectedColNames="euribor3m,pdays";
```

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
keepOriginal	可选，是否保留原始列(keepOriginal=true时，处理过的列重命名	-	默认为false

	（”stdized_”前缀），原始列保留，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

实例

测试数据

新建数据SQL

```
drop table if exists standardize_test_input;
create table standardize_test_input(
col_string string,
col_bigint bigint,
col_double double,
col_boolean boolean,
col_datetime datetime);
insert overwrite table standardize_test_input
select
*
from
(
select
'01' as col_string,
10 as col_bigint,
10.1 as col_double,
True as col_boolean,
cast('2016-07-01 10:00:00' as datetime) as col_datetime
from dual
union all
select
cast(null as string) as col_string,
11 as col_bigint,
10.2 as col_double,
False as col_boolean,
cast('2016-07-02 10:00:00' as datetime) as col_datetime
from dual
union all
select
'02' as col_string,
cast(null as bigint) as col_bigint,
10.3 as col_double,
True as col_boolean,
cast('2016-07-03 10:00:00' as datetime) as col_datetime
from dual
```

```
union all
select
'03' as col_string,
12 as col_bigint,
cast(null as double) as col_double,
False as col_boolean,
cast('2016-07-04 10:00:00' as datetime) as col_datetime
from dual
union all
select
'04' as col_string,
13 as col_bigint,
10.4 as col_double,
cast(null as boolean) as col_boolean,
cast('2016-07-05 10:00:00' as datetime) as col_datetime
from dual
union all
select
'05' as col_string,
14 as col_bigint,
10.5 as col_double,
True as col_boolean,
cast(null as datetime) as col_datetime
from dual
) tmp;
```

输入数据说明

```
+------------+------------+------------+-------------+--------------+
| col_string | col_bigint | col_double | col_boolean | col_datetime |
+------------+------------+------------+-------------+--------------+
| 01 | 10 | 10.1 | true | 2016-07-01 10:00:00 |
| NULL | 11 | 10.2 | false | 2016-07-02 10:00:00 |
| 02 | NULL | 10.3 | true | 2016-07-03 10:00:00 |
| 03 | 12 | NULL | false | 2016-07-04 10:00:00 |
| 04 | 13 | 10.4 | NULL | 2016-07-05 10:00:00 |
| 05 | 14 | 10.5 | true | NULL |
+------------+------------+------------+-------------+--------------+
```

运行命令

```
drop table if exists standardize_test_input_output;
drop table if exists standardize_test_input_model_output;
PAI -name Standardize
-project algo_public
-DoutputParaTableName="standardize_test_input_model_output"
-Dlifecycle="28"
-DoutputTableName="standardize_test_input_output"
-DinputTableName="standardize_test_input"
-DselectedColNames="col_double,col_bigint"
-DkeepOriginal="true";

drop table if exists standardize_test_input_output_using_model;
drop table if exists standardize_test_input_output_using_model_model_output;
```

```
PAI -name Standardize
-project algo_public
-DoutputParaTableName="standardize_test_input_output_using_model_model_output"
-DinputParaTableName="standardize_test_input_model_output"
-Dlifecycle="28"
-DoutputTableName="standardize_test_input_output_using_model"
-DinputTableName="standardize_test_input";
```

运行结果

standardize_test_input_output

```
+------------+------------+------------+-------------+--------------+--------------------+--------------------+
| col_string | col_bigint | col_double | col_boolean | col_datetime | stdized_col_bigint | stdized_col_double |
+------------+------------+------------+-------------+--------------+--------------------+--------------------+
| 01 | 10 | 10.1 | true | 2016-07-01 10:00:00 | -1.2649110640673518 | -1.2649110640683832 |
| NULL | 11 | 10.2 | false | 2016-07-02 10:00:00 | -0.6324555320336759 | -0.6324555320341972 |
| 02 | NULL | 10.3 | true | 2016-07-03 10:00:00 | NULL | 0.0 |
| 03 | 12 | NULL | false | 2016-07-04 10:00:00 | 0.0 | NULL |
| 04 | 13 | 10.4 | NULL | 2016-07-05 10:00:00 | 0.6324555320336759 | 0.6324555320341859 |
| 05 | 14 | 10.5 | true | NULL | 1.2649110640673518 | 1.2649110640683718 |
+------------+------------+------------+-------------+--------------+--------------------+--------------------+
```

standardize_test_input_model_output

```
+------------+------------+
| feature | json |
+------------+------------+
| col_bigint | {"name": "standardize", "type":"bigint", "paras":{"mean":12, "std": 1.58113883008419}} |
| col_double | {"name": "standardize", "type":"double", "paras":{"mean":10.3, "std": 0.1581138830082909}} |
+------------+------------+
```

standardize_test_input_output_using_model

```
+------------+------------+------------+-------------+--------------+
| col_string | col_bigint | col_double | col_boolean | col_datetime |
+------------+------------+------------+-------------+--------------+
| 01 | -1.2649110640673515 | -1.264911064068383 | true | 2016-07-01 10:00:00 |
| NULL | -0.6324555320336758 | -0.6324555320341971 | false | 2016-07-02 10:00:00 |
| 02 | NULL | 0.0 | true | 2016-07-03 10:00:00 |
| 03 | 0.0 | NULL | false | 2016-07-04 10:00:00 |
| 04 | 0.6324555320336758 | 0.6324555320341858 | NULL | 2016-07-05 10:00:00 |
| 05 | 1.2649110640673515 | 1.2649110640683716 | true | NULL |
+------------+------------+------------+-------------+--------------+
```

standardize_test_input_output_using_model_model_output

```
+------------+------------+
| feature | json |
+------------+------------+
| col_bigint | {"name": "standardize", "type":"bigint", "paras":{"mean":12, "std": 1.58113883008419}} |
```

```
| col_double | {"name": "standardize", "type":"double", "paras":{"mean":10.3, "std": 0.1581138830082909}} |
+------------+------------+
```

类型转换

组件功能介绍

将表的字段类型转成另一个类型

PAI 命令实例

```
PAI -name type_transform
-project algo_public
-DinputTable="pai_dense"
-DselectedCols="gail,loss,work_year"
-Dpre_type="double"
-Dnew_type="bigint"
-DoutputTable="pai_temp_2250_20272_1"
-Dlifecycle="28"
```

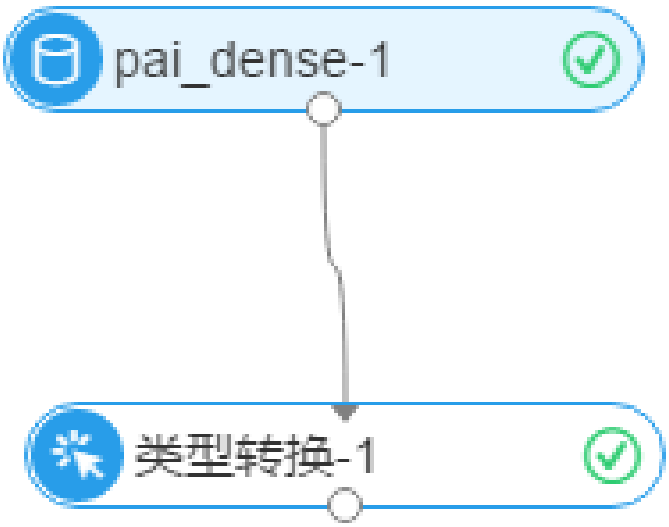
算法参数

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，类型转换的结果表	-	-
selectedCols	必选，需要类型转换特征列，必须是同一个类型	-	-
pre_type	原字段类型，必须与selectedCols勾选的字段类型一致，否则会报字段类型不一致错误	-	-
new_type	新的字段类型，用户设置		100

lifecycle	outputTable结果表生命周期，可选，默认7		7

使用DEMO

1. 拖拽一个读数据表组件，配置数据表pai_dense结构如下



Field	Type	Label	Comment
age	bigint		
workclass	string		
fwlght	bigint		
edu	string		
edu_num	bigint		
married	string		
c	string		
family	string		
race	string		
sex	string		
gail	double		
loss	double		
work_year	double		
country	string		
income	string		

2. 拖拽一个类型转换组件, 在右侧参数配置栏中：勾选需要转换的特征，比如下图勾选了3个原数据类型为double的列(主要double类型必须与勾选的字段类型一致)，转换成bigint类型

3. 右键点击运行后，可以看到输出结果表字段类型变化如下

```
+-------------------------------------------------------------------------------+
| Field           | Type       | Label | Comment                                |
+-------------------------------------------------------------------------------+
| age             | bigint     |       |                                        |
| workclass       | string     |       |                                        |
| fwlght          | bigint     |       |                                        |
| edu             | string     |       |                                        |
| edu_num         | bigint     |       |                                        |
| married         | string     |       |                                        |
| c               | string     |       |                                        |
| family          | string     |       |                                        |
| race            | string     |       |                                        |
| sex             | string     |       |                                        |
| gail            | bigint     |       |                                        |
| loss            | bigint     |       |                                        |
| work_year       | bigint     |       |                                        |
| country         | string     |       |                                        |
| income          | string     |       |                                        |
+-------------------------------------------------------------------------------+
```

KV2Table

给定kv（key:value）格式，转成普通表格式，key转换成表的某列名，value转成该列在对应行的值。

kv表格式定义: Key是列名的index，value支持bigint或double，这样能直接作为稀疏格式输入其他包括逻辑回归、线性回归等算法组件，key也支持string类型。在该组件中可以输入用户定义的key_map表，是列名和key的映射，但无论是否输入这张表，该组件都会输出key_map表记录转化后列名和key的映射。

kv
1:10;2:20;3:30

KeyMap表格式定义：包含列名和index的映射以及类型信息的三元组表：col_name，col_index，col_datatype，这三列类型要求是string，其中col_datatype缺失时默认值为“double”。

col_name	col_index	col_datatype
col1	1	bigint
col2	2	double

PAI命令行

```
PAI -name KVToTable
-project algo_public
-DinputTableName=test
-DoutputTableName=test_out
-DoutputKeyMapTableName=test_keymap_out
-DkvColName=kv;
```

参数说明

参数名称	参数描述	取值范围	是否必选，默认值
inputTableName	输入表名	-	必选，不能为空表
kvColName	kv列名	限选一列	必选
outputTableName	输出表名	-	必选

outputKeyMapTableName	输出索引表名	-	必选
inputKeyMapTableName	输入索引表名	-	可选，“”
appendColName	附加列名	可选多列	可选，“”
inputTablePartitions	输入表分区	-	可选，” ”
kvDelimiter	key和value之间分隔符	-	可选，默认”:”
itemDelimiter	kv对之间分隔符	-	可选，默认”,”
top1200	是否只截取前1200列	-	可选，默认true，该选项为false时，超过最大列数会报错
lifecycle	生命周期	>=-1整数	可选，默认-1，不设生命周期
coreNum	核数目	>0整数	可选，默认-1，会根据输入数据量计算所起instance的数量
memSizePerCore	内存数	(100,64*1024)	可选，默认-1，会根据输入数据量计算所需内存大小

具体示例

数据生成

```
    drop table if exists test;
create table test as
select
*
from
(
select '1:1,2:2,3:-3.3' as kv from dual
union all
select '1:10,2:20,3:-33.3' as kv from dual
) tmp;
```

PAI命令行

```
 PAI -name KVToTable
-project algo_public
-DinputTableName=test
-DoutputTableName=test_out
-DoutputKeyMapTableName=test_keymap_out
-DkvColName=kv;
```

输出说明

输出表

```
+------------+------------+------------+
| kv_1 | kv_2 | kv_3 |
+------------+------------+------------+
| 1.0 | 2.0 | -3.3 |
| 10.0 | 20.0 | -33.3 |
+------------+------------+------------+
```

输出映射表

```
+------------+------------+------------+
| col_name | col_index | col_type |
+------------+------------+------------+
| kv_1 | 1 | double |
| kv_2 | 2 | double |
| kv_3 | 3 | double |
+------------+------------+------------+
```

算法规模

转化后的列包含append列和kv所转化的列，先输出kv列再输出append列，当总列数超过odps最大列数限制，输出top1200选项为True，则输出最大列数，否则报错，目前odps的最大列数为1200列

数据量不超过1亿条记录

常见问题

若有输入key_map表，转化的列是key_map表中的key和kv表中的key的交集

转化的列类型只支持数值类型

若有输入key_map表，转化后key列类型和key_map表中一致，若无，类型为double

没有输入key*map表，转化后key列名的命名规则为：kv列的列名+“”*+key， 若key中包含”%&()*+-./;<>=?”中任一字符，都不符合odps的命名规则，会报错

列名冲突规则：若指定了append列，且append列名和转化后key列名相同，会报错

同一行有重复key处理：value值相加

列名长度超过128个字符将被截断成128个字符

特征工程

目录

主成分分析

- 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，降维后数据信息保留的百分比	-	0.9
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...	-67.75761249391427	-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
91.83250242...	-42.22581270993...	-69.85867560905872...	-78.51539738335026...

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...	-0.3068447200 53...	-0.38529019595...	-0.1519122...	0.0083997057...	1.33346902 3...	0.190495574717606...	0.7425523616980203
prin2	0.1344...	-0.248552...	-0.1027813425...	-0.3555146611 17...	0.882898436451...	0.01908078...	-0.057503451...	0.953306884...	0.136186697819829...	0.8787390595178498
prin3	-0.216...	0.2034146...	0.07125284421...	-0.7329421363 87...	-0.16599040564...	0.21796215...	0.5426739826...	0.426458636...	0.060922662369309...	0.939661721887 1589

特征尺度变换

功能说明

- 支持常见的 **尺度变化函数** log2,log10,ln,abs,sqrt。支持 稠密或稀疏

PAI 命令

```
PAI -name fe_scale_runner -project algo_public
-Dlifecycle=28
-DscaleMethod=log2
-DscaleCols=nr_employed
-DinputTable=pai_dense_10_1
-DoutputTable=pai_temp_2262_20380_1;
```

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	-	输入表的所有partition

	name1=value1/name2=value2；如果指定多个分区，中间用’,’分开		
outputTable	必选，尺度缩放后结果表	-	-
scaleCols	必选，勾选需要缩放的特征，稀疏特征自动化筛选，只能勾选数值类特征		-
labelCol	可选, 标签字段,如果设该签列，可视化特征到目标变量的x-y分布直方图	-	-
categoryCols	可选，将勾选的字段视作枚举特征处理，不支持缩放		“”
scaleMethod	可选，缩放方法，默认log2， 支持log2,log10,ln,abs,sqrt		SameDistance
scaleTopN	当scaleCols没有勾选时，自动挑选的TopN个需要缩放特征特征，默认10		10
isSparse	是否是k:v的稀疏特征，可选，默认稠密数据		100
itemSpliter	稀疏特征item分隔符，可选，默认逗号		“,”
kvSpliter	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

实例

输入数据

```
create table if not exists pai_dense_10_1 as
select
nr_employed
from bank_data limit 10;
```

参数配置

勾选nr_employed 做特征尺度变化的特征，只支持数值类特征尺度变化函数，勾选log2

运行结果

nr_employed
12.352071021075528
12.34313018339218
12.285286613666395
12.316026916036957
12.309533196497519
12.352071021075528
12.316026916036957
12.316026916036957
12.309533196497519
12.316026916036957

特征离散

离散模块功能介绍

- 支持 **稠密或稀疏的 数值类特征** 离散
- 支持 等频离散 和 等距离离散(默认)

pai 命令

```
PAI -name fe_discrete_runner -project algo_public
-DdiscreteMethod=SameFrequecy
-Dlifecycle=28
```

```
-DmaxBins=5
-DinputTable=pai_dense_10_1
-DdiscreteCols=nr_employed
-DoutputTable=pai_temp_2262_20382_1;
```

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，离散后结果表	-	-
discreteCols	必选，勾选需要离散的特征，如果是稀疏特征自动化筛选		“”
labelCol	可选, 标签字段,如果设该签列，可视化特征到目标变量的x-y分布直方图	-	-
categoryCols	可选，将勾选的字段视作枚举特征处理，不支持离散		“”
discreteMethod	可选，离散方法，默认SameDistance 目前支持:SameDistance(等距离散),SameFrequecy(等频离散)		SameDistance
discreteTopN	当discreteCols没有勾选时，自动挑选的TopN个需要离散特征特征，默认10		10
maxBins	离散区间大小，默认100		100
isSparse	是否是k:v的稀疏特征，可选，默认稠密数据		100
itemSpliter	稀疏特征item分隔符，可选，默认逗号		“,”
kvSpliter	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

建模示例

输入数据

```
create table if not exists pai_dense_10_1 as
select
nr_employed
from bank_data limit 10;
```

参数配置

输入数据为pai_dense_10_1离散特征勾选 nr_employed，用 **等距离离散** 方法离散成5个区间, 结果如下

运行结果

nr_employed
4.0
3.0
1.0
3.0
2.0
4.0
3.0
3.0

2.0
3.0

特征异常平滑

组件功能介绍

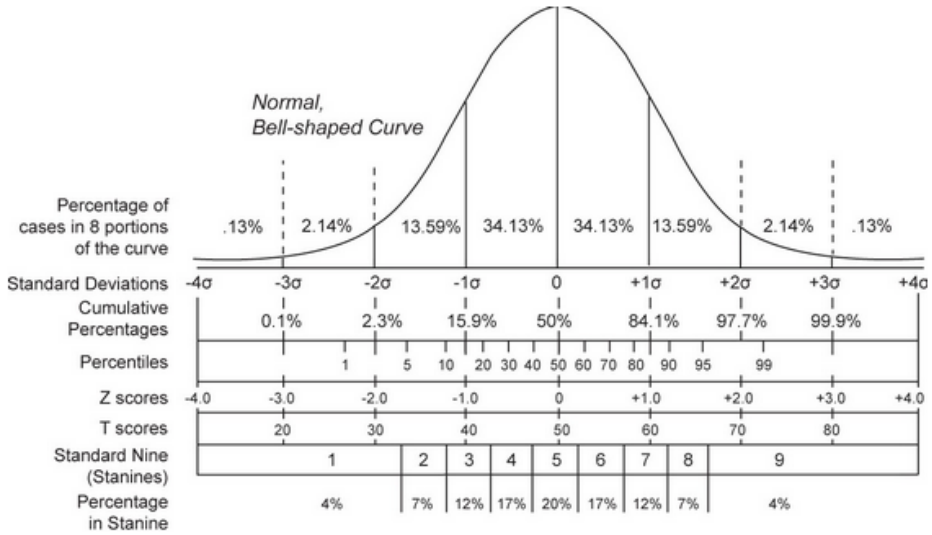
- **功能**：将输入特征中含有异常的数据平滑到一定区间,**支持稀疏和稠密**

ps: 特征平滑组件只是将异常取值的特征值修正成正常值，本身不过滤或删除任何记录，输入数据维度和条数都不变.

平滑方法介绍

- Zscore平滑：如果特征分布遵循正态分布，考虑噪音一般集在-3xalpha 和 3xalpha 之外，ZScore是将该范围数据平滑到[-3xalpha,3xalpha]。

eg: 某个特征遵循特征分布，均值为0，标准差为3，因此-10的特征值会被识别为异常而修正为-3x3+0=-9，同理,10会被修正为3x3+0



- 百分位平滑: 将数据分布在[minPer, maxPer]分位之外的数据平滑平滑到minPer/maxPer这两个分位点

eg: age特征取值0-200，设置minPer为0，maxPer为50%，那么在0-100之外的特征取值都会被修正成0或者100

- 阈值平滑: 将数据分布在[minThresh, maxThresh]之外的数据平滑到minThresh和maxThresh这两个数据点.

eg: age特征取值0-200，设置minThresh=10，maxThresh=80，那么在0-80之外的特征取值都会被修正成

0或者80

三 pai 命令

```
PAI -name fe_soften_runner -project algo_public
-DminThresh=5000 -Dlifecycle=28
-DsoftenMethod=min-max-thresh
-DsoftenCols=nr_employed
-DmaxThresh=6000
-DinputTable=pai_dense_10_1
-DoutputTable=pai_temp_2262_20381_1;
```

四 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，平滑后结果表	-	-
labelCol	可选, 标签字段,如果设该签列，可视化特征到目标变量的x-y分布直方图	-	-
categoryCols	可选，将勾选的字段视作枚举特征处理		“”
softenCols	必选，勾选需要平滑的特征，但特征是稀疏特征时自动化筛选		-
softenMethod	可选，平滑方法，默认zscore 目前支持:min-max-thresh(阈值平滑), min-max-per(百分位平滑)		zscore
softenTopN	当softenCols没有勾选时，自动挑选的TopN个需要的平滑特征特征，默认10		10
cl	置信水平，当平滑方法是zscore方生效		10

minPer	最低百分位，当平滑方法是min-max-thresh方生效		0.0
maxPer	最高百分位，当平滑方法是min-max-thresh方生效		1.0
minThresh	阈值最小值，默认-9999，表示不设置，当平滑方法是min-max-per方生效		-9999
maxThresh	阈值最大值，默认-9999，表示不设置，当平滑方法是min-max-per方生效		-9999
isSparse	是否是k:v的稀疏特征，可选，默认稠密数据		100
itemSpliter	稀疏特征item分隔符，可选，默认逗号		“,”
kvSpliter	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

五 实例

1. 输入数据

输入数据

```
create table if not exists pai_dense_10_1 as
select
nr_employed
from bank_data limit 10;
```

nr_employed
5228.1
5195.8
4991.6
5099.1
5076.2
5228.1
5099.1

5099.1
5076.2
5099.1

参数配置

平滑特征列勾选nr_employed，参数配置中选择阈值平滑，下限5000，上限6000

运行结果

nr_employed
5228.1
5195.8
5000.0
5099.1
5076.2
5228.1
5099.1
5099.1
5076.2

5099.1

随机森林特征重要性

组件功能

使用原始数据和随机森林模型，计算特征重要性.

PAI 命令

```
pai -name feature_importance -project algo_public
-DinputTableName=pai_dense_10_10
-DmodelName=xlab_m_random_forests_1_20318_v0
-DoutputTableName=erkang_test_dev.pai_temp_2252_20319_1
-DlabelColName=y
-
DfeatureColNames="pdays,previous,emp_var_rate,cons_price_idx,cons_conf_idx,euribor3m,nr_employed,age,campaign,poutcome"
-Dlifecycle=28 ;
```

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

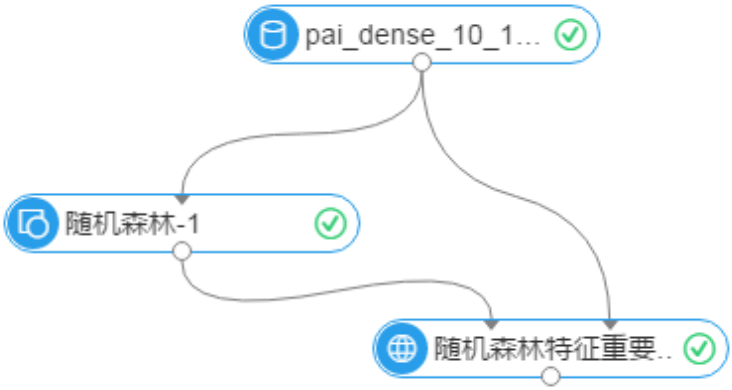
实例

训练数据

```
drop table if exists pai_dense_10_10;
creat table if not exists pai_dense_10_10 as
select
age,campaign,pdays, previous, poutcome, emp_var_rate, cons_price_idx, cons_conf_idx, euribor3m, nr_employed, y
from bank_data limit 10;
```

参数配置

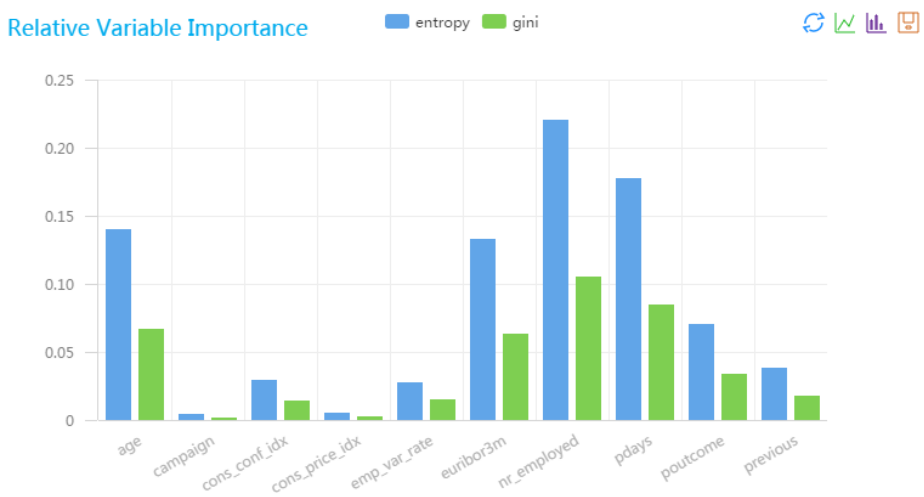
实例流程图如下，数据源为pai_dense_10_10, y列为随机森林的标签列，其他列为特征列，强制转换列勾选age和campaign，表示这两个特征当做枚举特征处理，其他采用默认参数。 运行成功如下



运行结果

colname	gini	entropy
age	0.06625000000000003	0.13978726292803723
campaign	0.0017500000000000003	0.004348515545596772
cons_conf_idx	0.013999999999999999	0.02908409497018851
cons_price_idx	0.002	0.0049804499913461255
emp_var_rate	0.014700000000000003	0.026786360680260933
euribor3m	0.06300000000000003	0.1321936348846039
nr_employed	0.10499999999999998	0.2203227248076733
pdays	0.0845	0.17750329234397513
poutcome	0.03360000000000001	0.07050327193845542
previous	0.017700000000000004	0.03810381005801592

随机森林特征重要性组件上 **右键查看可视化分析**,效果如下所示



GBDT特征重要性

组件功能

计算梯度渐进决策树(GBDT)特征重要性

PAI 命令

```
pai -name gbdt_importance -project algo_public
-DmodelName=xlab_m_GBDT_LR_1_20307_v0
-Dlifecycle=28 -DoutputTableName=pai_temp_2252_20308_1 -DlabelColName=y
-
DfeatureColNames=pdays,previous,emp_var_rate,cons_price_idx,cons_conf_idx,euribor3m,nr_employed,age,campaign
-DinputTableName=pai_dense_10_9;
```

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

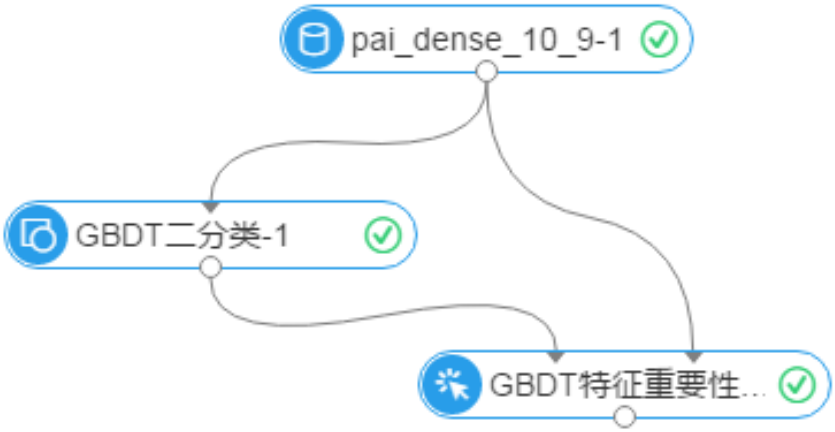
实例

输入数据

```
drop table if exists pai_dense_10_9;
create table if not exists pai_dense_10_9 as
select
age,campaign,pdays, previous, emp_var_rate, cons_price_idx, cons_conf_idx, euribor3m, nr_employed, y
from bank_data limit 10;
```

参数配置

实例流程图如下，输入数据为pai_dense_10_9, GBDT二分类组件选择标签列y，其他字段作为特征列，组件参数配置中**叶节点最小样本数**配置为1，运行

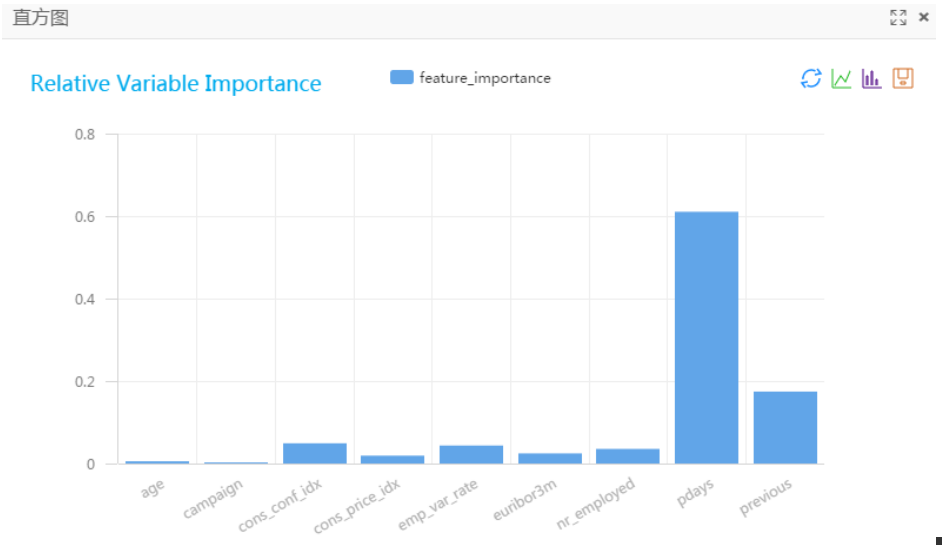


运行结果

colname	feature_importance
age	0.004667214954427797
campaign	0.001962038566773853
cons_conf_idx	0.04857761873887033
cons_price_idx	0.01925292649801252
emp_var_rate	0.044881269590771274
euribor3m	0.025034606434306696
nr_employed	0.036085457464908766

pdays	0.639121250405536
previous	0.18041761734639272

右键查看可视化分析报告



线性模型特征重要性

组件功能

计算线性模型的特征重要性，包括线性回归和二分类逻辑回归。 支持稀疏和稠密。

PAI 命令

```
PAI -name regression_feature_importance -project algo_public
-DmodelName=xlab_m_logisticregressi_20317_v0
-DoutputTableName=pai_temp_2252_20321_1
-DlabelColName=y
-
DfeatureColNames=pdays,previous,emp_var_rate,cons_price_idx,cons_conf_idx,euribor3m,nr_employed,age,campaign
-DenableSparse=false -DinputTableName=pai_dense_10_9;
```

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	-	-
modelName	必选，输入的模型名	-	-

labelColName	必选，label所在列	-	-
feaureColNames	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	-	默认选全表
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
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

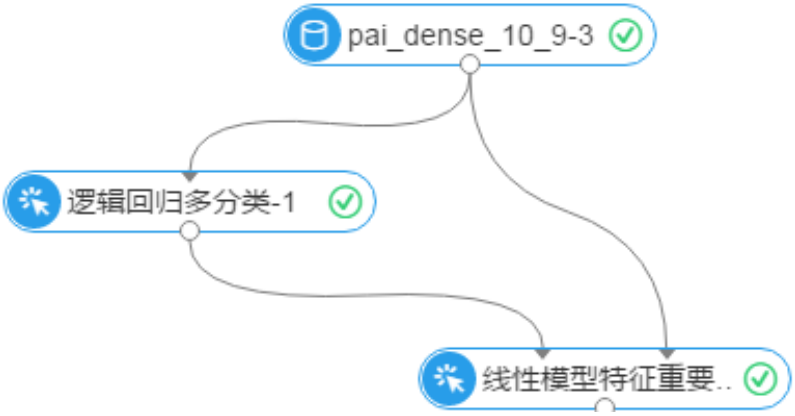
实例

输入数据

```
create table if not exists pai_dense_10_9 as
select
age,campaign,pdays, previous, emp_var_rate, cons_price_idx, cons_conf_idx, euribor3m, nr_employed, y
from bank_data limit 10;
```

参数配置

建模流程如下图示， 逻辑回归多分类组件选择标签列为y，其他字段为特征列，其他参数默认，运行



运行效果

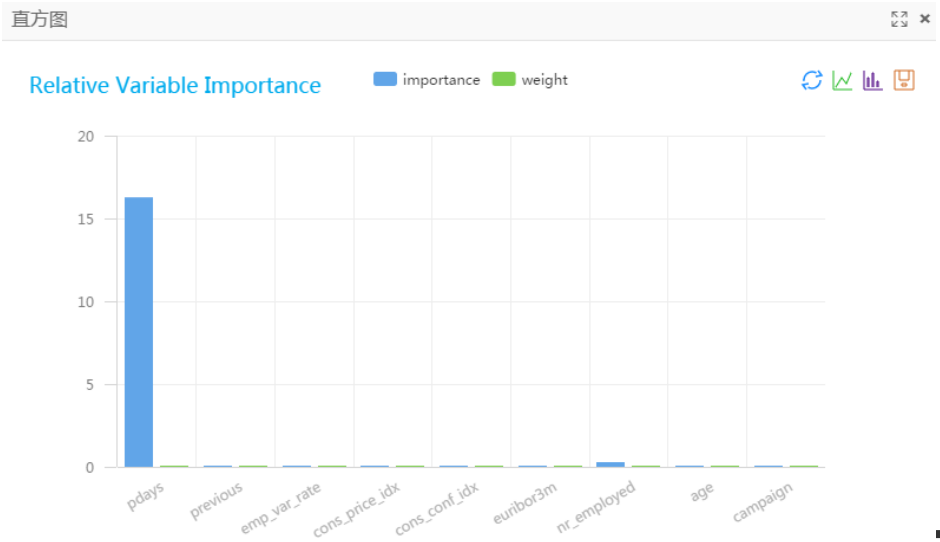
colname	weight	importance

pdays	0.033942600256583334	16.31387797440866
previous	0.00004248130342485344	0.000030038817725357177
emp_var_rate	0.00006720242617694611	0.00010554561260753949
cons_price_idx	0.00012311047229142307	0.00006581255124425219
cons_conf_idx	0.00017227965471819213	0.0008918770542818432
euribor3m	0.00006113758212679113	0.00010427128177450988
nr_employed	0.0034541377310490697	0.26048098230126043
age	0.00009618162708080744	0.0009267659744232966
campaign	0.000019142551785274455	0.00004179353660529855

指标计算公式

列名	公式	
weight	abs(w_)	
importance	abs(w_j) * STD(f_i)	

在线性模型特征重要性组件上，**右键查看可视化分析**



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
-DOutputTableName=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 |
+------------+------------+
```

基于GBDT的过滤式特征选择

组件功能

根据用户不同的特征选择方法，选择并过滤出TopN的特征数据，同时保存所有特征重要性表(右输出)。支持稀疏和稠密数据

PAI 命令

```
PAI -name fe_select_runner -project algo_public
-DfeatImportanceTable=pai_temp_2260_22603_2
-DselectMethod=iv
-DselectedCols=pdays,previous,emp_var_rate,cons_price_idx,cons_conf_idx,euribor3m,nr_employed,age,campaign
-DtopN=5
-DlabelCol=y
-DmaxBins=100
-DinputTable=pai_dense_10_9
-DoutputTable=pai_temp_2260_22603_1;
```

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(Information Value)，Gini增益，信息增益，Lasso	iv,GiniGain,InfoGain，Lasso	iv
topN	挑选的TopN个特征，如果topN答应输入特征数，则输出所有特征，可选，默认10		10
isSparse	是否是k:v的稀疏特征，可选，默认稠密数据		100
itemSpliter	稀疏特征item分隔符，可选，默认逗号		“,”
kvSpliter	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

实例

输入数据

```
create table if not exists pai_dense_10_9 as
select
```

```
age,campaign,pdays, previous, emp_var_rate, cons_price_idx, cons_conf_idx, euribor3m, nr_employed, y
from bank_data limit 10;
```

参数配置

标签列勾选y字段，其他字段勾选为特征列, 特征选择方法选IV，topN特征为5，表示过滤top5的特征

运行结果

左输出是过滤后的数据

pdays	nr_employed	emp_var_rate	cons_conf_idx	cons_price_idx	y
999.0	5228.1	1.4	-36.1	93.444	0.0
999.0	5195.8	-0.1	-42.0	93.2	0.0
6.0	4991.6	-1.7	-39.8	94.055	1.0
999.0	5099.1	-1.8	-47.1	93.075	0.0
3.0	5076.2	-2.9	-31.4	92.201	1.0
999.0	5228.1	1.4	-42.7	93.918	0.0
999.0	5099.1	-1.8	-46.2	92.893	0.0
999.0	5099.1	-1.8	-46.2	92.893	0.0
3.0	5076.2	-2.9	-40.8	92.963	1.0
999.0	5099.1	-1.8	-47.1	93.075	0.0

右输出是特征重要性表

特征重要性表字段结构如下,featureName存放特征名称，weight表示特征选择方法计算出来的权重。

featname	weight
pdays	30.675544191232486
nr_employed	29.08332850085075
emp_var_rate	29.08332850085075
cons_conf_idx	28.02710269740324
cons_price_idx	28.02710269740324
euribor3m	27.829058450563718
age	27.829058450563714
previous	14.319325030742775
campaign	10.658129656314467

weight计算公式

选择方法	weght含有
IV	信息价值
GiniGain	Gini增益
InfoGain	信息熵增益
Lasso	线性模型权重绝对值

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;
```

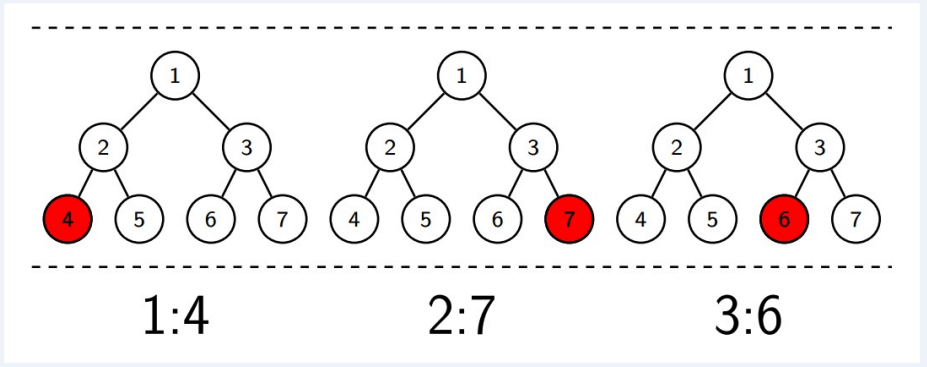
运行结果

```
+------------+------------+------------+------------+------------+------------+------------+------------+------------
+
| 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 |
+------------+------------+------------+------------+------------+------------+------------+------------+------------
+
```

特征编码

一 特征编码概念

1. 由决策树和Ensemble算法挖掘**新特征**的一种策略. 特征来自一个或多个特征组成的决策树分支，比如下图左边树的分支 节点1->节点2->节点4 形成一个特征. 很明显，该编码策略可以有效的将 非线性特征 转换为 线性特征.



二 pai 命令

```
PAI -name fe_encode_runner -project algo_public
-DinputTable="pai_temp_2159_19087_1"
-DencodeModel="xlab_m_GBDT_LR_1_19064"
-DselectedCols="pdays,previous,emp_var_rate,cons_price_idx,cons_conf_idx,euribor3m,nr_employed,age,campaign"
-DlabelCol="y"
-DoutputTable="pai_temp_2159_19061_1";
```

三 参数说明

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
encodeModel	必选，编码的输入GBDT二分类模型	-	-
outputTable	必选，尺度缩放后结果表	-	-
selectedCols	必选，勾选GBDT参与编码的特征,一般是GBDT组件的训练特征		-
labelCol	必选, 标签字段	-	-
lifecycle	结果表生命周期，可选，默认7		7

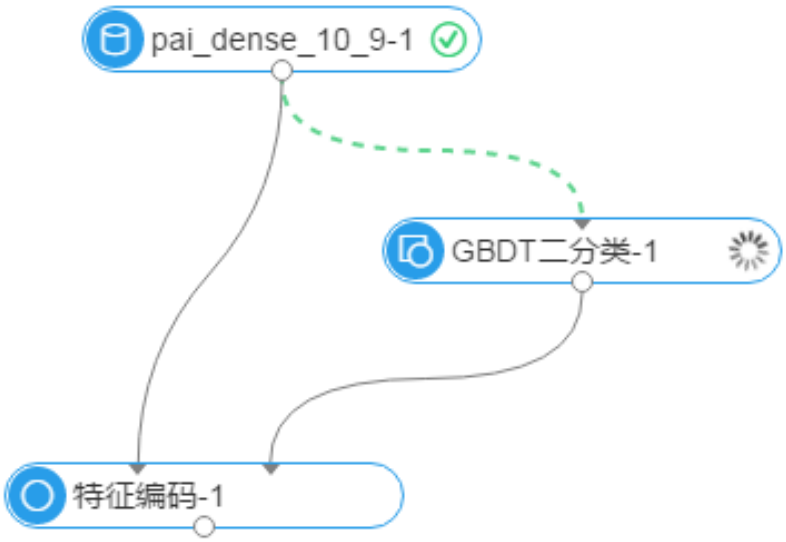
实例

输入数据

```
create table if not exists pai_dense_10_9 as
select
age,campaign,pdays, previous, emp_var_rate, cons_price_idx, cons_conf_idx, euribor3m, nr_employed, y
from bank_data limit 10;
```

参数说明

建模流程如下，一般配合GBDT二分类组件使用



出于演示方便，GBDT二分类树棵数设置为5，深度设置为3，y字段为标签列，其他字段为特征列，运行如下.

运行结果

kv	y
2:1,5:1,8:1,8:1,12:1,15:1,18:1,18:1,28:1,34:1,34:1,41:1,50:1,53:1,53:1,63:1,72:1,72:1	0.0
2:1,5:1,6:1,6:1,12:1,15:1,16:1,16:1,28:1,34:1,34:1,41:1,50:1,51:1,51:1,63:1,72:1,72:1	0.0
2:1,3:1,3:1,12:1,13:1,13:1,28:1,34:1,34:1,36:1,39:1,39:1,55:1,61:1,61:1	1.0
2:1,3:1,3:1,12:1,13:1,13:1,20:1,21:1,22:1,22:1,41:1,42:1,43:1,46:1,46:1,63:1,64:1,67:1,68:1,68:1	0.0
0:1,0:1,10:1,10:1,28:1,29:1,32:1,32:1,36:1,37:1,37:1,55:1,56:1,59:1,59:1	1.0
2:1,5:1,8:1,8:1,12:1,15:1,18:1,18:1,20:1,26:1,26:1,41:1,42:1,48:1,48:1,63:1,64:1,67:1,70:1,70:1	0.0
2:1,3:1,3:1,12:1,13:1,13:1,20:1,21:1,24:1,24:1,41:1,42:1,43:1,44:1,44:1,63:1,64:1,65:1,65:1	0.0
2:1,3:1,3:1,12:1,13:1,13:1,20:1,21:1,24:1,24:1,41:1,42:1,43:1,44:1,44:1,63:1,64:1,65:1,65:1	0.0
0:1,0:1,10:1,10:1,28:1,29:1,30:1,30:1,36:1,37:1,37:1,55:1,56:1,57:1,57:1	1.0
2:1,3:1,3:1,12:1,13:1,13:1,20:1,21:1,22:1,22:1,41:1,42:1,43:1,46:1,46:1,63:1,64:1,67:1,68:1,68:1	0.0

one-hot编码

一 功能组件

one-hot编码，也称独热编码，对于每一个特征，如果它有m个可能值，那么经过独热编码后，就变成了m个二元特征。并且，这些特征互斥，每次只有一个激活。因此，数据会变成稀疏的,输出结果也是k:v的稀疏结构。

二 PAI命令

```
PAI -name fe_binary_runner -project algo_public
-DinputTable=one_hot
-DbinaryCols=edu_num
-DlabelCol=income
-DbinaryReserve=false
-DbinStrategy=noDealStrategy
-DbinaryIndexTable=pai_temp_2458_23436_3
-DmodelTable=pai_temp_2458_23436_2
-DoutputTable=pai_temp_2458_23436_1
-Dlifecycle=28;
```

三 参数说明

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
binaryCols	必选 one-hot编码字段，必须是枚举类特征,字段类型可以是任意类型		-
binStrategy	必选，编码策略，本组件提供2种编码策略：简单二值化(noDealStrategy)，自动二值化(autoStrategy),分类标签必须设置		-
labelCol	分类标签类. **按纯度二值化策略下必选**，其他策略下可选	-	-
impurityMergeThresh	可选，按纯度二值化策略中，前后二值化特征		0.1

	合并,纯度提升阈值		
densityMergeThresh	可选，按密度二值化策略中，二值化特征密度(占比)合并的阈值		0.1
binaryReserve	可选, 是否输出表中保留原独热编码特征	-	-
outputTable	必选,one-hot后的结果表，编码结果保存在kv字段里	-	-
binaryIndexTable	必选, kv序列化表，编码特征名与kv中Key的映射关系	-	-
modelTable	必选, one-hot编码的映射逻辑,Json格式存放	-	-
lifecycle	结果表生命周期，可选，默认7		7

编码策略

- noDealStrategy： 简单二值化，比如sex特征取值 Female|Male|Unknowed. 那么二值化出来特征是sex_female, sex_male 和 sex_Unknowed
- autoStrategy：自动二值化. 在简单二值化的基础上，利用逻辑回归算法计算每个one-hot特征，将one-hot特征权重为0的特征合并成独立一项，命名other

实例

1 输入数据

edu_num	income
13	<=50K
13	<=50K
9	<=50K
7	<=50K
13	<=50K
14	<=50K
5	<=50K
9	>50K
14	>50K
13	>50K

2. 组件参数

勾选edu_num作为二值化特征， 其他全部采用默认参数，即简单二值化策略，

3. 结果

编码后结果(outputTable)

income	kv
<=50K	0:1
<=50K	0:1
<=50K	4:1
<=50K	3:1
<=50K	0:1
<=50K	1:1
<=50K	2:1
>50K	4:1
>50K	1:1
>50K	0:1

kv的序列列表(binaryIndexTable)

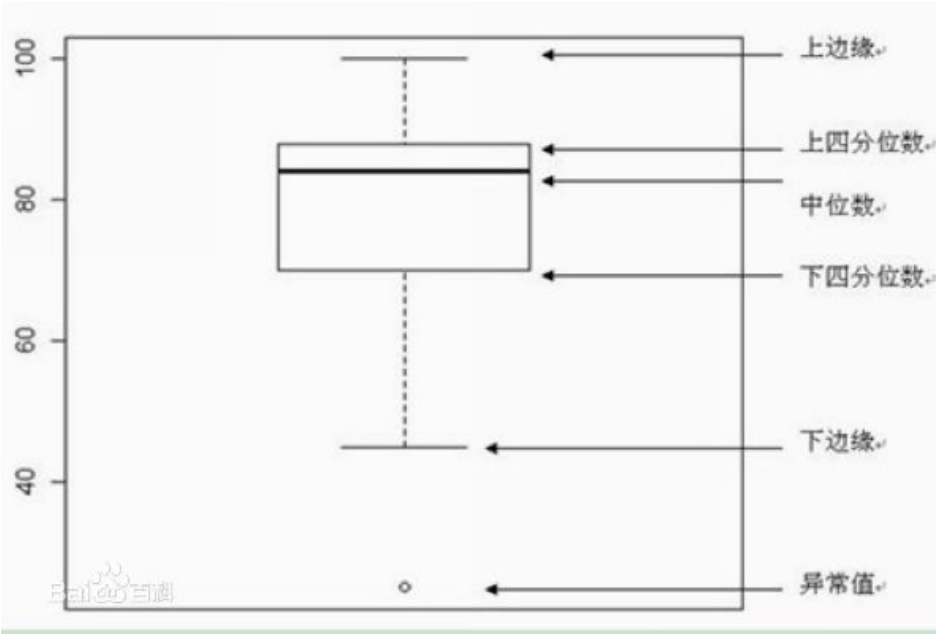
featname	featindex
edu_num	0
13	0
14	1
5	2
7	3
9	4

异常检测

一 组件功能

支持一下功能

针对连续值特征： 按箱线图最大值和最小值检测异常特征.



针对枚举值特征: 按照枚举特征的取值频率，按照阈值过滤异常特征

二 PAI 参数

```
PAI -name fe_detect_runner -project algo_public
-DselectedCols="emp_var_rate,cons_price_rate,cons_conf_idx,euribor3m,nr_employed" \
-Dlifecycle="28"
-DdetectStrategy="boxPlot"
-DmodelTable="pai_temp_2458_23565_2"
-DinputTable="pai_bank_data"
-DoutputTable="pai_temp_2458_23565_1";
```

三 参数说明

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
selectedCols	必选 输入特征，字段类型可以是任意类型		-
detectStrategy	必选，支持boxPlot和		boxPlot

	avf 选项，boxPlot是针对连续特征做检测，avf针对枚举类特征做检测		
outputTable	必选,过滤检测到的异常特征后数据集	-	-
modelTable	必选, 异常检测模型	-	-
lifecycle	结果表生命周期，可选，默认7		7

特征重要性过滤

一 组件功能

为线性特征重要性/GBDT 特征重要性/ 随机森林特征重要性等重要性评估组件提供过滤功能.支持过滤topN的特征

二 PAI 命令

```
PAI -name fe_filter_runner -project algo_public -
DselectedCols=pdays,previous,emp_var_rate,cons_price_idx,cons_conf_idx,euribor3m,nr_employed,age,campaign,poutcome
-DinputTable=pai_dense_10_10
-DweightTable=pai_temp_2252_20319_1
-DtopN=5
-DmodelTable=pai_temp_2252_20320_2
-DoutputTable=pai_temp_2252_20320_1;
```

三 组件参数说明

参数名称	参数描述	参数值可选项	默认值
inputTable	必选，输入原始数据表	-	-
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
weightTable	必选，特征重要性的权重表(即线性特征重要性/GBDT特征重要性/随机森林特征重要性	-	-

	的输出表)		
outputTable	必选, 过滤出TopN个特征的输出表	-	-
modelTable	必选，特征过滤产出的模型文件	-	-
selectedCols	可选, 默认输入表的所有字段	-	-
topN	挑选的TopN个特征，默认10		10
lifecycle	输出表生命周期，可选，默认7		7

四 建模示例

输入数据

特征重要性过滤组件左边输入数据是原始数据表，右边是特征重要性表输出是过滤后的TopN特征数据

ps: 特征重要性表有一定格式要求: 第一字段存储特征名称，第二字段存储该特征对应的权重值,一般为double类型，比如随机森林特征重要性的输出表

Field	Type	Label	Comment
colname	string		
gini	double		
entropy	double		

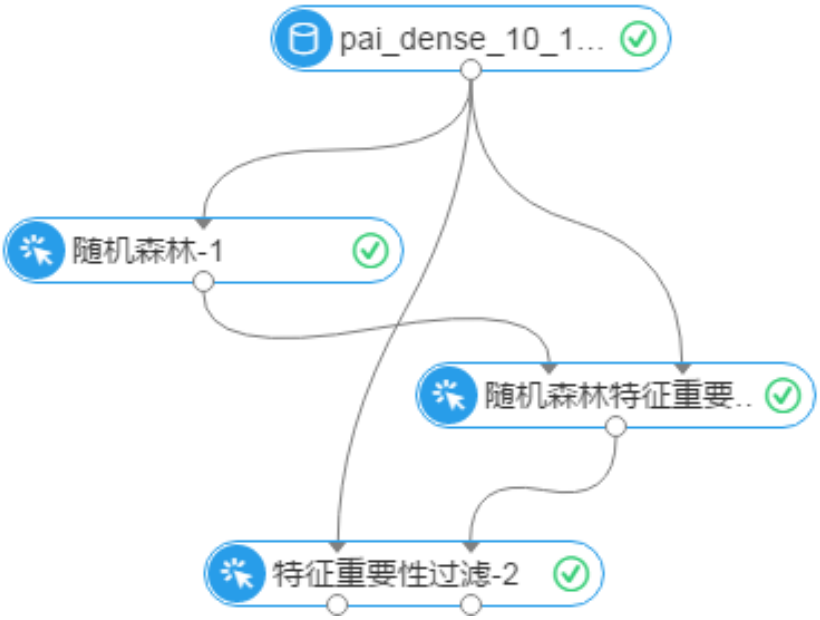
原始输入数据

```
creat table if not exists pai_dense_10_10 as
select
age,campaign,pdays, previous, poutcome, emp_var_rate, cons_price_idx, cons_conf_idx, euribor3m, nr_employed, y
from bank_data limit 10;
```

特征重要性表为随机森林特征重要性组件的输出表.

参数配置

建模流程图(**该组件一般是配合特征重要性组件使用**，比如线性特征重要性/GBDT 特征重要性/ 随机森林特征重要性)



输入数据为pai_dense_10_10

随机森林和随机森林重要性组件都是勾选y字段为标签类，其他字段为特征列

特征重要性过滤组件参数配置topN为5，表示只过滤top5的特征

运行结果

左输出数据结果

nr_employed	pdays	age	euribor3m	poutcome

5228.1	999.0	44	4.963	nonexistent
5195.8	999.0	53	4.021	nonexistent
4991.6	6.0	28	0.729	success
5099.1	999.0	39	1.405	nonexistent
5076.2	3.0	55	0.869	success
5228.1	999.0	30	4.961	nonexistent
5099.1	999.0	37	1.327	nonexistent
5099.1	999.0	39	1.313	nonexistent
5076.2	3.0	36	1.266	success
5099.1	999.0	27	1.41	failure

右输出数据为过滤模型，暂无数据

统计分析

目录

经验概率密度图

箱线图

散点图

相关系数矩阵

正态检验

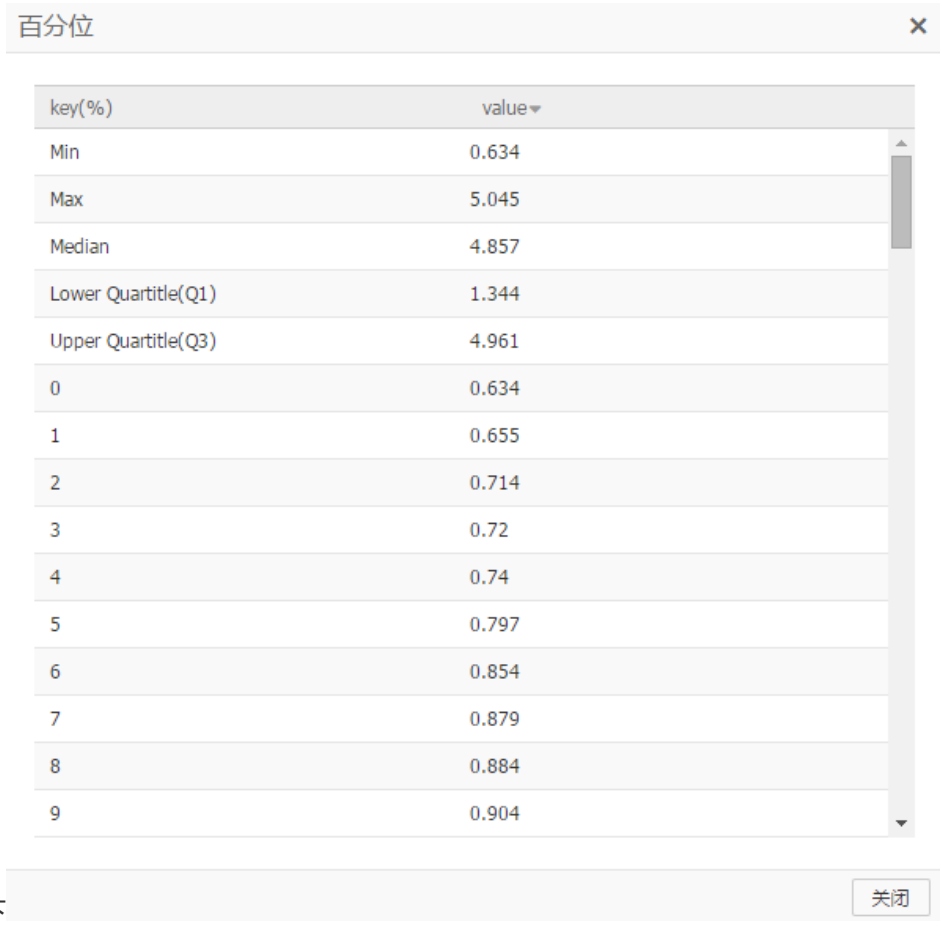
洛伦兹曲线

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

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

PAI 命令

```
PAI -name stat_summary
-project algo_public
-DinputTableName=test_data
-DoutputTableName=test_summary_out
-DinputTablePartitions="ds='20160101'"
-DselectColNames=col0,col1,col2
-Dlifecycle=1
```

参数说明

参数名称	参数描述	参数值可选项	默认值
inputTableName	必选，输入表名	-	-
outputTableName	必选，推荐结果的输出表名	-	-
inputTablePartitions	可选，输入表的分区	-	“”
selectColNames	可选，指定需要统计的列名	-	“”
lifecycle	可选，输出结果表的生命周期	-	不设生命周期
coreNum	可选，指定instance的总数	-	-1
memSizePerCore	可选，指定memory大小，范围在100~64*1024之间	-	-1

输入格式：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

输出格式：输出统计结果的全部字段如下

列名	描述
colname	列名
datatype	类型
totalcount	总数
count	非NULL数量
missingcount	NULL数量
nancount	NAN数量
positiveinfinitycount	正无穷数量
negativeinfinitycount	负无穷数量
min	最小值
max	最大值
mean	平均值
variance	方差
standarddeviation	标准差
standarderror	标准误差
skewness	偏度
kurtosis	峰度
moment2	二阶矩
moment3	三阶矩
moment4	四阶矩
centralmoment2	二阶中心距
centralmoment3	三阶中心距
centralmoment4	四阶中心距
sum	总和
sum2	平方和
sum3	立方和
sum4	四次方和

实例

测试数据

新建数据SQL

```
drop table if exists summary_test_input;
create table summary_test_input as
select
*
from
(
select 'a' as col1, 1 as col2, 0.001 as col3 from dual
union all
select 'b' as col1, 2 as col2, 100.01 as col3 from dual
) tmp;
```

运行命令

```
PAI -name stat_summary
-project algo_public
-DinputTableName=summary_test_input
-DoutputTableName=summary_test_input_out
-DselectColNames=col1,col2,col3
-Dlifecycle=1;
```

运行结果

```
    | colname    | datatype   | totalcount | count      | missingcount | nancount   | positiveinfinitycount |
negativeinfinitycount | min        | max        | mean       | variance   | standarddeviation | standarderror | skewness   |
kurtosis   | moment2    | moment3    | moment4    | centralmoment2 | centralmoment3 | centralmoment4 | sum        |
sum2       | sum3       | sum4       |
| col1 | string | 2 | 2 | 0 | 0 | 0 | 0 | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| col2 | bigint | 2 | 2 | 0 | 0 | 0 | 0 | 1 | 2 | 1.5 | 0.5 | 0.7071067811865476 | 0.5 | 0 | -2 | 2.5 | 4.5 | 8.5 | 0.25 | 0 | 0.0625 |
3 | 5 | 9 | 17 |
| col3 | double | 2 | 2 | 0 | 0 | 0 | 0 | 0.001 | 100.01 | 50.0055 | 5000.900040500001 | 70.71704207968544 |
50.00450000000001 | 2.327677906939552e-16 | -1.999999999999999 | 5001.000050500001 | 500150.0150005006 |
50020003.00020002 | 2500.45002025 | 2.91038304567337e-11 | 6252250.303768232 | 100.011 | 10002.000101 |
1000300.030001001 | 100040006.0004 |
```

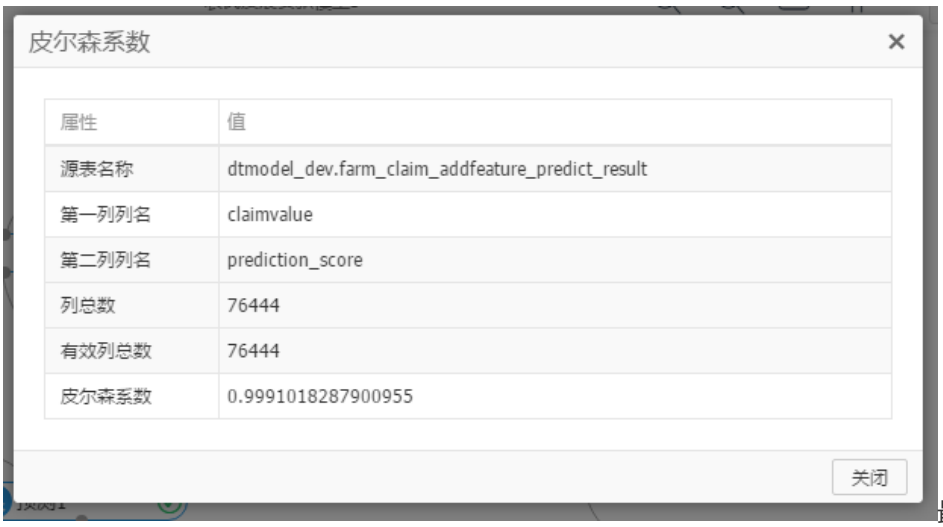
皮尔森系数

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=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
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 可调节步长大小，以及滑动查看直方图

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，infomation gain，infomation gain ratio等指标。其中计算每个离散值对应的gini，entropy，计算单列对应的gini gain，infomation gain，infomation gain ratio。

$$I_G(f) = \sum_{i=1}^{m} f_i(1 - fi)$$

gini index:

$$I_E(f) = -\sum_{i=1}^{m} f_i log_2 f_i$$

entropy:

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

示例

测试数据

新建数据SQL

```
drop table if exists enum_feature_selection_test_input;
create table enum_feature_selection_test_input
as
select
*
from
(
select
'00' as col_string,
1 as col_bigint,
0.0 as col_double
from dual
union all
select
cast(null as string) as col_string,
0 as col_bigint,
0.0 as col_double
from dual
union all
select
'01' as col_string,
0 as col_bigint,
1.0 as col_double
from dual
union all
select
'01' as col_string,
1 as col_bigint,
cast(null as double) as col_double
from dual
union all
select
'01' as col_string,
1 as col_bigint,
1.0 as col_double
from dual
union all
select
'00' as col_string,
0 as col_bigint,
0.0 as col_double
from dual
) tmp;
```

输入数据说明

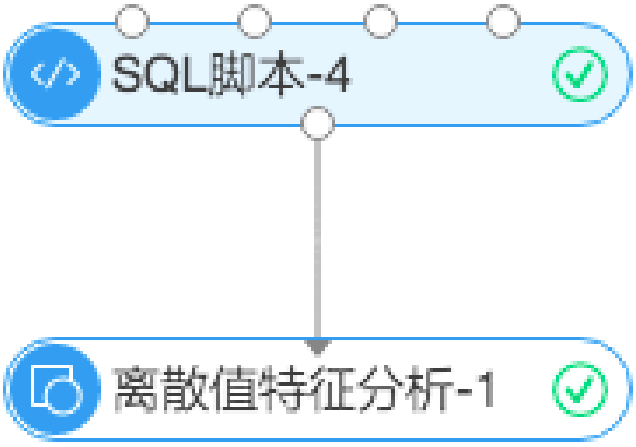
```
+------------+------------+------------+
| col_string | col_bigint | col_double |
+------------+------------+------------+
| 01 | 1 | 1.0 |
| 01 | 0 | 1.0 |
| 01 | 1 | NULL |
```

```
| NULL | 0 | 0.0 |
| 00 | 1 | 0.0 |
| 00 | 0 | 0.0 |
+------------+------------+------------+
```

运行命令

```
drop table if exists enum_feature_selection_test_input_enum_value_output;
drop table if exists enum_feature_selection_test_input_cnt_output;
drop table if exists enum_feature_selection_test_input_value_output;
PAI -name enum_feature_selection -project algo_public -DitemDelimiter=":" -Dlifecycle="28" -
DoutputValueTableName="enum_feature_selection_test_input_value_output" -DkvDelimiter="," -
DlabelColName="col_bigint" -DfeatureColNames="col_double,col_string" -
DoutputEnumValueTableName="enum_feature_selection_test_input_enum_value_output" -DenableSparse="false" -
DinputTableName="enum_feature_selection_test_input" -
DoutputCntTableName="enum_feature_selection_test_input_cnt_output";
```

界面



参数界面

界面运行结果

运行结果

enum_feature_selection_test_input_cnt_output

```
+------------+------------+------------+------------+
| colname | colvalue | labelvalue | cnt |
+------------+------------+------------+------------+
| col_double | NULL | 1 | 1 |
| col_double | 0 | 0 | 2 |
| col_double | 0 | 1 | 1 |
| col_double | 1 | 0 | 1 |
```

```
| col_double | 1 | 1 | 1 |
| col_string | NULL | 0 | 1 |
| col_string | 00 | 0 | 1 |
| col_string | 00 | 1 | 1 |
| col_string | 01 | 0 | 1 |
| col_string | 01 | 1 | 2 |
+------------+------------+------------+------------+
```

enum_feature_selection_test_input_value_output

```
+------------+------------+------------+------------+------------+---------------+
| colname | gini | entropy | infogain | ginigain | infogainratio |
+------------+------------+------------+------------+------------+---------------+
| col_double | 0.3888888888888889 | 0.792481250360578 | 0.20751874963942196 | 0.1111111111111111 |
0.14221913160264427 |
| col_string | 0.38888888888888884 | 0.792481250360578 | 0.20751874963942196 | 0.11111111111111116 |
0.14221913160264427 |
+------------+------------+------------+------------+------------+---------------+
```

enum_feature_selection_test_input_enum_value_output

```
+------------+------------+------------+------------+
| colname | colvalue | gini | entropy |
+------------+------------+------------+------------+
| col_double | NULL | 0.0 | 0.0 |
| col_double | 0 | 0.22222222222222224 | 0.4591479170272448 |
| col_double | 1 | 0.16666666666666666 | 0.3333333333333333 |
| col_string | NULL | 0.0 | 0.0 |
| col_string | 00 | 0.16666666666666666 | 0.3333333333333333 |
| col_string | 01 | 0.2222222222222222 | 0.4591479170272448 |
+------------+------------+------------+------------+
```

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。

```
{
"AlternativeHypthesis": "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
}
```

卡方检验

卡方拟合性检验是检验单个多项分类名义型变量各分类间的实际观测次数与理论次数之间是否一致的问题，其零假设是观测次数与理论次数之间无差异。

pai命令示例

```
PAI -name chisq_test
-project algo_public
-DinputTableName=pai_chisq_test_input
-DcolName=f0
-DprobConfig=0:0.3,1:0.7
-DoutputTableName=pai_chisq_test_output0
```

```
-DoutputDetailTableName=pai_chisq_test_output0_detail
```

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	输入表	表名	必选
colName	需要做卡方检验的列	列名	必选
outputTableName	输出表	不存在的表名	必选
outputDetailTableName	输出detail表	不存在的表名	必选
inputTablePartitions	输入表的分区列表	分区列表	可选, 默认值:””
probConfig	类别概率配置	kv对，格式 类别:概率,类别:概率...所有概率和为1	可选, 默认为所有类别概率相等

示例

测试数据

```
create table pai_chisq_test_input as
select * from
(
select '1' as f0,'2' as f1 from dual
union all
select '1' as f0,'3' as f1 from dual
union all
select '1' as f0,'4' as f1 from dual
union all
select '0' as f0,'3' as f1 from dual
union all
select '0' as f0,'4' as f1 from dual
)tmp;
```

pai命令

```
PAI -name chisq_test
-project algo_public
-DinputTableName=pai_chisq_test_input
-DcolName=f0
-DprobConfig=0:0.3,1:0.7
-DoutputTableName=pai_chisq_test_output0
-DoutputDetailTableName=pai_chisq_test_output0_detail
```

输出说明

输出表outputTableName，只有一行一列，是json格式。

```
{
"Chi-Square": {
"comment": "皮尔逊卡方",
"df": 1,
"p-value": 0.75,
"value": 0.2380952380952381
}
}
```

输出表outputDetailTableName，对应列：类别，观察频率(observed)，期望频率(expected)，标准误差(residuals = (observed-expected) / sqrt(expected))

f0	f1	observed	expected	residuals
0	2	0.0	0.4	-0.6324555320336759
0	3	1.0	0.8	0.22360679774997894
0	4	1.0	0.8	0.22360679774997894
1	2	1.0	0.6000000000000001	0.5163977794943221
1	3	1.0	1.2000000000000002	-0.1825741858350555
1	4	1.0	1.2000000000000002	-0.1825741858350555

数据视图

一 组件功能

- 提供数据透视功能，通过数据视图组件，用户可以可视化了解特征取值分布，特征与标签列的分布情况. 了解特征特点方便后续数据分析.支持稀疏和稠密.

二 PAI 命令

```
PAI -name fe_meta_runner -project algo_public -DinputTable="pai_dense_10_10" -
DoutputTable="pai_temp_2263_20384_1" -DmapTable="pai_temp_2263_20384_2" -
DselectedCols="pdays,previous,emp_var_rate,cons_price_idx,cons_conf_idx,euribor3m,nr_employed,age,campaign,p
outcome" -DlabelCol="y" -DcategoryCols="previous" -Dlifecycle="28"-DmaxBins="5" ;
```

三 算法参数说明

参数key名称	参数描述	必/选填	默认值
inputTable	输入数据表名	必填	-
inputTablePartitions	输入表分区	选填	-

outputTable	输出表名	必填	-
mapTable	输出映射表，数据视图对String类字符串会做一个统计映射成数字(转换成Int方便机器学习识别和训练)	必填	-
selectedCols	输入表选择列名类型	必填	-
categoryCols	把Int或者Double字段当做枚举特征，可选		“”
maxBins	连续性特征等距离划分最大区间数，可选		100
isSparse	是否是k:v的稀疏特征，可选，默认稠密数据		100
itemSpliter	稀疏特征item分隔符，可选，默认逗号		“,”
kvSpliter	稀疏特征item分隔符，可选，默认冒号		“:”
lifecycle	输出表生命周期（单位：天）	选填	28

四 建模示例

1. 输入数据

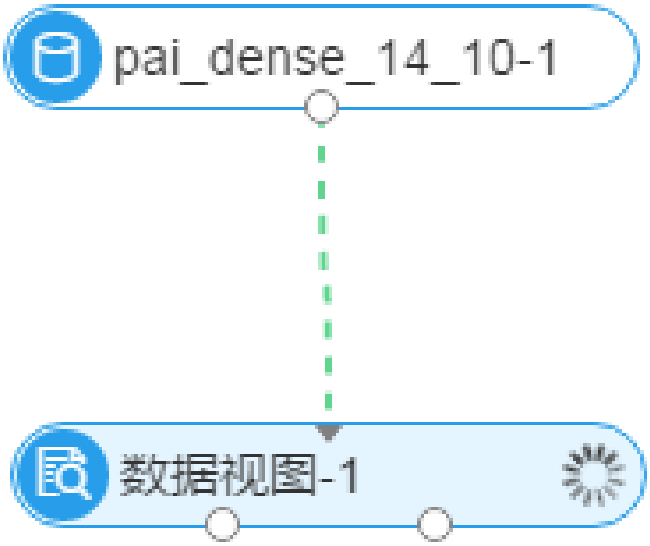
age	workclass	fwlght	edu	edu_num	married	c	family	race	sex	gail	loss	work_year	country	income
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174.0	0.0	40.0	United-States	<=50K
50	Self-emp-not-	833111	Bachelors	13	Married-civ-sp	Exec-managager	Husband	White	Male	0.0	0.0	13.0	United-States	<=50K

	inc				ouse	ial							
38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0.0	0.0	40.0	United-States	<=50K
53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0.0	0.0	40.0	United-States	<=50K
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0.0	0.0	40.0	Cuba	<=50K
37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0.0	0.0	40.0	United-States	<=50K
49	Private	16011	9th	5	Marrie	Other-	Not-in	Black	Femal	0.0	0.0	16.0	Jamaica	<=50

		87			d-spouse-absent	service	-family		e				K
52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0.0	0.0	45.0	United-States	>50K
31	Private	457811	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	140844.0	0.0	50.0	United-States	>50K
42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	51780.0	0.0	40.0	United-States	>50K

2. 建模DAG

建模DAG



- 数据视图配置：勾选income为目标列，其他14个字段为特征列，其中bigint类型的edu_num字段做枚举值处理

3. 建模效果

- 左边输入数据，你会发现原本是String字段的family，race,sex，income等都被映射成数值，这是 **为了方便数据被机器学习算法训练(某种程度有数据格式转换的功能)**

gail	loss	work_year	age	fwlght	edu_num	workclass	edu	married	c	family	race	sex	country	income
21740	0.0	40.0	39.0	775160	1.0	3.0	3.0	4.0	1.0	2.0	2.0	2.0	3.0	0.0
0.0	0.0	13.0	50.0	8331110	1.0	2.0	3.0	2.0	2.0	1.0	2.0	2.0	3.0	0.0
0.0	0.0	40.0	38.0	21564660	5.0	1.0	4.0	1.0	3.0	2.0	2.0	2.0	3.0	0.0
0.0	0.0	40.0	53.0	23472210	4.0	1.0	1.0	2.0	3.0	1.0	1.0	2.0	3.0	0.0
0.0	0.0	40.0	28.0	338400 90	1.0	1.0	3.0	2.0	5.0	3.0	1.0	1.0	1.0	0.0
0.0	0.0	40.0	37.0	28458220	2.0	1.0	5.0	2.0	2.0	3.0	2.0	1.0	3.0	0.0
0.	0.	1	4	1	3.	1.	2.	3.	4.	2.	1.	1.	2.	0.

0	0	6.0	9.0	601870	0	0	0	0	0	0	0	0	0	0
0.0	0.0	450	520	20964 2.0	5.0	2.0	4.0	2.0	2.0	1.0	2.0	2.0	3.0	1.0
140840	0.0	500	310	457810	2.0	1.0	5.0	4.0	5.0	2.0	2.0	1.0	3.0	1.0
51780	0.0	400	420	1594449	1.0	1.0	3.0	2.0	2.0	1.0	2.0	2.0	3.0	1.0

- 右边的输出是映射表.

feature_name	feature_value	map_id
income	<=50K	
0		
income	>50K	
1		
edu_num	13	1
edu_num	14	2
edu_num	5	3
edu_num	7	4
edu_num	9	5
workclass	Private	1
workclass	Self-emp-not-inc	2
workclass	State-gov	3
edu	11th	1
edu	9th	2

edu	Bachelors	3
edu	HS-grad	4
edu	Masters	5
married	Divorced	1
married	Married-civ-spouse	2
married	Married-spouse-absent	3
married	Never-married	4
c	Adm-clerical	1
c	Exec-managerial	2
c	Handlers-cleaners	3
c	Other-service	4
c	Prof-specialty	5
family	Husband	1
family	Not-in-family	2
family	Wife	3
race	Black	1
race	White	2
sex	Female	1
sex	Male	2
country	Cuba	1
country	Jamaica	2
country	United-States	3

协方差

在概率论和统计学中，协方差用于衡量两个变量的总体误差。而方差是协方差的一种特殊情况，即当两个变量是相同的情况。期望值分别为E(X) = μ 与 E(Y) = ν 的两个实数随机变量X与Y之间的协方差定义为：cov(X, Y) = E((X - μ) (Y - ν))

PAI命令行

```
PAI -name cov
-project algo_public
-DinputTableName=maple_test_cov_basic12x10_input
-DoutputTableName=maple_test_cov_basic12x10_output
-DcoreNum=6
```

```
-DmemSizePerCore=110;
```

参数说明

参数名称	参数描述	取值范围	是否必选，默认值
inputTableName	输入表	表名	必选
inputTablePartitions	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		可选， 默认值选择所有分区
outputTableName	输出表名列表	表名	必选
selectedColNames	输入表选择列名类型	列名	可选默认选择全部列
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期
coreNum	节点个数	与参数memSizePerCore配对使用，正整数，范围[1, 9999]	可选， 默认自动计算
memSizePerCore	单个节点内存大小，单位M	正整数，范围[1024, 64*1024]	可选， 默认自动计算

经验概率密度图

经验分布是当无法得到精确的参数分布时，需要从数据中估计概率分布从而得到非参数分布。算法中采用内核分布估计样本数据的概率密度，和直方图类似都是产生函数描述样本数据的分布，区别是内核分布叠加各部分的贡献而产生连续平滑的分布曲线，而直方图是离散地描述，采用内核分布时，非样本的数据点概率密度并非0，而是各样本抽样点在某种内核分布下的概率密度加权叠加，在这版实现中，内核分布固定采用高斯分布。

内核分布更详细介绍请见 wiki。

经验分布更详细介绍请见 wiki

PAI命令行

```
PAI -name empirical_pdf
-project algo_public
-DinputTableName="test_data"
-DoutputTableName="test_epdf_out"
-DfeatureColNames="col0,col1,col2"
-DinputTablePartitions="ds='20160101'"
```

```
-Dlifecycle=1
-DintervalNum=100
```

参数说明

参数名称	参数描述	取值范围	是否必选，默认值
inputTableName	输入表名	-	必选
outputTableName	输出表名	-	必选
featureColNames	输入列	可选多列，类型为double或bigint	必选
labelColName	输入label列，把feature列按照这一列所有的label值进行分组计算	只能选一列，类型为bigint或string，label值数量不能超过100	可选， “”
inputTablePartitions	输入表的分区	-	可选，” ”
intervalNum	计算频次区间数，越大精度越高	[1,1E14)	可选，-1，会根据各列数据的取值范围进行区间划分自动计算区间数
lifecycle	生命周期	-	可选，-1，不设生命周期
coreNum	核数目	>0	可选，-1，会根据输入数据量计算所起instance的数量
memSizePerCore	内存数	(100,64*1024)	可选，-1，会根据输入数据量计算所需内存大小

具体示例

数据生成

```
    drop table if exists epdf_test;
create table epdf_test as
select
*
from
(
select 1.0 as col1 from dual
union all
select 2.0 as col1 from dual
union all
select 3.0 as col1 from dual
union all
select 4.0 as col1 from dual
union all
select 5.0 as col1 from dual
) tmp;
```

PAI命令行

```
PAI -name empirical_pdf
-project algo_public
-DinputTableName=epdf_test
-DoutputTableName=epdf_test_out
-DfeatureColNames=col1;
```

输入说明：选择需要计算的列，可选择多列，同时可选label列按照每个label值把这些列进行切分成多组，例如label列中包含label值0、1，需要计算的列会分成label=0的一组，和label=1的一组，分别画出概率密度。若不选label列，feature列则会一整列进行计算。

输出说明：图和结果表，结果表的字段如下，不选label列时，label字段输出NULL

列名	数据类型
colName	string
label	string
x	double
pdf	double

输出表

```
   +------------+------------+------------+------------+
| colname | label | x | pdf |
+------------+------------+------------+------------+
| col1 | NULL | 1.0 | 0.12775155176809325 |
| col1 | NULL | 1.0404050505050506 | 0.1304256933829622 |
| col1 | NULL | 1.0808101010101012 | 0.13306325897429525 |
| col1 | NULL | 1.1212151515151518 | 0.1356613897616418 |
| col1 | NULL | 1.1616202020202024 | 0.1382173796574596 |
| col1 | NULL | 1.202025252525253 | 0.1407286844875733 |
| col1 | NULL | 1.2424303030303037 | 0.14319293014274642 |
| col1 | NULL | 1.2828353535353543 | 0.14560791960033242 |
| col1 | NULL | 1.3232404040404049 | 0.14797163876379316 |
| col1 | NULL | 1.3636454545454555 | 0.1502822610772349 |
| col1 | NULL | 1.404050505050506 | 0.1525381508819247 |
| col1 | NULL | 1.4444555555555567 | 0.1547378654919243 |
| col1 | NULL | 1.4848606060606073 | 0.1568801559764068 |
| col1 | NULL | 1.525265656565658 | 0.15896396664681753 |
| col1 | NULL | 1.5656707070707085 | 0.16098843325768245 |
| col1 | NULL | 1.6060757575757592 | 0.1629528799404685 |
| col1 | NULL | 1.6464808080808098 | 0.16485681490034038 |
| col1 | NULL | 1.6868858585858604 | 0.16669992491584543 |
| col1 | NULL | 1.727290909090911 | 0.16848206869138338 |
| col1 | NULL | 1.7676959595959616 | 0.17020326912168932 |
| col1 | NULL | 1.8081010101010122 | 0.17186370453638117 |
| col1 | NULL | 1.8485060606060628 | 0.17346369900080946 |
| col1 | NULL | 1.8889111111111134 | 0.17500371175692428 |
| col1 | NULL | 1.929316161616164 | 0.17648432589456017 |
| col1 | NULL | 1.9697212121212146 | 0.17790623634938396 |
```

col1	NULL	2.0101262626262653	0.1792702373286898
col1	NULL	2.050531313131316	0.18057720927022053
col1	NULL	2.0909363636363665	0.18182810544221673
col1	NULL	2.131341414141417	0.18302393829491406
col1	NULL	2.1717464646464677	0.18416576567472337
col1	NULL	2.2121515151515183	0.1852546770123305
col1	NULL	2.252556565656569	0.18629177959496213
col1	NULL	2.2929616161616195	0.18727818503109434
col1	NULL	2.33336666666667	0.18821499601297229
col1	NULL	2.3737717171717208	0.18910329347850022
col1	NULL	2.4141767676767714	0.18994412426940221
col1	NULL	2.454581818181822	0.19073848937711185
col1	NULL	2.4949868686868726	0.19148733286168018
col1	NULL	2.535391919191923	0.1921915315221827
col1	NULL	2.575796969696974	0.19285188538972659
col1	NULL	2.6162020202020244	0.19346910910630113
col1	NULL	2.656607070707075	0.19404382424446043
col1	NULL	2.6970121212121256	0.1945765526142701
col1	NULL	2.7374171717171762	0.19506771059517916
col1	NULL	2.777822222222227	0.19551760452158667
col1	NULL	2.8182272727272775	0.19592642714194602
col1	NULL	2.858632323232328	0.1962942551623821
col1	NULL	2.8990373737373787	0.1966210478770638
col1	NULL	2.9394424242424293	0.1969066468790639
col1	NULL	2.97984747474748	0.19715077683721793
col1	NULL	3.0202525252525305	0.19735304731663747
col1	NULL	3.060657575757581	0.19751295561309964
col1	NULL	3.1010626262626317	0.19762989056457925
col1	NULL	3.1414676767676823	0.19770313729675995
col1	NULL	3.181872727272733	0.19773188285349683
col1	NULL	3.2222777777777836	0.19771522265793107
col1	NULL	3.262682828282834	0.19765216774530828
col1	NULL	3.303087878787885	0.19754165270453194
col1	NULL	3.3434929292929354	0.19738254426210697
col1	NULL	3.383897979797986	0.19717365043938664
col1	NULL	3.4243030303030366	0.19691373021193162
col1	NULL	3.4647080808080872	0.1966015035982942
col1	NULL	3.505113131313138	0.19623566210464843
col1	NULL	3.5455181818181885	0.19581487945135703
col1	NULL	3.585923232323239	0.19533782250778076
col1	NULL	3.6263282828282897	0.1948031623623475
col1	NULL	3.6667333333333403	0.1942095854560816
col1	NULL	3.707138383838391	0.19355580470939734
col1	NULL	3.7475434343434415	0.19284057057394655
col1	NULL	3.787948484848492	0.19206268194364004
col1	NULL	3.8283535353535427	0.19122099686158253
col1	NULL	3.8687585858585933	0.19031444296253852
col1	NULL	3.909163636363644	0.1893420275936375
col1	NULL	3.9495686868686946	0.18830284755928747
col1	NULL	3.989973737373745	0.1871960984396676
col1	NULL	4.030378787878796	0.18602108343567092
col1	NULL	4.070783838383846	0.18477722169674377
col1	NULL	4.111188888888897	0.1834640560916829
col1	NULL	4.151593939393948	0.1820812603860928
col1	NULL	4.191998989898998	0.18062864579383914
col1	NULL	4.232404040404049	0.179106166873458

```
| col1 | NULL | 4.272809090909099 | 0.17751392674406796 |
| col1 | NULL | 4.31321414141415 | 0.17585218159888508 |
| col1 | NULL | 4.353619191919201 | 0.17412134449794325 |
| col1 | NULL | 4.394024242424251 | 0.1723219884250765 |
| col1 | NULL | 4.434429292929302 | 0.17045484859762067 |
| col1 | NULL | 4.4748343434343525 | 0.16852082402064342 |
| col1 | NULL | 4.515239393939403 | 0.1665209782808102 |
| col1 | NULL | 4.555644444444454 | 0.16445653957824907 |
| col1 | NULL | 4.596049494949504 | 0.16232889999798905 |
| col1 | NULL | 4.636454545454555 | 0.16013961402571825 |
| col1 | NULL | 4.6768595959596055 | 0.1578903963157465 |
| col1 | NULL | 4.717264646464656 | 0.15558311872216193 |
| col1 | NULL | 4.757669696969707 | 0.1532198066072439 |
| col1 | NULL | 4.798074747474757 | 0.1508026344442397 |
| col1 | NULL | 4.838479797979808 | 0.14833392073462115 |
| col1 | NULL | 4.878884848484859 | 0.14581612226291346 |
| col1 | NULL | 4.919289898989909 | 0.1432518277151203 |
| col1 | NULL | 4.95969494949496 | 0.1406437506896507 |
| col1 | NULL | 5.00010000000001 | 0.13799472213247665 |
+------------+------------+------------+------------+
```

算法规模

若选择label列，则label不能超过100个

箱线图

一 箱线图

- 可视化多个连续值特征与某个枚举类特征的箱线图和扰动关系

二 PAI 命令

```
PAI -name box_plot -project algo_public
-DinputTable="boxplot"
-DcontinueCols="age"
-DcategoryCol="y"
-DoutputTable="pai_temp_6075_97181_1"
-DsampleSize="1000"
-Dlifecycle="7";
```

三 参数说明

参数key名称	参数描述	必/选填	默认值
inputTable	输入数据表名	必填	-
inputTablePartitions	输入表分区	选填	-
outputTable	输出表名,存放箱线图和采样的样本	必填	-

continueCols	必选，连续值特征，支持多选	必填	-
categoryCol	必选，枚举特征列，单选	必填	-
sampleSize	绘制每个特征的扰动情况是的样本采样数		1000
lifecycle	输出表生命周期（单位：天）	选填	28

四 实例

输入数据

```
create table boxplot as select age, y from bank_data limit 100;
```

age	y
50	0
53	0
28	1
39	0
55	1
30	0
37	0
39	0
36	1
27	0
34	0
41	0
55	1
33	0
26	0
52	0
35	1
27	1
28	0
26	0

41	0
35	0
40	0
32	0
41	0
34	0
49	0
37	0
35	0
38	0
47	0
46	0
27	0
29	1
32	0
36	0
29	0
47	0
44	0
54	0
36	0
42	0
44	0
72	1
48	0
36	0
35	0
43	0
56	0
42	0
31	0
32	0
33	0

31	0
39	0
30	0
24	0
24	0
38	0
26	0
41	0
34	0
30	1
37	0
68	0
31	0
48	0
33	0
59	0
44	0
28	0
50	0
33	0
45	0
40	0
45	0
43	0
54	0
53	0
35	0
30	0
25	0
35	0
54	1
30	0
38	0

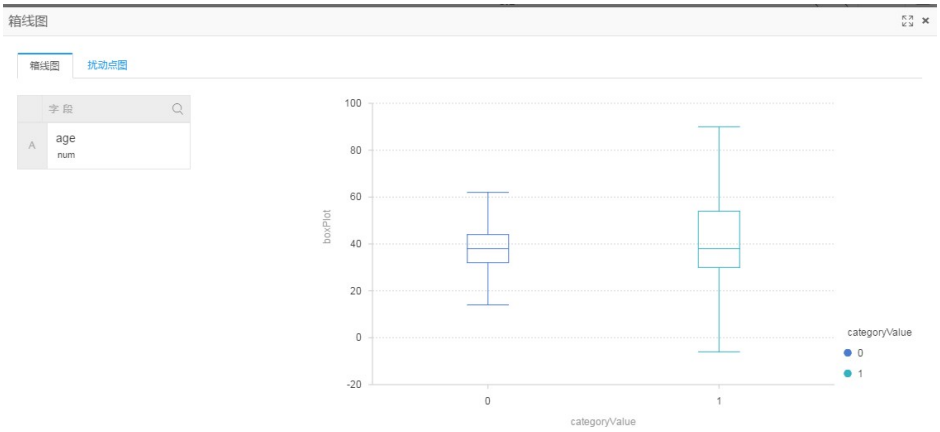
35	0
47	0
32	0
27	0
40	1
31	0
42	0
40	0
31	0
57	0
38	1
39	0
37	0
44	0

参数配置

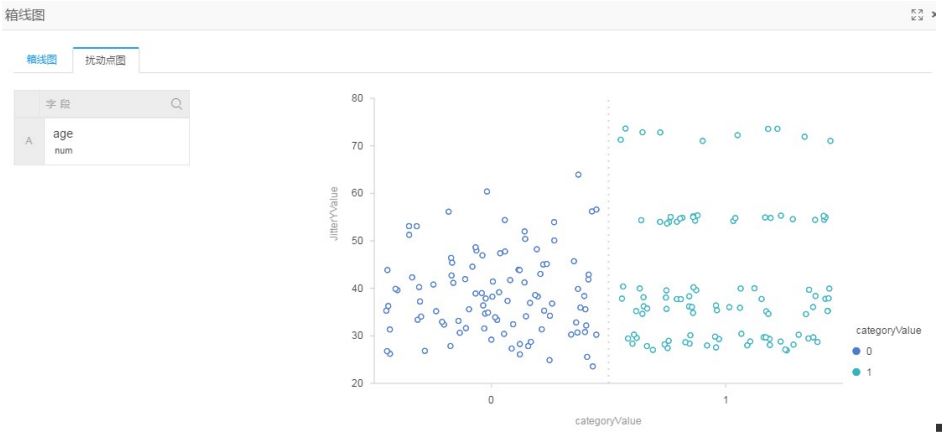
选择age为连续类型特征，y为枚举类特征,其他默认

运行效果

箱线图分布情况



扰动点



散点图

一 组件功能说明

在回归分析中，数据点在直角坐标系平面上的分布图

二 PAI 命令

```
PAI -name scatter_diagram -project algo_public
-DselectedCols=emp_var_rate,cons_price_rate,cons_conf_idx,euribor3m
-DsampleSize=1000
-DlabelCol=y
-DmapTable=pai_temp_2447_22859_2
-DinputTable=scatter_diagram
-DoutputTable=pai_temp_2447_22859_1;
```

三 参数说明

参数key名称	参数描述	必/选填	默认值
inputTable	输入数据表名	必填	-
inputTablePartitions	输入表分区	选填	-
outputTable	输出表名,存放采样样本	必填	-
mapTable	输出信息表，存放每个特征的最小最大值，枚举取值等	必填	-
selectedCols	输入表选择列名类型，用于绘制两两特征之间的散点图，最多勾选5个特征	必填	-
labelCol	把Int或者String字段		“”

	当做枚举标签列，可选		
sampleSize	对输入数据进行采样，可选		1000
lifecycle	输出表生命周期（单位：天）	选填	28

四 实例

输入数据

```
create table scatter_diagram as select emp_var_rate,cons_price_rate, cons_conf_idx,euribor3m,y from pai_bank_data limit 10
```

emp_var_rate	cons_price_rate	cons_conf_idx	euribor3m	y
1.4	93.918	-42.7	4.962	0
-0.1	93.2	-42.0	4.021	0
-1.7	94.055	-39.8	0.729	1
-1.8	93.075	-47.1	1.405	0
-2.9	92.201	-31.4	0.869	1
1.4	93.918	-42.7	4.961	0
-1.8	92.893	-46.2	1.327	0
-1.8	92.893	-46.2	1.313	0
-2.9	92.963	-40.8	1.266	1
-1.8	93.075	-47.1	1.41	0
1.1	93.994	-36.4	4.864	0
1.4	93.444	-36.1	4.964	0
1.4	93.444	-36.1	4.965	1
-1.8	92.893	-46.2	1.291	0
1.4	94.465	-41.8	4.96	0
1.4	93.918	-42.7	4.962	0
-1.8	93.075	-47.1	1.365	1
-0.1	93.798	-40.4	4.86	1
1.1	93.994	-36.4	4.86	0
1.4	93.918	-42.7	4.96	0
-1.8	93.075	-47.1	1.405	0

1.4	94.465	-41.8	4.967	0
1.4	93.918	-42.7	4.963	0
1.4	93.918	-42.7	4.968	0
1.4	93.918	-42.7	4.962	0
-1.8	92.893	-46.2	1.344	0
-3.4	92.431	-26.9	0.754	0
-1.8	93.075	-47.1	1.365	0
-1.8	92.893	-46.2	1.313	0
1.4	93.918	-42.7	4.961	0
1.4	94.465	-41.8	4.961	0
-1.8	92.893	-46.2	1.327	0
-1.8	92.893	-46.2	1.299	0
-2.9	92.963	-40.8	1.268	1
1.4	93.918	-42.7	4.963	0
-1.8	92.893	-46.2	1.334	0
1.4	93.918	-42.7	4.96	0
-1.8	93.075	-47.1	1.405	0
1.4	94.465	-41.8	4.96	0
1.4	93.444	-36.1	4.962	0
1.1	93.994	-36.4	4.86	0
1.1	93.994	-36.4	4.857	0
1.4	93.918	-42.7	4.961	0
-3.4	92.649	-30.1	0.715	1
1.4	93.444	-36.1	4.966	0
-0.1	93.2	-42.0	4.076	0
1.4	93.444	-36.1	4.965	0
-1.8	92.893	-46.2	1.354	0
1.4	93.444	-36.1	4.967	0
1.4	94.465	-41.8	4.959	0
-1.8	92.893	-46.2	1.354	0
1.4	94.465	-41.8	4.958	0
-1.8	92.893	-46.2	1.354	0
1.4	94.465	-41.8	4.864	0

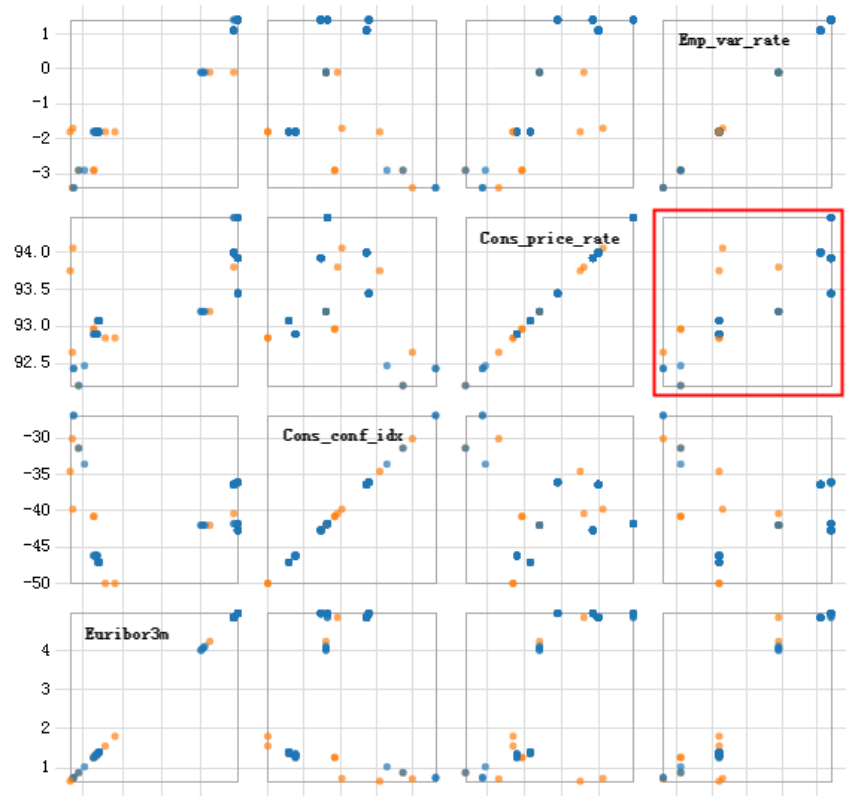
1.1	93.994	-36.4	4.859	0
1.1	93.994	-36.4	4.857	0
-1.8	92.893	-46.2	1.27	0
1.1	93.994	-36.4	4.857	0
1.1	93.994	-36.4	4.859	0
1.4	94.465	-41.8	4.959	0
1.1	93.994	-36.4	4.856	0
-1.8	93.075	-47.1	1.405	0
-1.8	92.843	-50.0	1.811	1
-0.1	93.2	-42.0	4.021	0
-2.9	92.469	-33.6	1.029	0
1.4	93.918	-42.7	4.962	0
-1.8	93.075	-47.1	1.365	0
1.1	93.994	-36.4	4.857	0
-1.8	92.893	-46.2	1.259	0
1.1	93.994	-36.4	4.857	0
1.4	94.465	-41.8	4.866	0
-2.9	92.201	-31.4	0.883	0
-0.1	93.2	-42.0	4.076	0
1.1	93.994	-36.4	4.857	0
1.4	93.918	-42.7	4.96	0
1.4	93.444	-36.1	4.962	0
1.1	93.994	-36.4	4.858	0
1.1	93.994	-36.4	4.857	0
1.1	93.994	-36.4	4.856	0
1.4	93.918	-42.7	4.968	0
1.4	93.444	-36.1	4.966	0
1.4	94.465	-41.8	4.962	0
1.4	93.444	-36.1	4.963	0
-1.8	92.843	-50.0	1.56	1
1.4	93.918	-42.7	4.96	0
1.4	93.444	-36.1	4.963	0
-3.4	92.431	-26.9	0.74	0

1.1	93.994	-36.4	4.856	0
1.4	93.918	-42.7	4.962	0
1.1	93.994	-36.4	4.856	0
-0.1	93.2	-42.0	4.245	1
1.1	93.994	-36.4	4.857	0
-1.8	93.075	-47.1	1.405	0
-1.8	92.893	-46.2	1.327	0
-0.1	93.2	-42.0	4.12	0
1.4	94.465	-41.8	4.958	0
-1.8	93.749	-34.6	0.659	1
1.1	93.994	-36.4	4.858	0
1.1	93.994	-36.4	4.858	0
1.4	93.444	-36.1	4.963	0

参数配置

散点图参数配置：特征列选择select emp_var_rate,cons_price_rate, cons_conf_idx,euribor3m， 可选的标签列选择y

运行效果



可以直观的看出特征与特征直接，分类标签的分布情况

相关系数矩阵

相关系数算法用于计算一个矩阵中每一列之间的相关系数，范围在[-1,1]之间。计算的时候，count数按两列间同时非空的元素个数计算，两两列之间可能不同。

PAI命令行

```
PAI -name corrcoef
-project algo_public
-DinputTableName=maple_test_corrcoef_basic12x10_input
-DoutputTableName=maple_test_corrcoef_basic12x10_output
-DcoreNum=1
-DmemSizePerCore=110;
```

参数说明

参数名称	参数描述	取值范围	是否必选，默认值
inputTableName	输入表	表名	必选
inputTablePartitions	输入表中指定哪些分区参与训练， 格式为: Partition_name=valu		可选， 默认值选择所有分区

	e. 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		
outputTableName	输出表名列表	表名	必选
selectedColNames	输入表选择列名类型	列名	可选默认选择全部列
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期
coreNum	节点个数	与参数memSizePerCore配对使用，正整数，范围[1, 9999] 详细介绍	可选， 默认自动计算
memSizePerCore	单个节点内存大小，单位M	正整数，范围[1024, 64*1024] 详细介绍	可选， 默认自动计算

具体示例

数据生成

col0: double	col1: bigint	col2: double	col3: bigint	col4: double	col5: bigint	col6: double	col7: bigint	col8: double	col9: double
19	95	33	52	115	43	32	98	76	40
114	26	101	69	56	59	116	23	109	105
103	89	7	9	65	118	73	50	55	81
79	20	63	71	5	24	77	31	21	75
87	16	66	47	25	14	42	99	108	57
11	104	38	37	106	51	3	91	80	97
84	30	70	46	8	6	94	22	45	48
35	17	107	64	10	78	53	34	90	96
13	61	39	1	29	117	112	2	82	28
62	4	102	88	100	36	67	54	12	85
49	27	44	93	68	110	60	72	86	58
92	119	0	113	41	15	74	83	18	111

PAI命令行

```
PAI -name corrcoef
-project algo_public
-DinputTableName=maple_test_corrcoef_basic12x10_input
```

```
-DoutputTableName=maple_test_corrcoef_basic12x10_output
-DcoreNum=1
-DmemSizePerCore=110;
```

输出表

columnsnames	col0	col1	col2	col3	col4	col5	col6	col7	col8	col9
col0	1	-0.21156572518207 24	0.05983062597065 61	0.25999035706846 93	-0.34832491882255 86	-0.28716254396809 926	0.47880162127435 116	-0.13646519484213 326	-0.19500158764680 092	0.38973902409490 85
col1	-0.21156572518207 24	1	-0.84444773778985 85	-0.17507636221594 533	0.40943384150571 377	0.09135976026101 403	-0.30185063746265 74	0.40733726912808 044	-0.11827739124590 071	0.12433851389455 183
col2	0.05983062597065 61	-0.84444773778985 85	1	0.18518346647293 102	-0.20934839228057 014	-0.18964175123896 59	0.17993774988632 13	-0.38588856764699 48	0.20254569203773 892	0.13476160753756 655
col3	0.25999035706846 93	-0.17507636221594 533	0.18518346647293 102	1	0.03988018649854 009	-0.43737887418329 147	-0.05381829642526 718 4	0.29008564415869 86	-0.36075479100756 88	0.49120190749304 49
col4	-0.34832491882255 86	0.40943384150571 377	-0.20934839228057 014	0.03988018649854 009	1	0.14656052092468 75	-0.50160303643479 55	0.54960243257111 17	0.01374325611539 412 2	0.07497231559184 887
col5	-0.28716254396809 926	0.09135976026101 403	-0.18964175123896 59	-0.43737887418329 147	0.14656052092468 75	1	0.16729809310873 522	-0.29890655828796 964	0.36185181010146 17	-0.17139609572868 85

col6	0.47880162127435116	-0.3018506374626574	0.17993774988632013	-0.05381829642526718 4	-0.50160303643479 55	0.16729809310873522	1	-0.81650198801564 62	-0.11173420918721436	-0.10363860378347944
col7	-0.13646519484213326	0.40733726912808044	-0.38588856764699 48	0.29008564415869 86	0.54960243257111 17	-0.29890655828796964	-0.81650198801564 62	1	0.07435907471544469	0.11711976051999162
col8	-0.19500158764680092	-0.11827739124590071	0.20254569203773892	-0.36075479100756 88	0.01374325611539412 2	0.36185181010146 17	-0.11173420918721436	0.07435907471544469	1	-0.18463012549540175
col9	0.38973902409490 85	0.12433851389455183	0.13476160753756655	0.49120190749304 49	0.07497231559184887	-0.17139609572868 85	-0.10363860378347944	0.11711976051999162	-0.18463012549540175	1

正态检验

正态性检验是检验观测值是否服从正态分布，本组件由三种检验方法组成，包括Anderson-Darling Test, 详见wiki，Kolmogorov-Smirnov Test，详见wiki 以及QQ图，详见wiki，用户可以自选某一种或多种检验方法。

算法说明

原假设H0：观测值服从正态分布，H1：观测值不服从正态分布

KS的p值计算方法采用渐进计算KS分布的CDF，无论样本量多大都采用的是该方法，详见wiki

QQ图在样本量>1000时，会采样进行计算和画图输出，因此图中的数据点不一定覆盖所有样本

PAI命令行

```
PAI -name normality_test
-project algo_public
```

```
-DinputTableName=test
-DoutputTableName=test_out
-DselectedColNames=col1,col2
-Dlifecycle=1;
```

参数说明

参数名称	参数描述	取值范围	是否必选，默认值
inputTableName	输入表	-	必选
outputTableName	输出表	-	必选
selectedColNames	选择字段列	可选多列，类型为double或bigint	可选
inputTablePartitions	输入表分区	-	可选，“ ”
enableQQplot	使用QQ图	-	可选，默认true
enableADtest	Anderson-Darling检验	-	可选，默认true
enableKStest	Kolmogorov-Smirnov检验	-	可选，默认true
lifecycle	生命周期	>=-1整数	可选，默认-1，不设生命周期
coreNum	核数目	>0整数	可选，默认-1，会根据输入数据量计算所起instance的数量
memSizePerCore	内存数	(100,64*1024)	可选，默认-1，会根据输入数据量计算所需内存大小

具体示例

数据生成

```
    drop table if exists normality_test_input;
create table normality_test_input as
select
*
from
(
select 1 as x from dual
union all
select 2 as x from dual
union all
select 3 as x from dual
union all
select 4 as x from dual
union all
select 5 as x from dual
union all
```

```
select 6 as x from dual
union all
select 7 as x from dual
union all
select 8 as x from dual
union all
select 9 as x from dual
union all
select 10 as x from dual
) tmp;
```

PAI命令行

```
PAI -name normality_test
-project projectxlib4
-DinputTableName=normality_test_input
-DoutputTableName=normality_test_output
-DselectedColNames=x
-Dlifecycle=1;
```

输入说明

- 输入格式：选择需要计算的列，可选择多列，但类型必须为double或bigint。

  输出格式

  图和结果表，结果表的字段如下。结果表有两个分区，p=’test’的分区是AD检验或KS检验的结果，只当enableADtest为true或enableKStest为true时会输出数据。

- p=’plot’是QQ图的数据，只当enableQQplot为true时会输出数据，并复用p=’test’的列，即当p=’plot’时，testvalue列记录原观测数据（QQ图的x轴），pvalue列记录如果服从正态分布的话预期的数据（QQ图的y轴）。

列名	数据类型	含义
colName	string	列名
testname	string	检验名
testvalue	double	检验值/QQ图x轴
pvalue	double	检验的p值/QQ图y轴
p	double	分区名

输出表

```
+------------+------------+------------+------------+------------+
| colname | testname | testvalue | pvalue | p |
+------------+------------+------------+------------+------------+
| x | NULL | 1.0 | 0.8173291742279805 | plot |
```

```
| x | NULL | 2.0 | 2.470864450785345 | plot |
| x | NULL | 3.0 | 3.5156067948020056 | plot |
| x | NULL | 4.0 | 4.3632330349313095 | plot |
| x | NULL | 5.0 | 5.128868067945126 | plot |
| x | NULL | 6.0 | 5.871131932054874 | plot |
| x | NULL | 7.0 | 6.6367669650686905 | plot |
| x | NULL | 8.0 | 7.4843932051979944 | plot |
| x | NULL | 9.0 | 8.529135549214654 | plot |
| x | NULL | 10.0 | 10.182670825772018 | plot |
| x | Anderson_Darling_Test | 0.1411092332197832 | 0.9566579606430077 | test |
| x | Kolmogorov_Smirnov_Test | 0.09551932503797644 | 0.9999888659426232 | test |
+------------+------------+------------+------------+------------+
```

洛伦兹曲线

洛伦兹曲线研究的是国民收入在国民之间的分配问题。为了研究国民收入在国民之间的分配问题，美国统计学家（或说奥地利统计学家）M.O.洛伦兹（Max Otto Lorenz，1903- ）1907年（或说1905年）提出了著名的洛伦兹曲线。意大利经济学家基尼在此基础上定义了基尼系数。

画一个矩形，矩形的高衡量社会财富的百分比，将之分为N等份，每一等分为1/N的社会总财富。在矩形的长上，将所有家庭从最贫者到最富者自左向右排列，也分为N等分，第一个等份代表收入最低的1/N的家庭。在这个矩形中，将每1/N的家庭所有拥有的财富的占比累积起来，并将相应的点画在图中，便得到了一条曲线就是洛伦兹曲线。

PAI命令行

```
PAI -name LorenzCurve
-project algo_public
-DinputTableName=maple_test_lorenz_basic10_input
-DcolName=col0
-DoutputTableName=maple_test_lorenz_basic10_output -DcoreNum=20
-DmemSizePerCore=110;
```

参数说明

参数名称	参数描述	取值范围	是否必选，默认值
inputTableName	输入表	表名	必选
outputTableName	输出表	必选	-
colName	列名，逗号分隔	可选选中全表	
N	分位数	可选100	
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name1=value1/name2=value2; 如果是指		可选， 默认值选择所有分区

	定多个分区， 中间用,分开		
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期
coreNum	节点个数	与参数memSizePerCore配对使用，正整数，范围[1, 9999] [详细介绍](	可选， 默认自动计算
memSizePerCore	单个节点内存大小，单位M	正整数，范围[1024, 64*1024]	可选， 默认自动计算

具体示例

数据生成

col0:double
4
7
2
8
6
3
9
5
0
1
10

PAI命令行

```
PAI -name LorenzCurve
-project algo_public
-DinputTableName=maple_test_lorenz_basic10_input
-DcolName=col0
-DoutputTableName=maple_test_lorenz_basic10_output
-DcoreNum=20
-DmemSizePerCore=110;
```

输出说明

输出表

quantile	col0

0	0
1	0.01818181818181818
2	0.01818181818181818
3	0.01818181818181818
4	0.01818181818181818
5	0.01818181818181818
6	0.01818181818181818
7	0.01818181818181818
8	0.01818181818181818
9	0.01818181818181818
10	0.01818181818181818
11	0.05454545454545454
12	0.05454545454545454
13	0.05454545454545454
14	0.05454545454545454
…	…
85	0.8181818181818182
86	0.8181818181818182
87	0.8181818181818182
88	0.8181818181818182
89	0.8181818181818182
90	1
91	1
92	1
93	1
94	1
95	1
96	1
97	1
98	1
99	1
100	1

机器学习

目录

PS-SMART二分类

PS-SMART多分类

PS-SMART回归

PS线性回归

线性支持向量机

背景

支持向量机（SVM）是90 年代中期发展起来的基于统计学习理论的一种机器学习方法，通过寻求结构化风险最小来提高学习机泛化能力，实现经验风险和置信范围的最小化，从而达到在统计样本量较少的情况下，亦能获得良好统计规律的目的。算法的详细介绍可以参考wiki

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

算法组件

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

设置算法参数

- 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

PAI命令

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

参数设置

参数名称	参数描述	参数值可选项	默认值
inputTableName	输入表	-	-
inputTableParitions	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	-	输入表的所有分区
modelName	必选，输出的模型名称	-	-
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	-	-
positiveLabel	可选，正例的值	-	在label的取值中随机选择一个
negativeCost	可选，负例权重值，即负例惩罚因子	(0, ∞)	默认1.0
positiveCost	可选，正例权重值。即正例惩罚因子	(0, ∞)	默认1.0
∞	可选，收敛系数	(0, 1)	0.001

示例

训练数据

id	y	f0	f1	f2	f3	f4	f5	f6	f7
1	-1	-0.294118	0.487437	0.180328	-0.292929	-1	0.0014902	-0.53117	-0.0333333
2	+1	-0.882353	-0.145729	0.0819672	-0.414141	-1	-0.207153	-0.766866	-0.666667
3	-1	-0.0588235	0.839196	0.0491803	-1	-1	-0.305514	-0.492741	-0.633333
4	+1	-0.882353	-0.105528	0.0819672	-0.535354	-0.777778	-0.162444	-0.923997	-1
5	-1	-1	0.376884	-0.344262	-0.292929	-0.602837	0.28465	0.887276	-0.6

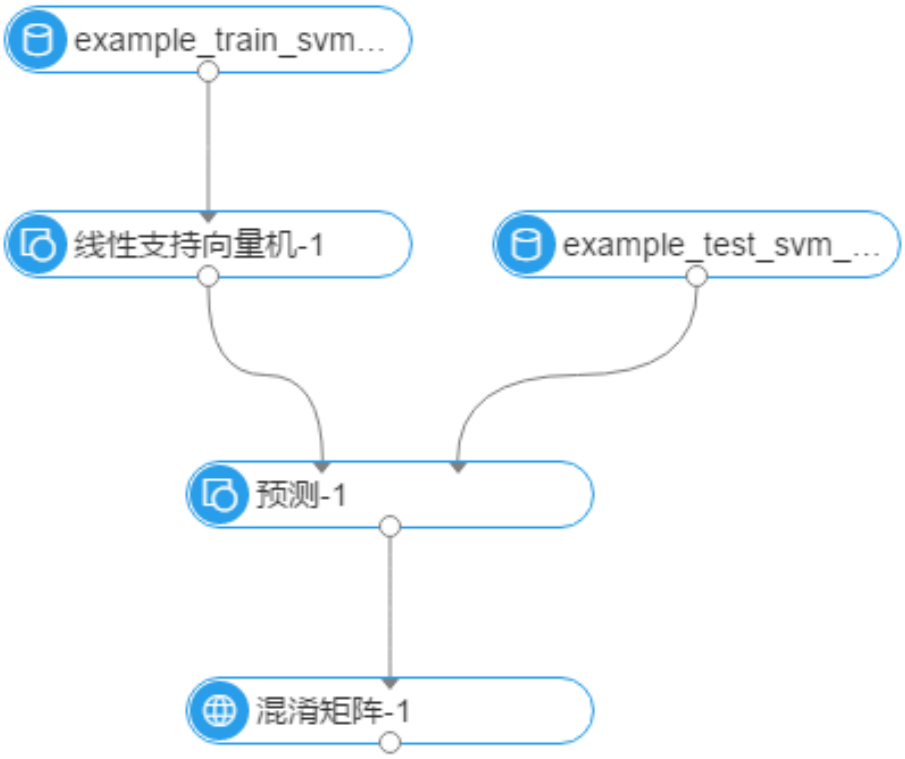
6	+1	-0.411765	0.165829	0.213115	-1	-1	-0.23696	-0.894962	-0.7
7	-1	-0.647059	-0.21608	-0.180328	-0.353535	-0.791962	-0.0760059	-0.854825	-0.833333
8	+1	0.176471	0.155779	-1	-1	-1	0.052161	-0.952178	-0.733333
9	-1	-0.764706	0.979899	0.147541	-0.0909091	0.283688	-0.0909091	-0.931682	0.0666667
10	-1	-0.0588235	0.256281	0.57377	-1	-1	-1	-0.868488	0.1

测试数据

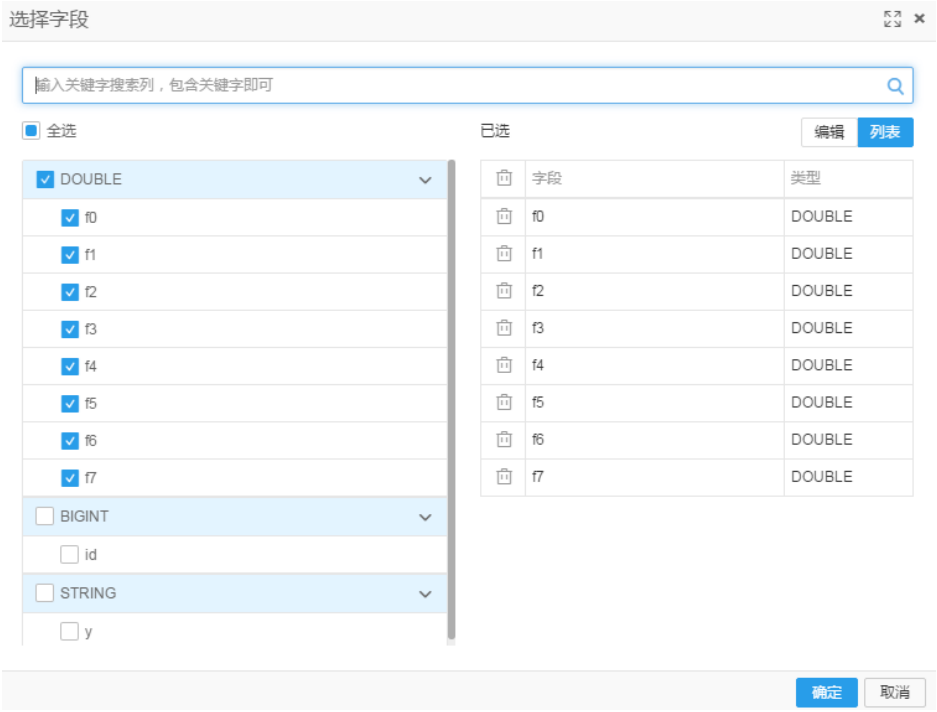
id	y	f0	f1	f2	f3	f4	f5	f6	f7
1	+1	-0.882353	0.0854271	0.442623	-0.616162	-1	-0.19225	-0.725021	-0.9
2	+1	-0.294118	-0.0351759	-1	-1	-1	-0.293592	-0.904355	-0.766667
3	+1	-0.882353	0.246231	0.213115	-0.272727	-1	-0.171386	-0.981213	-0.7
4	-1	-0.176471	0.507538	0.278689	-0.414141	-0.702128	0.0491804	-0.475662	0.1
5	-1	-0.529412	0.839196	-1	-1	-1	-0.153502	-0.885568	-0.5
6	+1	-0.882353	0.246231	-0.0163934	-0.353535	-1	0.0670641	-0.627669	-1
7	-1	-0.882353	0.819095	0.278689	-0.151515	-0.307329	0.19225	0.00768574	-0.966667
8	+1	-0.882353	-0.0753769	0.0163934	-0.494949	-0.903073	-0.418778	-0.654996	-0.866667
9	+1	-1	0.527638	0.344262	-0.212121	-0.356974	0.23696	-0.836038	-0.8

10	+1	-0.882353	0.115578	0.0163934	-0.737374	-0.56974	-0.28465	-0.948762	-0.933333

创建实验svm_example



选择特征列

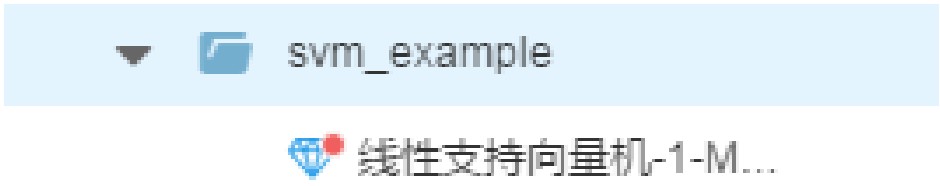


选择标签列

配置SVM的参数


字段设置
参数设置
正样本的标签值 留空, label值中随机选择一个
+1
正例惩罚因子 可选 (0, +inf)
1.0
负例惩罚因子 可选 (0, +inf)
1.0
收敛系数
0.001

运行实验


svm_example
线性支持向量机-1-M...

生成模型如下：

预测结果如下：

id	y	prediction_result	prediction_score	prediction_detail
1	+1	+1	0.157594271402295...	{"+1": 0.1575942714022959, "-1": -0.1575942714022959}
2	+1	+1	0.6614698832250897	{"+1": 0.6614698832250897, "-1": -0.6614698832250897}
3	+1	-1	-0.033364749587674...	{"+1": -0.03336474958767432, "-1": 0.03336474958767432}
4	-1	-1	-0.7993936496277533	{"+1": -0.7993936496277533, "-1": 0.7993936496277533}
5	-1	-1	-0.062511633256538...	{"+1": -0.06251163325653875, "-1": 0.06251163325653875}
6	+1	+1	0.099662488068409...	{"+1": 0.09966248806840972, "-1": -0.09966248806840972}
7	-1	-1	-0.7519810414882623	{"+1": -0.7519810414882623, "-1": 0.7519810414882623}
8	+1	+1	0.180293970573986...	{"+1": 0.1802939705739862, "-1": -0.1802939705739862}
9	+1	-1	-0.1196222721567562	{"+1": -0.1196222721567562, "-1": 0.1196222721567562}
10	+1	+1	0.4110490612942082	{"+1": 0.4110490612942082, "-1": -0.4110490612942082}

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。 逻辑回归多分类最多支持100类。

参数设置

逻辑回归组件的参数

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

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" -Depsilon="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

示例

二分类

测试数据

新建数据SQL

```
drop table if exists lr_test_input;
create table lr_test_input
as
select
*
from
(
select
cast(1 as double) as f0,
cast(0 as double) as f1,
cast(0 as double) as f2,
cast(0 as double) as f3,
cast(0 as bigint) as label
from dual
union all
select
cast(0 as double) as f0,
cast(1 as double) as f1,
cast(0 as double) as f2,
cast(0 as double) as f3,
cast(0 as bigint) as label
from dual
union all
select
cast(0 as double) as f0,
cast(0 as double) as f1,
cast(1 as double) as f2,
cast(0 as double) as f3,
cast(1 as bigint) as label
from dual
union all
select
cast(0 as double) as f0,
```

```
cast(0 as double) as f1,
cast(0 as double) as f2,
cast(1 as double) as f3,
cast(1 as bigint) as label
from dual
union all
select
cast(1 as double) as f0,
cast(0 as double) as f1,
cast(0 as double) as f2,
cast(0 as double) as f3,
cast(0 as bigint) as label
from dual
union all
select
cast(0 as double) as f0,
cast(1 as double) as f1,
cast(0 as double) as f2,
cast(0 as double) as f3,
cast(0 as bigint) as label
from dual
) a;
```

输入数据说明

```
+------------+------------+------------+------------+------------+
| f0 | f1 | f2 | f3 | label |
+------------+------------+------------+------------+------------+
| 1.0 | 0.0 | 0.0 | 0.0 | 0 |
| 0.0 | 0.0 | 1.0 | 0.0 | 1 |
| 0.0 | 0.0 | 0.0 | 1.0 | 1 |
| 0.0 | 1.0 | 0.0 | 0.0 | 0 |
| 1.0 | 0.0 | 0.0 | 0.0 | 0 |
| 0.0 | 1.0 | 0.0 | 0.0 | 0 |
+------------+------------+------------+------------+------------+
```

运行命令

```
drop offlinemodel if exists lr_test_model;
drop table if exists lr_test_prediction_result;
PAI -name logisticregression_binary -project algo_public -DmodelName="lr_test_model" -DitemDelimiter="," -
DregularizedLevel="1" -DmaxIter="100" -DregularizedType="None" -Depsilon="0.000001" -DkvDelimiter=":" -
DlabelColName="label" -DfeatureColNames="f0,f1,f2,f3" -DenableSparse="false" -DgoodValue="1" -
DinputTableName="lr_test_input";
PAI -name prediction -project algo_public -DdetailColName="prediction_detail" -DmodelName="lr_test_model" -
DitemDelimiter="," -DresultColName="prediction_result" -Dlifecycle="28" -
DoutputTableName="lr_test_prediction_result" -DscoreColName="prediction_score" -DkvDelimiter=":" -
DinputTableName="lr_test_input" -DenableSparse="false" -DappendColNames="label";
```

界面

参数界面

界面运行结果

运行结果

lr_test_prediction_result

```
+------------+------------------+------------------+------------------+
| label | prediction_result | prediction_score | prediction_detail |
+------------+------------------+------------------+------------------+
| 0 | 0 | 0.9999998793434426 | {
"0": 0.9999998793434426,
"1": 1.206565574533681e-07} |
| 1 | 1 | 0.999999799574135 | {
"0": 2.004258650156743e-07,
"1": 0.999999799574135} |
| 1 | 1 | 0.999999799574135 | {
"0": 2.004258650156743e-07,
"1": 0.999999799574135} |
| 0 | 0 | 0.9999998793434426 | {
"0": 0.9999998793434426,
"1": 1.206565574533681e-07} |
| 0 | 0 | 0.9999998793434426 | {
"0": 0.9999998793434426,
"1": 1.206565574533681e-07} |
| 0 | 0 | 0.9999998793434426 | {
"0": 0.9999998793434426,
"1": 1.206565574533681e-07} |
+------------+------------------+------------------+------------------+
```

多分类

测试数据

新建数据SQL

```
drop table if exists multi_lr_test_input;
create table multi_lr_test_input
as
select
*
from
(
select
cast(1 as double) as f0,
cast(0 as double) as f1,
cast(0 as double) as f2,
cast(0 as double) as f3,
cast(0 as bigint) as label
from dual
union all
select
cast(0 as double) as f0,
cast(1 as double) as f1,
cast(0 as double) as f2,
```

```
cast(0 as double) as f3,
cast(0 as bigint) as label
from dual
union all
select
cast(0 as double) as f0,
cast(0 as double) as f1,
cast(1 as double) as f2,
cast(0 as double) as f3,
cast(2 as bigint) as label
from dual
union all
select
cast(0 as double) as f0,
cast(0 as double) as f1,
cast(0 as double) as f2,
cast(1 as double) as f3,
cast(1 as bigint) as label
from dual
) a;
```

输入数据说明

```
+------------+------------+------------+------------+------------+
| f0 | f1 | f2 | f3 | label |
+------------+------------+------------+------------+------------+
| 1.0 | 0.0 | 0.0 | 0.0 | 0 |
| 0.0 | 0.0 | 1.0 | 0.0 | 2 |
| 0.0 | 0.0 | 0.0 | 1.0 | 1 |
| 0.0 | 1.0 | 0.0 | 0.0 | 0 |
+------------+------------+------------+------------+------------+
```

运行命令

```
drop offlinemodel if exists multi_lr_test_model;
drop table if exists multi_lr_test_prediction_result;
PAI -name logisticregression_multi -project algo_public -DmodelName="multi_lr_test_model" -DitemDelimiter="," -
DregularizedLevel="1" -DmaxIter="100" -DregularizedType="None" -Depsilon="0.000001" -DkvDelimiter=":" -
DlabelColName="label" -DfeatureColNames="f0,f1,f2,f3" -DenableSparse="false" -
DinputTableName="multi_lr_test_input";
PAI -name prediction -project algo_public -DdetailColName="prediction_detail" -
DmodelName="multi_lr_test_model" -DitemDelimiter="," -DresultColName="prediction_result" -Dlifecycle="28" -
DoutputTableName="multi_lr_test_prediction_result" -DscoreColName="prediction_score" -DkvDelimiter=":" -
DinputTableName="multi_lr_test_input" -DenableSparse="false" -DappendColNames="label";
```

界面

参数界面

界面运行结果

运行结果

multi_lr_test_prediction_result

```
+------------+------------------+------------------+------------------+
| label | prediction_result | prediction_score | prediction_detail |
+------------+------------------+------------------+------------------+
| 0 | 0 | 0.9999997274902165 | {
"0": 0.9999997274902165,
"1": 2.324679066261573e-07,
"2": 2.324679066261569e-07} |
| 0 | 0 | 0.9999997274902165 | {
"0": 0.9999997274902165,
"1": 2.324679066261573e-07,
"2": 2.324679066261569e-07} |
| 2 | 2 | 0.9999999155958832 | {
"0": 2.0188339798509 94e-07,
"1": 2.324679066261573e-07,
"2": 0.9999999155958832} |
| 1 | 1 | 0.9999999155958832 | {
"0": 2.0188339798509 94e-07,
"1": 0.9999999155958832,
"2": 2.324679066261569e-07} |
+------------+------------------+------------------+------------------+
```

GBDT二分类

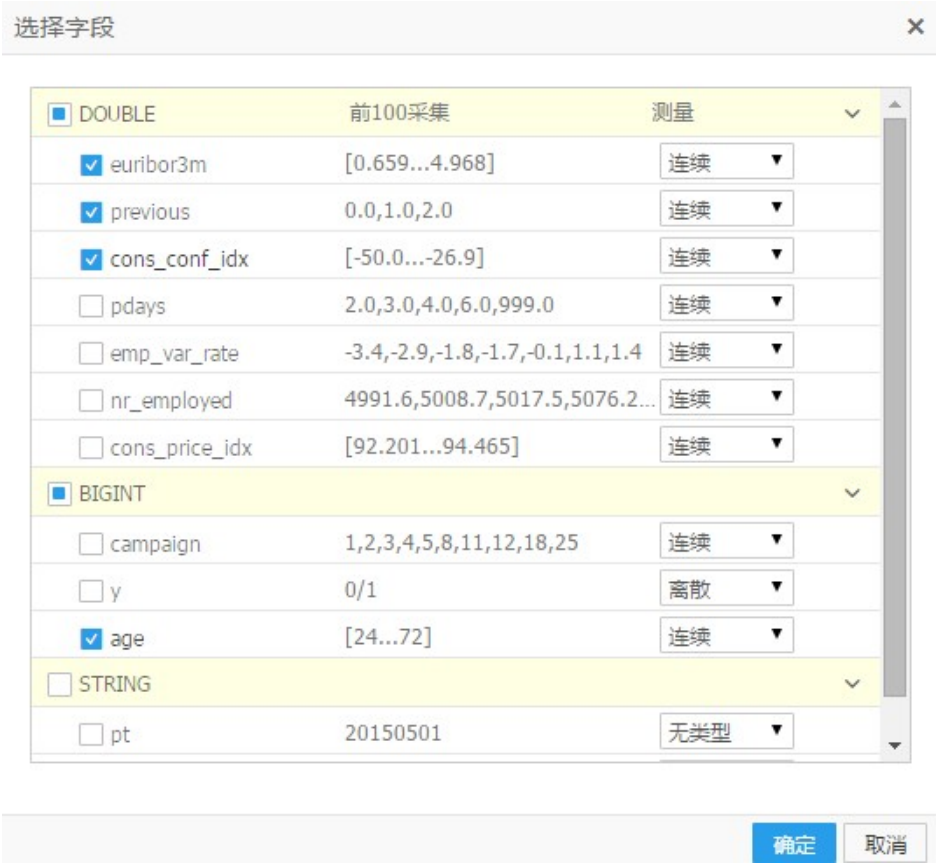
在GBDT回归与排序基础上，用于二分类问题，既设定阈值：大于阈值为正例，反之为负例。

组件介绍

向画布拖入GBDT二分类组件训练，调整参数，如：

- 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：



- 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

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

训练采集特征比例

- 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）

注意:

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。

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" -DsampleRatio="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" -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
- 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

示例

测试数据

新建数据SQL

```
drop table if exists gbdt_lr_test_input;
create table gbdt_lr_test_input
as
select
*
from
(
select
cast(1 as double) as f0,
cast(0 as double) as f1,
cast(0 as double) as f2,
cast(0 as double) as f3,
cast(0 as bigint) as label
```

```
from dual
union all
select
cast(0 as double) as f0,
cast(1 as double) as f1,
cast(0 as double) as f2,
cast(0 as double) as f3,
cast(0 as bigint) as label
from dual
union all
select
cast(0 as double) as f0,
cast(0 as double) as f1,
cast(1 as double) as f2,
cast(0 as double) as f3,
cast(1 as bigint) as label
from dual
union all
select
cast(0 as double) as f0,
cast(0 as double) as f1,
cast(0 as double) as f2,
cast(1 as double) as f3,
cast(1 as bigint) as label
from dual
union all
select
cast(1 as double) as f0,
cast(0 as double) as f1,
cast(0 as double) as f2,
cast(0 as double) as f3,
cast(0 as bigint) as label
from dual
union all
select
cast(0 as double) as f0,
cast(1 as double) as f1,
cast(0 as double) as f2,
cast(0 as double) as f3,
cast(0 as bigint) as label
from dual
) a;
```

输入数据说明

```
+------------+------------+------------+------------+------------+
| f0 | f1 | f2 | f3 | label |
+------------+------------+------------+------------+------------+
| 1.0 | 0.0 | 0.0 | 0.0 | 0 |
| 0.0 | 0.0 | 1.0 | 0.0 | 1 |
| 0.0 | 0.0 | 0.0 | 1.0 | 1 |
| 0.0 | 1.0 | 0.0 | 0.0 | 0 |
| 1.0 | 0.0 | 0.0 | 0.0 | 0 |
| 0.0 | 1.0 | 0.0 | 0.0 | 0 |
+------------+------------+------------+------------+------------+
```

运行命令

```
drop offlinemodel if exists gbdt_lr_test_model;
drop table if exists gbdt_lr_test_prediction_result;
PAI -name gbdt_lr -project algo_public -DfeatureSplitValueMaxSize="500" -DrandSeed="1" -Dshrinkage="1" -
DmaxLeafCount="30" -DlabelColName="label" -DinputTableName="gbdt_lr_test_input" -
DminLeafSampleCount="1" -DsampleRatio="1" -DmaxDepth="10" -DmodelName="gbdt_lr_test_model" -
DmetricType="0" -DfeatureRatio="1" -DtestRatio="0" -DfeatureColNames="f0,f1,f2,f3" -DtreeCount="5";
PAI -name prediction -project algo_public -DdetailColName="prediction_detail" -
DmodelName="gbdt_lr_test_model" -DitemDelimiter="," -DresultColName="prediction_result" -Dlifecycle="28" -
DoutputTableName="gbdt_lr_test_prediction_result" -DscoreColName="prediction_score" -DkvDelimiter=":" -
DinputTableName="gbdt_lr_test_input" -DenableSparse="false" -DappendColNames="label";
```

界面

参数界面

界面运行结果

运行结果

gbdt_lr_test_prediction_result

```
+------------+------------------+------------------+------------------+
| label | prediction_result | prediction_score | prediction_detail |
+------------+------------------+------------------+------------------+
| 0 | 0 | 0.9984308925552831 | {
"0": 0.9984308925552831,
"1": 0.0015691074447169 43} |
| 0 | 0 | 0.9984308925552831 | {
"0": 0.9984308925552831,
"1": 0.0015691074447169 43} |
| 1 | 1 | 0.9982721832240973 | {
"0": 0.0017271677590272 4,
"1": 0.9982721832240973} |
| 1 | 1 | 0.9982721832240973 | {
"0": 0.0017271677590272 4,
"1": 0.9982721832240973} |
| 0 | 0 | 0.9984308925552831 | {
"0": 0.9984308925552831,
"1": 0.0015691074447169 43} |
| 0 | 0 | 0.9984308925552831 | {
"0": 0.9984308925552831,
"1": 0.0015691074447169 43} |
+------------+------------------+------------------+------------------+
```

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那一类作为该行的类

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=pai_knn_test_input
-DtrainFeatureColNames=f0,f1
-DtrainLabelColName=class
-DpredictTableName=pai_knn_test_input
-DpredictFeatureColNames=f0,f1
-DoutputTableName=pai_knn_test_output
-Dk=2;
```

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

示例

测试数据

```
create table pai_knn_test_input as
select * from
(
select 1 as f0,2 as f1, 'good' as class from dual
union all
select 1 as f0,3 as f1, 'good' as class from dual
union all
select 1 as f0,4 as f1, 'bad' as class from dual
union all
select 0 as f0,3 as f1, 'good' as class from dual
union all
select 0 as f0,4 as f1, 'bad' as class from dual
)tmp;
```

PAI命令

```
pai -name knn
-DtrainTableName=pai_knn_test_input
-DtrainFeatureColNames=f0,f1
-DtrainLabelColName=class
-DpredictTableName=pai_knn_test_input
-DpredictFeatureColNames=f0,f1
-DoutputTableName=pai_knn_test_output
-Dk=2;
```

输出说明 f0,f1为结果附件列

prediction_result: 分类结果；prediction_score: 分类结果对应概率；prediction_detail: 最近K个结论以及对应的概率

f0	f1	prediction_result	prediction_score	prediction_detail
1	4	bad	1.0	{"bad": 1}
0	4	bad	1.0	{"bad": 1}
0	3	bad	0.5	{ "bad": 0.5, "good": 0.5}
1	3	good	1.0	{"good": 1}
1	2	good	1.0	{"good": 1}

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单棵树输出的类别的众数而定.

单棵树算法可以选择id3，c4.5，cart

随机森林更多详细介绍请见维基百科链接wiki

PAI 命令

```
PAI -name randomforests
-project algo_public
```

```
-DinputTableName="pai_rf_test_input"
-DmodelName="pai_rf_test_model"
-DforceCategorical="f1"
-DlabelColName="class"
-DfeatureColNames="f0,f1"
-DmaxRecordSize="100000"
-DminNumPer="0"
-DminNumObj="2"
-DtreeNum="3";
```

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	输入表	表名	必选
inputTablePartitions	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		可选， 默认值选择所有分区
labelColName	输入表中标签列的列名	列名	必选
modelName	输出的模型名		必选
treeNum	森林中树的个数	正整数，(0, 1000]	必选
weightColName	输入表中权重列的列名		可选, 默认无权重列
featureColNames	输入表中用于训练的特征的列名		可选，默认除labelColName、weightColName外其他所有列
excludedColNames	用于反选特征列，该参数不可以与featureColNames并存		可选， 默认为空
forceCategorical	feature默认解析规则：string、boolean、datetime类型的列解析为离散类型，double、bigint类型的列解析为连续类型；若有将bigint解析为categorical的情况，通过参数forceCategorical指定		可选，默认int为连续类型
algorithmTypes	单颗树的算法在森林中的位置	如果有则长度为2.比如有n棵树，algorithmTypes=[a,b], 则[0,a) 是 id3, [a,b) 是cart， [b,n) 是	可选，默认算法在森林中均分

		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。	
randomColNum	单颗树在生成时，每次分裂时选择的随机的特征个数	[1-N]，其中N为feature 个数	可选，默认log2N
minNumObj	叶节点数据的最小个数	正整数	可选，默认2
minNumPer	叶节点数据个数占父节点的最小比例	[0,1]	可选，默认0.0
maxTreeDeep	单颗树的最大深度	[1, ∞)	可选，默认∞
maxRecordSize	森林中单颗树输入的随机数据的个数	(1000, 1000000]	可选，默认100000

示例

测试数据

```
create table pai_rf_test_input as
select * from
(
select 1 as f0,2 as f1, "good" as class from dual
union all
select 1 as f0,3 as f1, "good" as class from dual
union all
select 1 as f0,4 as f1, "bad" as class from dual
union all
select 0 as f0,3 as f1, "good" as class from dual
union all
select 0 as f0,4 as f1, "bad" as class from dual
)tmp;
```

pai命令

```
 PAI -name randomforests
-project algo_public
-DinputTableName="pai_rf_test_input"
-Dmodelbashame="pai_rf_test_model"
-DforceCategorical="f1"
-DlabelColName="class"
-DfeatureColNames="f0,f1"
-DmaxRecordSize="100000"
-DminNumPer="0"
-DminNumObj="2"
-DtreeNum="3";
```

输出说明

模型PMML

```
<?xml version="1.0" encoding="utf-8"?
<PMML xmlns="http://www.dmg.org/PMML-4_2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="4.2" xsi:schemaLocation="http://www.dmg.org/PMML-4_2 http://www.dmg.org/v4-2/pmml-4-2.xsd"
<Header copyright="Copyright (c) 2014, Alibaba Inc." description=""
<Application name="ODPS/PMML" version="0.1.0"/
<TimestampTue, 12 Jul 2016 07:04:48 GMT</Timestamp
</Header
<DataDictionary numberOfFields="2"
<DataField name="f0" optype="continuous" dataType="integer"/
<DataField name="f1" optype="continuous" dataType="integer"/
<DataField name="class" optype="categorical" dataType="string"
<Value value="bad"/
<Value value="good"/
</DataField
</DataDictionary
<MiningModel modelName="xlab_m_random_forests_1_75078_v0" functionName="classification"
algorithmName="RandomForests"
<MiningSchema
<MiningField name="f0" usageType="active"/
<MiningField name="f1" usageType="active"/
<MiningField name="class" usageType="target"/
</MiningSchema
<Segmentation multipleModelMethod="majorityVote"
<Segment id="0"
<True/
<TreeModel modelName="xlab_m_random_forests_1_75078_v0" functionName="classification"
algorithmName="RandomForests"
<MiningSchema
<MiningField name="f0" usageType="active"/
<MiningField name="f1" usageType="active"/
<MiningField name="class" usageType="target"/
</MiningSchema
<Node id="1"
<True/
<ScoreDistribution value="bad" recordCount="2"/
<ScoreDistribution value="good" recordCount="3"/
<Node id="2" score="good"
<SimplePredicate field="f1" operator="equal" value="2"/
<ScoreDistribution value="good" recordCount="1"/
</Node
<Node id="3" score="good"
<SimplePredicate field="f1" operator="equal" value="3"/
<ScoreDistribution value="good" recordCount="2"/
</Node
<Node id="4" score="bad"
<SimplePredicate field="f1" operator="equal" value="4"/
<ScoreDistribution value="bad" recordCount="2"/
</Node
</Node
</TreeModel
</Segment
```

```
<Segment id="1"
<True/
<TreeModel modelName="xlab_m_random_forests_1_75078_v0" functionName="classification"
algorithmName="RandomForests"
<MiningSchema
<MiningField name="f0" usageType="active"/
<MiningField name="f1" usageType="active"/
<MiningField name="class" usageType="target"/
</MiningSchema
<Node id="1"
<True/
<ScoreDistribution value="bad" recordCount="2"/
<ScoreDistribution value="good" recordCount="3"/
<Node id="2" score="good"
<SimpleSetPredicate field="f1" booleanOperator="isIn"
<Array n="2" type="integer"2 3</Array
</SimpleSetPredicate
<ScoreDistribution value="good" recordCount="3"/
</Node
<Node id="3" score="bad"
<SimpleSetPredicate field="f1" booleanOperator="isNotIn"
<Array n="2" type="integer"2 3</Array
</SimpleSetPredicate
<ScoreDistribution value="bad" recordCount="2"/
</Node
</Node
</TreeModel
</Segment
<Segment id="2"
<True/
<TreeModel modelName="xlab_m_random_forests_1_75078_v0" functionName="classification"
algorithmName="RandomForests"
<MiningSchema
<MiningField name="f0" usageType="active"/
<MiningField name="f1" usageType="active"/
<MiningField name="class" usageType="target"/
</MiningSchema
<Node id="1"
<True/
<ScoreDistribution value="bad" recordCount="2"/
<ScoreDistribution value="good" recordCount="3"/
<Node id="2" score="bad"
<SimplePredicate field="f0" operator="lessOrEqual" value="0.5"/
<ScoreDistribution value="bad" recordCount="1"/
<ScoreDistribution value="good" recordCount="1"/
</Node
<Node id="3" score="good"
<SimplePredicate field="f0" operator="greaterThan" value="0.5"/
<ScoreDistribution value="bad" recordCount="1"/
<ScoreDistribution value="good" recordCount="2"/
</Node
</Node
</TreeModel
</Segment
</Segmentation
</MiningModel
```

```
</PMML
```

模型可视化

```
随机森林输出
```

朴素贝叶斯

背景

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

算法组件

- 特征列：支持double、string与bigint数据类型
- 标签列：只能选择非特征列的其它列，支持double、string与bigint类型

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";
```

参数设置

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	输入表的所有partition

		name1=value1/name2=value2；如果是指定多个分区，中间用’,’分开	
modelName	必选，输出的模型名	-	-
labelColName	必选，输入表中标签列的列名	-	-
featureColNames	可选，输入表中用于训练的特征的列名	-	除label列外其他所有列
excludedColNames	可选，用于反选特征列，该参数不可以与featureColNames并存	-	空列
forceCategorical	可选，feature默认解析规则：string、boolean、datetime类型的列解析为离散类型，double、bigint类型的列解析为连续类型；若有将bigint解析为categorical的情况，通过参数forceCategorical指定	-	int为连续类型

示例

训练数据

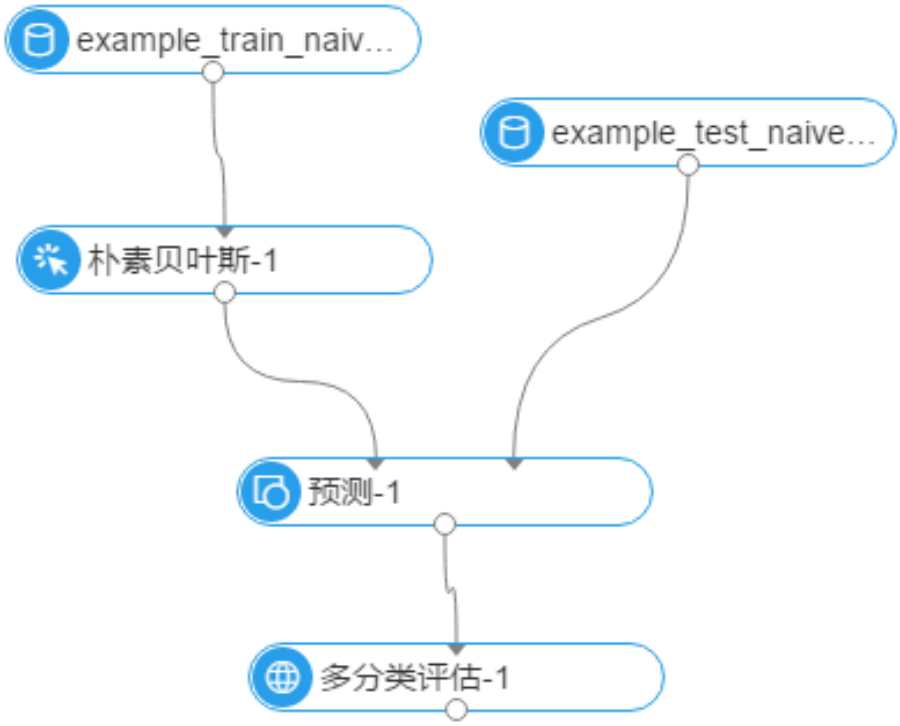
id	y	f0	f1	f2	f3	f4	f5	f6	f7
1	-1	-0.294118	0.487437	0.180328	-0.292929	-1	0.0014902	-0.53117	-0.0333333
2	+1	-0.882353	-0.145729	0.0819672	-0.414141	-1	-0.207153	-0.766866	-0.666667
3	-1	-0.0588235	0.839196	0.0491803	-1	-1	-0.305514	-0.492741	-0.633333
4	+1	-0.882353	-0.105528	0.0819672	-0.535354	-0.777778	-0.162444	-0.923997	-1
5	-1	-1	0.376884	-0.344262	-0.292929	-0.602837	0.28465	0.887276	-0.6
6	+1	-0.411765	0.165829	0.213115	-1	-1	-0.23696	-0.894962	-0.7

7	-1	-0.647059	-0.21608	-0.180328	-0.353535	-0.791962	-0.0760059	-0.854825	-0.833333
8	+1	0.176471	0.155779	-1	-1	-1	0.052161	-0.952178	-0.733333
9	-1	-0.764706	0.979899	0.147541	-0.0909091	0.283688	-0.0909091	-0.931682	0.0666667
10	-1	-0.0588235	0.256281	0.57377	-1	-1	-1	-0.868488	0.1

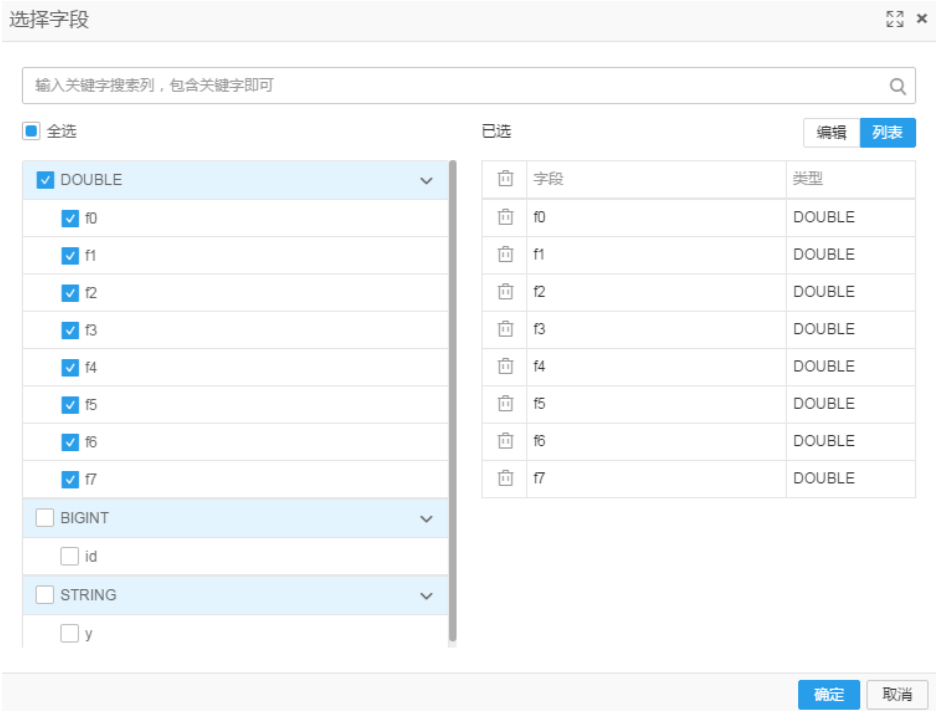
测试数据

id	y	f0	f1	f2	f3	f4	f5	f6	f7
1	+1	-0.882353	0.0854271	0.442623	-0.616162	-1	-0.19225	-0.725021	-0.9
2	+1	-0.294118	-0.0351759	-1	-1	-1	-0.293592	-0.904355	-0.766667
3	+1	-0.882353	0.246231	0.213115	-0.272727	-1	-0.171386	-0.981213	-0.7
4	-1	-0.176471	0.507538	0.278689	-0.414141	-0.702128	0.0491804	-0.475662	0.1
5	-1	-0.529412	0.839196	-1	-1	-1	-0.153502	-0.885568	-0.5
6	+1	-0.882353	0.246231	-0.0163934	-0.353535	-1	0.0670641	-0.627669	-1
7	-1	-0.882353	0.819095	0.278689	-0.151515	-0.307329	0.19225	0.00768574	-0.966667
8	+1	-0.882353	-0.0753769	0.0163934	-0.494949	-0.903073	-0.418778	-0.654996	-0.866667
9	+1	-1	0.527638	0.344262	-0.212121	-0.356974	0.23696	-0.836038	-0.8
10	+1	-0.882353	0.115578	0.0163934	-0.737374	-0.56974	-0.28465	-0.948762	-0.933333

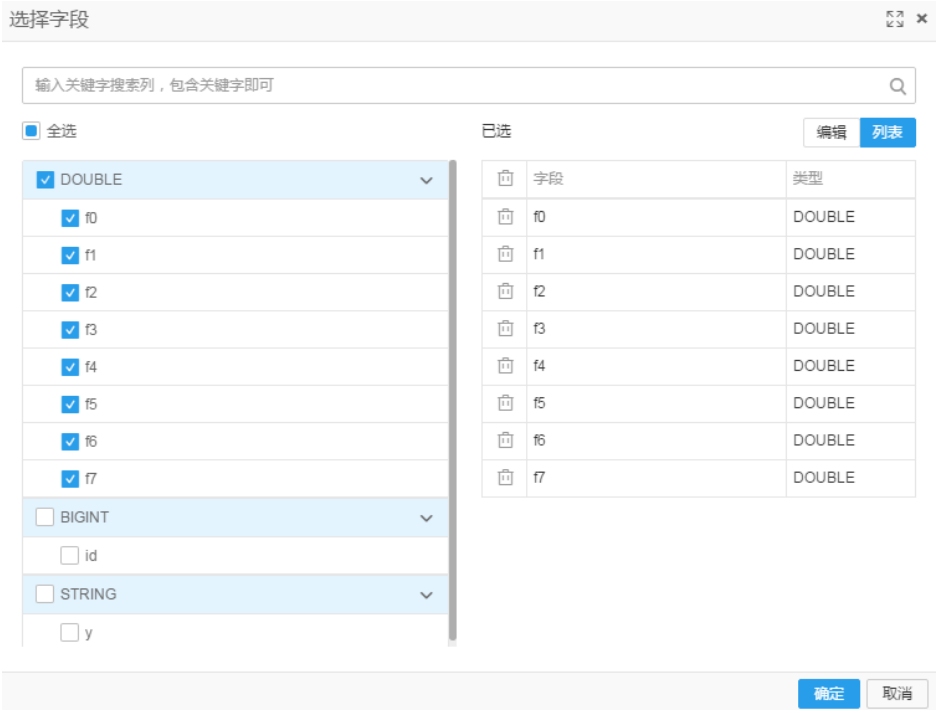
创建实验



选择特征列

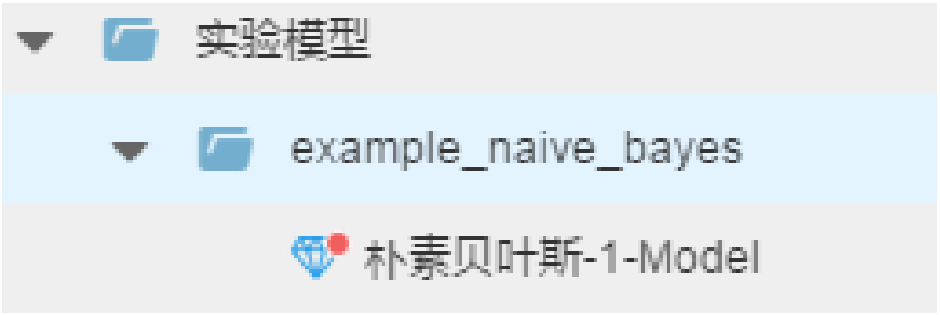


选择标签列



运行实验

生成的模型如下：



预测结果如下：

id	y	prediction_result	prediction_score	prediction_detail
1	+1	+1	1.1253341740304466	{"+1": 1.125334174030447, "-1": -4.116083356256129}
2	+1	+1	2.1096017126455773	{"+1": 2.109601712645577, "-1": -9.431111061253313}
3	+1	+1	1.0119555555558313	{"+1": 1.011955555555831, "-1": -3.065398632191728}
4	-1	-1	-2.435966936637894	{"+1": -33.18118539397123, "-1": -2.435966936637894}
5	-1	-1	-8.569186264886811	{"+1": -10.87241674956984, "-1": -8.569186264886811}
6	+1	-1	-3.7653558619317407	{"+1": -4.842928791659737, "-1": -3.765355861931741}
7	-1	-1	-4.895262140022778	{"+1": -89.31454432308786, "-1": -4.895262140022778}
8	+1	+1	-2.701748842959141	{"+1": -2.701748842959141, "-1": -3.683850999783979}
9	+1	-1	-4.321549531283424	{"+1": -20.76204601810873, "-1": -4.321549531283424}
10	+1	+1	-3.0880514229489764	{"+1": -3.088051422948976, "-1": -3.667255025859788}

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.

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的详细介绍请见维基百科链接 wiki

pai命令示例

```
pai -name kmeans
-project algo_public
-DinputTableName=pai_kmeans_test_input
-DselectedColNames=f0,f1
-DcenterCount=3
-Dloop=10
-Daccuracy=0.00001
-DdistanceType=euclidean
-DinitCenterMethod=random
-Dseed=1
-DmodelName=pai_kmeans_test_input_output_model
-DidxTableName=pai_kmeans_test_input_output_idx
-DclusterCountTableName=pai_kmeans_test_input_output_cc
```

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	输入表	表名	必选
selectedColNames	输入表中用于训练的列名，以逗号分隔，支持int和double类型	列名	可选，默认值选择所有列
inputTablePartitions	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		可选， 默认值选择所有分区
centerCount	聚类数	正整数， [1, 1000]	必选
loop	最大迭代次数	正整数， [1, 1000]	可选，默认值100
accuracy	算法终止条件，如果两次迭代之间变化低于该值， 算法终止		可选， 默认值0.0
distanceType	距离度量方式	euclidean(欧式距离)， cosine(夹角余弦)， cityblock(曼哈顿	可选， 默认值euclidean

		距离) 详细介绍	
initCenterMethod	质心初始化方法	random(随机采样)，topk(输入表前k行)，uniform(均匀分布)，kmpp(kmeans++)，external(指定初始质心表) 详细介绍	可选，默认值random
initCenterTableName	初始质心表名	表名	当initCenterMethod为external时生效
seed	初始随机种子	正整数	可选，默认值为当前时间。seed设置为固定值，每次聚类结果是稳定的
enableSparse	输入表数据是否为稀疏格式	true， false	可选， 默认值false
itemDelimiter	当输入表数据为稀疏格式时，kv间的分割符		可选， 默认值为空格
kvDelimiter	当输入表数据为稀疏格式时，key和value的分割符		可选， 默认值冒号
appendColNames	inputTableName表的哪些列附加输出到idxTableName表，列名以逗号分隔		可选，默认值无附件列
modelName	输出模型	模型名	必选
idxTableName	输出聚类结果表，和输入表对应，并指明聚类后每条record所属的类号	表名	必选
idxTablePartition	输出聚类结果表的分区	表名	可选，默认不输出分区
clusterCountTableName	输出聚类统计表，统计各个聚类包含的点的数目		可选， 模型不输出
centerTableName	输出聚类中心表		可选， 即将下线，建议使用参数modelName

距离度量方式

参数名称	参数描述
euclidean	$d(x-c)=(x-c)(x$

cosine	$d(x-c)=0.5-0.5*\frac{x$
cityblock	$d(x-c)=\lvert x-$

质心初始化方法

参数名称	参数描述
random	从输入数据表中随机采样出K个初始中心点，初始随机种子可以有参数seed指定
topk	从输入表中读取前K行作为初始中心点
uniform	从输入数据表，按最小到最大值，均匀计算出K个初始中心点
kmpp	使用k-means++算法选出K个初始中心点，详细介绍请见维基百科链接wiki
external	指定额外的初始中心表

示例

测试数据

```
create table pai_kmeans_test_input as
select * from
(
select 1 as f0,2 as f1 from dual
union all
select 1 as f0,3 as f1 from dual
union all
select 1 as f0,4 as f1 from dual
union all
select 0 as f0,3 as f1 from dual
union all
select 0 as f0,4 as f1 from dual
)tmp;
```

pai命令

```
pai -name kmeans
```

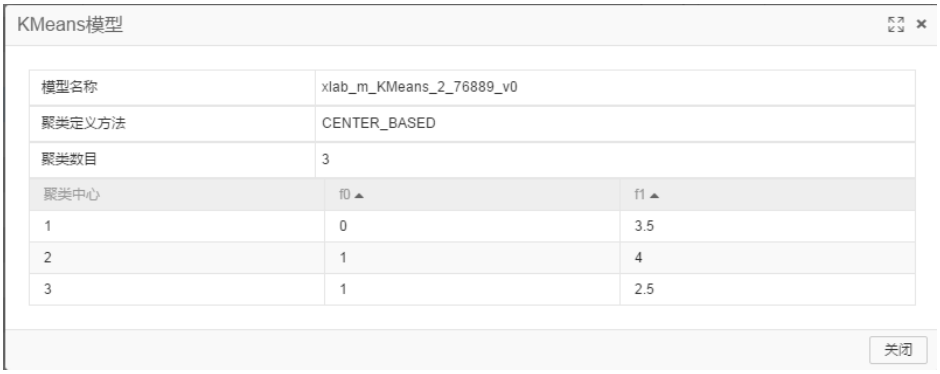
```
-project algo_public
-DinputTableName=pai_kmeans_test_input
-DselectedColNames=f0,f1
-DcenterCount=3
-Dloop=10
-Daccuracy=0.00001
-DdistanceType=euclidean
-DinitCenterMethod=random
-Dseed=1
-DmodelName=pai_kmeans_test_input_output_model
-DidxTableName=pai_kmeans_test_input_output_idx
-DclusterCountTableName=pai_kmeans_test_input_output_cc
```

输出说明

模型PMML

```
<?xml version="1.0" encoding="utf-8"?
<PMML xmlns="http://www.dmg.org/PMML-4_2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="4.2" xsi:schemaLocation="http://www.dmg.org/PMML-4_2 http://www.dmg.org/v4-2/pmml-4-2.xsd"
<Header copyright="Copyright (c) 2014, Alibaba Inc." description=""
<Application name="ODPS/PMML" version="0.1.0"/
<TimestampFri, 15 Jul 2016 03:09:38 GMT</Timestamp
</Header
<DataDictionary numberOfFields="2"
<DataField name="f0" optype="continuous" dataType="integer"/
<DataField name="f1" optype="continuous" dataType="integer"/
<DataField name="cluster_index" optype="continuous" dataType="integer"/
</DataDictionary
<ClusteringModel modelName="xlab_m_KMeans_2_76889_v0" functionName="clustering"
algorithmName="kmeans" modelClass="centerBased" numberOfClusters="3"
<MiningSchema
<MiningField name="f0" usageType="active"/
<MiningField name="f1" usageType="active"/
</MiningSchema
<ComparisonMeasure kind="distance" compareFunction="absDiff"
<squaredEuclidean/
</ComparisonMeasure
<ClusteringField field="f0" compareFunction="absDiff"/
<ClusteringField field="f1" compareFunction="absDiff"/
<Cluster
<Array n="2" type="real"0 3.5</Array
</Cluster
<Cluster
<Array n="2" type="real"1 4</Array
</Cluster
<Cluster
<Array n="2" type="real"1 2.5</Array
</Cluster
</ClusteringModel
</PMML
```

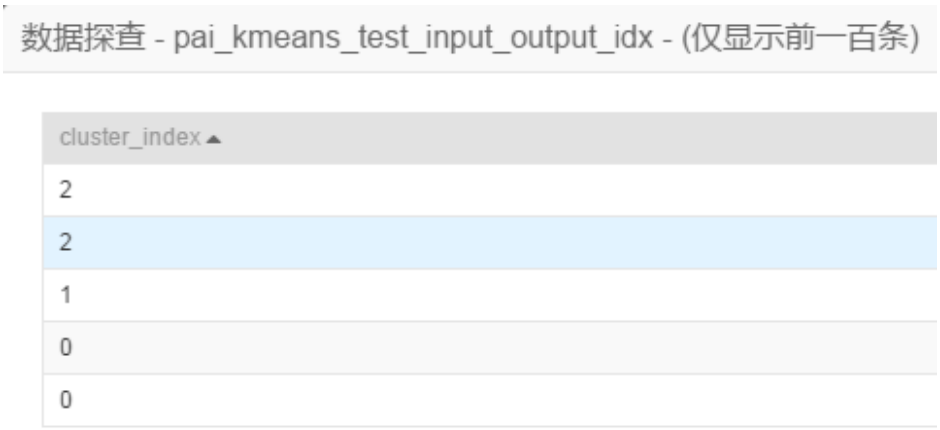
模型可视化


KMeans模型

模型名称	xlab_m_KMeans_2_76889_v0	
聚类定义方法	CENTER_BASED	
聚类数目	3	
聚类中心	f0	f1
1	0	3.5
2	1	4
3	1	2.5

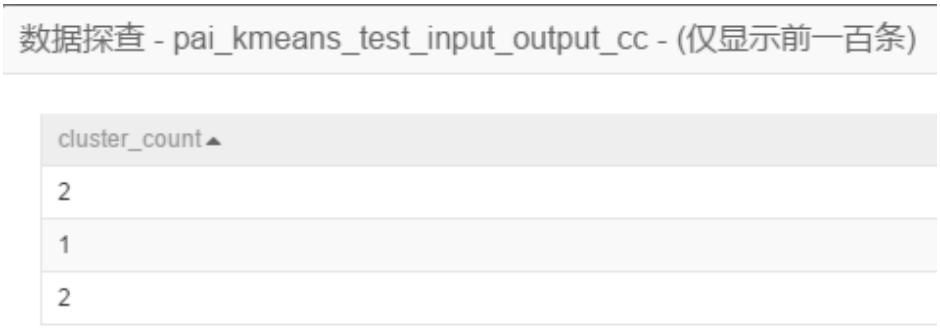
关闭

聚类结果表：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


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

cluster_index
2
2
1
0
0

聚类统计表：行数据等于聚类个数，每行的值表示当前聚类包含的点个数


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

cluster_count
2
1
2

线性回归

线性回归是分析因变量和多个自变量之间线性关系的模型。详细请见（https://en.wikipedia.org/wiki/Linear_regression ）

PAI命令

```
PAI -name linearregression
-project algo_public
-DinputTableName=lm_test_input
-DfeatureColNames=x
```

```
-DlabelColName=y
-DmodelName=lm_test_input_model_out;
```

算法参数

参数名称	参数描述	参数值可选项	默认值
inputTableName	必选，输入表名	-	-
modelName	必选，输出的模型名	-	-
outputTableName	可选，输出的模型评估表名	enableFitGoodness为true时必须指定outputTableName	“”
labelColName	必选，因变量	double或者bigint类型，限选一列	-
featureColNames	必选，自变量	非稀疏格式下double或者bigint类型，稀疏格式下string类型，可选多列	-
inputTablePartitions	可选，输入表的分区	-	“”
maxIter	可选， 最大迭代次数	-	100
epsilon	可选，最小似然误差	-	0.000001
enableSparse	可选，是否是稀疏格式	[true, false]	false
enableFitGoodness	可选，是否是要做模型评估，指标包括R-squared, AdjustedR-Squared, AIC, 自由度，残差的标准差，偏差	[true, false]	false
enableCoefficientEstimate	可选，是否要做回归系数评估，指标包括t值，p值，置信区间[2.5%, 97.5%]，只在enableFitGoodness为true时这个参数才会生效，否则该参数都处理为false	[true, false]	false
itemDelimiter	可选，稀疏格式kv对之间的分隔符，只有在enableSparse为true时生效	-	命令行默认值为空格”“, web页面默认值为”,”
kvDelimiter	可选，稀疏格式key和value之间的分隔符，只有在enableSparse为true时生效	-	默认值为”:”
lifecycle	可选，模型评估输出表的生命周期	0	-1
coreNum	可选，指定instance的	[1, 800)	默认自动计算

	总数		
memSizePerCore	可选，指定memory大小	[1024, 20*1024]	默认自动计算

实例

测试数据

新建数据SQL

```
 drop table if exists lm_test_input;
create table lm_test_input as
select
*
from
(
select 10 as y, 1.84 as x1, 1 as x2, '0:1.84 1:1' as sparsecol1 from dual
union all
select 20 as y, 2.13 as x1, 0 as x2, '0:2.13' as sparsecol1 from dual
union all
select 30 as y, 3.89 as x1, 0 as x2, '0:3.89' as sparsecol1 from dual
union all
select 40 as y, 4.19 as x1, 0 as x2, '0:4.19' as sparsecol1 from dual
union all
select 50 as y, 5.76 as x1, 0 as x2, '0:5.76' as sparsecol1 from dual
union all
select 60 as y, 6.68 as x1, 2 as x2, '0:6.68 1:2' as sparsecol1 from dual
union all
select 70 as y, 7.58 as x1, 0 as x2, '0:7.58' as sparsecol1 from dual
union all
select 80 as y, 8.01 as x1, 0 as x2, '0:8.01' as sparsecol1 from dual
union all
select 90 as y, 9.02 as x1, 3 as x2, '0:9.02 1:3' as sparsecol1 from dual
union all
select 100 as y, 10.56 as x1, 0 as x2, '0:10.56' as sparsecol1 from dual
) tmp;
```

运行命令

```
PAI -name linearregression
-project algo_public
-DinputTableName=lm_test_input
-DlabelColName=y
-DfeatureColNames=x1,x2
-DmodelName=lm_test_input_model_out
-DoutputTableName=lm_test_input_conf_out
-DenableCoefficientEstimate=true
-DenableFitGoodness=true
-Dlifecycle=1;

pai -name prediction
```

```
-project algo_public
-DmodelName=lm_test_input_model_out
-DinputTableName=lm_test_input
-DoutputTableName=lm_test_input_predict_out
-DappendColNames=y;
```

运行结果

lm_test_input_conf_out

```
+------------+------------+------------+------------+--------------------+------------+
| colname | value | tscore | pvalue | confidenceinterval | p |
+------------+------------+------------+------------+--------------------+------------+
| Intercept | -6.42378496687763 | -2.2725755951390028 | 0.06 | {"2.5%": -11.964027, "97.5%": -0.883543} |
coefficient |
| x1 | 10.260063429838898 | 23.270944360826963 | 0.0 | {"2.5%": 9.395908, "97.5%": 11.124219} | coefficient |
| x2 | 0.35374498323846265 | 0.2949247320997519 | 0.81 | {"2.5%": -1.997160, "97.5%": 2.704650} | coefficient |
| rsquared | 0.9879675667384592 | NULL | NULL | NULL | goodness |
| adjusted_rsquared | 0.9845297286637332 | NULL | NULL | NULL | goodness |
| aic | 59.331109494251805 | NULL | NULL | NULL | goodness |
| degree_of_freedom | 7.0 | NULL | NULL | NULL | goodness |
| standardErr_residual | 3.765777749448906 | NULL | NULL | NULL | goodness |
| deviance | 99.26757440771128 | NULL | NULL | NULL | goodness |
+------------+------------+------------+------------+--------------------+------------+
```

lm_test_input_predict_out

```
+------------+-------------------+-----------------+-------------------+
| y | prediction_result | prediction_score | prediction_detail |
+------------+-------------------+-----------------+-------------------+
| 10 | NULL | 12.808476727264404 | {"y": 12.8084767272644} |
| 20 | NULL | 15.43015013867922 | {"y": 15.43015013867922} |
| 30 | NULL | 33.48786177519568 | {"y": 33.48786177519568} |
| 40 | NULL | 36.565880804147355 | {"y": 36.56588080414735} |
| 50 | NULL | 52.674180388994415 | {"y": 52.67418038899442} |
| 60 | NULL | 62.82092871092313 | {"y": 62.82092871092313} |
| 70 | NULL | 71.34749583130122 | {"y": 71.34749583130122} |
| 80 | NULL | 75.75932310613193 | {"y": 75.75932310613193} |
| 90 | NULL | 87.1832221199846 | {"y": 87.18322211998461} |
| 100 | NULL | 101.92248485222113 | {"y": 101.9224848522211} |
+------------+-------------------+-----------------+-------------------+
```

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，勾选后可选择非输非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

训练采集样本比例

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, regresssion 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p1，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" -Dtau="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

示例

测试数据

新建数据SQL

```
drop table if exists gbdt_ls_test_input;
create table gbdt_ls_test_input
as
select
*
from
(
select
cast(1 as double) as f0,
cast(0 as double) as f1,
```

```
cast(0 as double) as f2,
cast(0 as double) as f3,
cast(0 as bigint) as label
from dual
union all
select
cast(0 as double) as f0,
cast(1 as double) as f1,
cast(0 as double) as f2,
cast(0 as double) as f3,
cast(0 as bigint) as label
from dual
union all
select
cast(0 as double) as f0,
cast(0 as double) as f1,
cast(1 as double) as f2,
cast(0 as double) as f3,
cast(1 as bigint) as label
from dual
union all
select
cast(0 as double) as f0,
cast(0 as double) as f1,
cast(0 as double) as f2,
cast(1 as double) as f3,
cast(1 as bigint) as label
from dual
union all
select
cast(1 as double) as f0,
cast(0 as double) as f1,
cast(0 as double) as f2,
cast(0 as double) as f3,
cast(0 as bigint) as label
from dual
union all
select
cast(0 as double) as f0,
cast(1 as double) as f1,
cast(0 as double) as f2,
cast(0 as double) as f3,
cast(0 as bigint) as label
from dual
) a;
```

输入数据说明

```
+------------+------------+------------+------------+------------+
| f0 | f1 | f2 | f3 | label |
+------------+------------+------------+------------+------------+
| 1.0 | 0.0 | 0.0 | 0.0 | 0 |
| 0.0 | 0.0 | 1.0 | 0.0 | 1 |
| 0.0 | 0.0 | 0.0 | 1.0 | 1 |
| 0.0 | 1.0 | 0.0 | 0.0 | 0 |
```

```
| 1.0 | 0.0 | 0.0 | 0.0 | 0 |
| 0.0 | 1.0 | 0.0 | 0.0 | 0 |
+------------+------------+------------+------------+------------+
```

运行命令

```
drop offlinemodel if exists gbdt_ls_test_model;
drop table if exists gbdt_ls_test_prediction_result;
PAI -name GBDT -project algo_public -DfeatureSplitValueMaxSize="500" -DlossType="3" -DrandSeed="0" -
DnewtonStep="1" -Dshrinkage="0.5" -DmaxLeafCount="32" -DlabelColName="label" -
DinputTableName="gbdt_ls_test_input" -DminLeafSampleCount="1" -DsampleRatio="1" -DmaxDepth="10" -
DmetricType="0" -DmodelName="gbdt_ls_test_model" -DfeatureRatio="1" -Dp="1" -Dtau="0.6" -DtestRatio="0" -
DfeatureColNames="f0,f1,f2,f3" -DtreeCount="10";
PAI -name prediction -project algo_public -DdetailColName="prediction_detail" -
DmodelName="gbdt_ls_test_model" -DitemDelimiter="," -DresultColName="prediction_result" -Dlifecycle="28" -
DoutputTableName="gbdt_ls_test_prediction_result" -DscoreColName="prediction_score" -DkvDelimiter=":" -
DinputTableName="gbdt_ls_test_input" -DenableSparse="false" -DappendColNames="label";
```

运行结果

gbdt_ls_test_prediction_result

```
+------------+------------------+-----------------+------------------+
| label | prediction_result | prediction_score | prediction_detail |
+------------+------------------+-----------------+------------------+
| 0 | NULL | 0.0 | {"label": 0} |
| 0 | NULL | 0.0 | {"label": 0} |
| 1 | NULL | 0.9990234375 | {"label": 0.9990234375} |
| 1 | NULL | 0.9990234375 | {"label": 0.9990234375} |
| 0 | NULL | 0.0 | {"label": 0} |
| 0 | NULL | 0.0 | {"label": 0} |
+------------+------------------+-----------------+------------------+
```

协同过滤etrec

etrec是一个item base的协同过滤算法，输入为两列或者三列，两列的情况下，第一列为user，第二列item。三列的情况下：第一列为user，第二列item，第三列为payload。输出为item之间相似度topK待补充参数说明

示例

测试数据

新建数据SQL

```
drop table if exists etrec_test_input;
create table etrec_test_input
as
select
*
```

```
from
(
select
cast(0 as string) as user,
cast(0 as string) as item
from dual
union all
select
cast(0 as string) as user,
cast(1 as string) as item
from dual
union all
select
cast(1 as string) as user,
cast(0 as string) as item
from dual
union all
select
cast(1 as string) as user,
cast(1 as string) as item
from dual
) a;
```

输入数据说明

```
+------------+------------+
| user | item |
+------------+------------+
| 0 | 0 |
| 0 | 1 |
| 1 | 0 |
| 1 | 1 |
+------------+------------+
```

运行命令

```
drop table if exists etrec_test_result;
PAI -name etrec -project algo_public -DsimilarityType="wbcosine" -Dweight="1" -DminUserBehavior="2" -
Dlifecycle="28" -DtopN="2000" -Dalpha="0.5" -DoutputTableName="etrec_test_result" -
DinputTableFormat="user-item" -DmaxUserBehavior="500" -DinputTableName="etrec_test_input" -
Doperator="add" -DselectedColNames="user,item";
```

运行结果

etrec_test_result

```
+------------+------------+
| itemid | similarity |
+------------+------------+
| 0 | 1:1 |
| 1 | 0:1 |
+------------+------------+
```

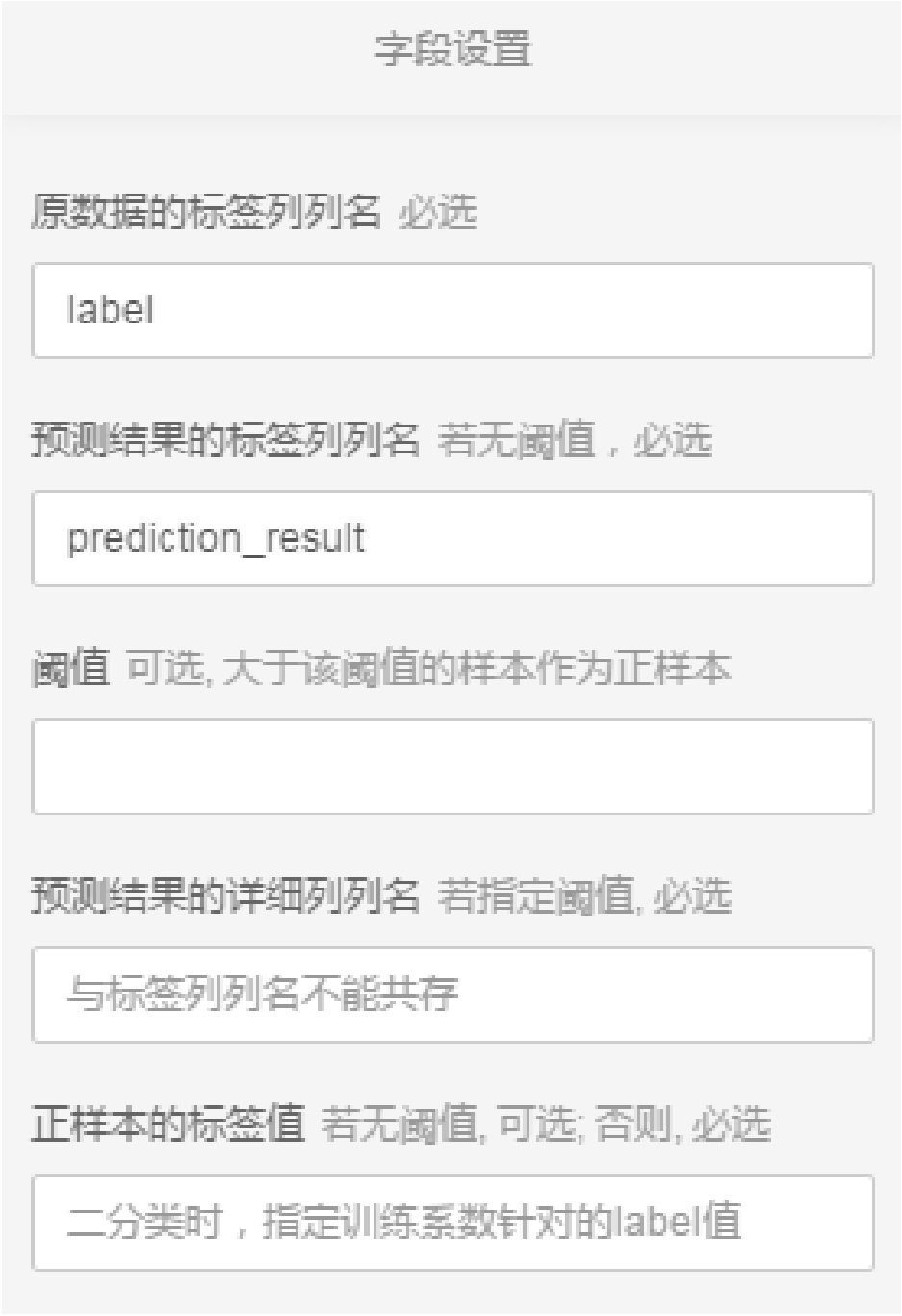
混淆矩阵

背景

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。详细定义可以参考ConfusionMatrix

算法组件

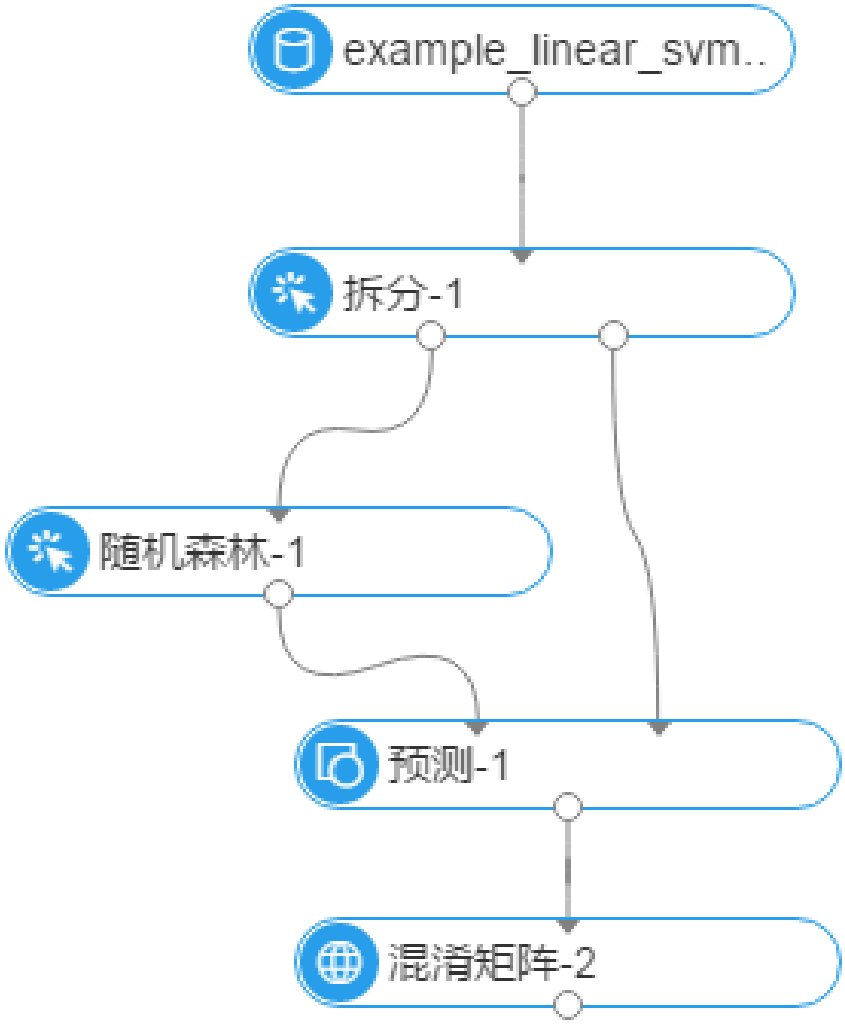
参数说明



- 注意：预测结果的标签列 与 详细列 这两个参数不能共存，只能填一个

组件的连接方式

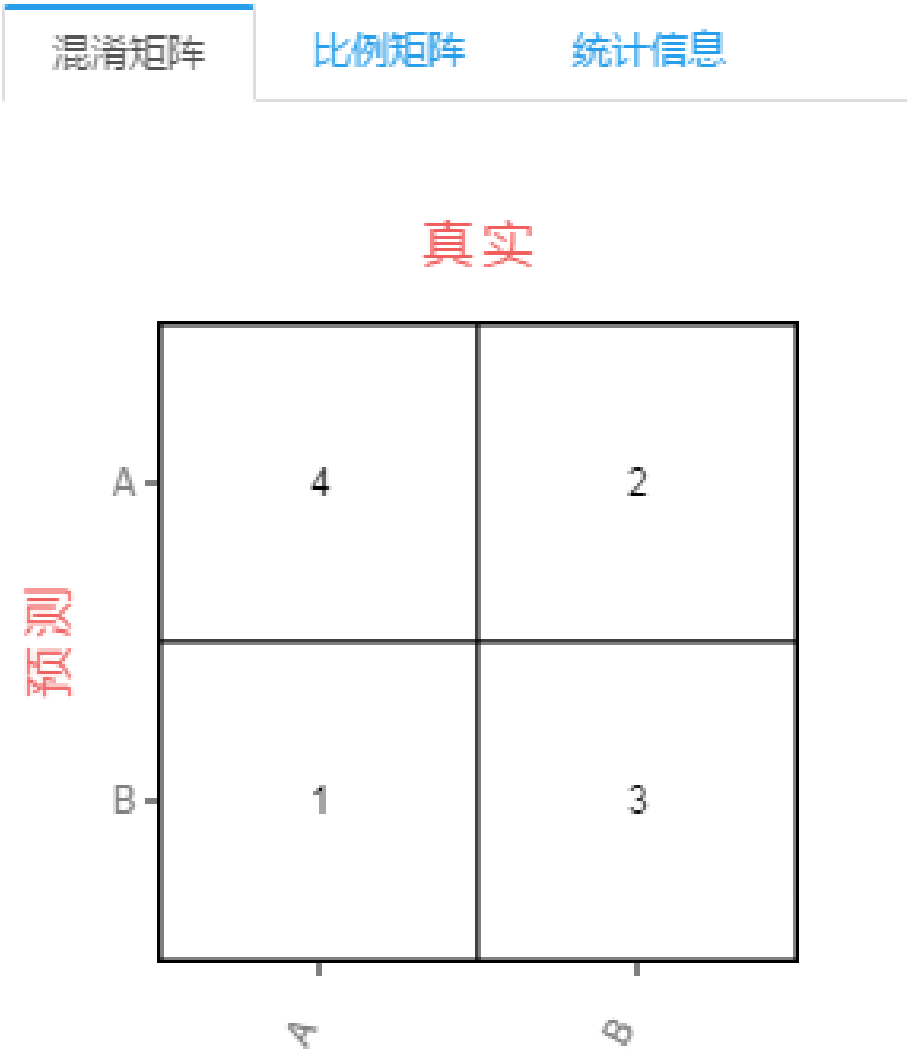
一般上层组件为分类预测，示例如下：



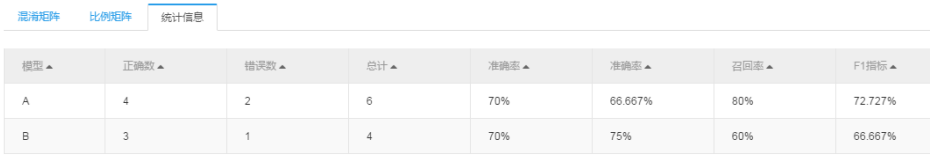
评估报告

右击混淆矩阵组件可以查看评估报告：

- 混淆矩阵



- 每个Label的指标，准确率、召回率等



模型	正确数	错误数	总计	准确率	准确率	召回率	F1指标
A	4	2	6	70%	66.667%	80%	72.727%
B	3	1	4	70%	75%	60%	66.667%

PAI命令

- 不指定阈值的示例

```
pai -name confusionmatrix -project algo_public \
-DinputTableName=wpbc_pred \
-DlabelColName=label \
-DpredictionColName=prediction_result \
-DoutputTableName=wpbc_confu;
```

- 指定阈值的示例

```
pai -name confusionmatrix -project algo_public \
-DinputTableName=wpbc_pred \
-DlabelColName=label \
-DpredictionDetailColName=prediction_detail \
-DgoodValue=N \
-Dthreshold=0.8 \
-DoutputTableName=wpbc_confu;
```

参数设置

参数key名称	参数描述	参数value可选项	默认值
inputTableName	必选，输入表的表名，即预测输出表	-	-
labelColName	必选，原始label列名	-	-
outputTableName	必选，输出表名，存储混淆矩阵	-	-
inputTablePartition	可选，输入表的partition	-	输入表的所有partition
predictionColName	可选，预测结果label列名，当不需要指定阈值时，该参数必须	-	-
predictionDetailColName	可选，预测结果表detail列名，当指定阈值时，该参数必须	-	-
threshold	可选，划分为goodValue的阈值	-	0.5
goodValue	可选，二分类时，指定训练系数针对的label值，当指定阈值时，该参数必须	-	-
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期

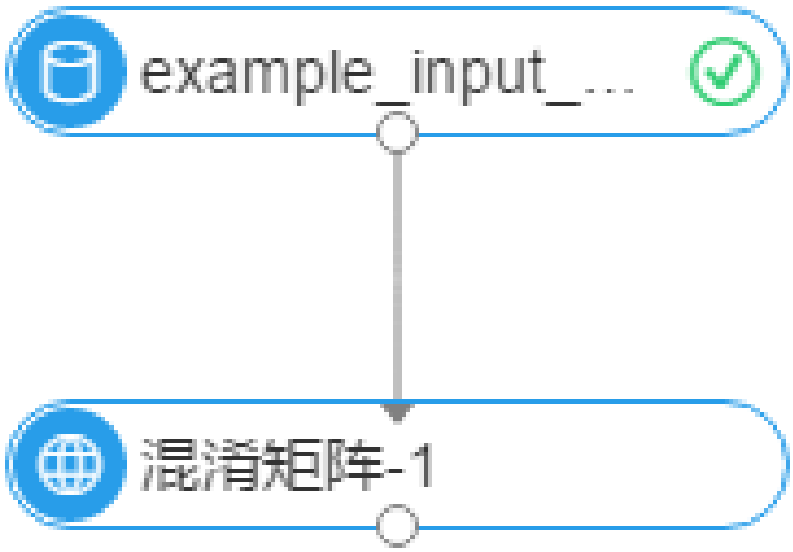
示例

测试数据

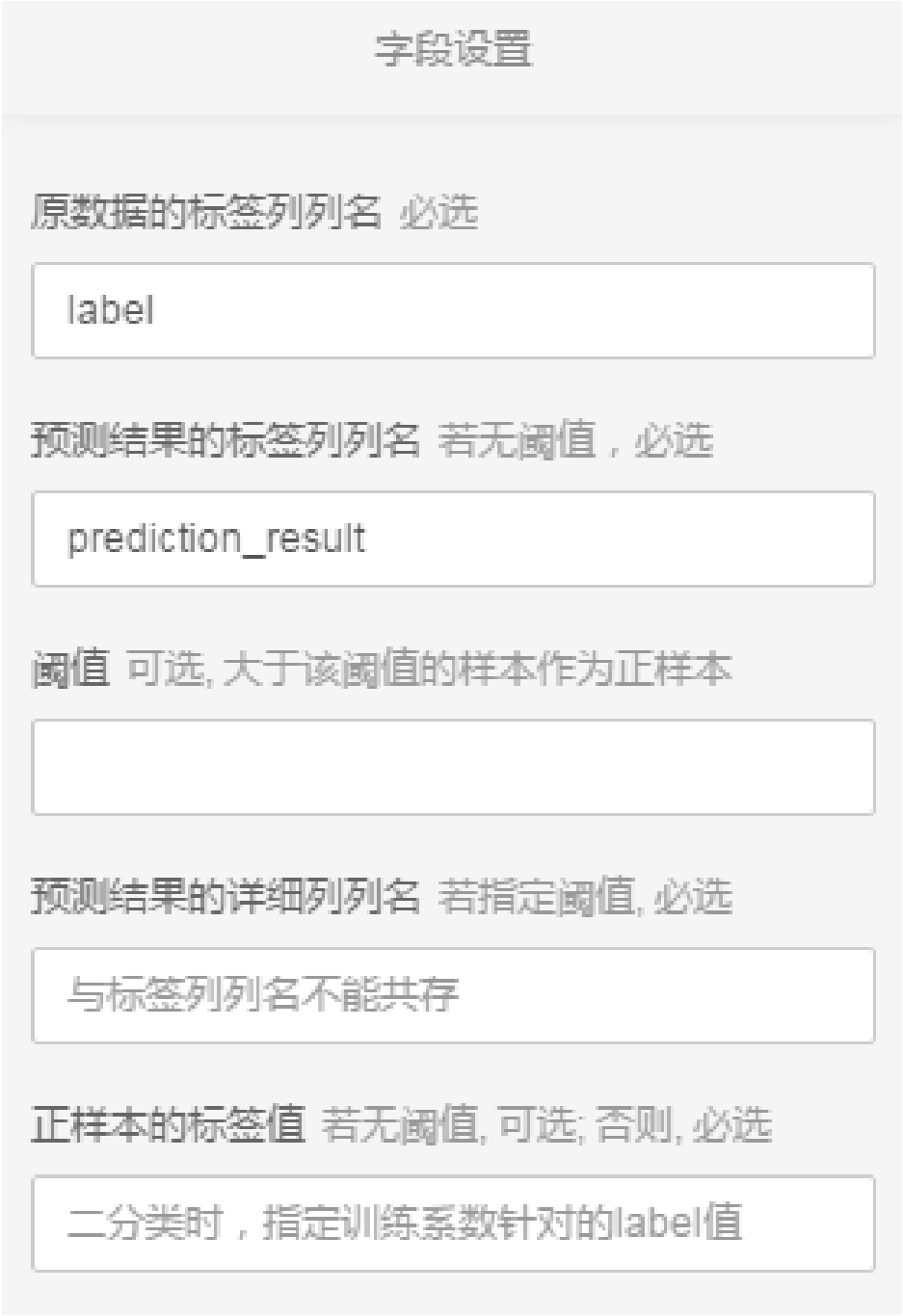
id	label	prediction_result
0	A	A
1	A	B
2	A	A
3	A	A

4	B	B
5	B	B
6	B	A
7	B	B
8	B	A
9	A	A

创建实验

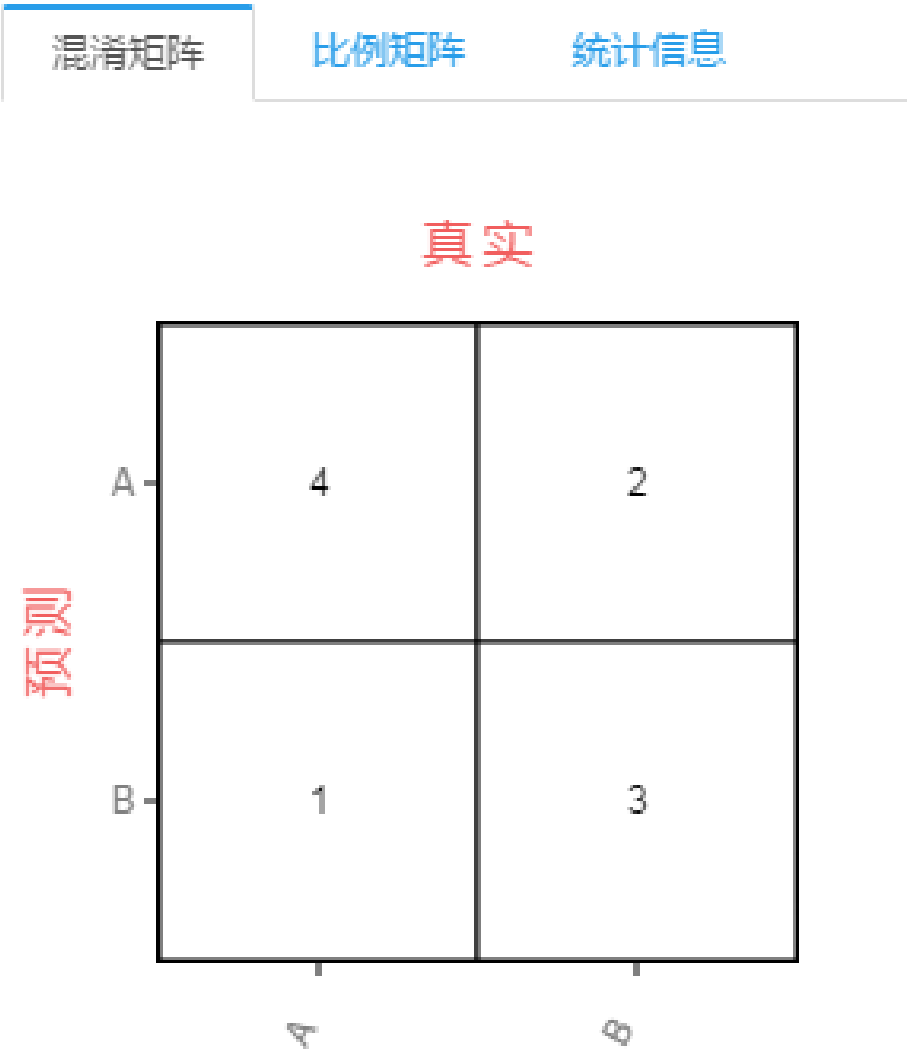


配置参数

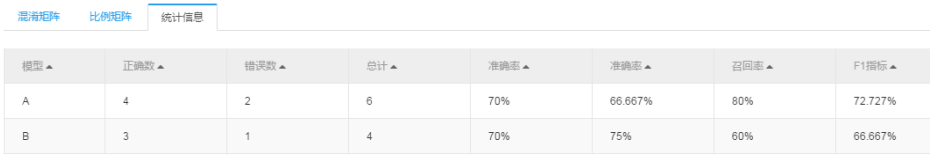

字段设置
原数据的标签列列名 必选
label
预测结果的标签列列名 若无阈值，必选
prediction_result
阈值 可选, 大于该阈值的样本作为正样本
预测结果的详细列列名 若指定阈值, 必选
与标签列列名不能共存
正样本的标签值 若无阈值, 可选; 否则, 必选
二分类时，指定训练系数针对的label值

运行实验

混淆矩阵如下：



每个Label的统计信息如下：



模型	正确数	错误数	总计	准确率	准确率	召回率	F1指标
A	4	2	6	70%	66.667%	80%	72.727%
B	3	1	4	70%	75%	60%	66.667%

多分类评估

背景

基于分类模型的预测结果和原始结果，评价多分类算法模型的优劣，指标包括Accuracy, kappa, F1-Score等

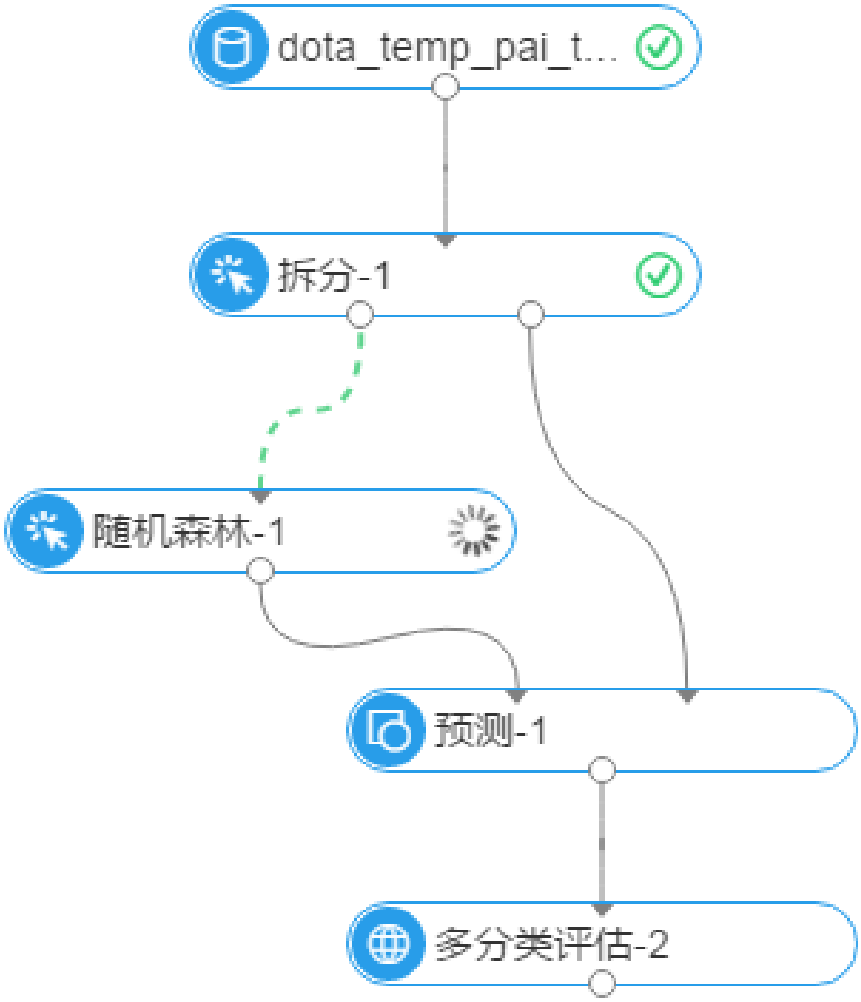
算法组件

参数说明

- 通过【原始分类结果列】，可以下拉选择”原始Label”列
  - 注意：支持的最大分类数为1000
- 通过【预测分类结果列】，可以选择”预测Label”列，这个值一般默认为prediction_result
- 通过【预测结果概率列】，可以选择”预测Label的概率列”；一般情况下，该字段的名为prediction_detail
  - 注意此参数仅支持随机森林预测

组件的连接方式

多分类评估组件需连接预测组件，不支持回归模型。连接示例如下：



评估报告说明

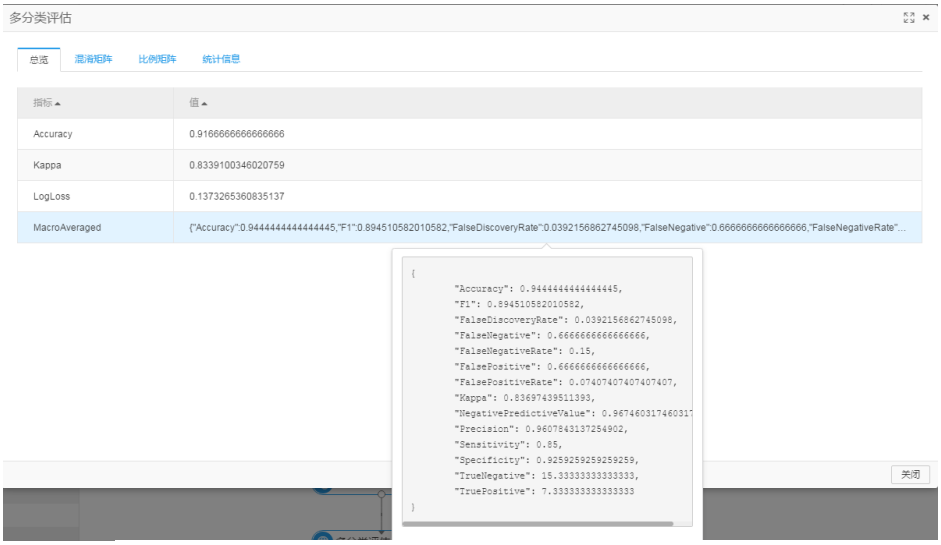
- 按照one-vs-all的方式计算每个label的指标

多分类评估

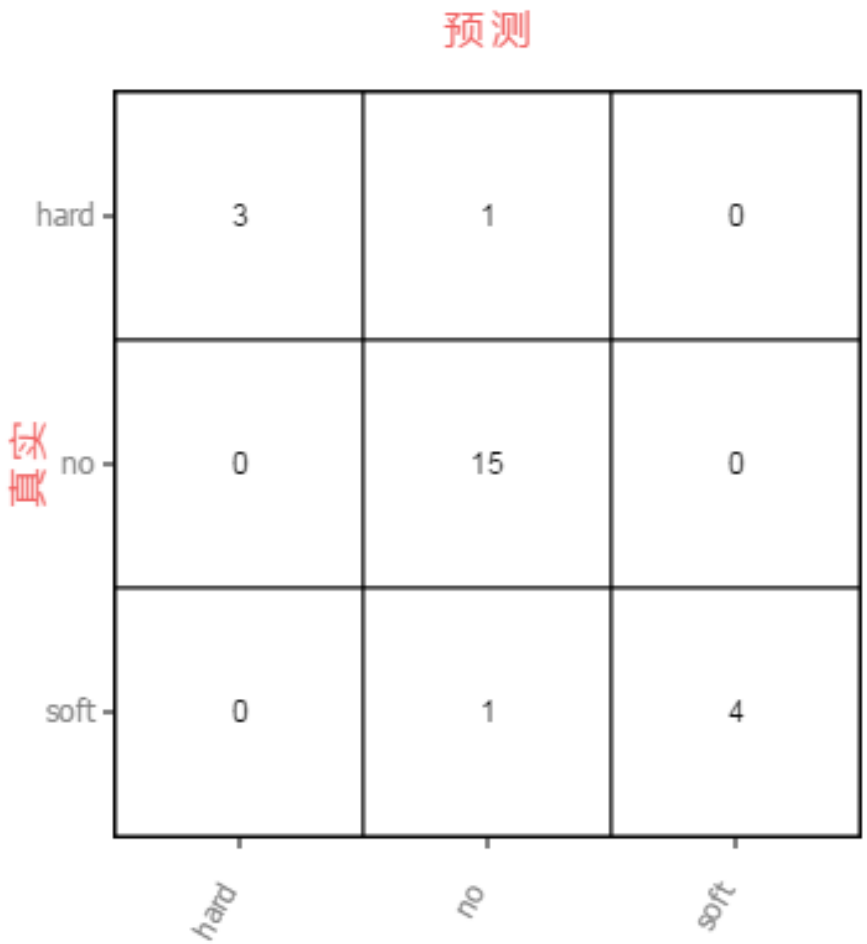
总览 混淆矩阵 比例矩阵 统计信息

模型	TruePositive	TrueNegative	FalsePositive	FalseNegative	Sensitivity	Specificity	Precision	Accuracy	F1	Kappa
hard	3	20	0	1	0.75	1	1	0.9583333...	0.857...	0.8333...
no	15	7	2	0	1	0.7777777...	0.8823529...	0.9166666...	0.9375	0.8139...
soft	4	19	0	1	0.8	1	1	0.9583333...	0.888...	0.8636...

- 下面是汇总的指标，其中MacroAveraged一项，是每个Label指标的平均值


多分类评估
总览
混淆矩阵
比例矩阵
统计信息
指标
值
Accuracy
0.9166666666666666
Kappa
0.8339100346020759
LogLoss
0.1373265360835137
MacroAveraged
{"Accuracy":0.9444444444444445,"F1":0.894510582010582,"FalseDiscoveryRate":0.039215686274509...
{
"Accuracy": 0.9444444444444445,
"F1": 0.894510582010582,
"FalseDiscoveryRate": 0.0392156862745098,
"FalseNegative": 0.6666666666666666,
"FalseNegativeRate": 0.15,
"FalsePositive": 0.6666666666666666,
"FalsePositiveRate": 0.0740740740740741,
"Kappa": 0.83697439511393,
"NegativePredictiveValue": 0.967460317460317
"Precision": 0.9607843137254902,
"Sensitivity": 0.85,
"Specificity": 0.9259259259259259,
"TrueNegative": 15.333333333333333,
"TruePositive": 7.333333333333333
}
关闭


总览
混淆矩阵
比例矩阵
统计信息


预测
hard
no
soft
3
1
0
0
15
0
0
1
4
hard
no
soft

- 混淆矩阵

PAI命令

```
PAI -name MultiClassEvaluation -project algo_public \
–DinputTableName="test_input" \
-DoutputTableName="test_output" \
-DlabelColName="label" \
-DpredictionColName="prediction_result" \
-Dlifecycle=30;
```

参数设置

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
outputTableName	必选，输出表表名	-	-
labelColName	必选，输入表原始label列名	-	-
predictionColName	必选，预测结果label列名	-	-
predictionDetailColName	可选，预测结果的概率列，格式如：{“A”:0.2, “B”:0.3, “C”，0.5}	-	默认为空
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期
coreNum	可选，计算的核心数	正整数	系统自动分配
memSizePerCore	可选，每个核心的内存（单位为兆）	正整数，范围(0, 65536)	系统自动分配

结果表的JSON格式说明

```
{
"LabelNumber": 3,
"LabelList": ["A", "B", "C"],
"ConfusionMatrix": [ // 混淆矩阵 [actual][predict]
[100, 10, 20],
[30, 50, 9],
[7, 40, 90] ],
"ProportionMatrix": [ // 按行的占比 [actual][predict]
[0.6, 0.2, 0.2],
[0.3, 0.6, 0.1],
[0.1, 0.4, 0.5] ],
"ActualLabelFrequencyList": [ // 每个label的真实的数目
200, 300, 600],
"ActualLabelProportionList": [ // 每个label的真实的占比
0.1, 0.2, 0.7],
"PredictedLabelFrequencyList": [ // 预测的每个label的数目
```

```
300, 400, 400],
"PredictedLabelProportionList": [ // 预测的每个label的占比
0.2, 0.1, 0.7],
"OverallMeasures": { // 汇总的指标
"Accuracy": 0.70,
"Kappa" : 0.3,
"MacroList": { // 每个label的指标的均值
"Sensitivity": 0.4,
"Specificity": 0.3,
},
"MicroList": { // 根据每个label的TP, TN, FP, FN的和，计算的指标
"Sensitivity": 0.4,
"Specificity": 0.3,
},
"LabelFrequencyBasedMicro": { // 每个label的指标的按频率的加权平均值
"Sensitivity": 0.4,
"Specificity": 0.3,
},
},
"LabelMeasuresList": [ // 每个label的指标
{
"Accuracy": 0.6,
"Sensitivity": 0.4,
"Specificity": 0.3,
"Kappa": 0.3
},
{
"Accuracy": 0.6,
"Sensitivity": 0.4,
"Specificity": 0.3,
"Kappa": 0.3
},
]
}
```

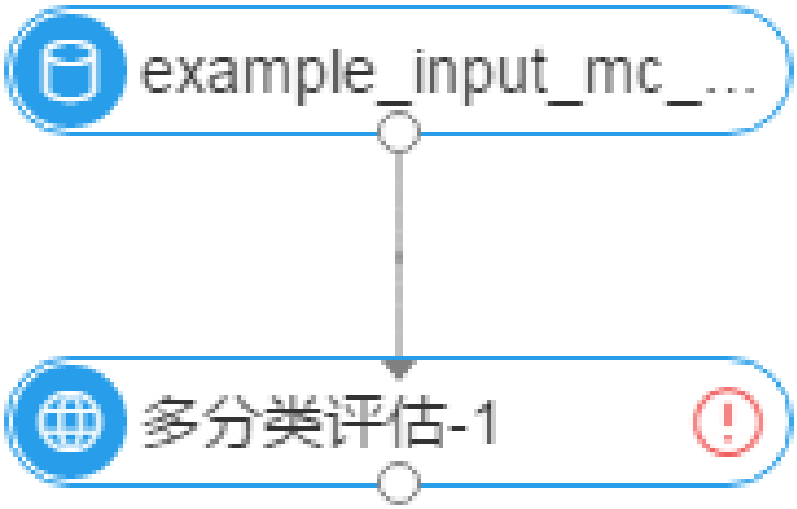
示例

测试数据，数据表example_input_mc_eval

id	label	prediction	detail
0	A	A	{“A”:0.6, “B”:0.4}
1	A	B	{“A”:0.45, “B”:0.55}
2	A	A	{“A”:0.7, “B”:0.3}
3	A	A	{“A”:0.9, “B”:0.1}
4	B	B	{“A”:0.2, “B”:0.8}
5	B	B	{“A”:0.1, “B”:0.9}

6	B	A	{“A”:0.52, “B”:0.48}
7	B	B	{“A”:0.4, “B”:0.6}
8	B	A	{“A”:0.6, “B”:0.4}
9	A	A	{“A”:0.75, “B”:0.25}

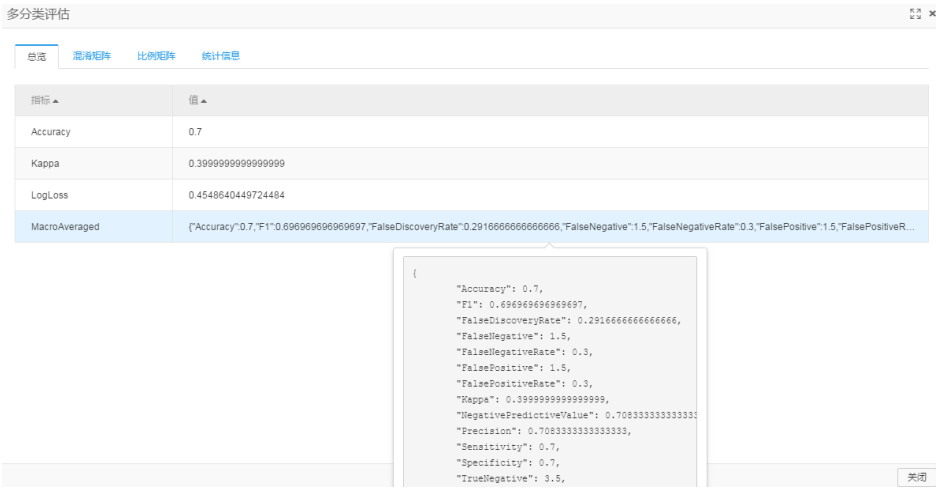
创建实验



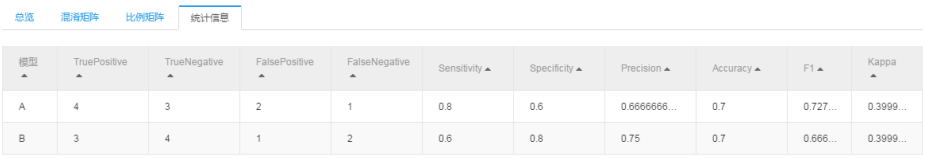
配置参数

运行实验

- 汇总报告如下：



- 每个label的统计信息如下：



- 具体输出JSON如下：


```
{
"ActualLabelFrequencyList": [5,
5],
"ActualLabelProportionList": [0.5,
0.5],
"ConfusionMatrix": [[4,
1],
[2,
3]],
"LabelList": ["A",
"B"],
"LabelMeasureList": [{
"Accuracy": 0.7,
"Auc": 0.9,
"F1": 0.7272727272727273,
"FalseDiscoveryRate": 0.3333333333333333,
"FalseNegative": 1,
"FalseNegativeRate": 0.2,
"FalsePositive": 2,
"FalsePositiveRate": 0.4,
"Kappa": 0.3999999999999999,
"NegativePredictiveValue": 0.75,
"Precision": 0.6666666666666666,
"Sensitivity": 0.8,
"Specificity": 0.6,
"TrueNegative": 3,
"TruePositive": 4},
{
"Accuracy": 0.7,
"Auc": 0.9,
"F1": 0.6666666666666666,
"FalseDiscoveryRate": 0.25,
"FalseNegative": 2,
"FalseNegativeRate": 0.4,
"FalsePositive": 1,
"FalsePositiveRate": 0.2,
"Kappa": 0.3999999999999999,
"NegativePredictiveValue": 0.6666666666666666,
"Precision": 0.75,
"Sensitivity": 0.6,
"Specificity": 0.8,
"TrueNegative": 4,
"TruePositive": 3}],
"LabelNumber": 2,
"OverallMeasures": {
"Accuracy": 0.7,
"Kappa": 0.3999999999999999,
```

"LabelFrequencyBasedMicro": {
"Accuracy": 0.7,
"F1": 0.696969696969697,
"FalseDiscoveryRate": 0.2916666666666666,
"FalseNegative": 1.5,
"FalseNegativeRate": 0.3,
"FalsePositive": 1.5,
"FalsePositiveRate": 0.3,
"Kappa": 0.3999999999999999,
"NegativePredictiveValue": 0.7083333333333333,
"Precision": 0.7083333333333333,
"Sensitivity": 0.7,
"Specificity": 0.7,
"TrueNegative": 3.5,
"TruePositive": 3.5},
"LogLoss": 0.4548640449724484,
"MacroAveraged": {
"Accuracy": 0.7,
"F1": 0.696969696969697,
"FalseDiscoveryRate": 0.2916666666666666,
"FalseNegative": 1.5,
"FalseNegativeRate": 0.3,
"FalsePositive": 1.5,
"FalsePositiveRate": 0.3,
"Kappa": 0.3999999999999999,
"NegativePredictiveValue": 0.7083333333333333,
"Precision": 0.7083333333333333,
"Sensitivity": 0.7,
"Specificity": 0.7,
"TrueNegative": 3.5,
"TruePositive": 3.5},
"MicroAveraged": {
"Accuracy": 0.7,
"F1": 0.7,
"FalseDiscoveryRate": 0.3,
"FalseNegative": 3,
"FalseNegativeRate": 0.3,
"FalsePositive": 3,
"FalsePositiveRate": 0.3,
"Kappa": 0.3999999999999999,
"NegativePredictiveValue": 0.7,
"Precision": 0.7,
"Sensitivity": 0.7,
"Specificity": 0.7,
"TrueNegative": 7,
"TruePositive": 7}},
"PredictedLabelFrequencyList": [6,
4],
"PredictedLabelProportionList": [0.6,
0.4],
"ProportionMatrix": [[0.8,
0.2],
[0.4,
0.6]]}

二分类评估

评估模块支持计算AUC,KS及F1 score，同时输出数据用于画KS曲线，PR曲线，ROC曲线，LIFT chart, Gain chart，同时也支持分组评估

PAI命令

```
pai -name=evaluate -project=algo_public
-DoutputMetricTableName=output_metric_table
-DoutputDetailTableName=output_detail_table
-DinputTableName=input_data_table
-DlabelColName=label
-DscoreColName=score
```

算法参数

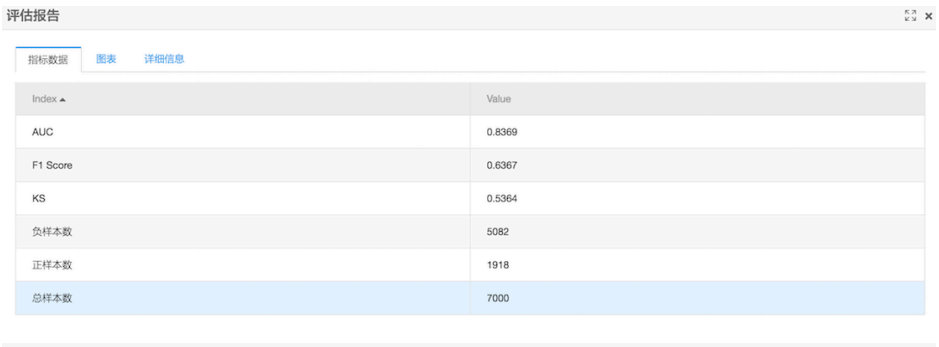
参数名称	参数描述	参数值可选项	默认值
inputTableName	必选，输入表名	-	-
inputTablePartitions	可选，输入表分区	-	默认选择全表
labelColName	必选，目标列的列名	-	-
scoreColName	必选，score列的列名	-	-
groupColName	可选，分组列的列名，用于分组评估场景	-	-
binCount	可选，计算KS，PR等指标时按等频分成多少个桶	-	默认1000
outputMetricTableName	必选，输出的指标表，两列，指标名和指标值，包含AUC,KS,F1 Score三行	-	-
outputDetailTableName	可选，输出用于画图的详细数据表	-	-
positiveLabel	可选，正样本的分类	-	默认为”1”
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

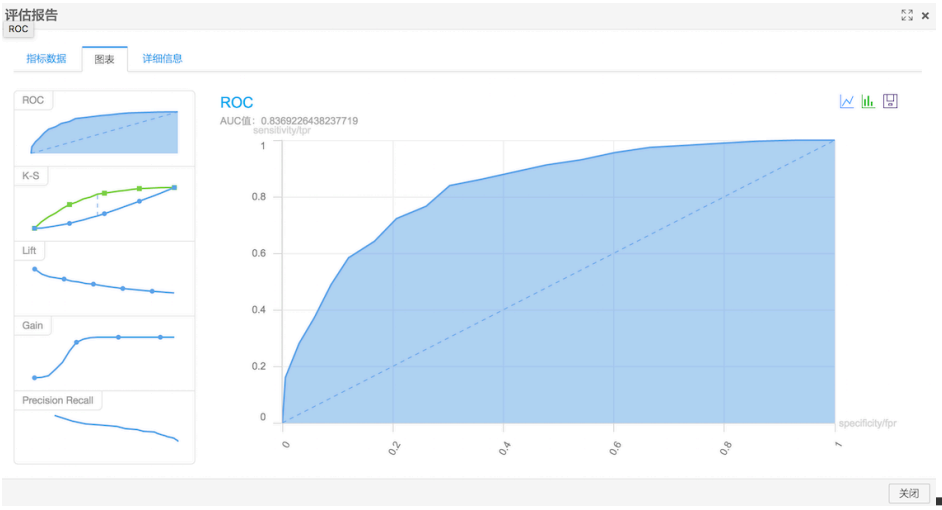
参数配置说明

- 通过【原始标签列列名】，可以下拉选择”原始Label”列
- 通过【分数列列名】，可以选择”预测分数”列，这个值一般默认为prediction_score
- 通过【正样本的标签值】，可以选择正样本所对应的标签值
- 通过【计算KS,PR等指标时按等频分成多少个桶】，可选择将数据等频划分多少个桶，默认为1000
- 通过【分组列列名】，选择进行分组评估时用于划分分组数据的分组列

结果展示

通过在”二分类评估“节点上点击右键在弹出菜单中选择”查看评估报告“可以查看可视化评估结果，如下图所示：





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，包含指标和残差直返图。其中指标包括SST, SSE, SSR, R2, R, MSE, RMSE, MAE, MAD, MAPE, count,yMean和predictMean。

PAI 命令

```
pai -name regression_evaluation -project algo_public
-DinputTableName=input_table
-DyColName=y_col
-DpredictionColName=prediction_col
-DindexOutputTableName=index_output_table
-DresidualOutputTableName=residual_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	-	-
indexOutputTableName	必选，回归指标输出表表名	-	-

residualOutputTableName	必选，残差直方图输出表表名	-	-
intervalNum	可选，直方图区间个数	-	100
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期
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

输出结果

回归指标输出表

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

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

pai命令示例

```
pai -name prediction
-DmodelName=nb_model
-DinputTableName=wpbc
-DoutputTableName=wpbc_pred
-DappendColNames=label;
```

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	数据输入表	表名	必选
modelName	模型	模型名字	必选
outputTableName	输出表	表名	必选
featureColNames	输入表中哪些列作为预测的feature，多列以逗号分隔	列名	可选，默认选择所有列
appendColNames	输出表中需要附加的预测输入表的列		可选，默认无附加列
inputTablePartitions	输入表中指定哪些分区参与预测，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		可选，默认选择输入表的所有partition
outputTablePartition	输出到输出表的分区		可选，默认输出表不进行分区
resultColName	输出表中result 列名		可选，默认prediction_result
scoreColName	输出表中score列名		可选，默认prediction_score
detailColName	输出表中detail列名		可选，默认prediction_detail
enableSparse	输入表数据是否为稀疏格式	true，false	可选，默认false
itemDelimiter	当输入表数据为稀疏格式时，kv间的分割符		可选，默认空格
kvDelimiter	当输入表数据为稀疏格式时，key和value的分割符		可选，默认冒号
lifecycle	指定输出表的生命周期	正整数	可选，默认没有生命周期

预测公式

朴素贝叶斯(NaiveBayes)

prediction_result 公式：

$$y' = \underset{y}{arg\,max}\ log(p(x)\prod_{i=1}^{n} p(x_i|y))$$

prediction_score 公式：

$$p = log(p(y_{max}) * \prod_{i=1}^{n} p(x_i|y_{max}))$$

分类变量：

$$p(x_i|y) = \frac{p(x_i \bigcap y)}{p(y)}$$

连续变量：

$$p(x_i|y) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

K均值(KMeans)

prediction_result 公式：

$$k' = \underset{k}{arg\,min}\ d(x_i - c_k)$$

prediction_score 公式:

$$d(x-c) = (x-c)\,(x-c)'$$

euclidean:

$$d(x-c) = 0.5 - 0.5 * \frac{xc'}{\sqrt{xx'}\sqrt{cc'}}$$

cosine:

$$d(x - c) = |x - c|$$

cityblock:

输出说明

分类	模型	prediction_result	prediction_score	prediction_detail
二分类	LogisticRegression模型	预测的label	预测label的概率	每个label及其对应的概率
	LinearSVM模型	预测的label	预测label的概率	每个label及其对应的概率
	RandomForests模型	预测的label	预测label的概率	每个label及其对应的概率
	GBDT_LR模型	预测的label	预测label的概率	每个label及其对应的概率
	NaiveBayes模型	预测的label	log(预测label的概率)	每个label及其对应的 log(概率)
	xgboost模型	预测的label	预测label的概率	每个label及其对应的概率
多分类	LogisticRegression模型	预测的label	预测label的概率	每个label及其对应的概率
	RandomForests模型	预测的label	预测label的概率	叶子节点存在的label及其概率
	NaiveBayes模型	预测的label	log(预测label的概率)	每个label及其对应的 log(概率)
回归	LinearRegression模型	空	回归值	label列名：回归值
	GBDT模型	空	回归值	label列名：回归值
	xgboost模型	空	回归值	label列名：回归值
聚类	KMeans模型	预测的中心点序号	到预测中心点的距离	到每个中心点的距离

示例

测试数据

```
create table pai_rf_test_input as
select * from
(
```

```
select 1 as f0,2 as f1, "good" as class from dual
union all
select 1 as f0,3 as f1, "good" as class from dual
union all
select 1 as f0,4 as f1, "bad" as class from dual
union all
select 0 as f0,3 as f1, "good" as class from dual
union all
select 0 as f0,4 as f1, "bad" as class from dual
)tmp;
```

构建模型

```
 PAI -name randomforests
-project algo_public
-DinputTableName="pai_rf_test_input"
-Dmodelbashame="pai_rf_test_model"
-DforceCategorical="f1"
-DlabelColName="class"
-DfeatureColNames="f0,f1"
-DmaxRecordSize="100000"
-DminNumPer="0"
-DminNumObj="2"
-DtreeNum="3";
```

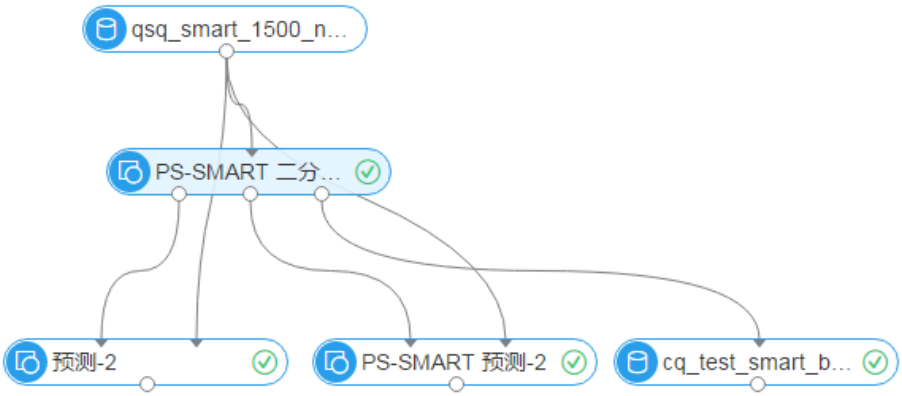
PAI预测

```
PAI -name prediction
-project algo_public
-DinputTableName=pai_rf_test_input
-DmodelName=pai_rf_test_model
-DresultColName=predict
```

PS-SMART二分类

PS是参数服务器（Parameter server）的简称。PS致力于解决大规模模型的离线、在线训练任务。SMART是Scalable Multiple Additive Regression Tree的缩写，是Gradient boosting decesion tree (GBDT)在PS上的一个实现。基于PS的Smart实现可以支持百亿样本、几十万特征的训练任务，可以在上千个节点上运行，且有failover功能，稳定性好。同时，PS-Smart支持多种数据格式、训练目标和评估目标，以及输出特征重要性，并包含直方图近似等加速训练的优化。

快速上手



图中我们使用训练数据学习了一个PS-SMART二分类模型。输出桩有3个，依次为

- 输出模型：offlinemodel，接统一的预测组件，目前不支持输出叶子节点编号
- 输出模型表：依然是二进制格式，不可读，是为了兼容已有PS-SMART预测组件，支持输出叶子节点编号，评估指标等功能。但对数据格式有较多要求，体验不佳，会逐渐改良或用其他组件代替。
- 输出特征重要性表：特征的重要性，有三种重要性类型可选（详见参数说明）

Pai命令行

训练

```
PAI -name ps_smart
-project algo_public
-DinputTableName="smart_binary_input"
-DmodelName="xlab_m_pai_ps_smart_bi_545859_v0"
-DoutputTableName="pai_temp_24515_545859_2"
-DoutputImportanceTableName="pai_temp_24515_545859_3"
-DlabelColName="label"
-DfeatureColNames="f0,f1,f2,f3,f4,f5"
-DenableSparse="false"
-Dobjective="binary:logistic"
-Dmetric="error"
-DfeatureImportanceType="gain"
-DtreeCount="5";
-DmaxDepth="5"
-Dshrinkage="0.3"
-Dl2="1.0"
-Dl1="0"
-Dlifecycle="3"
-DsketchEps="0.03"
-DsampleRatio="1.0"
-DfeatureRatio="1.0"
-DbaseScore="0.5"
-DminSplitLoss="0"
```

预测

```
PAI -name prediction
-project algo_public
-DinputTableName="smart_binary_input";
-DmodelName="xlab_m_pai_ps_smart_bi_545859_v0"
-DoutputTableName="pai_temp_24515_545860_1"
-DfeatureColNames="f0,f1,f2,f3,f4,f5"
-DappendColNames="label,qid,f0,f1,f2,f3,f4,f5"
-DenableSparse="false"
-Dlifecycle="28"
```

参数说明

数据参数

Pai命令选项	参数名称	参数描述	取值范围	是否必选，默认值
featureColNames	选择特征列	输入表中选择的用于训练的特征列名	列名，Dense格式只能选择数值类型（bigint或double），Sparse KV格式只能选择string类型，kv格式中key和value都必须是数值，不能是字符串。	必选
labelColName	选择标签列	输入表标签列列名	列名，支持string格式和数值格式的列，但内部存储的只能是数值类型，比如二分类时为0,1	必选
weightCol	选择权重列	可以给每行样本单独给出权重	列名，支持数值类型	可选，默认为空
enableSparse	是否稀疏格式	是否为稀疏格式，稀疏格式kv间分隔符为空格，key与value分隔符为冒号，比如1:0.3 3:0.9	[true, false]	可选，默认false
inputTableName	输入表名	-	-	必选
modelName	输出模型名	-	-	必选
outputImportanceTableName	输出特征重要性表名	-	-	可选，默认为空
inputTablePartitions	输入表分区	-	-	可选，格式如ds=1/pt=1

outputTableName	输出模型表名	输出到Odps表，也是二进制格式，不可读，可以使用Smart自带的预测组件，支持输出叶子节点编号	[true, false]	可选，默认false
lifecycle	输出表生命周期	-	正整数	可选，默认3

算法参数

Pai命令选项	参数名称	参数描述	取值范围	是否必选，默认值
objective	目标函数类型	目标函数类型直接决定了学习问题，需要正确选择。二分类时为binary:logistic		必选
metric	评估指标类型	训练集上的评估指标类型，输出在logview里coordinator的stdout中	logloss,error,auc	可选，默认为空
treeCount	树数量	树的棵树，训练时间与树的棵树呈正比	正整数	可选，默认为1
maxDepth	树最大深度	一棵树的最大深度，建议取5（即最多32个叶子节点）	正整数，[1,20]	可选，默认为5
sampleRatio	数据采样比例	构建每棵树时只采样一部分数据来学习，构建弱学习器，加快训练	(0,1]	可选，默认为1.0，不采样
featureRatio	特征采样比例	构建每棵树时只采样一部分特征来学习，构建弱学习器，加快训练	(0,1]	可选，默认为1.0，不采样
l1	L1惩罚项系数	控制叶子节点个数，该项越大，叶子节点数越少。过拟合时可以加大该项。	非负实数	可选，默认为0
l2	L2惩罚项系数	控制叶子节点大小，该项越大，叶子节点规模分布越均匀。过	非负实数	可选，默认为1.0

		拟合时可以加大该项。		
shrinkage	学习速率		(0,1]	可选，默认为0.3
sketchEps	近似Sketch精度	构造sketch时切割分位点的阈值，桶数为O(1.0/sketchEps)。这个值越小，切出来桶越多，一般不需要调整。	(0,1)	可选，默认为0.03
minSplitLoss	最小分裂损失变化	分裂节点所需要的最小损失变化，该值越大，分裂越保守	非负实数	可选，默认值0
featureNum	特征数量	特征的个数，或最大特征ID。当需要估计使用资源时需要填写	正整数	可选
baseScore	全局偏置项	所以样本的初始预测值	实数	可选，默认值0.5
featureImportanceType	特征重要性类型	计算特征重要性的类型。weight：模型中，该特征做为分裂特征的次数，gain：模型中，该特征带来的信息增益，cover：模型中，该特征在分裂节点覆盖的样本数。	可选的有”weight”，”gain”，”cover”	可选，默认”gain”

注意事项

1. objective在不同的学习问题中需要指定不同的值，二分类的web界面直接帮用户指定，不暴露给用户，命令行中需指定为binary:logistic。
2. metric的对应关系为：logloss对应negative loglikelihood for logistic regression, error对应binary classification error, auc对应Area under curve for classification。

执行调优

Pai命令选项	Pai web参数名称	参数描述	取值范围	是否必选，默认值
coreNum	核心数	核心个数，越多算法运行越快	正整数	可选，默认自动分配
memSizePerCore	每个核的内存大小（MB）	单个核心使用的内存数，1024代	正整数	可选，默认自动分配

		表1GB内存		

具体示例

数据生成

以Dense格式数据为例

```
drop table if exists smart_binary_input;
create table smart_binary_input lifecycle 3 as
select
*
from
(
select 0.72 as f0, 0.42 as f1, 0.55 as f2, -0.09 as f3, 1.79 as f4, -1.2 as f5, 0 as label from dual
union all
select 1.23 as f0, -0.33 as f1, -1.55 as f2, 0.92 as f3, -0.04 as f4, -0.1 as f5, 1 as label from dual
union all
select -0.2 as f0, -0.55 as f1, -1.28 as f2, 0.48 as f3, -1.7 as f4, 1.13 as f5, 1 as label from dual
union all
select 1.24 as f0, -0.68 as f1, 1.82 as f2, 1.57 as f3, 1.18 as f4, 0.2 as f5, 0 as label from dual
union all
select -0.85 as f0, 0.19 as f1, -0.06 as f2, -0.55 as f3, 0.31 as f4, 0.08 as f5, 1 as label from dual
union all
select 0.58 as f0, -1.39 as f1, 0.05 as f2, 2.18 as f3, -0.02 as f4, 1.71 as f5, 0 as label from dual
union all
select -0.48 as f0, 0.79 as f1, 2.52 as f2, -1.19 as f3, 0.9 as f4, -1.04 as f5, 1 as label from dual
union all
select 1.02 as f0, -0.88 as f1, 0.82 as f2, 1.82 as f3, 1.55 as f4, 0.53 as f5, 0 as label from dual
union all
select 1.19 as f0, -1.18 as f1, -1.1 as f2, 2.26 as f3, 1.22 as f4, 0.92 as f5, 0 as label from dual
union all
select -2.78 as f0, 2.33 as f1, 1.18 as f2, -4.5 as f3, -1.31 as f4, -1.8 as f5, 1 as label from dual
) tmp;
```

数据内容如下

序号	f0	f1	f2	f3	f4	f5	label
1	0.72	0.42	0.55	-0.09	1.79	-1.2	0
2	1.23	-0.33	-1.55	0.92	-0.04	-0.1	1
3	-0.2	-0.55	-1.28	0.48	-1.7	1.13	1
4	1.24	-0.68	1.82	1.57	1.18	0.2	0
5	-0.85	0.19	-0.06	-0.55	0.31	0.08	1
6	0.58	-1.39	0.05	2.18	-0.02	1.71	0
7	-0.48	0.79	2.52	-1.19	0.9	-1.04	1
8	1.02	-0.88	0.82	1.82	1.55	0.53	0
9	1.19	-1.18	-1.1	2.26	1.22	0.92	0
10	-2.78	2.33	1.18	-4.5	-1.31	-1.8	1

数据中有6列特征。

训练

如快速上手中所示，配置训练数据和训练组件。选择label为目标列，f0,f1,f2,f3,f4,f5为特征列。算法参数设置页面如下图

字段设置 参数设置 执行调优

评估指标类型 可选 ⓘ

Area under curve for classification

树数量 可选 正整数 ⓘ

5

树最大深度 可选 正整数 ⓘ

5

数据采样比例 可选 (0,1] ⓘ

1.0

特征采样比例 可选 (0,1] ⓘ

1.0

L1惩罚项系数 可选 非负实数 ⓘ

0

L2惩罚项系数 可选 非负实数 ⓘ

1.0

学习速率 可选 (0,1]

0.3

近似Sketch精度 可选 (0,1] ⓘ

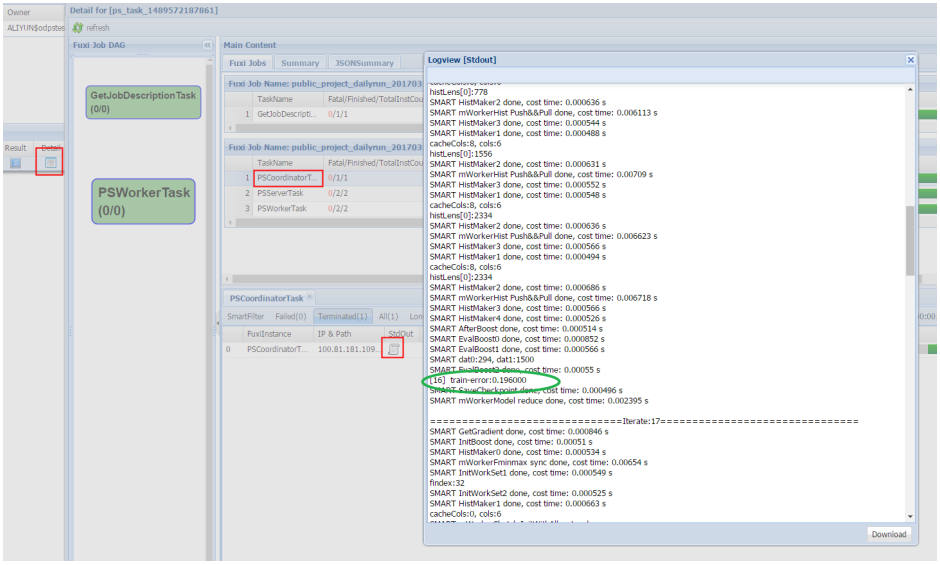
特征数量可以不填，算法会自动计算。如果您的特征数量较大，且需要算法较为精准地估计使用的资源，请在这里给出实际的值。

如果您想加速训练，可以通过执行调优页面设置核数目，核数目越多，算法运行越快。核内存一般不需要用户填写，算法可以较为精确地估计。另外，PS算法需要所有机器申请到资源才开始运行，当集群较忙时，申请较多资源可能会增加等待时间。

可以通过Logview(日志中以http://logview.odps.aliyun-inc.com:8080/logview开头的http链接)中Coordinator的stdout查看metric输出值。一个PS-Smart训练任务会有多个task，因而有多个logview，需要选取紧接着是PS开头输出的logview，如下图

红框中就是我们想要的PS任务的logview，可以通过绿色圆形中的显示来区分。

有了logview之后具体操作方式见下图



预测

使用统一预测组件预测

训练得到的输出模型以二进制方式存储，可以用来做预测。如快速上手中所示，配置预测组件的输入，也就是模型和测试数据，并正确配置参数如下图


字段设置
执行调优
特征列 默认全选
已选择 6 个字段
原样输出列 推荐添加label列,方便评估
已选择 7 个字段
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
key与value分隔符
:
kv对间的分隔符
,

Dense格式选择特征列（默认全选，多的列不影响预测）即可。如果是KV格式，需要选择稀疏格式并正确配置分隔符。Smart的kv对之间的分隔符是空格，需要将其置为空格或配置为\u0020（空格的转义表达形式）。

预测结果如下

序号	f0	f1	f2	f3	f4	f5	label	prediction_result	prediction_score	prediction_detail
1	0.72	0.42	0.55	-0.09	1.79	-1.2	0	0	0.5563381910324097	{"0": 0.5563382208347321, "1": 0.4436617791652679}
2	1.23	-0.33	-1.55	0.92	-0.04	-0.1	1	1	0.6076213121414185	{"0": 0.3923786878585815, "1": 0.6076213121414185}
3	-0.2	-0.55	-1.28	0.48	-1.7	1.13	1	1	0.5768264532089233	{"0": 0.4231735467910767, "1": 0.5768264532089233}
4	1.24	-0.68	1.82	1.57	1.18	0.2	0	0	0.7096900939941406	{"0": 0.709690123796463, "1": 0.290309876203537}
5	-0.85	0.19	-0.06	-0.55	0.31	0.08	1	1	0.7265769839286804	{"0": 0.2734230160713196, "1": 0.7265769839286804}
6	0.58	-1.39	0.05	2.18	-0.02	1.71	0	0	0.5573073625564575	{"0": 0.5573073327541351, "1": 0.4426926672458649}
7	-0.48	0.79	2.52	-1.19	0.9	-1.04	1	1	0.5777847170829773	{"0": 0.4222152829170227, "1": 0.5777847170829773}
8	1.02	-0.88	0.82	1.82	1.55	0.53	0	0	0.7096900939941406	{"0": 0.709690123796463, "1": 0.290309876203537}
9	1.19	-1.18	-1.1	2.26	1.22	0.92	0	0	0.7096900939941406	{"0": 0.709690123796463, "1": 0.290309876203537}
10	-2.78	2.33	1.18	-4.5	-1.31	-1.8	1	1	0.7265769839286804	{"0": 0.2734230160713196, "1": 0.7265769839286804}

prediction_detail列中1是正例，0是负例，后面是属于该类别的概率。

使用PS-Smart预测组件预测

训练得到的输出模型表存储的是二进制内，可以兼容原PS-Smart预测组件来做预测。如快速上手中所示，配置预测组件的输入，也就是模型和测试数据，并正确配置参数。包括数据格式，特征列和目标列以及类别数，ID列只能选择string格式的除特征列和目标列以外的列。参数设置中的损失函数必须显式选择为binary:logistic。预测结果如下

序号	original_label	prediction_score	leaf_index
1	0	0.4066115617752075	2 2 2 2 2
2	1	0.5709302425384521	2 2 1 1 2
3	1	0.6125051379203796	1 1 1 1 1
4	0	0.3217407166957855	2 1 2 2 1
5	1	0.6954338550567627	1 2 1 1 2
6	0	0.4794751703739166	2 1 1 1 1
7	1	0.5404139757156372	1 2 2 2 2
8	0	0.3217407166957855	2 1 2 2 1
9	0	0.3217407166957855	2 1 2 2 1
10	1	0.6954338550567627	1 2 1 1 2

其中prediction_score是预测的正例的概率，该值大于0.5，可以认为预测为正例，否则预测为负例。leaf_index列为预测的叶子节点编号，每个样本有N个数，N为树的棵树。每棵树对应一个数字，该数字表示样本落在这棵树上的叶子节点的编号。

注：

- 此处使用的输出模型表依然是二进制格式，不可读，是为了兼容已有PS-SMART预测组件，支持输出叶子节点编号，评估指标等功能。但对数据格式有较多要求，体验不佳，会逐渐改良或用其他组件代替。
- 此处要求必须给一列string格式的列作为label，可以填充一列字符串，不能为空或者NULL，可以将

一列特征通过类型转换组件转换为string格式传入。
- 参数设置中的损失函数必须显式选择为binary:logistic，默认继承不work。

查看特征重要性

可以将第三个输出桩输出到表来查看，或者直接在PS-Smart训练组件中点击右键查看数据-输出特征重要性表，查看重要性表内容，如下图

序号	id	value
1	0	0.5690338015556335
2	1	0.21714292466640472
3	4	0.21382322907447815

其中的id是传入的特征的序号，在这里是稠密格式，传入的特征是”f0,f1,f2,f3,f4,f5”，因此id为0指的就是f0，id为4指的就是f4。如果是KV格式，那么id就是Key-value pair中的key。value对应特征重要性类型，默认是gain，也就是模型中，该特征带来的信息增益的和。这里只出现了3个特征，是因为在树的分裂过程中只用到了这3个特征，可以认为没有出现的特征重要性为0。

常见问题

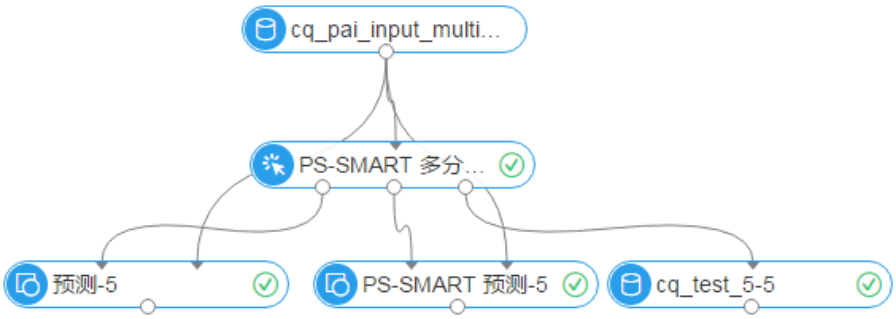
- PS-SMART二分类的目标列只支持数值类型（0代表负例，1代表正例），即使ODPS表这里是string类型，也必须存储的是数值。如果分类目标是”Good”,”Bad”这样的类别字符串，请用户将分类目标自行转换成1,0。
- KV格式时特征的ID必须是正整数，特征的值必须是实数。如果特征ID为字符串，需要使用序列化组件进行序列化操作。如果特征的值是类别型的字符串，需要进行特征离散化等特征工程处理。
- 虽然PS-SMART支持数十万特征的任务，但消耗资源较大且运行速度较慢，不建议使用这么大的特征规模。GBDT类算法适合直接使用连续特征进行训练，除了类别特征需要做one-hot编码（可以筛掉低频特征）外，其他连续型数值特征不建议做离散化，可以直接用于GBDT类算法的训练。
- PS-SMART算法有很多地方会引入随机性，比如data_sample_ratio，fea_sample_ratio选项分别对数据或特征做采样。除此之外，PS-SMART算法本身使用直方图做近似，当在集群上多个worker分布式执行时，由于局部sketch归并成全局sketch的顺序会有随机性，顺序不同会导致树结构的不同，但从理论上可以保证模型效果相差不大，请您放心使用。因此同样数据同样参数多次跑，结果不一致是正常的。

PS-SMART多分类

PS是参数服务器（Parameter server）的简称。PS致力于解决大规模模型的离线、在线训练任务。SMART是Scalable Multiple Additive Regression Tree的缩写，是Gradient boosting decesion tree (GBDT)在PS上的

一个实现。基于PS的Smart实现可以支持百亿样本、几十万特征的训练任务，可以在上千个节点上运行，且有failover功能，稳定性好。同时，PS-Smart支持多种数据格式、训练目标和评估目标，以及输出特征重要性，并包含直方图近似等加速训练的优化。

快速上手



图中我们使用训练数据学习了一个PS-SMART多分类模型。输出桩有3个，依次为

- 输出模型：offlinemodel，接统一的预测组件，目前不支持输出叶子节点编号
- 输出模型表：依然是二进制格式，不可读，是为了兼容已有PS-SMART预测组件，支持输出叶子节点编号，评估指标等功能。但对数据格式有较多要求，体验不佳，会逐渐改良或用其他组件代替。
- 输出特征重要性表：特征的重要性，有三种重要性类型可选（详见参数说明）

Pai命令行

训练

```
PAI -name ps_smart
-project algo_public
-DinputTableName="smart_multiclass_input"
-DmodelName="xlab_m_pai_ps_smart_bi_545859_v0"
-DoutputTableName="pai_temp_24515_545859_2"
-DoutputImportanceTableName="pai_temp_24515_545859_3"
-DlabelColName="label"
-DfeatureColNames="features"
-DenableSparse="true"
-Dobjective="multi:softprob"
-Dmetric="mlogloss"
-DfeatureImportanceType="gain"
-DtreeCount="5";
-DmaxDepth="5"
-Dshrinkage="0.3"
-Dl2="1.0"
-Dl1="0"
-Dlifecycle="3"
-DsketchEps="0.03"
-DsampleRatio="1.0"
-DfeatureRatio="1.0"
```

```
-DbaseScore="0.5"
-DminSplitLoss="0"
```

预测

```
PAI -name prediction
-project algo_public
-DinputTableName="smart_multiclass_input";
-DmodelName="xlab_m_pai_ps_smart_bi_545859_v0"
-DoutputTableName="pai_temp_24515_545860_1"
-DfeatureColNames="features"
-DappendColNames="label,features"
-DenableSparse="true"
-DkvDelimiter=":"
-Dlifecycle="28"
```

参数说明

数据参数

Pai命令选项	参数名称	参数描述	取值范围	是否必选，默认值
featureColNames	选择特征列	输入表中选择的用于训练的特征列名	列名，Dense格式只能选择数值类型（bigint或double），Sparse KV格式只能选择string类型，kv格式中key和value都必须是数值，不能是字符串。	必选
labelColName	选择标签列	输入表标签列列名	列名，支持string格式和数值格式的列，但内部存储的只能是数值类型，多分类时为0,1,2,...,n-1，n为类别数	必选
weightCol	选择权重列	可以给每行样本单独给出权重	列名，支持数值类型	可选，默认为空
enableSparse	是否稀疏格式	是否为稀疏格式，稀疏格式kv间分隔符为空格，key与value分隔符为冒号，比如1:0.3 3:0.9	[true, false]	可选，默认false
inputTableNam	输入表名	-	-	必选

e				
modelName	输出模型名	-	-	必选
outputImportanceTableName	输出特征重要性表名	-	-	可选，默认为空
inputTablePartitions	输入表分区	-	-	可选，格式如ds=1/pt=1
outputTableName	输出模型表名	输出到Odps表，也是二进制格式，不可读，可以使用Smart自带的预测组件，支持输出叶子节点编号	[true, false]	可选，默认false
lifecycle	输出表生命周期	-	正整数	可选，默认3

算法参数

Pai命令选项	参数名称	参数描述	取值范围	是否必选，默认值
classNum	类别数	多分类的类别数，如类别数为n，则标签列取值为0,1,2,...,n-1	非负整数，大于等于3	必选
objective	目标函数类型	目标函数类型直接决定了学习问题，需要正确选择。多分类时为multi:softprob		必选
metric	评估指标类型	训练集上的评估指标类型，输出在logview里coordinator的stdout中	mlogloss,merror	可选，默认为空
treeCount	树数量	树的棵树，训练时间与树的棵树呈正比	正整数	可选，默认为1
maxDepth	树最大深度	一棵树的最大深度，建议取5（即最多32个叶子节点）	正整数，[1,20]	可选，默认为5
sampleRatio	数据采样比例	构建每棵树时只采样一部分数据来学习，构建弱学习器，加快训练	(0,1]	可选，默认为1.0，不采样
featureRatio	特征采样比例	构建每棵树时只采样一部分特征	(0,1]	可选，默认为1.0，不采样

		来学习，构建弱学习器，加快训练		
l1	L1惩罚项系数	控制叶子节点个数，该项越大，叶子节点数越少。过拟合时可以加大该项。	非负实数	可选，默认为0
l2	L2惩罚项系数	控制叶子节点大小，该项越大，叶子节点规模分布越均匀。过拟合时可以加大该项。	非负实数	可选，默认为1.0
shrinkage	学习速率		(0,1]	可选，默认为0.3
sketchEps	近似Sketch精度	构造sketch时切割分位点的阈值，桶数为O(1.0/sketchEps)。这个值越小，切出来桶越多，一般不需要调整。	(0,1)	可选，默认为0.03
minSplitLoss	最小分裂损失变化	分裂节点所需要的最小损失变化，该值越大，分裂越保守	非负实数	可选，默认值0
featureNum	特征数量	特征的个数，或最大特征ID。当需要估计使用资源时需要填写	正整数	可选
baseScore	全局偏置项	所以样本的初始预测值	实数	可选，默认值0.5
featureImportanceType	特征重要性类型	计算特征重要性的类型。 weight：模型中，该特征做为分裂特征的次数，gain：模型中，该特征带来的信息增益，cover：模型中，该特征在分裂节点覆盖的样本数。	可选的有”weight”，”gain”，”cover”	可选，默认”gain”

注意事项

1. objective在不同的学习问题中需要指定不同的值，多分类的web界面直接帮用户指定，不暴露给用户，命令行中需指定为multi:softprob。

2. metric的对应关系为：mlogloss对应multiclass negative log likelihood, :merror对应multiclass classification error。

执行调优

Pai命令选项	Pai web参数名称	参数描述	取值范围	是否必选，默认值
coreNum	核心数	核心个数，越多算法运行越快	正整数	可选，默认自动分配
memSizePerCore	每个核的内存大小（MB）	单个核心使用的内存数，1024代表1GB内存	正整数	可选，默认自动分配

具体示例

数据生成

以KV格式数据为例

```
drop table if exists smart_multiclass_input;
create table smart_multiclass_input lifecycle 3 as
select
*
from
(
select 2 as label, '1:0.55 2:-0.15 3:0.82 4:-0.99 5:0.17' as features from dual
union all
select 1 as label, '1:-1.26 2:1.36 3:-0.13 4:-2.82 5:-0.41' as features from dual
union all
select 1 as label, '1:-0.77 2:0.91 3:-0.23 4:-4.46 5:0.91' as features from dual
union all
select 2 as label, '1:0.86 2:-0.22 3:-0.46 4:0.08 5:-0.60' as features from dual
union all
select 1 as label, '1:-0.76 2:0.89 3:1.02 4:-0.78 5:-0.86' as features from dual
union all
select 1 as label, '1:2.22 2:-0.46 3:0.49 4:0.31 5:-1.84' as features from dual
union all
select 0 as label, '1:-1.21 2:0.09 3:0.23 4:2.04 5:0.30' as features from dual
union all
select 1 as label, '1:2.17 2:-0.45 3:-1.22 4:-0.48 5:-1.41' as features from dual
union all
select 0 as label, '1:-0.40 2:0.63 3:0.56 4:0.74 5:-1.44' as features from dual
union all
select 1 as label, '1:0.17 2:0.49 3:-1.50 4:-2.20 5:-0.35' as features from dual
) tmp;
```

数据内容如下

序号	label	features
1	2	1:0.55 2:-0.15 3:0.82 4:-0.99 5:0.17
2	1	1:-1.26 2:1.36 3:-0.13 4:-2.82 5:-0.41
3	1	1:-0.77 2:0.91 3:-0.23 4:-4.46 5:0.91
4	2	1:0.86 2:-0.22 3:-0.46 4:0.08 5:-0.60
5	1	1:-0.76 2:0.89 3:1.02 4:-0.78 5:-0.86
6	1	1:2.22 2:-0.46 3:0.49 4:0.31 5:-1.84
7	0	1:-1.21 2:0.09 3:0.23 4:2.04 5:0.30
8	1	1:2.17 2:-0.45 3:-1.22 4:-0.48 5:-1.41
9	0	1:-0.40 2:0.63 3:0.56 4:0.74 5:-1.44
10	1	1:0.17 2:0.49 3:-1.50 4:-2.20 5:-0.35

数据中有5维特征。

训练

如**快速上手**中所示，配置训练数据和训练组件。选择label为目标列，features为特征列。数据参数和算法参数设置页面如下图


字段设置
参数设置
执行调优
是否稀疏格式 kv kv
选择特征列 必选
已选择 1 个字段
选择标签列 必选
已选择 1 个字段
选择权重列 可选
选择字段

字段设置 参数设置 执行调优

类别数 必选 非负整数 ?

3

评估指标类型 可选 ?

multiclass negative log likelihood

树数量 可选 正整数 ?

5

树最大深度 可选 正整数 ?

5

数据采样比例 可选 (0,1] ?

1.0

特征采样比例 可选 (0,1] ?

1.0

L1惩罚项系数 可选 非负实数 ?

0

L2惩罚项系数 可选 非负实数 ?

1.0

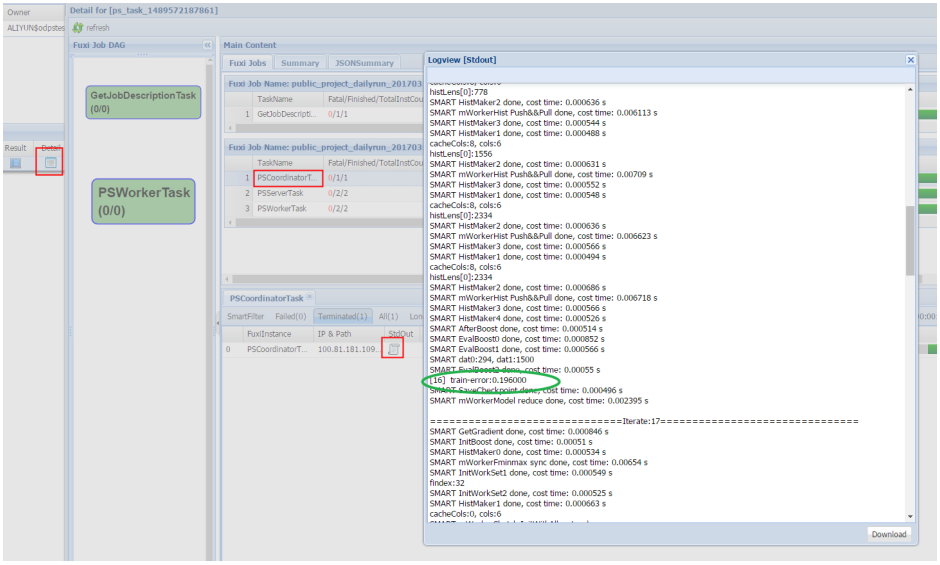
特征数量可以不填，算法会自动计算。如果您的特征数量较大，且需要算法较为精准地估计使用的资源，请在这里给出实际的值。

如果您想加速训练，可以通过执行调优页面设置核数目，核数目越多，算法运行越快。核内存一般不需要用户填写，算法可以较为精确地估计。另外，PS算法需要所有机器申请到资源才开始运行，当集群较忙时，申请较多资源可能会增加等待时间。

可以通过Logview(日志中以http://logview.odps.aliyun-inc.com:8080/logview开头的http链接)中Coordinator的stdout查看metric输出值。一个PS-Smart训练任务会有多个task，因而有多个logview，需要选取紧接着是PS开头输出的logview，如下图

红框中就是我们想要的PS任务的logview，可以通过绿色圆形中的显示来区分。

有了logview之后具体操作方式见下图



预测

使用统一预测组件预测

训练得到的输出模型以二进制方式存储，可以用来做预测。如快速上手中所示，配置预测组件的输入，也就是模型和测试数据，并正确配置参数如下图


字段设置
执行调优
特征列 默认全选
已选择 1 个字段
原样输出列 推荐添加label列,方便评估
已选择 2 个字段
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
key与value分隔符
:
kv对间的分隔符

Dense格式选择特征列（默认全选，多的列不影响预测）即可。如果是KV格式，需要选择稀疏格式并正确配置分隔符。Smart的kv对之间的分隔符是空格，需要将其置为空格或配置为\u0020（空格的转义表达形式）。

预测结果如下

序号	label	features	prediction_result	prediction_score	prediction_detail
1	2	1:0.55 2:-0....	1	0.46681538224220276	{"0": 0.1409690380096436, "1": 0.4668153822422028, "2": 0.3922155499458313}
2	1	1:-1.26 2:1....	1	0.7185402512550354	{"0": 0.1393321454524994, "1": 0.7185402512550354, "2": 0.1421276330947876}
3	1	1:-0.77 2:0....	1	0.6726031303405762	{"0": 0.1620725691318512, "1": 0.6726031303405762, "2": 0.165324330329895}
4	2	1:0.86 2:-0....	2	0.4884149134159088	{"0": 0.1755447387695312, "1": 0.3360402882099152, "2": 0.4884149134159088}
5	1	1:-0.76 2:0....	1	0.6707357168197632	{"0": 0.1629969924688339, "1": 0.6707357168197632, "2": 0.1662672907114029}
6	1	1:2.22 2:-0....	0	0.4126577377319336	{"0": 0.4126577377319336, "1": 0.2393952459096909, "2": 0.3479470014572144}
7	0	1:-1.21 2:0....	0	0.541054368019104	{"0": 0.541054368019104, "1": 0.2916863262653351, "2": 0.167259261012077 3}
8	1	1:2.17 2:-0....	1	0.5768934488296509	{"0": 0.1118654236197472, "1": 0.5768934488296509, "2": 0.311241090297699}
9	0	1:-0.40 2:0....	0	0.5392904877662659	{"0": 0.5392904877662659, "1": 0.2939955592155457, "2": 0.1667139828205109}
10	1	1:0.17 2:0....	1	0.7185402512550354	{"0": 0.1393321454524994, "1": 0.7185402512550354, "2": 0.1421276330947876}

prediction_detail列中0,1,2是对应的类别，后面是属于该类别的概率。predict_result是挑选的概率最大的类，predict_score是属于该类的概率。

使用PS-Smart预测组件预测

训练得到的输出模型表存储的是二进制内，可以兼容原PS-Smart预测组件来做预测。如快速上手中所示，配置预测组件的输入，也就是模型和测试数据，并正确配置参数。包括数据格式，特征列和目标列以及类别数，ID列只能选择string格式的除特征列和目标列以外的列。参数设置中的损失函数必须显式选择为multi:softprob。预测结果如下

序号	original_label	score_class_0	score_class_1	score_class_2	leaf_index
1	2	0.14096903800964355	0.46681538224220276	0.3922155499458313	1 1 2 1 1 2 1 1 2 1 2 2 1 2 2
2	1	0.1393321305513382	0.7185402512550354	0.1421276330947876	1 1 1 1 1 1 1 1 1 1 3 1 1 3 1
3	1	0.1620725691318512	0.6726031303405762	0.16532433032989502	1 1 1 1 1 1 1 1 1 1 3 1 1 2 1
4	2	0.17554473876953125	0.33604028820991516	0.4884149134159088	1 2 2 1 2 2 1 2 2 1 4 2 1 4 2
5	1	0.16299699246883392	0.6707357168197632	0.1662672907114029	1 1 1 1 1 1 1 1 1 1 2 1 1 3 1
6	1	0.4126577377319336	0.23939524590969086	0.34794700145721436	2 2 2 2 2 2 2 2 2 2 4 2 2 4 2
7	0	0.541054368019104	0.2916863262653351	0.16725926101207733	2 2 1 2 2 1 2 2 1 2 4 1 2 2 1
8	1	0.11186541616916656	0.5768935084342957	0.3112410604953766	1 1 2 1 1 2 1 1 2 1 3 2 1 3 2
9	0	0.5392904877662659	0.29399555921554565	0.16671398282051086	2 2 1 2 2 1 2 2 1 2 2 1 2 4 1
10	1	0.1393321305513382	0.7185402512550354	0.1421276330947876	1 1 1 1 1 1 1 1 1 1 3 1 1 3 1

其中score_class_k是预测的编号为k的类别的概率，取概率最大的类就可以作为预测的类别。leaf_index列为预测的叶子节点编号，每个样本有N*M个数，N为树的棵树，M为类别数，例子里是5*3=15个数。每棵树对应一个数字，该数字表示样本落在这棵树上的叶子节点的编号。

注：

- 此处使用的输出模型表依然是二进制格式，不可读，是为了兼容已有PS-SMART预测组件，支持输出叶子节点编号，评估指标等功能。但对数据格式有较多要求，体验不佳，会逐渐改良或用其他组件代替。
- 此处要求必须给一列string格式的列作为label，可以填充一列字符串，不能为空或者NULL，可以将一列特征通过类型转换组件转换为string格式传入。
- 参数设置中的损失函数必须显式选择为multi:softprob，默认继承不work。

查看特征重要性

可以将第三个输出桩输出到表来查看，或者直接在PS-Smart训练组件中点击右键查看数据-输出特征重要性表

，查看重要性表内容，如下图

序号	id	value
1	1	0.276059627532959
2	3	0.20854459702968597
3	4	0.31002077460289
4	5	0.20537501573562622

其中的id是传入的特征的序号，这里是KV格式，那么id就是Key-value pair中的key。如果稠密格式，假设传入的特征是”f0,f1,f2,f3,f4,f5”，因此id为0指的就是f0, id为4指的就是f4。value对应特征重要性类型，默认是gain，也就是模型中，该特征带来的信息增益的和。这里只出现了4个特征，是因为在树的分裂过程中只用到了这4个特征，可以认为没有出现的特征重要性为0。

常见问题

- PS-SMART多分类的目标列只支持正整数ID（分类数为n时，类别为0,1,2,...,n-1），即使ODPS表这里是string类型，也必须存储的是数值。如果分类目标是”Good”,”Medium”,”Bad”这样的类别字符串，请用户将分类目标自行转换成0,1,2,...,n-1。
- KV格式时特征的ID必须是正整数，特征的值必须是实数。如果特征ID为字符串，需要使用序列化组件进行序列化操作。如果特征的值是类别型的字符串，需要进行特征离散化等特征工程处理。
- 虽然PS-SMART支持数十万特征的任务，但消耗资源较大且运行速度较慢，不建议使用这么大的特征规模。GBDT类算法适合直接使用连续特征进行训练，除了类别特征需要做one-hot编码（可以筛掉低频特征）外，其他连续型数值特征不建议做离散化，可以直接用于GBDT类算法的训练。
- PS-SMART算法有很多地方会引入随机性，比如data_sample_ratio，fea_sample_ratio选项分别对数据或特征做采样。除此之外，PS-SMART算法本身使用直方图做近似，当在集群上多个worker分布式执行时，由于局部sketch归并成全局sketch的顺序会有随机性，顺序不同会导致树结构的不同，但从理论上可以保证模型效果相差不大，请您放心使用。因此同样数据同样参数多次跑，结果不一致是正常的。

PS-SMART回归

PS是参数服务器（Parameter server）的简称。PS致力于解决大规模模型的离线、在线训练任务。SMART是Scalable Multiple Additive Regression Tree的缩写，是Gradient boosting decesion tree (GBDT)在PS上的一个实现。基于PS的Smart实现可以支持百亿样本、几十万特征的训练任务，可以在上千个节点上运行，且有failover功能，稳定性好。同时，PS-Smart支持多种数据格式、训练目标和评估目标，以及输出特征重要性，并包含直方图近似等加速训练的优化。

快速上手

图中我们使用训练数据学习了一个PS-SMART回归模型。输出桩有3个，依次为

- 输出模型：offlinemodel，接统一的预测组件，目前不支持输出叶子节点编号
- 输出模型表：依然是二进制格式，不可读，是为了兼容已有PS-SMART预测组件，支持输出叶子节点编号，评估指标等功能。但对数据格式有较多要求，体验不佳，会逐渐改良或用其他组件代替。
- 输出特征重要性表：特征的重要性，有三种重要性类型可选（详见参数说明）

Pai命令行

训练

```
PAI -name ps_smart
-project algo_public
-DinputTableName="smart_regression_input"
-DmodelName="xlab_m_pai_ps_smart_bi_545859_v0"
-DoutputTableName="pai_temp_24515_545859_2"
-DoutputImportanceTableName="pai_temp_24515_545859_3"
-DlabelColName="label"
-DfeatureColNames="features"
-DenableSparse="true"
-Dobjective="reg:linear"
-Dmetric="rmse"
-DfeatureImportanceType="gain"
-DtreeCount="5";
-DmaxDepth="5"
-Dshrinkage="0.3"
-Dl2="1.0"
-Dl1="0"
-Dlifecycle="3"
-DsketchEps="0.03"
-DsampleRatio="1.0"
-DfeatureRatio="1.0"
-DbaseScore="0.5"
-DminSplitLoss="0"
```

预测

```
PAI -name prediction
-project algo_public
-DinputTableName="smart_regression_input";
-DmodelName="xlab_m_pai_ps_smart_bi_545859_v0"
-DoutputTableName="pai_temp_24515_545860_1"
-DfeatureColNames="features"
-DappendColNames="label,features"
-DenableSparse="true"
-Dlifecycle="28"
```

参数说明

数据参数

Pai命令选项	参数名称	参数描述	取值范围	是否必选，默认值
featureColNames	选择特征列	输入表中选择的用于训练的特征列名	列名，Dense格式只能选择数值类型（bigint或double），Sparse KV格式只能选择string类型，kv格式中key和value都必须是数值，不能是字符串。	必选
labelColName	选择标签列	输入表标签列列名	列名，支持string格式和数值格式的列，但内部存储的只能是数值类型，比如回归时为0,1	必选
weightCol	选择权重列	可以给每行样本单独给出权重	列名，支持数值类型	可选，默认为空
enableSparse	是否稀疏格式	是否为稀疏格式，稀疏格式kv间分隔符为空格，key与value分隔符为冒号，比如1:0.3 3:0.9	[true, false]	可选，默认false
inputTableName	输入表名	-	-	必选
modelName	输出模型名	-	-	必选
outputImportanceTableName	输出特征重要性表名	-	-	可选，默认为空
inputTablePartitions	输入表分区	-	-	可选，格式如ds=1/pt=1

outputTableName	输出模型表名	输出到Odps表，也是二进制格式，不可读，可以使用Smart自带的预测组件，支持输出叶子节点编号	[true, false]	可选，默认false
lifecycle	输出表生命周期	-	正整数	可选，默认3

算法参数

Pai命令选项	参数名称	参数描述	取值范围	是否必选，默认值
objective	目标函数类型	目标函数类型直接决定了学习问题，需要正确选择。回归时有多种损失函数供选择，详见注意事项		必选，默认为Linear regression
metric	评估指标类型	训练集上的评估指标类型，需要与目标函数类型对应，输出在logview里coordinator的stdout中，详见注意事项及示例		可选，默认为空
treeCount	树数量	树的棵树，训练时间与树的棵树呈正比	正整数	可选，默认为1
maxDepth	树最大深度	一棵树的最大深度，建议取5（即最多32个叶子节点）	正整数，[1,20]	可选，默认为5
sampleRatio	数据采样比例	构建每棵树时只采样一部分数据来学习，构建弱学习器，加快训练	(0,1]	可选，默认为1.0，不采样
featureRatio	特征采样比例	构建每棵树时只采样一部分特征来学习，构建弱学习器，加快训练	(0,1]	可选，默认为1.0，不采样
l1	L1惩罚项系数	控制叶子节点个数，该项越大，叶子节点数越少。过拟合时可以加大该项。	非负实数	可选，默认为0

l2	L2惩罚项系数	控制叶子节点大小，该项越大，叶子节点规模分布越均匀。过拟合时可以加大该项。	非负实数	可选，默认为1.0
shrinkage	学习速率		(0,1]	可选，默认为0.3
sketchEps	近似Sketch精度	构造sketch时切割分位点的阈值，桶数为O(1.0/sketchEps)。这个值越小，切出来桶越多，一般不需要调整。	(0,1)	可选，默认为0.03
minSplitLoss	最小分裂损失变化	分裂节点所需要的最小损失变化，该值越大，分裂越保守	非负实数	可选，默认值0
featureNum	特征数量	特征的个数，或最大特征ID。当需要估计使用资源时需要填写	正整数	可选
baseScore	全局偏置项	所以样本的初始预测值	实数	可选，默认值0.5
featureImportanceType	特征重要性类型	计算特征重要性的类型。 weight：模型中，该特征做为分裂特征的次数，gain：模型中，该特征带来的信息增益，cover：模型中，该特征在分裂节点覆盖的样本数。	可选的有”weight”，”gain”，”cover”	可选，默认”gain”
tweedieVarPower	Tweedie分布指数	Tweedie分布的方差和均值关系的指数$Var(y) ~ E(y)^tweedie_variance_power$	(1,2)	可选，默认1.5

注意事项

- objective在不同的学习问题中需要指定不同的值，回归的web界面提供多种目标函数供选择，具体介绍如下

```
reg:linear（Linear regression）注：label范围(-∞, +∞)
reg:logistic（Logistic regression）注：label范围[0,1]
```

count:poisson (Poisson regression for count data, output mean of poisson distribution) 注：label范围大于等于0
reg:gamma (Gamma regression for for modeling insurance claims severity, or for any outcome that might be [gamma-distributed](https://en.wikipedia.org/wiki/Gamma_distribution#Applications)) 注：label范围大于等于0
reg:tweedie (Tweedie regression for modeling total loss in insurance, or for any outcome that might be [Tweedie-distributed](https://en.wikipedia.org/wiki/Tweedie_distribution#Applications).) 注：label范围大于等于0

- 对应的metric为

rmse(rooted mean square error，对应objective reg:linear)
mae(mean absolute error，对应objective reg:linear)
poisson-nloglik(negative loglikelihood for poisson regression，对应objective count:poisson)
gamma-deviance(Residual deviance for gamma regression，对应objective reg:gamma)
gamma-nloglik(Negative log-likelihood for gamma regression，对应objective reg:gamma)
tweedie-nloglik(tweedie-nloglik@1.5，negative log-likelihood for Tweedie regression, at a specified value of the tweedie_variance_power parameter)

执行调优

Pai命令选项	Pai web参数名称	参数描述	取值范围	是否必选，默认值
coreNum	核心数	核心个数，越多算法运行越快	正整数	可选，默认自动分配
memSizePerCore	每个核的内存大小（MB）	单个核心使用的内存数，1024代表1GB内存	正整数	可选，默认自动分配

具体示例

数据生成

以Sparse KV格式数据为例

```
drop table if exists smart_regression_input;
create table smart_regression_input as
select
*
from
(
select 2.0 as label, '1:0.55 2:-0.15 3:0.82 4:-0.99 5:0.17' as features from dual
union all
select 1.0 as label, '1:-1.26 2:1.36 3:-0.13 4:-2.82 5:-0.41' as features from dual
union all
select 1.0 as label, '1:-0.77 2:0.91 3:-0.23 4:-4.46 5:0.91' as features from dual
union all
select 2.0 as label, '1:0.86 2:-0.22 3:-0.46 4:0.08 5:-0.60' as features from dual
union all
select 1.0 as label, '1:-0.76 2:0.89 3:1.02 4:-0.78 5:-0.86' as features from dual
```

```
union all
select 1.0 as label, '1:2.22 2:-0.46 3:0.49 4:0.31 5:-1.84' as features from dual
union all
select 0.0 as label, '1:-1.21 2:0.09 3:0.23 4:2.04 5:0.30' as features from dual
union all
select 1.0 as label, '1:2.17 2:-0.45 3:-1.22 4:-0.48 5:-1.41' as features from dual
union all
select 0.0 as label, '1:-0.40 2:0.63 3:0.56 4:0.74 5:-1.44' as features from dual
union all
select 1.0 as label, '1:0.17 2:0.49 3:-1.50 4:-2.20 5:-0.35' as features from dual
) tmp;
```

数据内容如下

序号 ▲	label ▲	features ▲
1	2.0	1:0.55 2:-0.15 3:0.82 4:-0.99 5:0.17
2	1.0	1:-1.26 2:1.36 3:-0.13 4:-2.82 5:-0.41
3	1.0	1:-0.77 2:0.91 3:-0.23 4:-4.46 5:0.91
4	2.0	1:0.86 2:-0.22 3:-0.46 4:0.08 5:-0.60
5	1.0	1:-0.76 2:0.89 3:1.02 4:-0.78 5:-0.86
6	1.0	1:2.22 2:-0.46 3:0.49 4:0.31 5:-1.84
7	0.0	1:-1.21 2:0.09 3:0.23 4:2.04 5:0.30
8	1.0	1:2.17 2:-0.45 3:-1.22 4:-0.48 5:-1.41
9	0.0	1:-0.40 2:0.63 3:0.56 4:0.74 5:-1.44
10	1.0	1:0.17 2:0.49 3:-1.50 4:-2.20 5:-0.35

数据中特征ID从1开始编号，最大特征ID为5。

训练

如快速上手中所示，配置训练数据和训练组件。选择label为目标列，features为特征列。以Linear regression为例，算法参数设置页面如下图

目标函数类型 必选 ⓘ

Linear regression

评估指标类型 可选 ⓘ

rooted mean square error

树数量 可选 正整数 ⓘ

5

树最大深度 可选 正整数 ⓘ

5

数据采样比例 可选 (0,1] ⓘ

1.0

特征采样比例 可选 (0,1] ⓘ

1.0

L1惩罚项系数 可选 非负实数 ⓘ

0

L2惩罚项系数 可选 非负实数 ⓘ

1.0

学习速率 可选 (0,1]

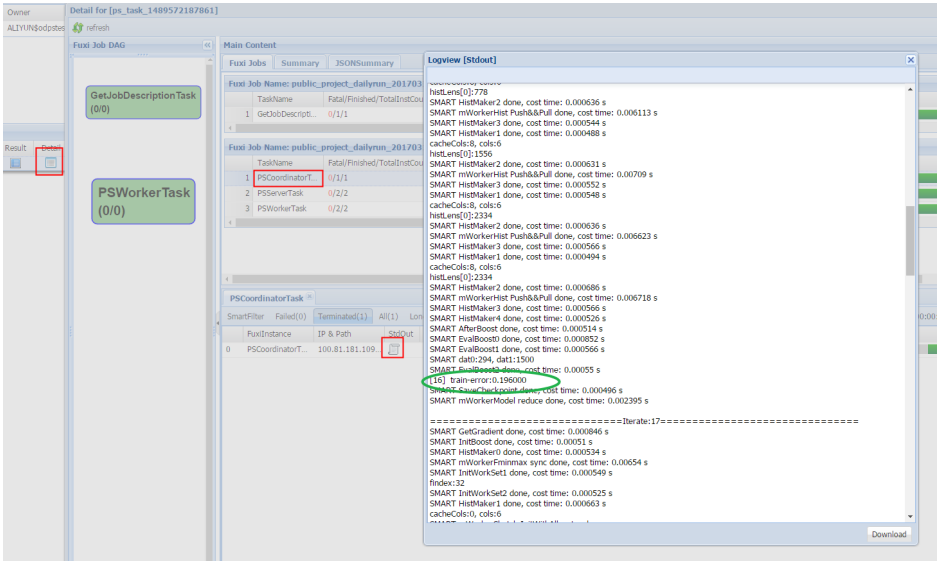
特征数量可以不填，算法会自动计算。如果您的特征数量较大，且需要算法较为精准地估计使用的资源，请在这里给出实际的值。

如果您想加速训练，可以通过执行调优页面设置核数目，核数目越多，算法运行越快。核内存一般不需要用户填写，算法可以较为精确地估计。另外，PS算法需要所有机器申请到资源才开始运行，当集群较忙时，申请较多资源可能会增加等待时间。

可以通过Logview(日志中以http://logview.odps.aliyun-inc.com:8080/logview开头的http链接)中Coordinator的stdout查看metric输出值。一个PS-Smart训练任务会有多个task，因而有多个logview，需要选取紧接着是PS开头输出的logview，如下图

红框中就是我们想要的PS任务的logview，可以通过绿色圆形中的显示来区分。

有了logview之后具体操作方式见下图



预测

使用统一预测组件预测

训练得到的输出模型以二进制方式存储，可以用来做预测。如快速上手中所示，配置预测组件的输入，也就是模型和测试数据，并正确配置参数如下图


字段设置
执行调优
特征列 默认全选
已选择 1 个字段
原样输出列 推荐添加label列,方便评估
已选择 2 个字段
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
key与value分隔符
:
kv对间的分隔符

Dense格式选择特征列（默认全选，多的列不影响预测）即可。如果是KV格式，需要选择稀疏格式并正确配置分隔符。Smart的kv对之间的分隔符是空格，需要将其置为空格或配置为\u0020（空格的转义表达形式）。

预测结果如下

序号	label	features	prediction_result	prediction_score	prediction_detail
1	2.0	1:0.55 2:-0....	1.467519998550415	1.467519998550415	{"label": 1.467519998550415}
2	1.0	1:-1.26 2:1....	0.8999134302139282	0.8999134302139282	{"label": 0.8999134302139282}
3	1.0	1:-0.77 2:0....	0.8999134302139282	0.8999134302139282	{"label": 0.8999134302139282}
4	2.0	1:0.86 2:-0....	1.467519998550415	1.467519998550415	{"label": 1.467519998550415}
5	1.0	1:-0.76 2:0....	0.8999134302139282	0.8999134302139282	{"label": 0.8999134302139282}
6	1.0	1:2.22 2:-0....	0.8771200180053711	0.8771200180053711	{"label": 0.8771200180053711}
7	0.0	1:-1.21 2:0....	0.16383999586105347	0.16383999586105347	{"label": 0.1638399958610535}
8	1.0	1:2.17 2:-0....	0.8771200180053711	0.8771200180053711	{"label": 0.8771200180053711}
9	0.0	1:-0.40 2:0....	0.16383999586105347	0.16383999586105347	{"label": 0.1638399958610535}
10	1.0	1:0.17 2:0....	0.8999134302139282	0.8999134302139282	{"label": 0.8999134302139282}

predict_result中就是预测值。

使用PS-Smart预测组件预测

训练得到的输出模型表存储的是二进制内，可以兼容原PS-Smart预测组件来做预测。如快速上手中所示，配置预测组件的输入，也就是模型和测试数据，并正确配置参数。包括数据格式，特征列和目标列以及类别数，ID列只能选择string格式的除特征列和目标列以外的列。参数设置中的损失函数必须显式选择与训练时objective一致的损失函数。预测结果如下

序号	original_label	prediction_score	leaf_index
1	2	1.467519998550415	1 5 5 5 5
2	1	0.8999134302139282	1 3 3 3 3
3	1	0.8999134302139282	1 3 3 3 3
4	2	1.467519998550415	1 5 5 5 5
5	1	0.8999134302139282	1 3 3 3 3
6	1	0.8771200180053711	1 6 6 6 6
7	0	0.16383999586105347	2 2 2 2 2
8	1	0.8771200180053711	1 6 6 6 6
9	0	0.16383999586105347	2 2 2 2 2
10	1	0.8999134302139282	1 3 3 3 3

其中prediction_score是预测值。leaf_index列为预测的叶子节点编号，每个样本有N个数，N为树的棵树。每棵树对应一个数字，该数字表示样本落在这棵树上的叶子节点的编号。

注：

- 此处使用的输出模型表依然是二进制格式，不可读，是为了兼容已有PS-SMART预测组件，支持输出

叶子节点编号，评估指标等功能。但对数据格式有较多要求，体验不佳，会逐渐改良或用其他组件代替。
- 此处要求必须给一列string格式的列作为label，可以填充一列字符串，不能为空或者NULL，可以将一列特征通过类型转换组件转换为string格式传入。
- 参数设置中的损失函数必须显式选择与训练时objective一致的损失函数，默认继承不work。

查看特征重要性

可以将第三个输出桩输出到表来查看，或者直接在PS-Smart训练组件中点击右键查看数据-输出特征重要性表，查看重要性表内容，如下图

序号	id	value
1	1	0.1405973434448242
2	4	0.8594027161598206

其中的id是传入的特征的序号，这里是KV格式，那么id就是Key-value pair中的key。如果稠密格式，假设传入的特征是”f0,f1,f2,f3,f4,f5”，因此id为0指的就是f0, id为4指的就是f4。value对应特征重要性类型，默认是gain，也就是模型中，该特征带来的信息增益的和。这里只出现了2个特征，是因为在树的分裂过程中只用到了这2个特征，可以认为没有出现的特征重要性为0。

常见问题

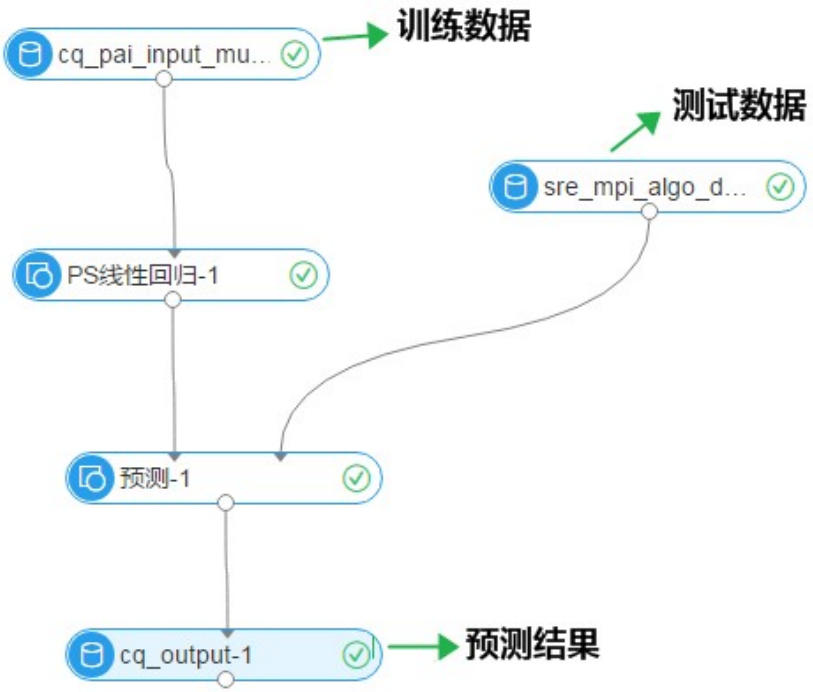
- PS-SMART回归的目标列只支持数值类型，即使ODPS表这里是string类型，也必须存储的是数值。
- KV格式时特征的ID必须是正整数，特征的值必须是实数。如果特征ID为字符串，需要使用序列化组件进行序列化操作。如果特征的值是类别型的字符串，需要进行特征离散化等特征工程处理。
- 虽然PS-SMART支持数十万特征的任务，但消耗资源较大且运行速度较慢，不建议使用这么大的特征规模。GBDT类算法适合直接使用连续特征进行训练，除了类别特征需要做one-hot编码（可以筛掉低频特征）外，其他连续型数值特征不建议做离散化，可以直接用于GBDT类算法的训练。
- PS-SMART算法有很多地方会引入随机性，比如data_sample_ratio，fea_sample_ratio选项分别对数据或特征做采样。除此之外，PS-SMART算法本身使用直方图做近似，当在集群上多个worker分布式执行时，由于局部sketch归并成全局sketch的顺序会有随机性，顺序不同会导致树结构的不同，但从理论上可以保证模型效果相差不大，请您放心使用。因此同样数据同样参数多次跑，结果不一致是正常的。

PS线性回归

线性回归（Linear regression）是经典的回归算法，分析因变量和多个自变量之间线性关系。PS是参数服务器（Parameter server）的简称。PS致力于解决大规模模型的离线、在线训练任务，能够使用 千亿行数样本 训练 百亿特征 模型并高效产出。PS线性回归支持千亿样本，十亿特征的训练任务，且支持L1，L2正则项。如果

您有更大规模的需求，请与算法作者联系。

快速上手



PAI命令行

训练

```
PAI -name ps_linearregression
-project algo_public
-DinputTableName="lm_test_input"
-DmodelName="linear_regression_model"
-DlabelColName="label"
-DfeatureColNames="features"
-Dl1Weight=1.0
-Dl2Weight=0.0
-DmaxIter=100
-Depsilon=1e-6
-DenableSparse=true
```

预测

```
drop table if exists logistic_regression_predict;
```

```
PAI -name prediction
-DmodelName="linear_regression_model"
-DoutputTableName="linear_regression_predict"
-DinputTableName="lm_test_input"
-DappendColNames="label,features"
-DfeatureColNames="features"
-DenableSparse=true
```

参数说明

数据参数

Pai命令选项	Pai web参数名称	参数描述	取值范围	是否必选，默认值
featureColNames	选择特征列	输入表中选择的用于训练的特征列名	列名，Dense格式只能选择数值类型（bigint或double），Sparse KV格式只能选择string类型	必选
labelColName	选择标签列	输入表标签列列名	列名，必须为数值类型（bigint或double）	必选
enableSparse	是否稀疏格式	当选择Sparse KV格式时，请不要使用0号特征id，建议您从1开始给特征编号。	[true, false]	可选，默认false
itemDelimiter	kv间的分隔符	当输入表数据为稀疏格式时，kv间的分隔符，默认为空格	-	可选，默认空格，' '
kvDelimiter	key与value的分隔符	当输入表数据为稀疏格式时，key与value的分隔符	-	可选，默认冒号，':'
inputTableName	输入表名	-	-	必选
modelName	输出模型名	-	-	必选
inputTablePartitions	输入表分区	-	-	可选，格式如ds=1/pt=1
enableModelIo	是否输出到OfflineModel	置为false时，输出到Odps表，可以查看模型权重	[true, false]	可选，默认true

算法参数

Pai命令选项	Pai web参数名称	参数描述	取值范围	是否必选，默认值
l1Weight	L1 weight	L1正则化系数。该项越大，模型非零元越少。过拟合时可以加大该项。	非负实数	可选，默认为1.0
l2Weight	L2 weight	L2正则化系数。该项越大，模型参数绝对值越小。过拟合时可以加大该项。	非负实数	可选，默认为 0
maxIter	最大迭代次数	最大迭代数，指定L-BFGS/OWL-QN的最大迭代次数，0代表不限制迭代次数	非负整数	可选，默认值100
epsilon	最小收敛误差	收敛误差，优化算法终止条件，为10次迭代Loss相对变化率均值，越小要求越严格，算法执行时间越长	0到1之间实数	可选，默认值1.0e-06
modelSize	最大特征ID	特征中最大的特征ID，特征维度。可以比实际值大，数值越大，内存占用越大。不填写时会启动SQL任务自动计算。	非负整数	可选，默认值0

最大迭代次数和最小收敛误差都对算法终止起作用，同时配置时，无论哪个终止条件触发，算法都会终止。

执行调优

Pai命令选项	Pai web参数名称	参数描述	取值范围	是否必选，默认值
coreNum	核心数	核心个数，越多算法运行越快	正整数	可选，默认自动分配
memSizePerCore	每个核的内存大小（MB）	单个核心使用的内存数，1024代表1GB内存	正整数	可选，默认自动分配，一般不需要配置，算法会精确估算出来

具体示例

数据生成

以Sparse KV格式数据为例

```
drop table if exists lm_test_input;
create table lm_test_input as
select
*
from
(
select 2 as label, '1:0.55 2:-0.15 3:0.82 4:-0.99 5:0.17' as features from dual
union all
select 1 as label, '1:-1.26 2:1.36 3:-0.13 4:-2.82 5:-0.41' as features from dual
union all
select 1 as label, '1:-0.77 2:0.91 3:-0.23 4:-4.46 5:0.91' as features from dual
union all
select 2 as label, '1:0.86 2:-0.22 3:-0.46 4:0.08 5:-0.60' as features from dual
union all
select 1 as label, '1:-0.76 2:0.89 3:1.02 4:-0.78 5:-0.86' as features from dual
union all
select 1 as label, '1:2.22 2:-0.46 3:0.49 4:0.31 5:-1.84' as features from dual
union all
select 0 as label, '1:-1.21 2:0.09 3:0.23 4:2.04 5:0.30' as features from dual
union all
select 1 as label, '1:2.17 2:-0.45 3:-1.22 4:-0.48 5:-1.41' as features from dual
union all
select 0 as label, '1:-0.40 2:0.63 3:0.56 4:0.74 5:-1.44' as features from dual
union all
select 1 as label, '1:0.17 2:0.49 3:-1.50 4:-2.20 5:-0.35' as features from dual
) tmp;
```

数据内容如下

序号	label	features
1	2	1:0.55 2:-0.15 3:0.82 4:-0.99 5:0.17
2	1	1:-1.26 2:1.36 3:-0.13 4:-2.82 5:-0.41
3	1	1:-0.77 2:0.91 3:-0.23 4:-4.46 5:0.91
4	2	1:0.86 2:-0.22 3:-0.46 4:0.08 5:-0.60
5	1	1:-0.76 2:0.89 3:1.02 4:-0.78 5:-0.86
6	1	1:2.22 2:-0.46 3:0.49 4:0.31 5:-1.84
7	0	1:-1.21 2:0.09 3:0.23 4:2.04 5:0.30
8	1	1:2.17 2:-0.45 3:-1.22 4:-0.48 5:-1.41
9	0	1:-0.40 2:0.63 3:0.56 4:0.74 5:-1.44
10	1	1:0.17 2:0.49 3:-1.50 4:-2.20 5:-0.35

数据中特征ID从1开始编号，最大特征ID为5。

训练

如**快速上手**中所示，配置训练数据和训练组件。选择label为目标列，features为特征列，注意选择数据格式为稀疏格式。算法参数设置页面如下图

最大特征ID可以不填，默认0，算法会启动sql来自动计算。如果想避免启动sql任务来计算，这里可以填大于等于5的数，Dense格式时也就是特征列数，kv格式时为最大的特征ID编号。

如果您想加速训练，可以通过执行调优页面设置核数目，核数目越多，算法运行越快。核内存一般不需要用户填写，算法可以较为精确地估计。另外，PS算法需要所有机器申请到资源才开始运行，当集群较忙时，申请较多资源可能会增加等待时间。

预测

训练得到的模型以二进制方式存储，可以用来做预测。如快速上手中所示，配置预测组件的输入，也就是模型和测试数据，并正确配置参数如下图

特征选择 执行调优

特征列 默认全选

已选择 1 个字段

原样输出列 推荐添加label列,方便评估

已选择 2 个字段

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

key与value分隔符

:

kv对间的分隔符

\u0020

数据格式需要与训练时保持一致，仍然选择KV格式，并正确配置分隔符。kv对之间的分隔符为空格，需要置为空格或配置为\u0020（空格的转义表达形式）。

预测结果如下

序号	label	features	prediction_result	prediction_score	prediction_detail
1	2	1:0.55 2:-0.15 3:0.82 4:-0.99 5:0.17	0.9950045235424285	0.9950045235424285	{"1": 0.9950045235424285}
2	1	1:-1.26 2:1.36 3:-0.13 4:-2.82 5:-0.41	0.9022041049173464	0.9022041049173464	{"1": 0.9022041049173464}
3	1	1:-0.77 2:0.91 3:-0.23 4:-4.46 5:0.91	1.228147671448144	1.228147671448144	{"1": 1.228147671448144}
4	2	1:0.86 2:-0.22 3:-0.46 4:0.08 5:-0.60	0.9043180917054381	0.9043180917054381	{"1": 0.9043180917054381}
5	1	1:-0.76 2:0.89 3:1.02 4:-0.78 5:-0.86	0.7116744886195954	0.7116744886195954	{"1": 0.7116744886195954}
6	1	1:2.22 2:-0.46 3:0.49 4:0.31 5:-1.84	1.135349294653747	1.135349294653747	{"1": 1.135349294653747}
7	0	1:-1.21 2:0.09 3:0.23 4:2.04 5:0.30	0.22724956563861654	0.22724956563861654	{"1": 0.2272495656386165}
8	1	1:2.17 2:-0.45 3:-1.22 4:-0.48 5:-1.41	1.2369534428175184	1.2369534428175184	{"1": 1.236953442817518}
9	0	1:-0.40 2:0.63 3:0.56 4:0.74 5:-1.44	0.5672805014883875	0.5672805014883875	{"1": 0.5672805014883875}
10	1	1:0.17 2:0.49 3:-1.50 4:-2.20 5:-0.35	1.0918540901263776	1.0918540901263776	{"1": 1.091854090126378}

只需要查看predict_result一列内容即可。

常见问题

- KV格式时特征的ID必须是正整数（不包含0），特征的值必须是实数。如果特征ID为字符串，需要使用序列化组件进行序列化操作。如果特征的值是类别型的字符串，需要进行特征离散化等特征工程处理。

聚类模型评估

基于原始数据和聚类模型，评价聚类模型的优劣，包含指标和图标。

pai命令行

```
PAI
-name cluster_evaluation
-project algo_public
-DinputTableName=pai_cluster_evaluation_test_input
-DselectedColNames=f0,f3
-DmodelName=pai_kmeans_test_model
-DoutputTableName=pai_ft_cluster_evaluation_out;
```

参数说明

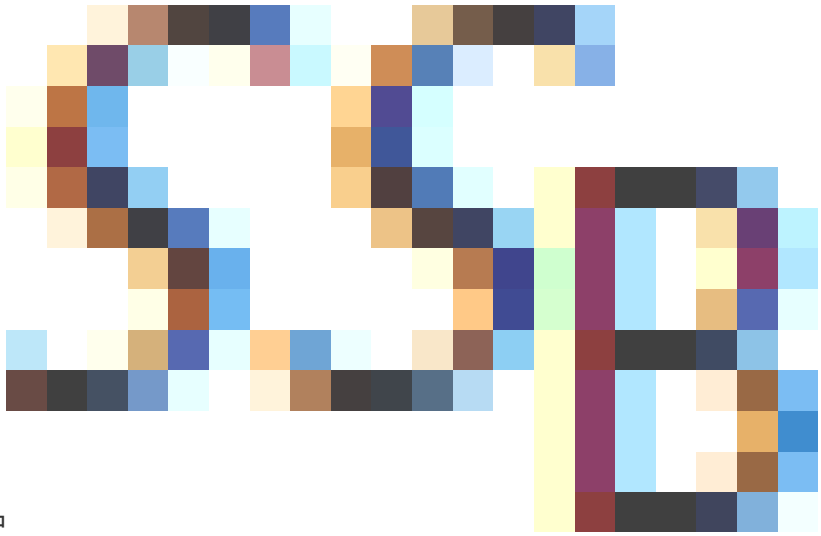
参数名称	参数描述	取值范围	是否必选，默认值/行为
inputTableName	输入表	表名	必选
selectedColNames	输入表中用于评估的列名，以逗号分隔，必须与模型存储的特征名一致	列名	可选，默认值选择所有列
inputTablePartitions	输入表中指定哪些分区		可选， 默认值选择所

	参与评估，格式为name1=value1/name2=value2; 如果是指定多个分区， 中间用’,’分开		有分区
enableSparse	输入表数据是否为稀疏格式	true， false	可选， 默认值false
itemDelimiter	当输入表数据为稀疏格式时，kv间的分割符		可选， 默认值为空格
kvDelimiter	当输入表数据为稀疏格式时，key和value的分割符		可选， 默认值冒号
modelName	输入聚类模型	模型名	必选
outputTableName	输出表	表名	必选
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期

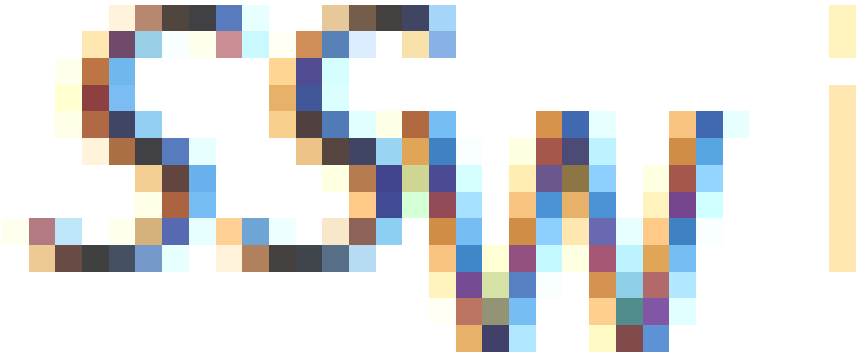
评估公式

- Calinski-Harabasz指标又称VRC (variance ratio criterion) 定义如下：

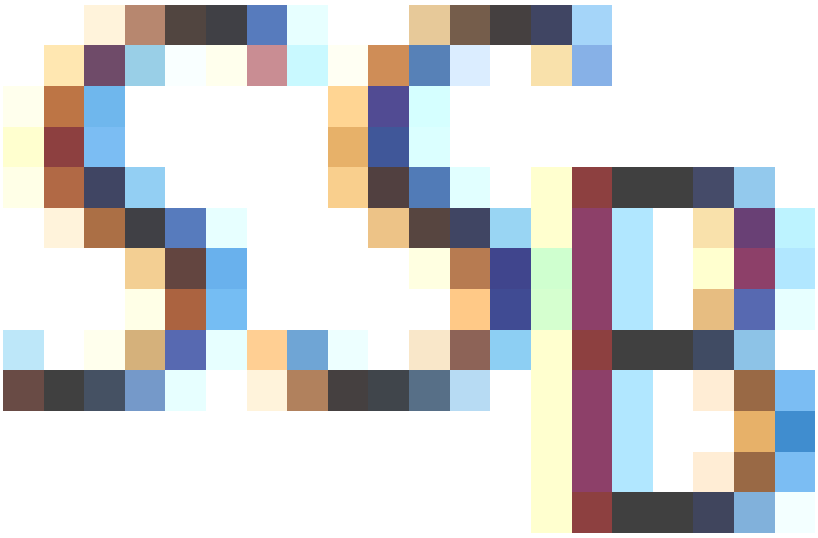
$$VRC_k = \frac{SS_B}{SS_W} \times \frac{(N-k)}{(k-1)},$$



其中 是整个聚类间的方差；



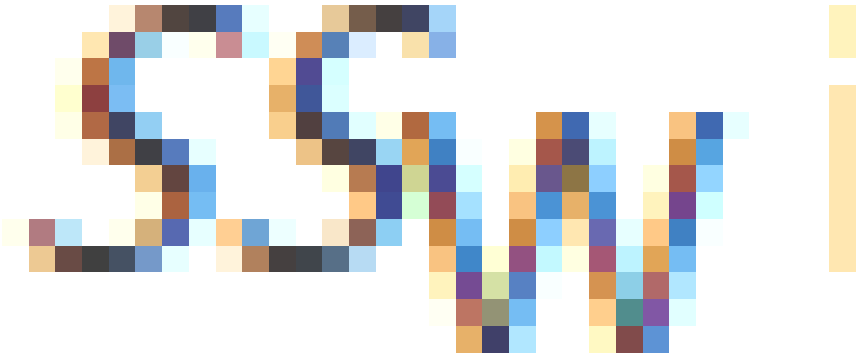
是整个聚类内的方差；N是记录总数，k聚类中心点个数



定义如下：

$$SS_B = \sum_{i=1}^{k} n_i \|m_i - m\|^2,$$

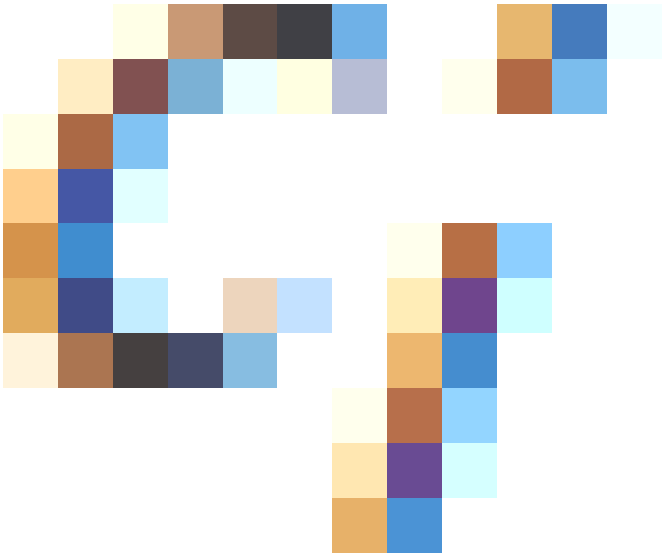
k聚类中心点个数，聚类i的中心点, m是输入数据的均值



定义如下：

$$SS_W = \sum_{i=1}^{k} \sum_{x \in c_i} \|x - m_i\|^2,$$

k聚类中心点个数, x是数据点，



第i个聚类，

聚类i的中心点

具体示例

- 测试数据

```
create table if not exists pai_cluster_evaluation_test_input  as
select * from
(
select 1 as id, 1 as f0,2 as f3 from dual
union all
select 2 as id, 1 as f0,3 as f3 from dual
union all
select 3 as id, 1 as f0,4 as f3 from dual
union all
select 4 as id, 0 as f0,3 as f3 from dual
union all
select 5 as id, 0 as f0,4 as f3 from dual
)tmp;
```

- 构造聚类模型

```
pai -name kmeans
-project algo_public
-DinputTableName=pai_cluster_evaluation_test_input
-DselectedColNames=f0,f3
-DcenterCount=3
-Dloop=10
-Daccuracy=0.00001
-DdistanceType=euclidean
-DinitCenterMethod=random
-Dseed=1
-DmodelName=pai_kmeans_test_model
-DidxTableName=pai_kmeans_test_idx
```

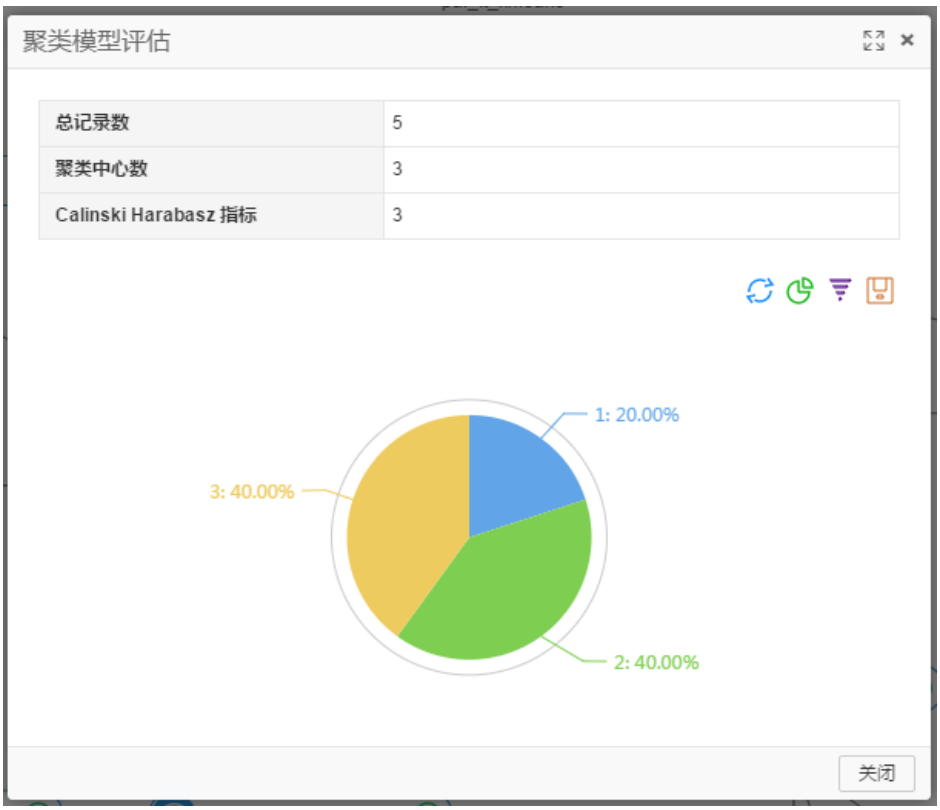
- pai命令示例

```
PAI
-name cluster_evaluation
-project algo_public
-DinputTableName=pai_cluster_evaluation_test_input
-DselectedColNames=f0,f3
-DmodelName=pai_kmeans_test_model
-DoutputTableName=pai_ft_cluster_evaluation_out;
```

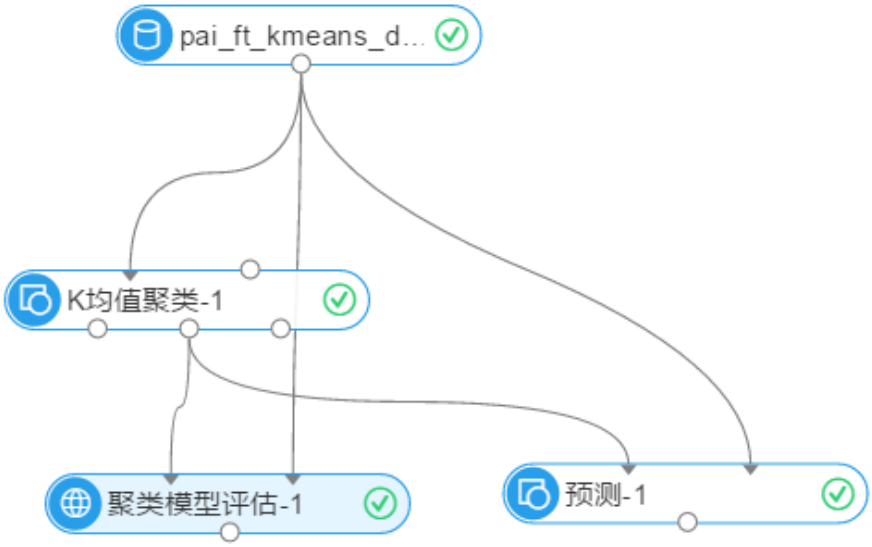
- 输出说明

输出表outputTableName，字段分别为：| column name | comment || ———— | ———— || count | 总记录数 || centerCount| 聚类中心数 || calinhara| Calinski Harabasz指标 [计算公式] (#评估公式) || clusterCounts| 各个聚类包含的点的数目|

PaiWeb展示



PaiWeb-Pipeline



深度学习

目录

深度学习框架说明

阿里云机器学习平台上支持深度学习框架，同时后端提供了功能强大的GPU（型号M40）计算集群，用户可以使用这些框架及硬件资源来运行深度学习算法。 目前支持的框架包括 TensorFlow（支持1.0、1.1、1.2版本），MXNet 0.9.5， Caffe rc3。TensorFlow 和MXNet 支持用户自己编写的Python 代码， Caffe 支持用户自定义网络文件。

在使用深度学习框架训练数据之前，需要将训练的数据上传至阿里云对象存储OSS中，算法在运行时从指定的OSS目录中读取数据。需要注意的是阿里云机器学习目前只在华东2部署了GPU 集群，算法在执行时访问华东2 OSS中数据时不产生流量费用，访问其它地域的OSS会产生流量费用。

深度学习开通

目前机器学习平台深度学习相关功能处于公测阶段，深度学习组件包含TensorFlow、Caffe、MXNet三个框架，开通方式如下，进入机器学习控制台，在相应项目下勾选GPU资源即可使用。

开通GPU资源的项目会被分配到公共的资源池，可以动态的调用底层的GPU计算资源。

OSS上传数据说明

使用深度学习处理数据时，数据先存储到OSS的bucket中。第一步要创建OSS Bucket。 由于深度学习的GPU集群在华东2，建议您创建 OSS Bucket 时选择华东2地区。这样在数据传输时就可以使用阿里云经典网络，算法运行时不需要收取流量费用。Bucket 创建好之后，可以在OSS管理控制台 来创建文件夹，组织数据目录，上传数据了。

OSS支持多种方式上传数据， API或SDK详细见：
https://help.aliyun.com/document_detail/31848.html?spm=5176.doc31848.6.580.a6es2a

OSS还提供了大量的常用工具用来帮助用户更加高效的使用OSS。工具列表请参见：
https://help.aliyun.com/document_detail/44075.html?spm=5176.doc32184.6.1012.XlMMUx

建议您使用 ossutil 或 osscmd ，这是两个命令行工具，通过命令的方式来上传、下载文件，还支持断点续传。

注：在使用工具时需要配置 AccessKey 和ID，登录后，可以在Access Key 管理控制台创建或查看。

读OSSBucket

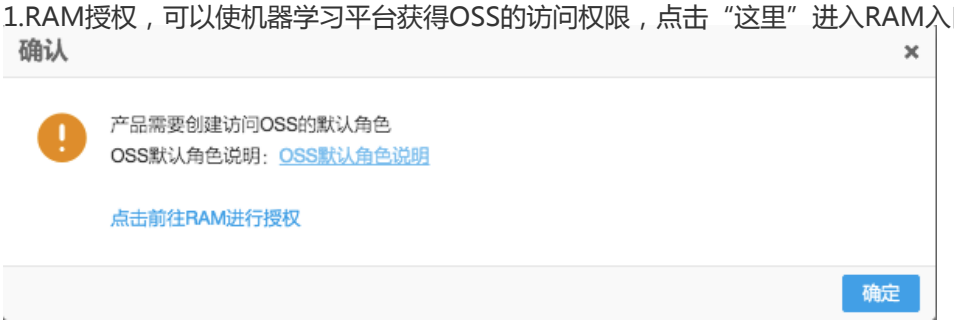
用户在机器学习平台中使用“读OSS Bucket”组件时，需要授予一个名称为“AliyunODPSPAIDefaultRole”的系统默认角色给数加的服务账号，当且仅当该角色被正确授权后，机器学习平台的算法才能正确地读、写OSS bucket。

- 注：由于机器学习平台运行在MaxCompute框架之上，与MaxCompute共用服务账号。在授权时，默认的角色授予给MaxCompute服务账号。

需要在“设置”菜单完成对OSS读写权限的授权，详情见RAM授权。

RAM授权

1.RAM授权，可以使机器学习平台获得OSS的访问权限，点击“这里”进入RAM入口，如图。



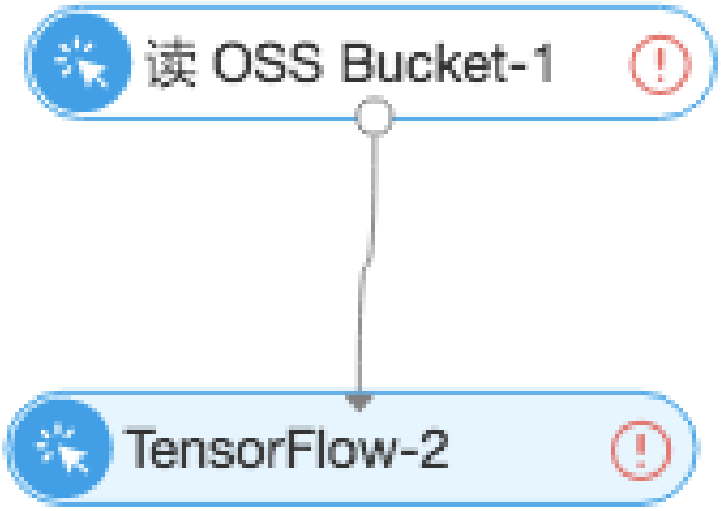
2.点击“前往RAM进行授权”，进入如下界面，点击同意即可。

注：如果您想查看AliyunODPSPAIDefaultRole的相关详细策略信息，可以登录RAM控制台来查看。 默认角色AliyunODPSPAIDefaultRole包含的权限信息如下：

权限名称（Action）	权限说明
oss:PutObject	上传文件或文件夹对象
oss:GetObject	获取文件或文件夹对象
oss:ListObjects	查询文件列表信息
oss:DeleteObjects	删除对象

3.回到阿里云机器学习界面，点击刷新，RAM信息会自动录入组件中。如图

4.在使用深度学习框架过程中，需要组件“读OSSBucket”与相应的深度学习组件相连，用来获得OSS的读写权限。



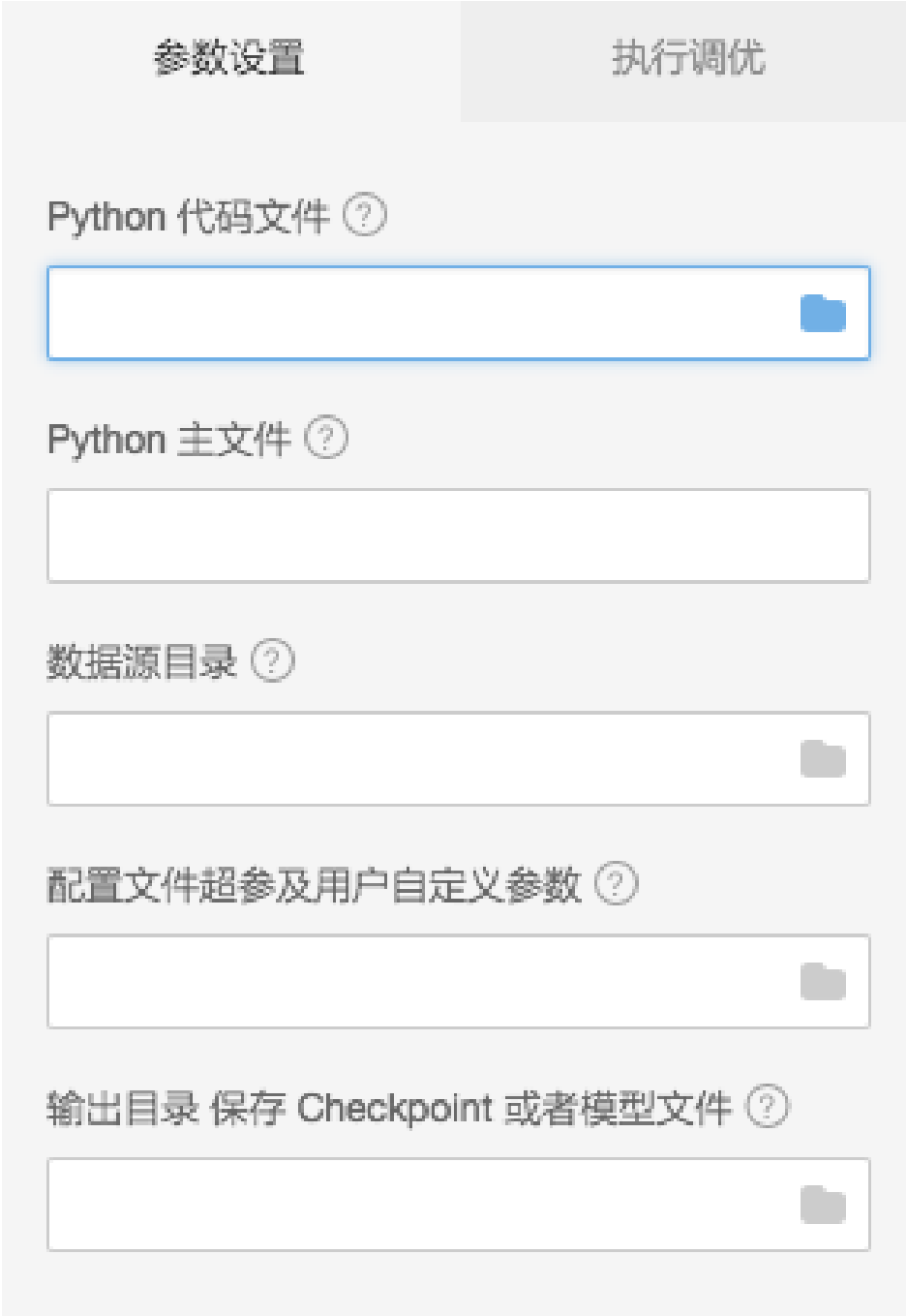
TensorFlow

背景

TensorFlow（以下简称TF）是Google开源的一套机器学习框架，算法开发者通过简单的学习就能快速上手。阿里云机器学习平台将TF框架集成到产品中。用户可以自由的利用TF进行代码编写，TF的计算引擎为GPU集群，用户可以灵活的对计算资源进行调整。

参数说明

(1)参数设置



- Python代码文件：程序执行文件，多个文件可通过tar.gz打包上传。
- Python主文件：指定代码文件压缩包中的主文件，可选。
- 数据源目录：选择OSS上的数据源。
- 配置文件超参及用户自定义参数：PAI Tensorflow支持用户通过Command传入相应的超参配置，这样用户可以在做模型试验的时候可以尝试不同的learning rate, batch size等。
- 输出目录：输出的模型路径。

（2）执行调优

用户可以根据自身任务的复杂程度指定GPU卡数

PAI命令

实际使用中，并不需要指定所有参数(不要直接复制下面的命令....)，各个参数的含义可以参考后面的表格

```
PAI -name tensorflow_ext -Dbuckets="oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_tensorflow/mnist/" -DgpuRequired="100" -Darn="acs:ram::166408185518****:role/aliyunodpspaidefaultrole" -Dscript="oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_tensorflow/mnist_ext.py";
```

各个参数的具体含义如下表：

参数名称	参数描述	参数值格式	默认值
script	必选，TF算法文件，可以是单个文件或者tar.gz压缩包	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_tensorflow/mnist_ext.py	-
entryFile	可选，算法入口文件名，当script为tar.gz压缩包时，该参数必填	train.py	空
buckets	必选，输入OSS bucket，可指定多个，以逗号分割, 每个bucket须以”/“结尾	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_tensorflow/mnist/	空
arn	必选， OSS role_arn		空
gpuRequired	必选，标识使用GPU资源量	200	100
checkpointDir	可选，TF checkpoint目录	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_tensorflow/mnist/	空

hyperParameters	可选，命令行超参数路径	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_tensorflow/mnist/hyper_parameters.txt	空

- 参数**script**和**entryFile**用于指定要执行的TF算法脚本，如果算法比较复杂，分成了多个文件，可以将多个算法文件打包成tar.gz格式，并利用entryFile参数指定该算法的入口文件。

  参数**checkpointDir**用于指定算法将要写入的OSS路径，**在Tensorflow保存模型时需要指定**。

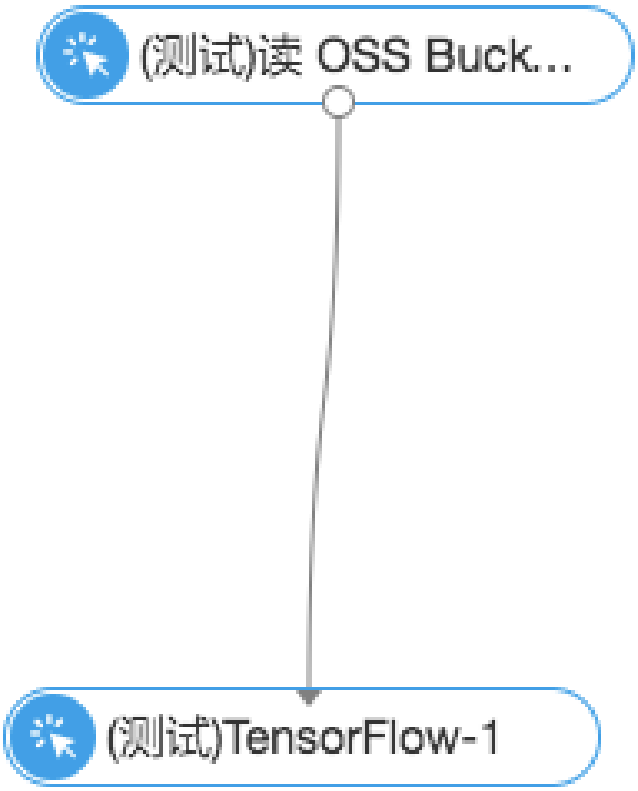
  参数**buckets**用于指定算法将要读取的OSS路径，使用OSS需要指定arn参数。

案例

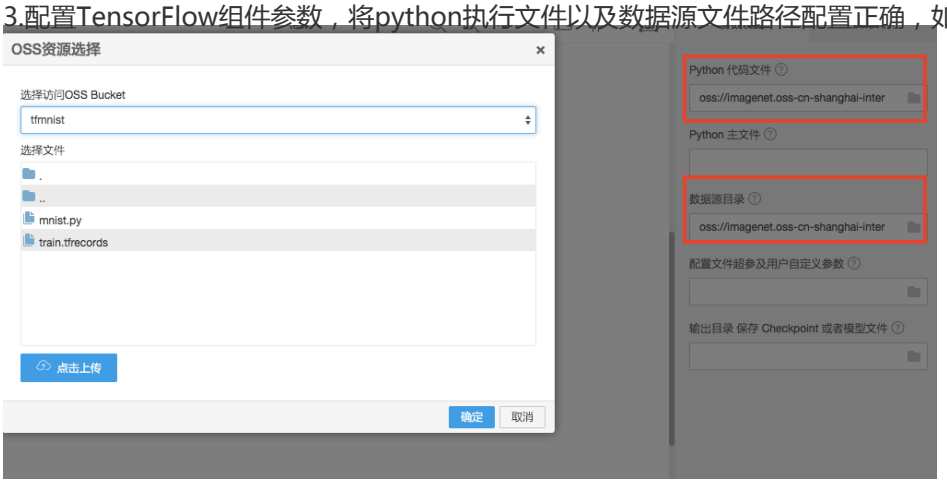
Mnist手写字识别是TF官方的案例，通过训练手写体1~9的数字生成模型，通过模型进行预测。

1.首先在OSS端上传数据的执行python文件以及训练数据集。本案例在OSS华东2 region创建bucket，bucket名为tfmnist，上传python脚本以及训练数据。

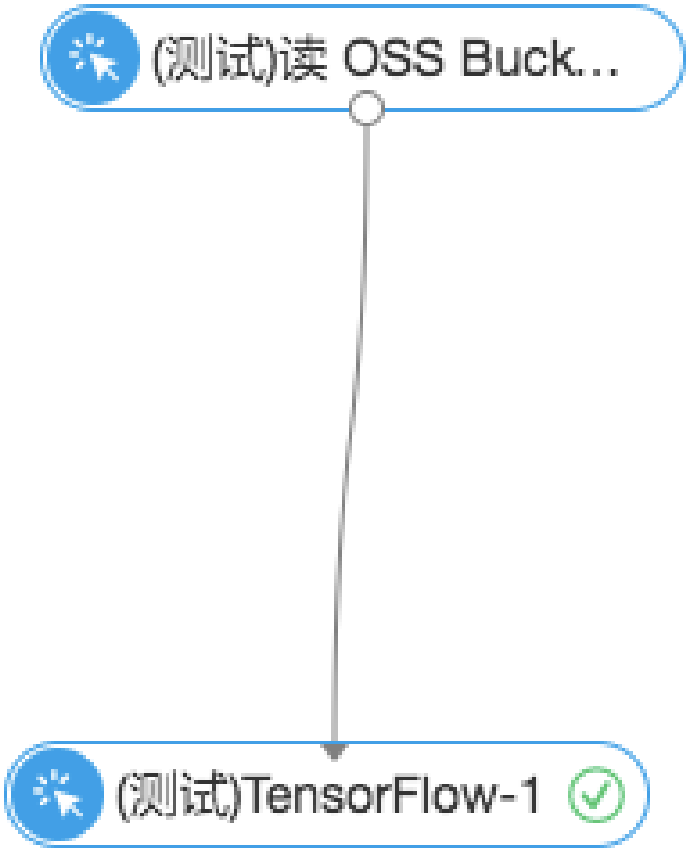
2.拖拽读OSS Bucket和TensorFlow组件，拼接成如下实验，需要设置好OSS Bucket的地区，并且完成RAM授权。如下图：



3.配置TensorFlow组件参数，将python执行文件以及数据源文件路径配置正确，如图：



4.点击运行，直到两个组件运行完成，如下图：



5.可以右键TensorFlow组件，查看运行日志。

```
[1] execute command : set biz_id=1664081855183111^alipay^LTAIwoqNk8Oucv9i^2017-02-15; PAI -
    name tensorflow_ext -project algo_public -Dbuckets="oss://imagenet.oss-cn-shanghai-
    internal.aliyuncs.com/smoke_tensorflow/mnist/" -DgpuRequired="100" -
    Darn="acs:ram::1664081855183111:role/aliyunodpspaidefaultrole" -Dscript="oss://imagenet.oss-
    cn-shanghai-internal.aliyuncs.com/smoke_tensorflow/mnist_ext.py";
[1] execute endpoint : http://service.odps.aliyun.com/api
[1] OK
[1] ID = 20170214170310476goyn17jc2
[1] Odps Instance Id = 20170214170310476goyn17jc2
[1] Sub Instance ID = 20170215010403a6dc5a02_2531_4931_9346_1783c2ccaef1
[1] http://logview.odps.aliyun.com/logview/?
    h=http://service.odps.aliyun.com/api&p=shequ&i=20170215010403a6dc5a02_2531_4931_9346_17
    83c2ccaef1&token=YTQzSjZQSFNLa05udFpHYVpLcFFGR2J6S2JVPSxPRFBTX09CTzoxNjY0MDgx
    ODU1MTgzMTExLDE0ODc2OTY2NDgseyJTdGF0ZW1lbnQiOlt7IkFjdGlvbiI6WyJvZHBzOlJlYWQiXS
    wiRWZmZWN0IjoiQWxsb3ciLCJSZXNvdXJjZSI6WyJhY3M6b2RwczoqOnByb2plY3RzL3NoZXF1L2l
    uc3RhbmNlcy8yMDE3MDIxNTAxMDQwM2E2ZGM1YTAyXzI1MzFfNDkzMV85MzQ2XzE3ODNjMmNj
    YWVmMSJdfV0sIlZlcnNpb24iOiIxIn0=
[1] train: 2017-02-15 01:04:08 TensorflowTask_job:0/0/0[0%]
[1] train: 2017-02-15 01:04:14 TensorflowTask_job:0/0/0[0%]
[1] train: 2017-02-15 01:04:19 TensorflowTask_job:1/0/1[0%]
```

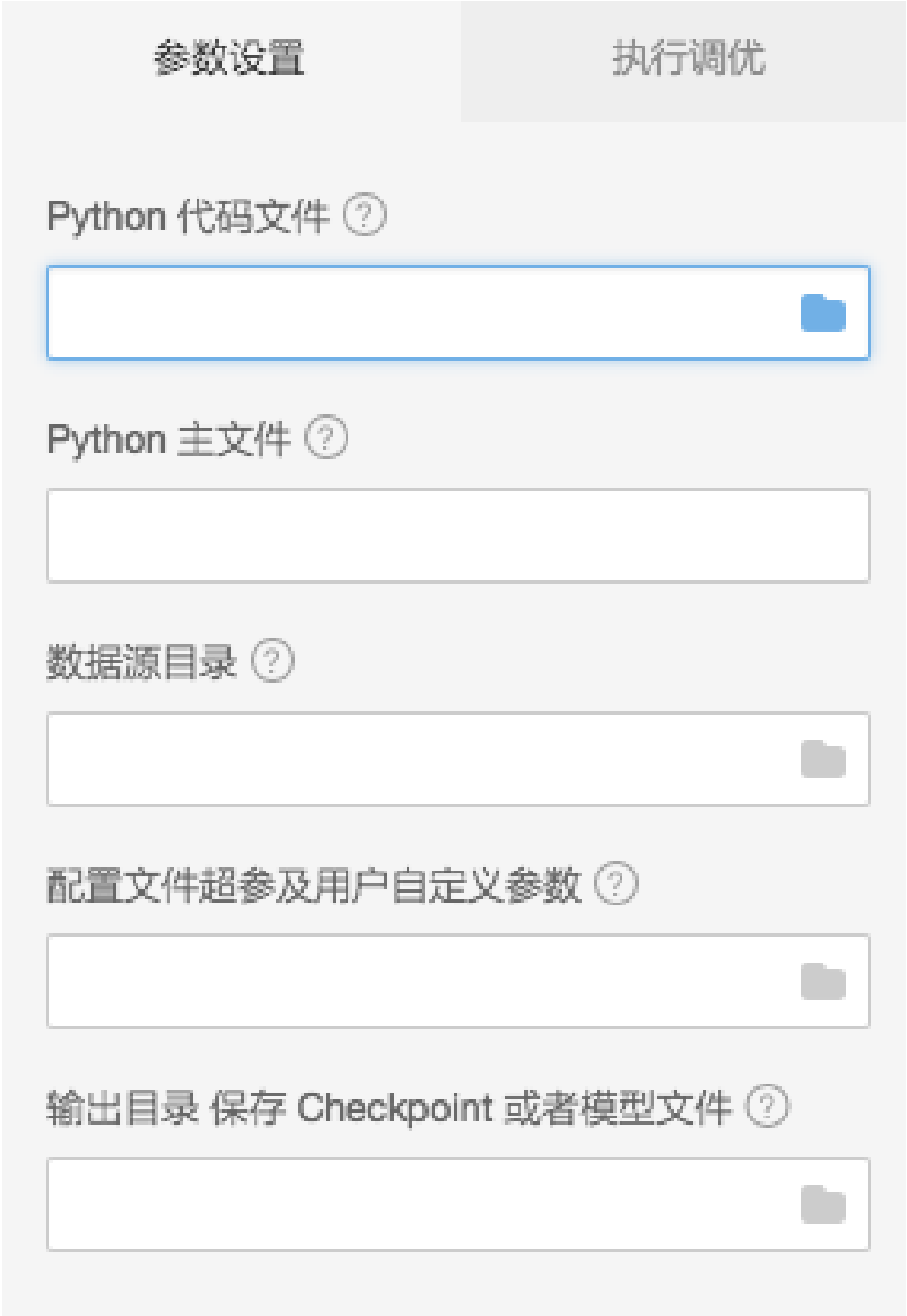
MXNet

背景

MXNet是一个深度学习框架，支持命令和符号编程，可以运行在CPU和GPU集群，MXNet是cxxnet的下一代，cxxnet借鉴了minerva的思想。

参数说明

(1)参数设置



- Python代码文件：程序执行文件，多个文件可通过tar.gz打包上传。
- Python主文件：指定代码文件压缩包中的主文件，可选。
- 数据源目录：选择OSS上的数据源。
- 配置文件超参及用户自定义参数：PAI MXNet支持用户通过Command传入相应的超参配置，这样用户可以在做模型试验的时候可以尝试不同的learning rate, batch size等。
- 输出目录：输出的模型路径。

（2）执行调优

用户可以根据自身任务的复杂程度指定GPU卡数

PAI命令

实际使用中，并不需要指定所有参数(不要直接复制下面的命令....)，各个参数的含义可以参考后面的表格

```
pai -name mxnet_ext -Dscript="oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/mxnet-ext-code/mxnet_cifar10_demo.tar.gz" -DentryFile="train_cifar10.py" -Dbuckets="oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com" -DcheckpointDir="oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/mxnet-ext-model/" -DhyperParameters="oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/mxnet-ext-code/hyperparam.txt.single" -Darn="acs:ram::1664081855183111:role/role-for-pai";
```

各个参数的具体含义如下表：

参数名称	参数描述	参数值格式	默认值	
script	必选，TF算法文件，可以是单个文件或者tar.gz压缩包	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_mxnet/mnist_ext.py	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_mxnet/mnist_ext.py	-
entryFile	可选，算法入口文件名，当script为tar.gz压缩包时，该参数必填	train.py	空	
buckets	必选，输入bucket，可多个，以逗号隔开, 每个bucket须以”/“结尾	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com	空	
hyperParameters	可选，命令行超参数路径	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/mxnet-ext-code/	空	

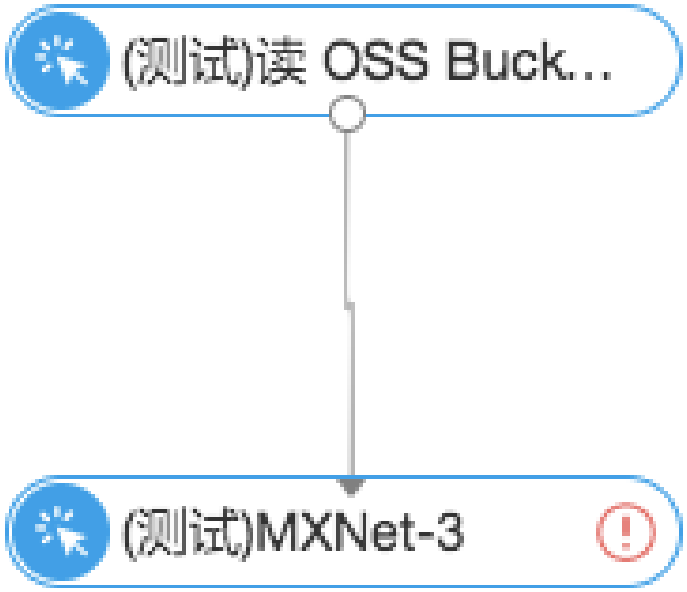
gpuRequired	可选，标识使用GPU资源量	200	100	
checkpointDir	可选，checkpoint目录	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/mxnet-ext-code/	空	

案例

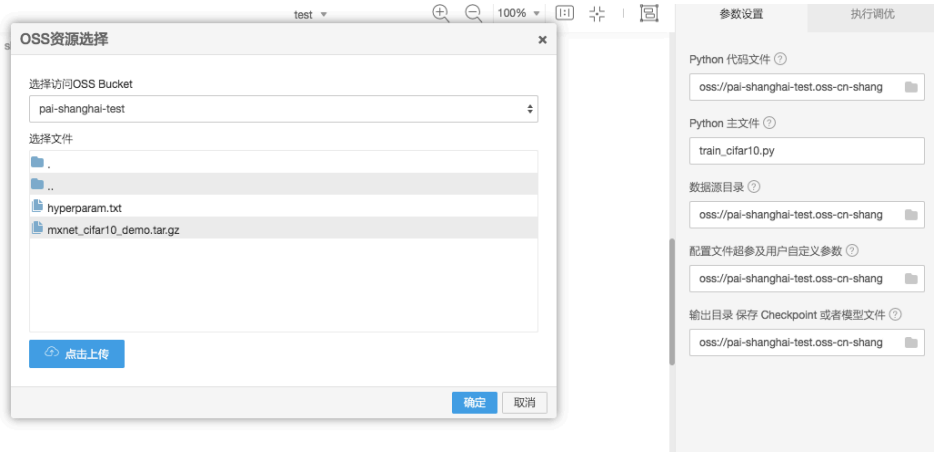
CIFAR-10是MXNet官方提供的基于图片的10分类场景的案例，通过对于6万张32*32的图片进行训练生成模型，可以对飞机、汽车、鸟、猫、鹿、狗、青蛙、马、船、卡车进行自动分类。详细内容见：https://www.cs.toronto.edu/~kriz/cifar.html

1.首先在OSS端上传数据的执行python文件以及训练数据集。本案例在OSS华东2 region创建bucket，bucket名为tfmnist，上传python脚本以及训练数据。

2.拖拽读OSS Bucket和MXNet组件，拼接成如下实验，需要设置好OSS Bucket的地区，并且完成RAM授权。如下图：



3.配置MXNet组件参数，将python执行文件以及数据源文件路径配置正确，如图：



- Python代码文件选择.tar.gz文件
- Python主文件选择tar包中的执行入口文件
- 超惨自定义参数文件选择.txt.single文件
- checkpoint为模型输出目录

4.点击运行，直到两个组件运行完成，如下图：

5.可以右键MXNet组件，查看运行日志。

```
    shanghai-test.oss-cn-shanghai-internal.aliyuncs.com" -DgpuRequired="100" -
    DhyperParameters="oss://pai-shanghai-test.oss-cn-shanghai-internal.aliyuncs.com/mxnet-ext-
    code/hyperparam.txt.single" -Darn="acs:ram::1664081855183111:role/aliyunodpspaidefaultrole" -
    Dscript="oss://pai-shanghai-test.oss-cn-shanghai-internal.aliyuncs.com/mxnet-ext-code/image-
    classification-ext.tar.gz";
[1] execute endpoint : http://service.odps.aliyun.com/api
[1] OK
[1] ID = 20170215024909 65gxhp37jc2
[1] Odps Instance Id = 20170215024909 65gxhp37jc2
[1] Sub Instance ID = 20170215104941 2de3b136_9ce2_4bc6_91e9_088fdd814849
[1] http://logview.odps.aliyun.com/logview/?
    h=http://service.odps.aliyun.com/api&p=shequ&i=20170215104941 2de3b136_9ce2_4bc6_91e9_08
    8fdd814849&token=aTZ1VjBlSk9qR3dGM0FjODNvZmZDc0JYaEs4PSxPRFBTX09CTzoxNjY0MDgx
    ODU1MTgzMTExLDE0ODc3MzE3ODUseyJTdGF0ZW1lbnQiOlt7IkFjdGlvbiI6WyJvZHBzOlJlYWQiXS
    wiRWZmZWN0IjoiQWxsb3ciLCJSZXNvdXJjZSI6WyJhY3M6b2RwczoqOnByb2plY3RzL3NoZXF1L2l
    uc3RhbmNlcy8yMDE3MDIxNTEwNDk0MTJkZTNiMTM2XzIjZTJfNGJjNi85MWU5XzA4OGZkZDgxND
    g0OSJdfV0sIlZlcnNpb24iOiIxIn0=
[1] train: 2017-02-15 10:49:45 MXNetWorker_job:0/0/0[0%]
[1] train: 2017-02-15 10:49:50 MXNetWorker_job:0/0/0[0%]
[1] train: 2017-02-15 10:49:55 MXNetWorker_job:1/0/1[0%]
```

6.最终在checkpoint地址下生成模型如图：

格式转换

目前PAI Caffe不支持自定义格式训练数据，数据需要通过格式转换组件进行转换方可使用。

输入桩接读oss组件。

参数

- 输入oss路径，oss的训练数据的file_list（如bucket.hz.aliyun.com/train_img/train_file_list.txt ）file_list格式如下：

```
bucket/ilsvrc12_val/ILSVRC2012_val_00029021.JPEG 817
bucket/ilsvrc12_val/ILSVRC2012_val_00021046.JPEG 913
bucket/ilsvrc12_val/ILSVRC2012_val_00041166.JPEG 486
bucket/ilsvrc12_val/ILSVRC2012_val_00029527.JPEG 327
bucket/ilsvrc12_val/ILSVRC2012_val_00042825.JPEG 138
```

- 输出oss目录，如bucket_name.oss-cn-hangzhou-zmf.aliyuncs.com/ilsvrc12_val_convert ，会输出转换后的data_file_list.txt和对应的数据文件。data_file_list格式如：

```
bucket/ilsvrc12_val_convert/train_data_00_01
bucket/ilsvrc12_val_convert/train_data_00_02
```

- 编码类型，选项，可选jpg，png，raw等。
- 是否shuffle，勾选
- 文件前缀，默认为data
- resize_height，默认为256
- resize_width，默认为256
- 是否灰度，默认为否
- 是否需要产生图片mean文件，默认否

转换组件PAI命令示例

```
pai -name convert_image_oss2oss
-Darn=acs:ram::1607128916545079:role/test-1
-DossImageList=bucket_name.oss-cn-hangzhou-zmf.aliyuncs.com/image_list.txt
-DossOutputDir=bucket_name.oss-cn-hangzhou-zmf.aliyuncs.com/your/dir
-DencodeType=jpg
-Dshuffle=true
-DdataFilePrefix=train
-DresizeHeight=256
-DresizeWidth=256
-DisGray=false
-DimageMeanFile=false
```

PAI命令参数

参数名称	参数描述	取值范围	是否必选，默认值/行为
ossHost	对应的oss host地址	形式如“oss-test.aliyun-inc.com”	可选，默认值为“oss-cn-hangzhou-zmf.aliyuncs.com”，即为对内oss使用的host
arn	OSS Bucket默认Role对应的ARN	形式如“acs:ram::XXXXXXXXXXXXXXXX:role/ossaccessroleforodps”，中间xxx代表生成的rolearn的16位数字	必选
ossImageList	图片文件列表	形式如“bucket_name/image_list.txt”	必选
ossOutputDir	输出oss目录	形式如“bucket_name/your/dir”	必选
encodeType	编码类型	如jpg,png,raw	可选，默认值为jpg
shuffle	是否shuffle数据	bool值	可选，默认值为true
dataFilePrefix	数据文件前缀	string类型，如train或val	必选
resizeHeight	图像resize的height	int类型，用户自定义	可选，默认值为256
resizeWidth	图像resize的width	int类型，用户自定义	可选，默认值为256
isGray	图像是否为灰度图	bool值	可选，默认值为false
imageMeanFile	是否需要生成imagemean文件	bool值	可选，默认值为false

Caffe

背景

caffe是一个清晰，可读性高，快速的深度学习框架。作者是贾扬清，加州大学伯克利的ph.D，现就职于Facebook。caffe的官网是http://caffe.berkeleyvision.org/。

参数说明

首先配置OSS访问权限

唯一的一个参数就是solver.prototxt文件的oss路径。其中solver由于并行化的修改，同开源caffe略有不同，需要注意一下几点

i. net: “bucket.hz.aliyun.com/alexnet/train_val.prototxt” net的文件位置是oss路径
ii. type: “ParallelSGD” type类型是ParallelSGD，注意这是一个字符串
iii. model_average_iter_interval: 1 多卡下表示同步的频率，1表示每轮都同步一次
iv. snapshot_prefix: “bucket/snapshot/alexnet_train” 模型输出到oss的目录

```
net: "bucket/alexnet/train_val.prototxt"
test_iter: 1000
test_interval: 1000
base_lr: 0.01
lr_policy: "step"
gamma: 0.1
stepsize: 100000
display: 20
max_iter: 450000
momentum: 0.9
weight_decay: 0.0005
```

```
snapshot: 10000
snapshot_prefix: "bucket/snapshot/alexnet_train"
solver_mode: GPU
type: "ParallelSGD"
model_average_iter_interval: 1
```

- train_val中的datalayer需使用BinaryDataLayer，请参考如下示例。

```
layer {
name: "data"
type: "BinaryData"
top: "data"
top: "label"
include {
phase: TRAIN
}
transform_param {
mirror: true
crop_size: 227
mean_file: "bucket/imagenet_mean.binaryproto"
}
binary_data_param {
source: "bucket/ilsvrc12_train_binary/data_file_list.txt"
batch_size: 256
num_threads: 10
}
}
layer {
name: "data"
type: "BinaryData"
top: "data"
top: "label"
include {
phase: TEST
}
transform_param {
mirror: false
crop_size: 227
mean_file: "bucket/imagenet_mean.binaryproto"
}
binary_data_param {
source: "bucket/ilsvrc12_val_binary/data_file_list.txt"
batch_size: 50
num_threads: 10
}
}
```

新的data Layer的名称为“ BinaryData”，其中也支持transform param对输入图像数据进行变换，参数和caffe原生参数保持一致；

其中binary_data_param为数据层本身的参数配置。

binary_data_param中包括以下特殊的参数：

source：数据来源，其中路径为oss中filelist的路径，从bucket名称开始，不包含oss://

num_threads：读取oss数据时并发的线程数目，默认值为10，用户可以根据自己的需求进行调整

PAI命令

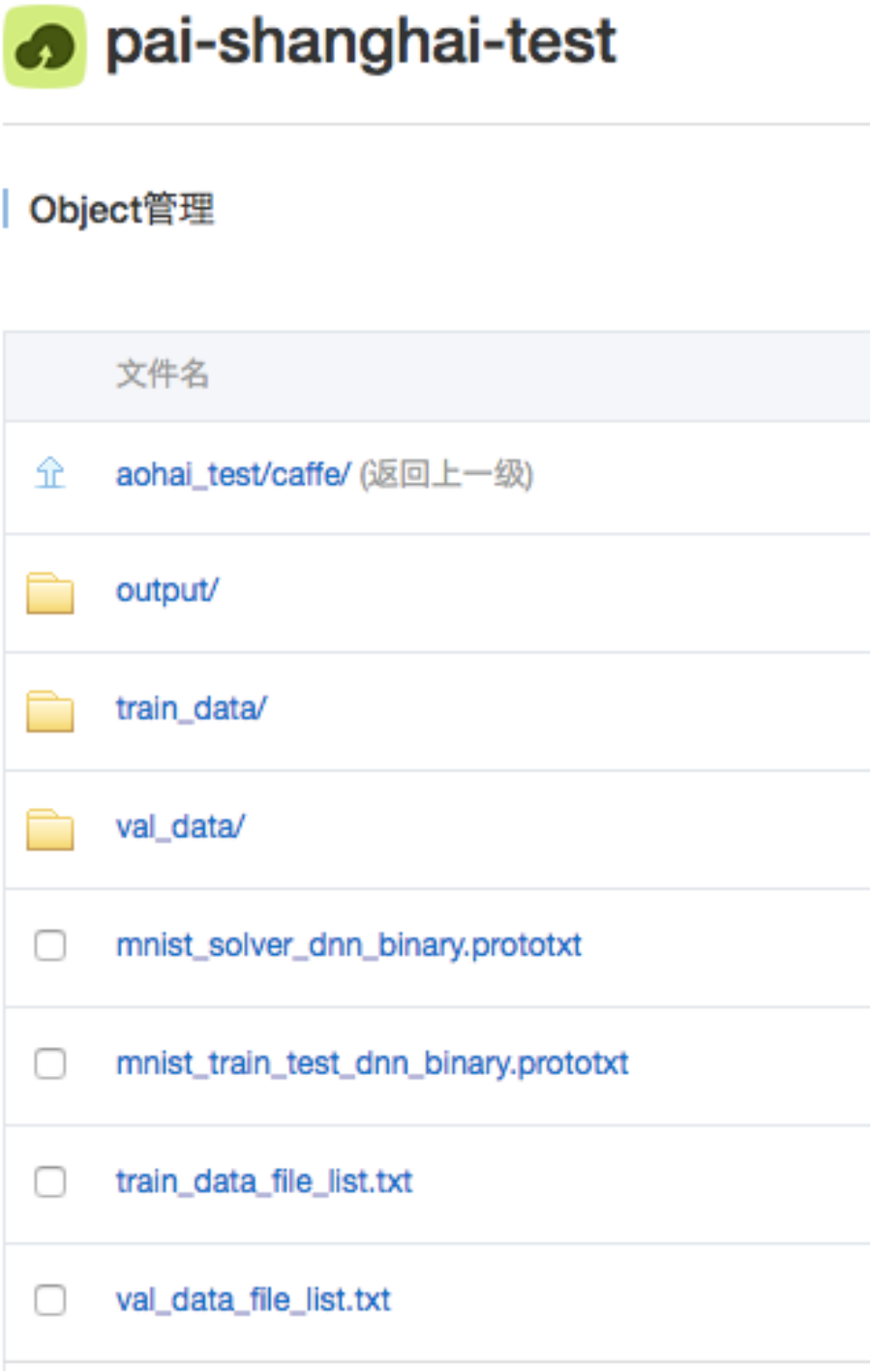
```
pai -name pluto_train_oss
-DossHost=oss-cn-hangzhou-zmf.aliyuncs.com
-Darn=acs:ram::1607128916545079:role/test-1
-DsolverPrototxtFile=bucket_name.oss-cn-hangzhou-zmf.aliyuncs.com/solver.prototxt
-DgpuRequired=1
```

PAI命令参数

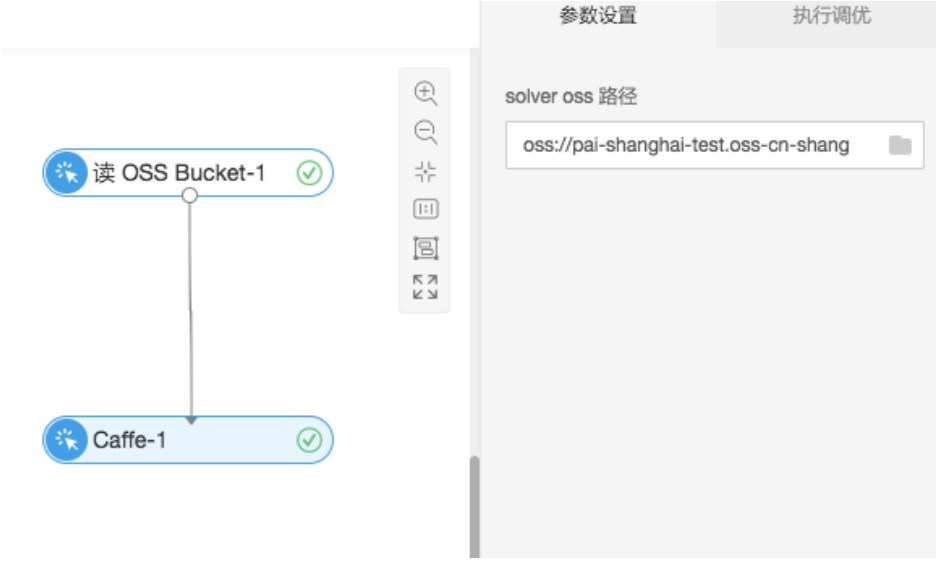
参数名称	参数描述	取值范围	是否必选，默认值/行为
ossHost	对应的oss host地址	形式如“oss-test.aliyun-inc.com”	可选，默认值为“oss-cn-hangzhou-zmf.aliyuncs.com”，即为对内oss使用的host
arn	OSS Bucket默认Role对应的ARN	形式如“acs:ram::XXXXXXXXXXXXXXXX:role/ossaccessroleforodps”，中间xxx代表生成的rolearn的16位数字	必选
solverPrototxtFile	solver文件	solver文件在oss中的路径，从bucket name开始	必选
gpuRequired	GPU卡个数	整型值	可选，默认值为1

案例

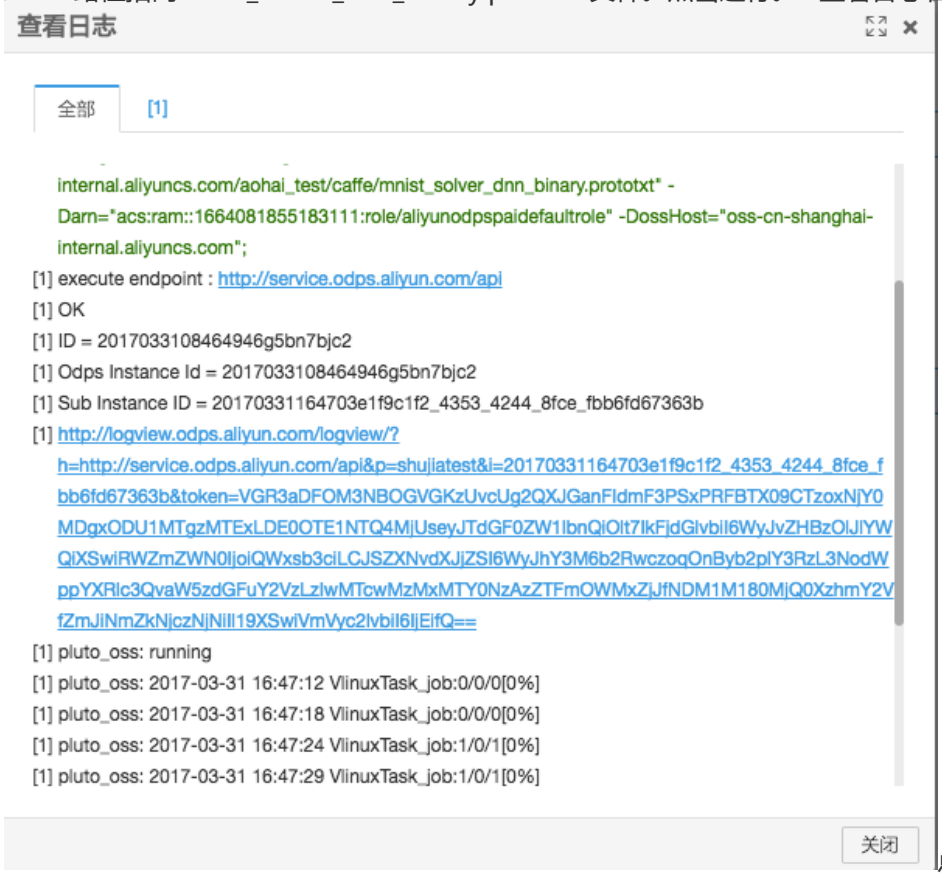
利用Caffe实现mnist的数据训练。1.准备数据源在本页的“深度学习案例代码及数据下载”页找到Caffe数据下载并解压。将数据导入OSS中，本案例路径如下图，请配合代码中的路径理解：



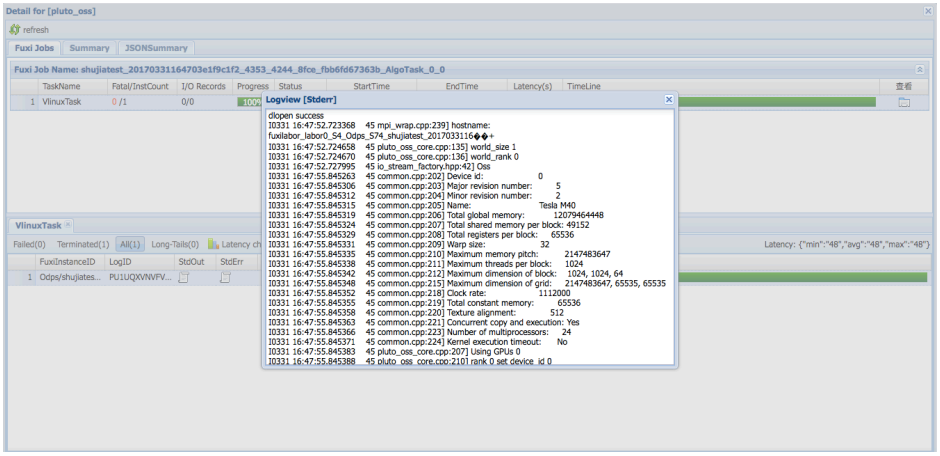
2.实现实验拖拉Caffe组件拼接成如下实验：



将solver oss路径指向mnist_solver_dnn_binary.prototxt文件。点击运行。3.查看日志右键Caffe组件，查看日志：



点击logview链接-》ODPS Tasks-》VlinuxTask-》StdErr，即可查看训练过程产生的日志：



深度学习案例代码及数据下载

TensorFlow相关下载

MXNet相关下载

Caffe相关下载

TensorFlow支持的第三方库

TensorFlow1.0.0版本第三方库

appdirs (1.4.3)
backports-abc (0.5)
backports.shutil-get-terminal-size (1.0.0)
backports.ssl-match-hostname (3.5.0.1)
bleach (2.0.0)
boto (2.48.0)
bz2file (0.98)
certifi (2017.7.27.1)
chardet (3.0.4)
configparser (3.5.0)
cycler (0.10.0)
decorator (4.1.2)
docutils (0.14)
easygui (0.98.1)
entrypoints (0.2.3)
enum34 (1.1.6)
funcsigs (1.0.2)
functools32 (3.2.3.post2)
gensim (2.3.0)
h5py (2.7.0)
html5lib (0.999999999)
idna (2.6)

iniparse (0.4)
ipykernel (4.6.1)
ipython (5.4.1)
ipython-genutils (0.2.0)
ipywidgets (7.0.0)
Jinja2 (2.9.6)
jsonschema (2.6.0)
jupyter (1.0.0)
jupyter-client (5.1.0)
jupyter-console (5.1.0)
jupyter-core (4.3.0)
Keras (2.0.6)
kitchen (1.1.1)
langtable (0.0.31)
MarkupSafe (1.0)
matplotlib (2.0.2)
mistune (0.7.4)
mock (2.0.0)
nbconvert (5.2.1)
nbformat (4.4.0)
networkx (1.11)
nose (1.3.7)
notebook (5.0.0)
numpy (1.13.1)
olefile (0.44)
pandas (0.20.3)
pandocfilters (1.4.2)
pathlib2 (2.3.0)
pbr (3.1.1)
pexpect (4.2.1)
pickleshare (0.7.4)
Pillow (4.2.1)
pip (9.0.1)
prompt-toolkit (1.0.15)
protobuf (3.1.0)
ptyprocess (0.5.2)
pycrypto (2.6.1)
pycurl (7.19.0)
Pygments (2.2.0)
pygobject (3.14.0)
pygpgme (0.3)
pyliblzma (0.5.3)
pyparsing (2.2.0)
python-dateutil (2.6.1)
pytz (2017.2)
PyWavelets (0.5.2)
pyxattr (0.5.1)
PyYAML (3.12)
pyzmq (16.0.2)
qtconsole (4.3.1)
requests (2.18.4)
scandir (1.5)
scikit-image (0.13.0)
scikit-learn (0.19.0)
scikit-sound (0.1.8)
scikit-stack (3.0)

scikit-surprise (1.0.3)
scikit-tensor (0.1)
scikit-video (0.1.2)
scipy (0.19.1)
setuptools (36.2.7)
simplegeneric (0.8.1)
singledispatch (3.4.0.3)
six (1.10.0)
slip (0.4.0)
slip.dbus (0.4.0)
smart-open (1.5.3)
subprocess32 (3.2.7)
tensorflow (1.0.0)
tensorlayer (1.6.1)
terminado (0.6)
testpath (0.3.1)
tflearn (0.3.2)
Theano (0.9.0)
torch (0.1.12.post2)
tornado (4.5.1)
traitlets (4.3.2)
urlgrabber (3.10)
urllib3 (1.22)
wcwidth (0.1.7)
webencodings (0.5.1)
wheel (0.29.0)
widgetsnbextension (3.0.0)
yum-langpacks (0.4.2)
yum-metadata-parser (1.1.4)

TensorFlow1.1.0版本第三方库

appdirs (1.4.3)
backports-abc (0.5)
backports.shutil-get-terminal-size (1.0.0)
backports.ssl-match-hostname (3.5.0.1)
bleach (2.0.0)
boto (2.48.0)
bz2file (0.98)
certifi (2017.7.27.1)
chardet (3.0.4)
configparser (3.5.0)
cycler (0.10.0)
decorator (4.1.2)
docutils (0.14)
easygui (0.98.1)
entrypoints (0.2.3)
enum34 (1.1.6)
funcsigs (1.0.2)
functools32 (3.2.3.post2)
gensim (2.3.0)
h5py (2.7.1)
html5lib (0.999999999)
idna (2.6)

iniparse (0.4)
ipykernel (4.6.1)
ipython (5.4.1)
ipython-genutils (0.2.0)
ipywidgets (7.0.0)
Jinja2 (2.9.6)
jsonschema (2.6.0)
jupyter (1.0.0)
jupyter-client (5.1.0)
jupyter-console (5.2.0)
jupyter-core (4.3.0)
jupyter-tensorboard (0.1.1)
Keras (2.0.8)
kitchen (1.1.1)
langtable (0.0.31)
MarkupSafe (1.0)
matplotlib (2.0.2)
mistune (0.7.4)
mock (2.0.0)
nbconvert (5.3.0)
nbformat (4.4.0)
networkx (1.11)
nose (1.3.7)
notebook (4.4.1)
numpy (1.13.1)
olefile (0.44)
pandas (0.20.3)
pandocfilters (1.4.2)
pathlib2 (2.3.0)
pbr (3.1.1)
pexpect (4.2.1)
pickleshare (0.7.4)
Pillow (4.2.1)
pip (9.0.1)
prompt-toolkit (1.0.15)
protobuf (3.1.0)
ptyprocess (0.5.2)
pycrypto (2.6.1)
pycurl (7.19.0)
Pygments (2.2.0)
pygobject (3.14.0)
pygpgme (0.3)
pyliblzma (0.5.3)
pyparsing (2.2.0)
python-dateutil (2.6.1)
pytz (2017.2)
PyWavelets (0.5.2)
pyxattr (0.5.1)
PyYAML (3.12)
pyzmq (16.0.2)
qtconsole (4.3.1)
requests (2.18.4)
scandir (1.5)
scikit-image (0.13.0)
scikit-learn (0.19.0)
scikit-sound (0.1.8)

scikit-stack (3.0)
scikit-surprise (1.0.3)
scikit-tensor (0.1)
scikit-video (0.1.2)
scipy (0.19.1)
setuptools (36.4.0)
simplegeneric (0.8.1)
singledispatch (3.4.0.3)
six (1.10.0)
slip (0.4.0)
slip.dbus (0.4.0)
smart-open (1.5.3)
subprocess32 (3.2.7)
tensorflow (1.1.0)
tensorlayer (1.6.2)
terminado (0.6)
testpath (0.3.1)
tflearn (0.3.2)
Theano (0.9.0)
torch (0.1.12.post2)
tornado (4.5.2)
traitlets (4.3.2)
urlgrabber (3.10)
urllib3 (1.22)
wcwidth (0.1.7)
webencodings (0.5.1)
Werkzeug (0.12.2)
wheel (0.29.0)
widgetsnbextension (3.0.2)
yum-langpacks (0.4.2)
yum-metadata-parser (1.1.4)

TensorFlow1.2.1版本第三方库

appdirs (1.4.3)
backports-abc (0.5)
backports.shutil-get-terminal-size (1.0.0)
backports.ssl-match-hostname (3.5.0.1)
backports.weakref (1.0rc1)
bleach (1.5.0)
boto (2.48.0)
bz2file (0.98)
certifi (2017.7.27.1)
chardet (3.0.4)
configparser (3.5.0)
cycler (0.10.0)
decorator (4.1.2)
docutils (0.14)
easygui (0.98.1)
entrypoints (0.2.3)
enum34 (1.1.6)
funcsigs (1.0.2)
functools32 (3.2.3.post2)
gensim (2.3.0)

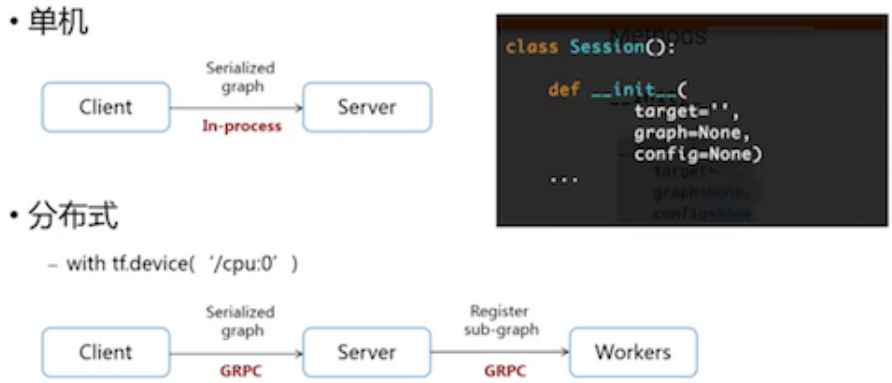
h5py (2.7.1)
html5lib (0.9999999)
idna (2.6)
iniparse (0.4)
ipykernel (4.6.1)
ipython (5.4.1)
ipython-genutils (0.2.0)
ipywidgets (7.0.0)
Jinja2 (2.9.6)
jsonschema (2.6.0)
jupyter (1.0.0)
jupyter-client (5.1.0)
jupyter-console (5.2.0)
jupyter-core (4.3.0)
jupyter-tensorboard (0.1.1)
Keras (2.0.8)
kitchen (1.1.1)
langtable (0.0.31)
Markdown (2.6.9)
MarkupSafe (1.0)
matplotlib (2.0.2)
mistune (0.7.4)
mock (2.0.0)
nbconvert (5.3.0)
nbformat (4.4.0)
networkx (1.11)
nose (1.3.7)
notebook (4.4.1)
numpy (1.13.1)
olefile (0.44)
pandas (0.20.3)
pandocfilters (1.4.2)
pathlib2 (2.3.0)
pbr (3.1.1)
pexpect (4.2.1)
pickleshare (0.7.4)
Pillow (4.2.1)
pip (9.0.1)
prompt-toolkit (1.0.15)
protobuf (3.1.0)
ptyprocess (0.5.2)
pycrypto (2.6.1)
pycurl (7.19.0)
Pygments (2.2.0)
pygobject (3.14.0)
pygpgme (0.3)
pyliblzma (0.5.3)
pyparsing (2.2.0)
python-dateutil (2.6.1)
pytz (2017.2)
PyWavelets (0.5.2)
pyxattr (0.5.1)
PyYAML (3.12)
pyzmq (16.0.2)
qtconsole (4.3.1)
requests (2.18.4)

```
scandir (1.5)
scikit-image (0.13.0)
scikit-learn (0.19.0)
scikit-sound (0.1.8)
scikit-stack (3.0)
scikit-surprise (1.0.3)
scikit-tensor (0.1)
scikit-video (0.1.2)
scipy (0.19.1)
setuptools (36.4.0)
simplegeneric (0.8.1)
singledispatch (3.4.0.3)
six (1.10.0)
slip (0.4.0)
slip.dbus (0.4.0)
smart-open (1.5.3)
subprocess32 (3.2.7)
tensorflow (1.2.1)
tensorlayer (1.6.2)
terminado (0.6)
testpath (0.3.1)
tflearn (0.3.2)
Theano (0.9.0)
torch (0.1.12.post2)
tornado (4.5.2)
traitlets (4.3.2)
urlgrabber (3.10)
urllib3 (1.22)
wcwidth (0.1.7)
webencodings (0.5.1)
Werkzeug (0.12.2)
wheel (0.29.0)
widgetsnbextension (3.0.2)
yum-langpacks (0.4.2)
yum-metadata-parser (1.1.4)
```

TensorFlow读取数据方法说明

低效的IO方式

最近通过观察PAI平台上TensoFlow用户的运行情况，发现大家在数据IO这方面还是有比较大的困惑，主要是因为很多同学没有很好的理解本地执行TensorFlow代码和分布式云端执行TensorFlow的区别。本地读取数据是server端直接从client端获得graph进行计算，而云端服务server在获得graph之后还需要将计算下发到各个worker处理(具体原理可以参考视频教程-Tensorflow高级篇：https://tianchi.aliyun.com/competition/new_articleDetail.html)。



本文通过读取一个简单的CSV文件为例，帮助大家快速了解如何使用TensorFlow高效的读取数据。CSV文件如下：

```
1,1,1,1,1
2,2,2,2,2
3,3,3,3,3
```

首先我们来看下大家容易产生问题的几个地方。

1.不建议用python本地读取文件的方式

PAI支持python的自带IO方式，但是需要将数据源和代码打包上传的方式使用，这种读取方式是将数据写入内存之后再计算，效率比较低，不建议使用。范例代码如下：

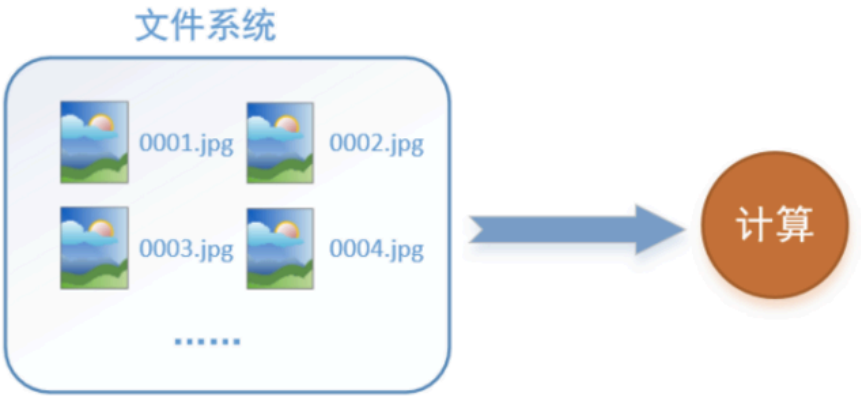
```
import csv
csv_reader=csv.reader(open('csvtest.csv'))
for row in csv_reader:
print(row)
```

2.尽量不要用第三方库的读取文件方法

很多同学使用第三方库的一些数据IO的方式进行数据读取，比如TFLearn、Panda的数据IO方式，这些方法很多都是通过封装PYTHON的读取方式实现的，所以在PAI平台使用的时候也会造成效率低下问题。

3.尽量不要用preload的方式读取文件

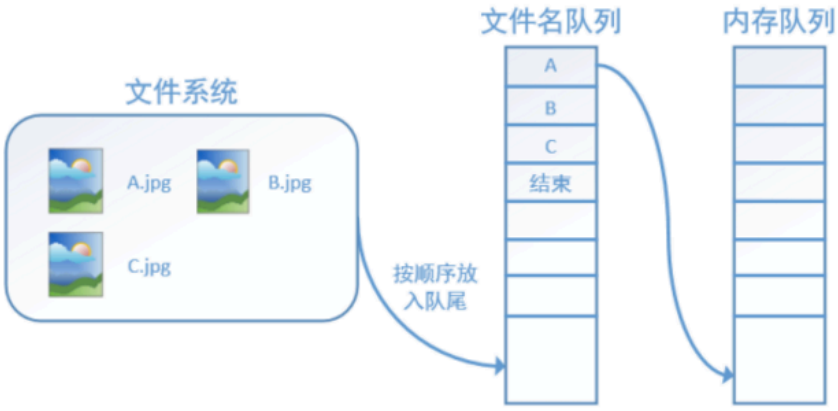
很多人在用PAI的服务的时候表示GPU并没有比本地的CPU速度快的明显，主要问题可能就出在数据IO这块。preload的方式是先把数据全部都读到内存中，然后再通过session计算，比如feed的读取方式。这样要先进行数据读取，再计算，不同步造成性能浪费，同时因为内存限制也无法支持大数据量的计算。举个例子：假设我们的硬盘中有一个图片数据集0001.jpg，0002.jpg，0003.jpg……我们只需要把它们读取到内存中，然后提供给GPU或是CPU进行计算就可以了。这听起来很容易，但事实远没有那么简单。事实上，我们必须要把数据先读入后才能进行计算，假设读入用时0.1s，计算用时0.9s，那么就意味着每过1s，GPU都会有0.1s无事可做，这就大大降低了运算的效率。



下面我们看下高效的读取方式。

高效的IO方式

高效的TensorFlow读取方式是将数据读取转换成OP，通过session run的方式拉去数据。另外，读取线程源源不断地将文件系统中的图片读入到一个内存的队列中，而负责计算的是另一个线程，计算需要数据时，直接从内存队列中取就可以了。这样就可以解决GPU因为IO而空闲的问题！



下面我们看下代码，如何在PAI平台通过OP的方式读取数据：

```
import argparse
import tensorflow as tf
import os
FLAGS=None
def main(_):
dirname = os.path.join(FLAGS.buckets, "csvtest.csv")
reader=tf.TextLineReader()
filename_queue=tf.train.string_input_producer([dirname])
key,value=reader.read(filename_queue)
record_defaults=[[''],[''],[''],[''],['']]
d1, d2, d3, d4, d5= tf.decode_csv(value, record_defaults, ',')

init=tf.initialize_all_variables()
```

```
with tf.Session() as sess:
sess.run(init)
coord = tf.train.Coordinator()
threads = tf.train.start_queue_runners(sess=sess,coord=coord)
for i in range(4):
print(sess.run(d2))
coord.request_stop()
coord.join(threads)

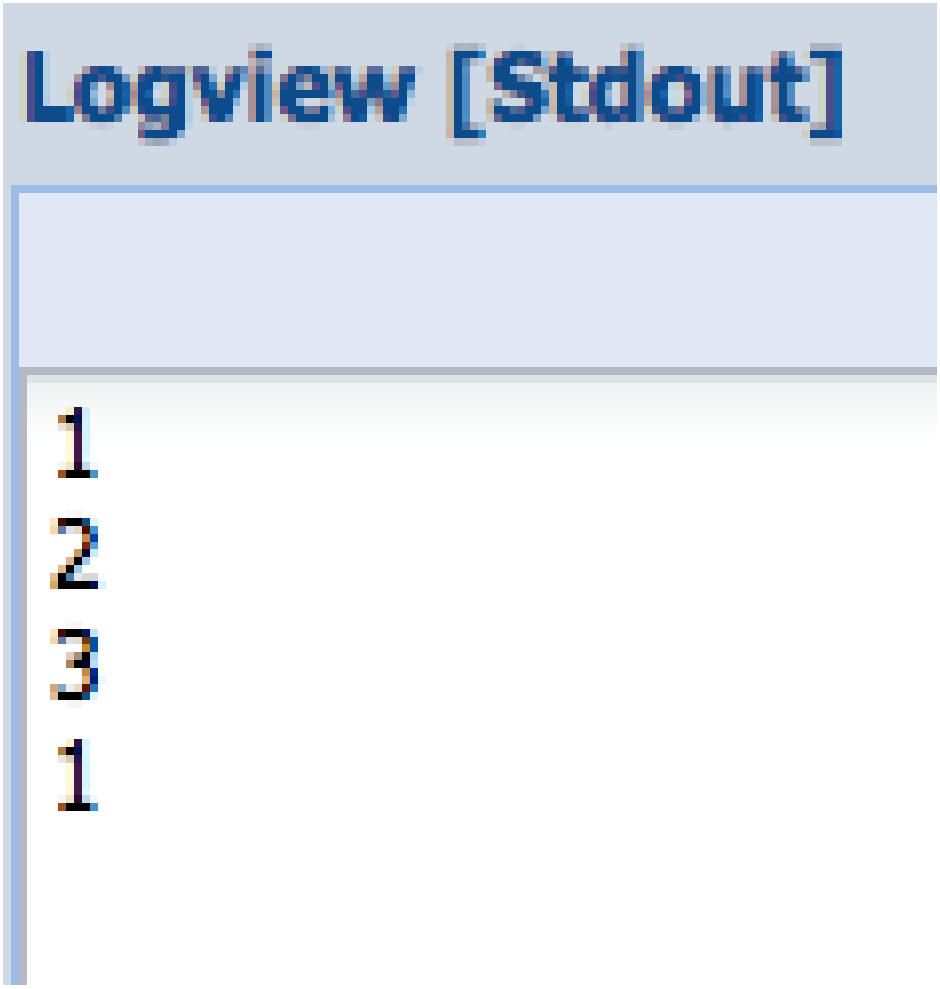
if __name__ == '__main__':
parser = argparse.ArgumentParser()
parser.add_argument('--buckets', type=str, default='',
help='input data path')
parser.add_argument('--checkpointDir', type=str, default='',
help='output model path')
FLAGS, _ = parser.parse_known_args()
tf.app.run(main=main)
```

- dirname:OSS文件路径，可以是数组，方便下一阶段shuffle
- reader：TF内置各种reader API，可以根据需求选用
- tf.train.string_input_producer：将文件生成队列
- tf.decode_csv：是一个splite功能的OP，可以拿到每一行的特定参数
- 通过OP获取数据，在session中需要tf.train.Coordinator()和 tf.train.start_queue_runners(sess=sess,coord=coord)

在代码中，我们的输入是3行5个字段：

```
1,1,1,1,1
2,2,2,2,2
3,3,3,3,3
```

我们循环输出4次，打印出第2个字段。结果如图：



输出结果也证明了数据结构是成队列。

其它

- PAI notebook功能上线，支持在线修改代码并且内置各种深度学习框架，欢迎使用：https://data.aliyun.com/product/learn
- 强烈推荐教程：https://yq.aliyun.com/articles/177245
- 本文参考了互联网上《十图详解TensorFlow数据读取机制（附代码）》一文，关于图片的读取方式也可以参考这篇文章，感谢原作者。

文本分析

目录

TF-IDF

PLDA

Word2Vec

Doc2Vec

SplitWord

三元组转kv

字符串相似度

字符串相似度-topN

停用词过滤

文本摘要

文章相似度

句子拆分

条件随机场

关键词抽取

ngram-count

语义向量距离

条件随机场

PMI

词频统计
功能介绍

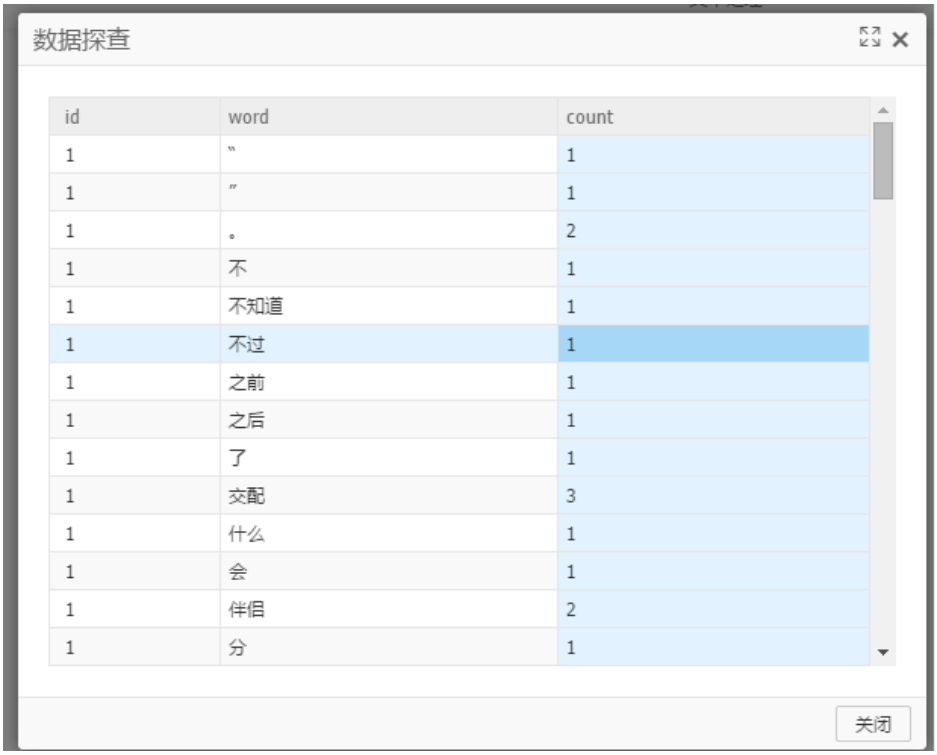
- 在对文章进行分词的基础上，按行保序输出对应文章ID列(docId)对应文章的词，统计指定文章ID列(docId)对应文章内容(docContent)的词频。

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

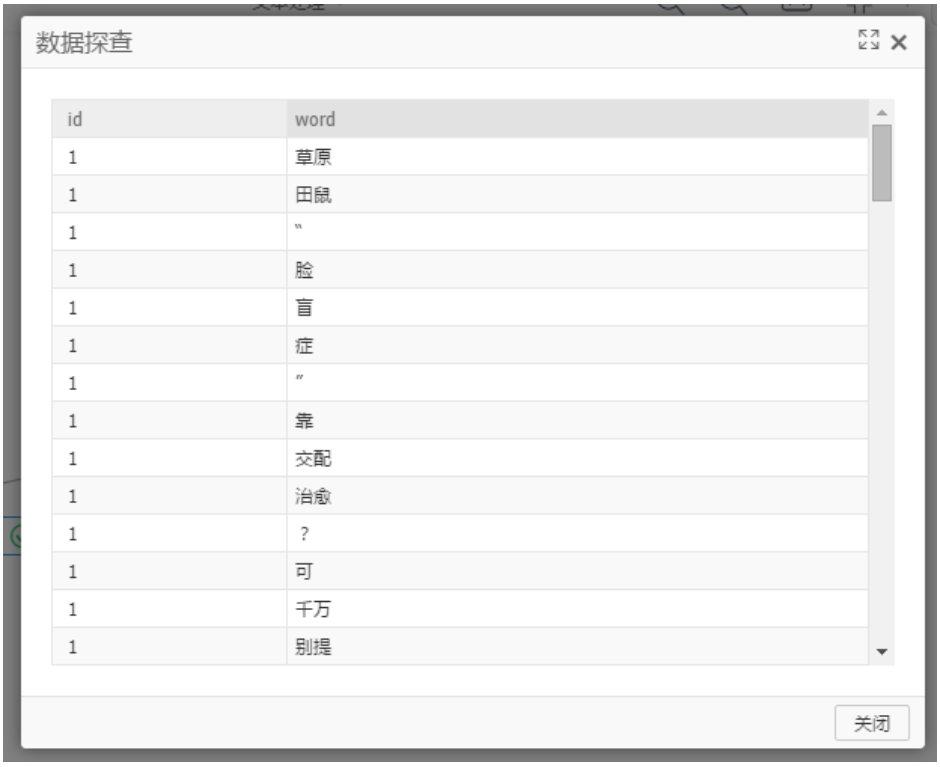
两个输出参数:

第一个输出端：输出表包含三个字段—id,word,count，如下图：



count—统计每个文档中，对应word词汇出现的次数

第二个输出端：输出包含两个字段—id,word,如下图：



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	-
outputTableNameM	输出保序词语表名	-	-

ulti			
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/name2=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

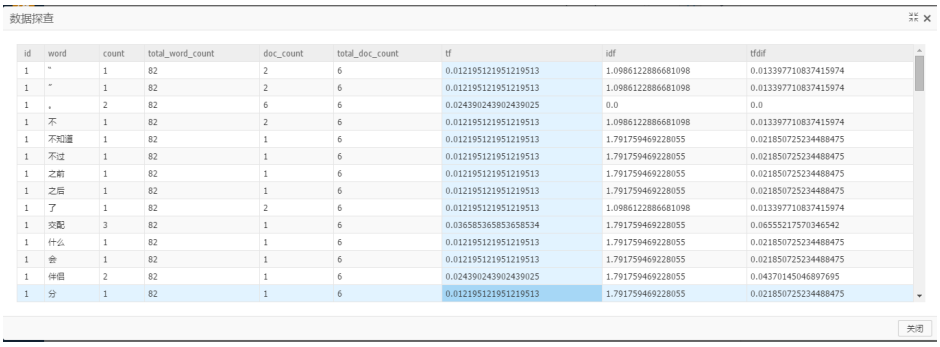
- TF-IDF（term frequency–inverse document frequency）是一种用于资讯检索与文本挖掘的常用加权技术。TF-IDF是一种统计方法，用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。TF-IDF加权的各种形式常被搜索引擎应用，作为文件与用户查询之间相关程度的度量或评级。
- 详细介绍，请参考：[维基百科tf-idf]
- 本组件是词频统计输出的基础上，计算各个word对于各个文章的tfidf值

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：



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
1	,	2	82	6	6	0.024390243902439025	0.0	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	交配	3	82	1	6	0.036585365853658534	1.791759469228055	0.06555217570346542
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	伴侣	2	82	1	6	0.024390243902439025	1.791759469228055	0.04370145046897695
1	分	1	82	1	6	0.012195121951219513	1.791759469228055	0.021850725234488475

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	必/选填	默认值
inputTableName	输入表名	必填	-
inputTablePartitions	输入表分区	选填	输入表的所有partition
docIdCol	标识文章id的列名，仅可指定一列	必填	-
wordCol	word列名，仅可指定一列	必填	-
countCol	count列名，仅可指定一列	必填	-
outputTableName	输出表名	必填	-
lifecycle	输出表生命周期（单位：天）	选填	无生命周期限制
coreNum	核心数，需和memSizePerCore同时设置才起作用	选填	自动计算
memSizePerCore	内存数，需和coreNum同时设置才起作用	选填	自动计算

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，实现数据的转换输入格式如下：

```
| id          | features |
+-------------+----------+
| 2           | 38:3.0,39:1.0,40:3.0,41:1.0,42:1.0,43:2.0,44:1.0,45:1.0,46:1.0,47:1.0,48:1.0,49:2.0,50:1.0,51:1.0,52:1.0,53:1.0,54:1.0,55:1.0,5
6:1.0,57:1.0,58:1.0,59:1.0,60:1.0,61:1.0,62:1.0,63:1.0,64:1.0,65:1.0,66:1.0,67:1.0,68:1.0,69:1.0,70:1.0,71:1.0,72:1.0,73:1.0,74:1.0,75:1.0,76:
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：

- 输出桩1：词频，算法内部抽样后每个词在主题出现次数
- 输出桩2：P(单词|主题)每个主题下词的概率
- 输出桩3：P(主题|单词)每个单词对应各个主题的概率
- 输出桩4：P(文档|主题)每个主题对应各个文档的概率
- 输出桩5：P(主题|文档)每个文档对应主题的概率
- 输出桩6：P(主题)每个主题的概率，表明在整个文档中的权重

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

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
pwzTableName	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

Doc2Vec

功能介绍

Doc2Vec可以将文章映射成向量。

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：输出文档向量表，词向量表，词汇表

pai命令示例

```
pai -name pai_doc2vec
-project algo_public
-DinputTableName=d2v_input
-DdocIdColName=docid
```

```
-DdocColName=text_seg
-DoutputWordTableName=d2v_word_output
-DoutputDocTableName=d2v_doc_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
docIdColName	文档id列名	列名	必选
docColName	文档内容，可以是用分词的结果，空格分割	列名	必选
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’
outputWordTableName	输出词向量表	表名	必选

outputDocTableName	输出文档向量表	表名	必选
lifecycle	输出表的生命周期	-	可选，默认值无生命周期
coreNum	核心数，需和memSizePerCore同时设置才起作用	-	自动计算
memSizePerCore	内存数，需和coreNum同时设置才起作用	-	自动计算

SplitWord

- 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

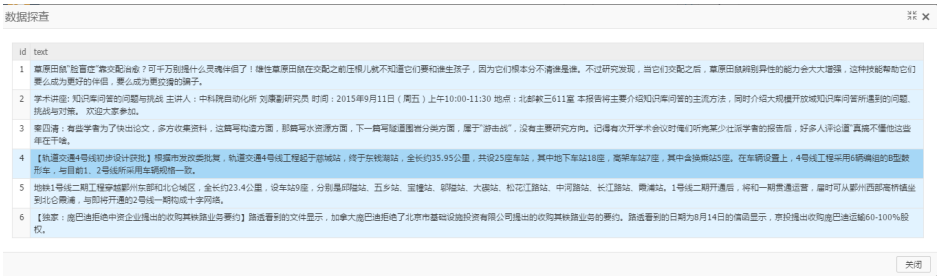
字段设置（略）

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：



输出结果如下：

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,INTE	默认为

		RNET_CHN	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
enableOrgnizationTagger	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
enableChnNumMerge	中文数字合并为一个检索单元	true,false	false
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

三元组转kv

功能介绍

- 给定三元组（row,col,value）类型为XXD 或 XXL, X表示任意类型, D表示Double, L表示bigint，转成kv格式（row,[col_id:value]），其中row和value类型和原始输入数据一致，col_id类型是bigint，并给出col的索引表映射到col_id
- 输入表形式如下

id	word	count
01	a	10
01	b	20

01	c	30

- 输出kv表如下，kv分隔符可以自定义

id	key_value
01	1:10;2:20;3:30

- 输出word的索引表如下

key	key_id
a	1
b	2
c	3

PAI命令示例

```
PAI -name triple_to_kv
-project algo_public
-DinputTableName=test_data
-DoutputTableName=test_kv_out
-DindexOutputTableName=test_index_out
-DidColName=id
-DkeyColName=word
-DvalueColName=count
-DinputTablePartitions=ds=test1
-DindexInputTableName=test_index_input
-DindexInputKeyColName=word
-DindexInputKeyIdColName=word_id
-DkvDelimiter=:
-DpairDelimiter=;
-Dlifecycle=3
```

算法参数

参数名称	参数描述	参数值可选项	默认值	备注
inputTableName	必选，输入表名	-	-	不能为空表
idColName	必选，转成kv表时保持不变的列名	-	-	-
keyColName	必选，kv中的key	-	-	-
valueColName	必选，kv中的value	-	-	-

outputTableName	必选，输出kv表名	-	-	-
indexOutputTableName	必选，输出key的索引表	-	-	-
indexInputTableName	可选，输入已有的索引表	-	“”	不能是空表，可以只有部分key的索引
indexInputKeyColName	可选，输入索引表key的列名	-	“”	输入indexInputTableName时必选此项
indexInputKeyIdColName	可选，输入索引表key索引号的列名	-	“”	输入indexInputTableName时必选此项
inputTablePartitions	可选，输入表的分区	-	“”	只能输入单个分区
kvDelimiter	可选，key和value之间分隔符	-	:	-
pairDelimiter	可选，kv对之间分隔符	-	;	-
lifecycle	可选，输出结果表的生命周期	-	不设生命周期	-
coreNum	可选，指定instance的总数	-	-1	默认会根据输入数据大小计算
memSizePerCore	可选，指定memory大小，范围在100~64*1024之间	-	-1	默认会根据输入数据大小计算

实例

测试数据

新建数据SQL

```
drop table if exists triple2kv_test_input;
create table triple2kv_test_input as
select
*
from
(
select '01' as id, 'a' as word, 10 as count from dual
union all
select '01' as id, 'b' as word, 20 as count from dual
union all
```

```
select '01' as id, 'c' as word, 30 as count from dual
union all
select '02' as id, 'a' as word, 100 as count from dual
union all
select '02' as id, 'd' as word, 200 as count from dual
union all
select '02' as id, 'e' as word, 300 as count from dual
) tmp;
```

运行命令

```
PAI -name triple_to_kv
-project algo_public
-DinputTableName=triple2kv_test_input
-DoutputTableName=triple2kv_test_input_out
-DindexOutputTableName=triple2kv_test_input_index_out
-DidColName=id
-DkeyColName=word
-DvalueColName=count
-Dlifecycle=1;
```

运行结果*triple2kv_test_input_out*

```
+------------+------------+
| id | key_value |
+------------+------------+
| 02 | 1:100;4:200;5:300 |
| 01 | 1:10;2:20;3:30 |
+------------+------------+
```

triple2kv_test_input_index_out

```
+------------+------------+
| key | key_id |
+------------+------------+
| a | 1 |
| b | 2 |
| c | 3 |
| d | 4 |
| e | 5 |
+------------+------------+
```

字符串相似度

功能介绍

计算字符串相似度在机器学习领域是一个非常基本的操作，主要用在信息检索，自然语言处理，生物信息学等领域。本算法支持Levenshtein Distance，Longest Common SubString，String Subsequence Kernel，Cosine，simhash_hamming五种相似度计算方式。支持两两计算和top n计算两种输入方式。

Levenshtein（Levenshtein Distance）支持距离和相似度两个参数，相似度=1-距离，距离在参数中表示为levenshtein，相似度在参数中表示为levenshtein_sim。

lcs（Longest Common SubString）支持距离和相似度两个参数，相似度=1-距离，距离在参数中表示为lcs，相似度在参数中表示为lcs_sim。

ssk（String Subsequence Kernel）支持相似度计算，在参数中表示为ssk。

参考：Lodhi, Huma; Saunders, Craig; Shawe-Taylor, John; Cristianini, Nello; Watkins, Chris (2002). “Text classification using string kernels”. Journal of Machine Learning Research: 419–444.

cosine（Cosine）支持相似度计算，在参数中表示为cosine。

参考：Leslie, C.; Eskin, E.; Noble, W.S. (2002), The spectrum kernel: A string kernel for SVM protein classification 7, pp. 566–575

simhash_hamming，其中SimHash算法是把原始的文本映射为64位的二进制指纹，HammingDistance则是计算二进制指纹在相同位置上不同的字符的个数，支持距离和相似度两个参数，相似度=1-距离/64.0，距离在参数中表示为simhash_hamming，相似度在参数中表示为simhash_hamming_sim。

SimHash详细介绍请见pdf；

HammingDistance详细介绍请见维基百科链接wiki

两两计算

PAI命令

```
PAI -name string_similarity
-project algo_public
-DinputTableName="pai_test_string_similarity"
-DoutputTableName="pai_test_string_similarity_output"
-DinputSelectedColName1="col0"
-DinputSelectedColName2="col1";
```

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	-	-
outputTableName	必选，输出表的表名	-	-
inputSelectedColName1	可选，相似度计算中第一列的列名	-	表中第一个为类型为string的列名
inputSelectedColName2	可选，相似度计算中第二列的列名	-	表中第二个为类型为string的列名
inputAppendColNames	可选，输出表追加的列名	-	不追加
inputTablePartitions	可选，输入表选中的分区	-	选择全表

outputColName	可选，输出表中相似度列的列名。列名中不能有特殊字符，只能用英文的a-z，A-Z及数字和下划线_，且以字母开头，名称的长度不超过128字节。	-	output
method	可选，相似度计算方法	levenshtein, levenshtein_sim, lcs, lcs_sim, ssk, cosine, simhash_hamming, simhash_hamming_sim	levenshtein_sim
lambda	可选，匹配字符串的权重，ssk中可用	(0, 1)	0.5
k	可选，子串的长度，ssk和cosine中可用	(0, 100)	2
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期
coreNum	可选，计算的核心数	正整数	系统自动分配
memSizePerCore	可选，每个核心的内存（单位为兆）	正整数，范围(0, 65536)	系统自动分配

示例

测试数据

```
create table pai_ft_string_similarity_input as select * from
(select 0 as id, "北京" as col0, "北京" as col1 from dual
union all
select 1 as id, "北京" as col0, "北京上海" as col1 from dual
union all
select 2 as id, "北京" as col0, "北京上海香港" as col1 from dual
)tmp;
```

pai命令

```
PAI -name string_similarity
-project sre_mpi_algo_dev
-DinputTableName=pai_ft_string_similarity_input
-DoutputTableName=pai_ft_string_similarity_output
-DinputSelectedColName1=col0
-DinputSelectedColName2=col1
-Dmethod=simhash_hamming
-DinputAppendColNames=col0,col1;
```

输出说明

方法simhash_hamming输出结果如下：

col0	col1	output
北京	北京	0
北京	北京上海	6
北京	北京上海香港	13

方法simhash_hamming_sim输出结果如下：

col0	col1	output
北京	北京	1
北京	北京上海	0.90625
北京	北京上海香港	0.796875

字符串相似度-topN

PAI命令

```
PAI -name string_similarity_topn
-project algo_public
-DinputTableName="pai_test_string_similarity_topn"
-DoutputTableName="pai_test_string_similarity_topn_output"
-DmapTableName="pai_test_string_similarity_map_topn"
-DinputSelectedColName="col0"
-DmapSelectedColName="col1";
```

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	-	-
mapTableName	必选，输入的映射表名	-	-
outputTableName	必选，输出表的表名	-	-
inputSelectedColName	可选，相似度计算中左表的列名	-	表中第一个为类型为string的列名
mapSelectedColName	可选，相似度计算中映射表的列名，左表中的每一行都会和映射表中所有的字符串计算出相似度，并最终已topn的方式给出结果	-	表中第一个为类型为string的列名

inputAppendColNames	可选，输入表在输出表追加的列名	-	不追加
inputAppendRenameColNames	可选，输入表在输出表追加的列名的别名，在inputAppendColNames不为空时有效	-	不使用别名
mapAppendColNames	可选，映射表在输出表追加的列名	-	不追加
mapAppendRenameColNames	可选，映射表在输出表追加的列名的别名	-	不使用别名
inputTablePartitions	可选，输入表选中的分区	-	选择全表
mapTablePartitions	可选，映射表中的分区	-	选择全表
outputColName	可选，输出表中相似度列的列名, 列名中不能有特殊字符，只能用英文的a-z，A-Z及数字和下划线_，且以字母开头，名称的长度不超过128字节。	-	output
method	可选，相似度计算方法	levenshtein_sim, lcs_sim, ssk, cosine, simhash_hamming_sim	levenshtein_sim
lambda	可选，匹配字符串的权重，ssk中可用	(0, 1)	0.5
k	可选，子串的长度，ssk和cosine中可用	(0, 100)	2
topN	可选，最终给出的相似度最大值的个数	(0, +∞)	10
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期
coreNum	可选，计算的核心数	正整数	系统自动分配
memSizePerCore	可选，每个核心的内存（单位为兆）	正整数，范围(0, 65536)	系统自动分配

示例

测试数据

```
create table pai_ft_string_similarity_topn_input as select * from
(select 0 as id, "北京" as col0 from dual
union all
select 1 as id, "北京上海" as col0 from dual
union all
```

```
select 2 as id, "北京上海香港" as col0 from dual
)tmp;
```

pai命令

```
PAI -name string_similarity_topn
-project sre_mpi_algo_dev
-DinputTableName=pai_ft_string_similarity_topn_input
-DmapTableName=pai_ft_string_similarity_topn_input
-DoutputTableName=pai_ft_string_similarity_topn_output
-DinputSelectedColName=col0
-DmapSelectedColName=col0
-DinputAppendColNames=col0
-DinputAppendRenameColNames=input_col0
-DmapAppendColNames=col0
-DmapAppendRenameColNames=map_col0
-Dmethod=simhash_hamming_sim;
```

输出说明

input_col0	map_col0	output
北京	北京	1
北京	北京上海	0.90625
北京	北京上海香港	0.796875
北京上海	北京上海	1
北京上海	北京	0.90625
北京上海	北京上海香港	0.828125
北京上海香港	北京上海香港	1
北京上海香港	北京上海	0.828125
北京上海香港	北京	0.796875

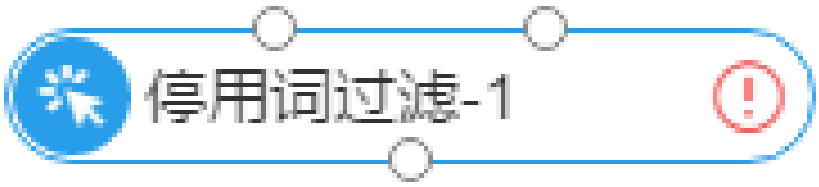
停用词过滤

功能介绍

停用词过滤，是文本分析中一个预处理方法。它的功能是过滤分词结果中的噪声（例如：的、是、啊等）。

参数设置

组件说明



两个输入桩，从左到右依次为：

- 输入表，即需要过滤的分词结果表；对应的参数名为inputTableName
- 停用词表，表的格式为一列，每行为一个停用词；对应的参数名为noiseTableName

参数界面说明

- 可以选择需要过渡的列

执行调化说明

- 可以自己配置并发计算核心数目与内存，默认系统自动分配

PAI命令

```
PAI -name FilterNoise -project algo_public \
-DinputTableName=”test_input” -DnoiseTableName=”noise_input” \
-DoutputTableName=”test_output” \
-DselectedColNames=”words_seg1,words_seg2” \
-Dlifecycle=30
```

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
noiseTableName	必选，停用词表	格式为一列，每一行一个词	-
noiseTablePartitions	可选，停用词表的分区		全表
outputTableName	必选，输出表	-	-
selectedColNames	必选，待过滤列，多列时以逗号为分隔	-	-
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期
coreNum	可选，计算的核心数	正整数	系统自动分配
memSizePerCore	可选，每个核心的内存	正整数，范围(0,	系统自动分配

	（单位为兆）	65536)	

示例

源数据

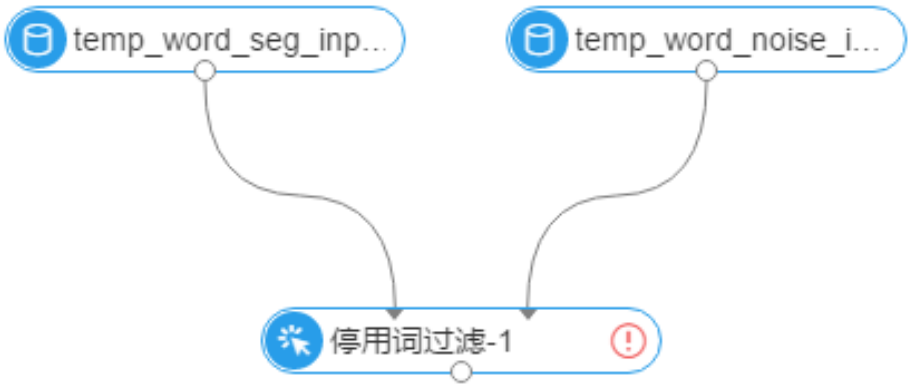
- 分词的结果表 temp_word_seg_input

doc_idx	s_idx	seg
1	1	在 金融 危机 最 严重 时，在 美国 股市 持续 暴跌 期间，2008年 10月 17日 巴菲特 在 《 纽约 时报 》 发表 文章 罕见 地 公开 宣称 " 我 正在 买入 美国 股票 "
1	2	股市 暴跌，别人 都 恐惧 得 要 死，纷纷 抛售
1	3	巴菲特 的 伯克希尔 公司 股票 投资 超过 600亿 美元，市值 也 是 大幅 下跌
1	4	他 不但 不 抛 反而 大量 买入 加上 收购 累计 投资 超过 500亿 美元
1	5	为什么 呢？这 就是 巴菲特 近 50 年 来 长期 业绩 远 远 战胜 市场 的 秘诀：反向 思考，反向 投资，而且 长期 坚持 如此
2	1	娱乐圈 有 很 多 明星 是 一夜 走红，比如 巩俐，她 尚 是 大 二 学生 时 就 在 1987年 拍摄 的 《 红高粱 》 而 一举 成为 国际 巨星
2	2	再 比如 很 多 明星 们 家境 很 好，出来 打拼 娱乐圈 无非 是 为了 兴趣，比如 tvb 的 何超仪，郭可盈，以及 内地 的 刘亦菲
2	3	然而 娱乐圈 也 有 很 多 明星 她们 却 并非 是 一夜 成名，也 非 家境 殷实，而是 一 步步 的 打拼 到 现在

停用词表 temp_word_noise_input

noise_word ▲
在
地
时
他
是
比如
而是
的

1 创建实验



2 选择待过滤列 seg

3 运行结果

doc_idx	s_idx	seg
1	1	金融 危机 最 严重 美国 股市 持续 暴跌 期间 2008年 10月 17日 巴菲特 纽约 时报 发表 文章 罕见 公开 宣称 我 正在 买入 美国 股票
1	2	股市 暴跌 别人 都 恐惧 得 要 死 纷纷 抛售
1	3	巴菲特 伯克希尔 公司 股票 投资 超过 600亿 美元 市值 大幅 下跌
1	4	不但 不 抛 反而 大量 买入 加上 收购 累计 投资 超过 500亿 美元
1	5	为什么 呢？这 就是 巴菲特 近 50 年 来 长期 业绩 远 远 战胜 市场 秘诀 : 反向 思考 反向 投资 而且 长期 坚持 如此
2	1	娱乐圈 有 很 多 明星 一夜 走红 巩俐 她 尚 大 二 学生 就 1987年 拍摄 红高粱 而 一举 成为 国际 巨星
2	2	再 很 多 明星 们 家境 很 好 出来 打拼 娱乐圈 无非 为了 兴趣 tvb 何超仪 郭可盈 以及 内地 刘亦菲
2	3	然而 娱乐圈 有 很 多 明星 她们 却 并非 一夜 成名 非 家境 殷实 一 步步 打拼 到 现在

文本摘要

所谓自动文摘就是利用计算机自动地从原始文献中提取文摘，文摘是全面准确地反映某一文献中心内容地简单连贯的短文。本算法基于TextRank，通过提取文档中已存在的句子形成摘要。

具体算法原理可以参考文章TextRank: Bringing Order into Texts

PAI命令行

```
PAI -name TextSummarization
-project algo_public
–DinputTableName="test_input"
-DoutputTableName="test_output"
-DdocIdCol="doc_id"
-DsentenceCol="sentence"
-DtopN=2
-Dlifecycle=30;
```

参数说明

参数名称	参数描述	参数可选项	默认值
inputTableName	必选，输入表名	-	-
inputTablePartitions	可选，输入表中指定参与计算的分区	-	输入表的所有partition
outputTableName	必选，输出表名	-	-
docIdCol	必选，标识文章id的列名	-	-
sentenceCol	必选，句子列，仅可指定一列	-	-
topN	可选，输出前多个关键句	-	3
similarityType	可选，句子相似度的计算方法	lcs_sim, levenshtein_sim, cosine, ssk	lcs_sim
lambda	可选，匹配字符串的权重，ssk中可用	(0, 1)	0.5

k	可选，子串的长度，ssk和cosine中可用	(0, 100)	2
dampingFactor	可选，阻尼系数	(0, 1)	0.85
maxIter	可选，最大迭代次数	[1, +]	100
epsilon	可选，收敛系数	(0, 正无穷)	0.000001
lifecycle	可选，输入出表的生命周期	正整数	无生命周期
coreNum	可选，参与计算的核心数	正整数	系统自动计算
memSizePerCore	可选，每个核心需要的内存	正整数	系统自动计算

句子相似度的可选项如下：（符号说明：A, B表示两个字符串，len(A)表示字符串A的长度）

- **lcs_sim**: 公式 1.0 - (最长公共子串(Longest Common Subsequence)的长度)/max(len(A), len(B))
- **levenshtein_sim**: 公式 1.0 - (编辑距离(Levenshtein Distance)) / max(len(A), len(B))
- **cosine**: 参考Lodhi, Huma; Saunders, Craig; Shawe-Taylor, John; Cristianini, Nello; Watkins, Chris (2002). “Text classification using string kernels”. Journal of Machine Learning Research: 419–444.
- **ssk**: 参考Leslie, C.; Eskin, E.; Noble, W.S. (2002), The spectrum kernel: A string kernel for SVM protein classification 7, pp. 566–575

输出格式说明

输出表为两列，分别是doc_id, abstract。示例如下：

doc_id	abstract
1000894	早在2008年，上交所便发布了上市公司社会责任披露相关指引，强制要求三类公司披露社会责任报告，同时鼓励其他有条件的上市公司进行自愿披露。统计显示，2012年，沪市上市公司共计379家披露社会责任报告，包括强制披露公司305家和自愿披露公司74家，合计占沪市全部上市公司的40%。胡汝银表示，下一步上交所将探索扩大社会责任报告的披露范围，修订细化有关社会责任报告披露的指引，并鼓励更多的机构推进社会责任产品创新。

文章相似度

文章相似度是在字符串相似度的基础上，基于词，计算两两文章或者句子之间的相似度，文章或者句子需要以空格分割的文本，计算方式和字符串相似度类似。

Levenshtein（Levenshtein Distance）支持距离和相似度两个参数，相似度=1-距离，距离在参数中表示为levenshtein，相似度在参数中表示为levenshtein_sim。

lcs（Longest Common SubString）支持距离和相似度两个参数，相似度=1-距离，距离在参数中表示为lcs，相似度在参数中表示为lcs_sim。

ssk（String Subsequence Kernel）支持相似度计算，在参数中表示为ssk。

参考：Lodhi, Huma; Saunders, Craig; Shawe-Taylor, John; Cristianini, Nello; Watkins, Chris (2002). “Text classification using string kernels”. Journal of Machine Learning Research: 419–444.

cosine（Cosine）支持相似度计算，在参数中表示为cosine。

参考：Leslie, C.; Eskin, E.; Noble, W.S. (2002), The spectrum kernel: A string kernel for SVM protein classification 7, pp. 566–575

simhash_hamming，其中SimHash算法是把原始的文本映射为64位的二进制指纹，HammingDistance则是计算二进制指纹在相同位置上不同的字符的个数，支持距离和相似度两个参数，相似度=1-距离/64.0，距离在参数中表示为simhash_hamming，相似度在参数中表示为simhash_hamming_sim。

SimHash详细介绍请见pdf；

HammingDistance详细介绍请见维基百科链接wiki

PAI命令行

```
PAI -name doc_similarity
-project algo_public
-DinputTableName="pai_test_doc_similarity"
-DoutputTableName="pai_test_doc_similarity_output"
-DinputSelectedColName1="col0"
-DinputSelectedColName2="col1"
```

参数说明

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	-	-
outputTableName	必选，输出表的表名	-	-
inputSelectedColName1	可选，相似度计算中第一列的列名	-	表中第一个为类型为string的列名
inputSelectedColName2	可选，相似度计算中第二列的列名	-	表中第二个为类型为string的列名
inputAppendColNames	可选，输出表追加的列名	-	不追加
inputTablePartitions	可选，输入表选中的分区	-	选择全表
outputColName	可选，输出表中相似度列的列名。列名中不能有特殊字符，只能用英文的a-z，A-Z及数字和下划线_，且以字母开头，名称的长度不超	-	output

	过128字节。		
method	可选，相似度计算方法	levenshtein, levenshtein_sim, lcs, lcs_sim, ssk, cosine, simhash_hamming, simhash_hamming_sim	levenshtein_sim
lambda	可选，匹配词组合的权重，ssk中可用	(0, 1)	0.5
k	可选，子串的长度，ssk和cosine中可用	(0, 100)	2
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期
coreNum	可选，计算的核心数	正整数	系统自动分配
memSizePerCore	可选，每个核心的内存（单位为兆）	正整数，范围(0, 65536)	系统自动分配

具体示例

数据生成

```
drop table if exists pai_doc_similarity_input;
create table pai_doc_similarity_input as
select * from
(
select 0 as id, "北京 上海" as col0, "北京 上海" as col1 from dual
union all
select 1 as id, "北京 上海" as col0, "北京 上海 香港" as col1 from dual
)tmp
```

PAI命令行

```
drop table if exists pai_doc_similarity_output;
PAI -name doc_similarity
-project algo_public
-DinputTableName=pai_doc_similarity_input
-DoutputTableName=pai_doc_similarity_output
-DinputSelectedColName1=col0
-DinputSelectedColName2=col1
-Dmethod=levenshtein_sim
-DinputAppendColNames=id,col0,col1;
```

输入说明 pai_doc_similarity_input

id	col0	col1
1	北京 上海	北京 上海 香港
0	北京 上海	北京 上海

输出说明 pai_doc_similarity_output

id	col0	col1	output
1	北京 上海	北京 上海 香港	0.6666666666666667
0	北京 上海	北京 上海	1.0

常见问题

这里计算相似度是基于分词的结果，即以空格分割的每个词作为相似度计算的一个单位。如果是以字符串整体输入的话，可以使用字符串相似度方法

参数method中，levenshtein、lcs、simhash_hamming为计算距离。levenshtein_sim、lcs_sim、ssk、cosine、simhash_hamming_sim为计算相似度。距离=1.0-相似度。

在cosine和ssk中，存在参数k，其意义为以k个词作为一个组合，进行相似度计算，所以这里会存在，k大于词的个数的情况下，相似度输出为0，即使两个字符串是一样的。这种情况下，调小k等于最小词个数即可

句子拆分

组件介绍

将一段文本，按标点进行句子拆分。

该组件主要用于文本摘要前的预处理，将一段文本拆分成一句一行的形式。

PAI命令行

```
PAI -name SplitSentences
-project algo_public
-DinputTableName="test_input"
-DoutputTableName="test_output"
-DdocIdCol="doc_id"
-DdocContent="content"
-Dlifecycle=30
```

参数说明

参数名称	参数描述	参数可选项	默认值
inputTableName	必选，输入表名	-	-
inputTablePartitions	可选，输入表中指定参与计算的分区	-	输入表的所有partition
outputTableName	必选，输出表名	-	-
docIdCol	必选，标识文章id的列名	-	-

docContent	必选，标示文章内容的列名，仅可指定一列	-	-
delimiter	可选，句子的间隔字符集合	-	。！？
lifecycle	可选，输入出表的生命周期	正整数	无生命周期
coreNum	可选，参与计算的核心数	正整数	系统自动计算
memSizePerCore	可选，每个核心需要的内存	正整数	系统自动计算

输出格式说明

输出表为两列，分别是doc_id, sentence。示例如下：

doc_id	sentence
1000894	早在2008年，上交所便发布了上市公司社会责任披露相关指引，强制要求三类公司披露社会责任报告，同时鼓励其他有条件的上市公司进行自愿披露。
1000894	统计显示，2012年，沪市上市公司共计379家披露社会责任报告，包括强制披露公司305家和自愿披露公司74家，合计占沪市全部上市公司的40%。

条件随机场

条件随机场（conditional random field, CRF）是给定一组输入随机变量条件下另一组输出随机变量的条件概率分布模型，其特点是假设输出随机变量构成马尔可夫随机场。条件随机场可以用于不同的预测问题，主要应用到标注问题中，其中线性链（linear chain）条件随机场是最典型的。

条件随机场的详细介绍请参见维基百科链接wiki

PAI命令示例

```
PAI -name=linearcrf
-project=algo_public
-DinputTableName=crf_input_table
-DidColName=sentence_id
-DfeatureColNames=word,f1
-DlabelColName=label
-DoutputTableName=crf_model
-Dlifecycle=28
-DcoreNum=10
```

算法参数

参数名称	参数描述	参数值可选项	默认值

inputTableName	必选，输入特征数据表	-	-
inputTablePartitions	可选，输入特征表选择的分区	-	默认选择全表
featureColNames	可选，输入表选择的特征列	-	默认选择全部，自动排除label列
labelColName	必选，目标列	-	-
idColName	必选，样本标号列	-	-
outputTableName	必选，输出模型表	-	-
outputTablePartitions	可选，输出模型表选择的分区	-	默认选择全表
template	可选，算法特征生成的模板	模板定义格式如下所示	模板默认值如下所示
freq	可选，过滤特征的参数，算法只保留出现次数大于等于freq的特征	正整数	默认为1
iterations	可选，优化的最大迭代次数	-	默认为100
l1Weight	可选，L1正则的参数权重	-	默认为1.0
l2Weight	可选，L2正则的参数权重	-	默认为1.0
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0.0001
lbfgsStep	可选，lbfgs优化过程中的历史长度，仅对lbfgs有效	-	默认为10
threadNum	可选，模型训练时并行启动线程的数量	-	默认为3
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

特征模板template定义

```
<template .=. <template_item,<template_item,...,<template_item
<template_item .=. [row_offset:col_index]/[row_offset:col_index]/.../[row_offset:col_index]
row_offset .=. integer
col_index .=. integer
```

算法默认template

```
[-2:0],[-1:0],[0:0],[1:0],[2:0],[-1:0]/[0:0],[0:0]/[1:0],[-2:1],[-1:1],[0:1],[1:1],[2:1],[-2:1]/[-1:1],[-1:1]/[0:1],[0:1]/[1:1],[1:1]/[2:1],[-2:1]/[-1:1]/[0:1],[-1:1]/[0:1]/[1:1],[0:1]/[1:1]/[2:1]
```

数据示例

sentence_id	word	f1	label
1	Rockwell	NNP	B-NP
1	International	NNP	I-NP
1	Corp	NNP	I-NP
1	‘s	POS	B-NP
…	…	…	…
823	Ohio	NNP	B-NP
823	grew	VBD	B-VP
823	3.8	CD	B-NP
823	%	NN	I-NP
823	.	.	O

预测算法PAI命令示例

```
PAI -name=crf_predict
-project=algo_public
-DinputTableName=crf_test_input_table
-DmodelTableName=crf_model
-DidColName=sentence_id
-DfeatureColNames=word,f1
-DlabelColName=label
-DoutputTableName=crf_predict_result
-DdetailColName=prediction_detail
-Dlifecycle=28
-DcoreNum=10
```

预测算法参数

参数名称	参数描述	参数值可选项	默认值
inputTableName	必选，输入特征数据表	-	-
inputTablePartitions	可选，输入特征表选择的分区	-	默认选择全表
featureColNames	可选，输入表选择的特征列	-	默认选择全部，自动排除label列
labelColName	可选，目标列	-	-

IdColName	必选，样本标号列	-	-
resultColName	可选，输出表中result列名	-	默认为prediction_result
scoreColName	可选，输出表中score列名	-	默认为prediction_score
detailColName	可选，输出表中detail列名	-	默认为空
outputTableName	必选，输出预测结果表	-	-
outputTablePartitions	可选，输出预测结果表选择的分区	-	默认选择全表
modelTableName	必选，算法模型表	-	-
modelTablePartitions	可选，算法模型表选择的分区	-	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

测试数据

sentence_id	word	f1	label
1	Confidence	NN	B-NP
1	in	IN	B-PP
1	the	DT	B-NP
1	pound	NN	I-NP
...	...	...	...
77	have	VBP	B-VP
77	announced	VBN	I-VP
77	similar	JJ	B-NP
77	increases	NNS	I-NP
77	.	.	O

说明: label列可以没有

关键词抽取

关键词抽取是自然语言处理中的重要技术之一，具体是指从文本里面把跟这篇文章意义最相关的一些词抽取出

来。本算法基于TextRank，它受到网页之间关系PageRank算法启发，利用局部词汇之间关系（共现窗口）构建网络，计算词的重要性，选取权重大的做为关键词。

PAI命令行

```
PAI -name KeywordsExtraction
-DinputTableName=maple_test_keywords_basic_input
-DdocIdCol=docid -DdocContent=word
-DoututTableName=maple_test_keywords_basic_output
-DtopN=19;
```

参数说明

参数名称	参数描述	取值范围	是否必选，默认值
inputTableName	输入表	表名	必选
inputTablePartitions	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		可选， 默认值选择所有分区
outputTableName	输出表名	必选	
docIdCol	标识文章id的列名，仅可指定一列	必选	
docContent	Word列,仅可指定一列	必选	
topN	输出前多少个关键词，当关键词个数小于全部词个数时，全部输出	可选5	
windowSize	TextRank算法的窗口大小	可选2	
dumpingFactor	TextRank算法的阻尼系数	可选0.85	
maxIter	TextRank算法的最大迭代次数	可选100	
epsilon	TextRank算法的收敛残差阈值	可选0.000001	
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期
coreNum	节点个数	与参数memSizePerCore配对使用，正整数，范围[1, 9999] 详细介绍	可选， 默认自动计算

memSizePerCore	单个节点内存大小，单位M	正整数，范围[1024, 64*1024] 详细介绍	可选， 默认自动计算

具体示例

数据生成输入表需采用空格分词，并过滤掉停用词（类似“的”、“地”、“得”、“了”、“个”）和所有的标点符号。

docid:string	word:string
doc0	翼身融合 飞机 是 未来 航空 领域 发展 一个 新 方向 国内外 诸多 研究 机构 已经 开展 对翼身融合 飞机 研究 而 其 全自动 外形 优化 算法 已 成为 新 研究 热点 国内 外 现有 成果 基础 之上 分析 比较 常用 建模 求解 平台 使用 方式 及 特点 设计 编写 翼身融合 飞机 外形 优化 几何 建模 网格 划分 流场 求解 外形 优化 模块 比 较 不同 算法 间 优劣 实现 翼身融合 飞机 概念设计 中 外形 优化 几何 建模 及 网格 生成 模块 实现 基于 超限 插值 网格 生成 算法 基于 样条 曲线 建模 方法 流场 求解 模块 包括 有限 差分 求解器 有限元 求解器和面元法 求解器 其中 有限 差分 求解器 主要 包括 基于 有限 差分法 势流 数学 建模 基于 笛卡尔 网格 变 步长 差分 格式 推导 笛卡尔 网格 生成 索引 算法 基于 笛卡尔 网格 诺 依曼 边界条件 表达 形式 推导 实现 基于 有限 差分 求解器 二维 翼型 气动 参数 计算 算例 有限元 求解器 主要 包括 基于 变分 原理 势流 有限元 理论 建模 二维 有限元 库塔 条件 表达式 推导 基于 最小 二乘 速度 求解 算法 设计 基于 Gmsh 二维 带尾迹 翼型 空间 网格 生成器 开发 实现 基于 有限元 求解器 二维 翼型 气动 参数 计算 算例 面元法 求解器 主要 包括 基于 面元法 势流 理论 建模 自动 尾迹 生成 算法 设计 基于 面元法 三维 翼身融合 体 流场 求解器 开发 基于 布拉 修斯 平板 解 阻力 估算 算法 设计 求解器 Fortran 语言 上 移 植 Python 和 Fortran 代码 混编 基于 OpenMP 和 CUDA 并行 加速 算法 设计 与 开发 实现 基于 面元法 求解器 三维 翼身融合 体 气动 参数 计算 算例 外形 优化 模块 实 现了 基于 自由 形状 变形 网格 变形 算法 遗传算法 差分 进化 算法 飞机 表面积 计算 算法 基于 矩 积分 飞机 体积 计算 算法 开发 基于 VTK 数据 可视化 格式 工具

PAI命令行

```
PAI -name KeywordsExtraction
-DinputTableName=maple_test_keywords_basic_input
-DdocIdCol=docid -DdocContent=word
-DoututTableName=maple_test_keywords_basic_output
-DtopN=19;
```

输入说明 输出说明

- 输出表

docid	keywords	weight
doc0	基于	0.041306752223538405
doc0	算法	0.03089845626854151
doc0	建模	0.021782865850562882
doc0	网格	0.020669749212693957
doc0	求解器	0.020245609506360847
doc0	飞机	0.019850761705313365
doc0	研究	0.014193732541852615
doc0	有限元	0.013831122054200538
doc0	求解	0.012924593244133104
doc0	模块	0.01280216562287212
doc0	推导	0.011907588923852495
doc0	外形	0.011505456605632607
doc0	差分	0.011477831662367547
doc0	势流	0.010969269350293957
doc0	设计	0.010830986516637251
doc0	实现	0.010747536556701583
doc0	二维	0.010695570768457084
doc0	开发	0.010527342662670088
doc0	新	0.010096978306668461

算法规模

新浪数据集，1080162篇文章，起30个结点，10分钟

ngram-count

ngram-count是语言模型中的其中一个步骤，完成的是基于词的基础上生成n-gram，并统计在全部语料集上，对应n-gram的个数。生成的是全局的个数，并不是单个文档的个数。 可以参考ngram-count，功能上是这个工具的子集

PAI命令行

```
PAI -name ngram_count
-project algo_public
```

```
-DinputTableName=pai_ngram_input
-DoutputTableName=pai_ngram_output
-DinputSelectedColNames=col0
-DweightColName=weight
-DcoreNum=2
-DmemSizePerCore=1000;
```

参数说明

参数名称	参数描述	参数可选项	默认值
inputTableName	必选，输入表	-	-
outputTableName	必选，输出表	-	-
inputSelectedColNames	可选，输入表选择列	-	第一个字符类型的列
weightColName	可选，权重列名	-	默认权重为1
inputTablePartitions	可选，输入表指定分区	-	默认选择全表
countTableName	可选，ngram-count以往的输出表，最终结果会merge这张表	-	-
countWordColName	可选，count表中词所在的列名	-	默认选择第二列
countCountColName	可选，count表中count所在的列	-	默认选择第三列
countTablePartitions	可选，count表指定分区	-	-
vocabTableName	可选，词袋表, 不在词袋中的词在结果中会被标识为\<unk\	-	-
vocabSelectedColName	可选，词袋所在的列名	-	默认选择第一个字符类型的列
vocabTablePartitions	可选，词袋表指定分区	-	-
order	可选，N-grams的最大长度	-	默认为3
lifecycle	可选，输出表的生命周期	-	-
coreNum	可选，核心个数	-	-
memSizePerCore	可选，单个核心使用的内存数	-	-

语义向量距离

基于算法语义向量结果（如word2vec生成的词向量），计算给定的词（或者句子）的扩展词（或者句子），即

计算其中某一向量距离最近的向量集合。其中一个用法是：基于word2vec生成的词向量结果，根据输入的词返回最为相似的词列表。

PAI命令行

```
PAI -name SemanticVectorDistance -project algo_public
-DinputTableName="test_input"
-DoutputTableName="test_output"
-DidColName="word"
-DvectorColNames="f0,f1,f2,f3,f4,f5"
-Dlifecycle=30
```

参数说明

参数名称	参数描述	参数可选项	默认值
inputTableName	必选，输入表名	-	-
inputTablePartitions	可选，输入表中指定参与计算的分区	-	输入表的所有partition
outputTableName	必选，输出表名	-	-
idTableName	可选，需要计算相近向量的id的列表所在表名，格式为一列，每一行一个id，默认为空，即input表中的所有向量参与计算	-	-
idTablePartitions	可选，id表中参与计算的分区列表，默认为所有分区	-	-
idColName	必选，id所在列名	-	3
vectorColNames	可选，向量的列名列表，如f1,f2,...	-	-
topN	可选，输出的距离最近的向量的数目	[1, 正无穷]	5
distanceType	可选，距离的计算方式	euclidean、cosine、manhattan	euclidean
distanceThreshold	可选，距离的阈值，当两个向量的距离小于此值时输出	(0, 正无穷)	正无穷大
lifecycle	可选，输入出表的生命周期	正整数	无生命周期
coreNum	可选，参与计算的核心数	正整数	系统自动计算
memSizePerCore	可选，每个核心需要的内存	正整数	系统自动计算

具体示例

输出表为四列，分别是original_id, near_id, distance, rank。示例如下：

original_id	near_id	distance	rank
hello	hi	0.2	1
hello	xxx	xx	2
Man	Woman	0.3	1
Man	xx	xx	2
..	...	...	...

条件随机场

条件随机场（conditional random field, CRF）是给定一组输入随机变量条件下另一组输出随机变量的条件概率分布模型，其特点是假设输出随机变量构成马尔可夫随机场。条件随机场可以用于不同的预测问题，主要应用到标注问题中，其中线性链（linear chain）条件随机场是最典型的。

条件随机场的详细介绍请参见维基百科链接wiki

PAI命令示例

```
PAI -name=linearcrf
-project=algo_public
-DinputTableName=crf_input_table
-DidColName=sentence_id
-DfeatureColNames=word,f1
-DlabelColName=label
-DoutputTableName=crf_model
-Dlifecycle=28
-DcoreNum=10
```

算法参数

参数名称	参数描述	参数值可选项	默认值
inputTableName	必选，输入特征数据表	-	-
inputTablePartitions	可选，输入特征表选择的分区	-	默认选择全表
featureColNames	可选，输入表选择的特征列	-	默认选择全部，自动排除label列
labelColName	必选，目标列	-	-
idColName	必选，样本标号列	-	-
outputTableName	必选，输出模型表	-	-
outputTablePartition	可选，输出模型表选择	-	默认选择全表

s	的分区		
template	可选，算法特征生成的模板	模板定义格式如下所示	模板默认值如下所示
freq	可选，过滤特征的参数，算法只保留出现次数大于等于freq的特征	正整数	默认为1
iterations	可选，优化的最大迭代次数	-	默认为100
l1Weight	可选，L1正则的参数权重	-	默认为1.0
l2Weight	可选，L2正则的参数权重	-	默认为1.0
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0.0001
lbfgsStep	可选，lbfgs优化过程中的历史长度，仅对lbfgs有效	-	默认为10
threadNum	可选，模型训练时并行启动线程的数量	-	默认为3
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

特征模板template定义

```
<template> .=. <template_item>,<template_item>,...,<template_item>
<template_item> .=. [row_offset:col_index]/[row_offset:col_index]/.../[row_offset:col_index]
row_offset .=. integer
col_index .=. integer
```

算法默认template

```
[-2:0],[-1:0],[0:0],[1:0],[2:0],[-1:0]/[0:0],[0:0]/[1:0],[-2:1],[-1:1],[0:1],[1:1],[2:1],[-2:1]/[-1:1],[-1:1]/[0:1],[0:1]/[1:1],[1:1]/[2:1],[-2:1]/[-1:1]/[0:1],[-1:1]/[0:1]/[1:1],[0:1]/[1:1]/[2:1]
```

数据示例

训练数据

sentence_id	word	f1	label
1	Rockwell	NNP	B-NP

1	International	NNP	I-NP
1	Corp	NNP	I-NP
1	‘s	POS	B-NP
…	…	…	…
823	Ohio	NNP	B-NP
823	grew	VBD	B-VP
823	3.8	CD	B-NP
823	%	NN	I-NP
823	.	.	O

预测算法PAI命令示例

```
PAI -name=crf_predict
-project=algo_public
-DinputTableName=crf_test_input_table
-DmodelTableName=crf_model
-DidColName=sentence_id
-DfeatureColNames=word,f1
-DlabelColName=label
-DoutputTableName=crf_predict_result
-DdetailColName=prediction_detail
-Dlifecycle=28
-DcoreNum=10
```

预测算法参数

参数名称	参数描述	参数值可选项	默认值
inputTableName	必选，输入特征数据表	-	-
inputTablePartitions	可选，输入特征表选择的分区	-	默认选择全表
featureColNames	可选，输入表选择的特征列	-	默认选择全部，自动排除label列
labelColName	可选，目标列	-	-
IdColName	必选，样本标号列	-	-
resultColName	可选，输出表中result列名	-	默认为prediction_result
scoreColName	可选，输出表中score列名	-	默认为prediction_score
detailColName	可选，输出表中detail列名	-	默认为空
outputTableName	必选，输出预测结果表	-	-

outputTablePartitions	可选，输出预测结果表选择的分区	-	默认选择全表
modelTableName	必选，算法模型表	-	-
modelTablePartitions	可选，算法模型表选择的分区	-	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

测试数据

sentence_id	word	f1	label
1	Confidence	NN	B-NP
1	in	IN	B-PP
1	the	DT	B-NP
1	pound	NN	I-NP
...	...	...	...
77	have	VBP	B-VP
77	announced	VBN	I-VP
77	similar	JJ	B-NP
77	increases	NNS	I-NP
77	.	.	O

说明: label列可以没有

PMI

互信息(Mutual Information)是信息论里一种有用的信息度量，它可以看成是一个随机变量中包含的关于另一个随机变量的信息量，或者说是一个随机变量由于已知另一个随机变量而减少的不肯定性。本算法统计若干文章中所有词的共现情况，计算两两之间的PMI（point mutual information）。互信息定义为：

```
PMI(x,y)=ln(p(x,y)/(p(x)p(y)))=ln(#(x,y)D/(#x#y))
```

其中，#(x,y)为pair(x,y)的count数，D为pair的总数。若x、y在同一个窗口出现，那么#x+=1;#y+=1;#(x,y)+=1。

PAI命令行

```
PAI -name PointwiseMutualInformation
-project algo_public
-DinputTableName=maple_test_pmi_basic_input
-DdocColName=doc
-DoutputTableName=maple_test_pmi_basic_output
-DminCount=0
-DwindowSize=2
-DcoreNum=1
-DmemSizePerCore=110;
```

参数说明

参数名称	参数描述	取值范围	是否必选，默认值
inputTableName	输入表	表名	必选
outputTableName	输出表	表名	必选
docColName	分词好的文档列名，分词用空格隔开	列名	必选
windowSize	窗口大小，例如5指当前词右边相邻的5个词（不包含当前词）。在窗口中出现的词被被认为与当前词相关。	[1, 句子长度]	可选，默认整行doc
minCount	截断的最小词频，出现次数少于该值的词会被过滤掉	[0, 2e63]	可选，默认5
inputTablePartitions	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		可选， 默认值选择所有分区
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期
coreNum	节点个数	与参数memSizePerCore配对使用，正整数，范围[1, 9999]	可选， 默认自动计算
memSizePerCore	单个节点内存大小，单位M	正整数，范围[1024, 64*1024]	可选， 默认自动计算

具体示例

数据生成

doc:string
w1 w2 w3 w4 w5 w6 w7 w8 w8 w9
w1 w3 w5 w6 w9
w0
w0 w0
w9 w1 w9 w1 w9

PAI命令行

```
PAI -name PointwiseMutualInformation
-project algo_public
-DinputTableName=maple_test_pmi_basic_input
-DdocColName=doc
-DoutputTableName=maple_test_pmi_basic_output
-DminCount=0
-DwindowSize=2
-DcoreNum=1
-DmemSizePerCore=110;
```

输入说明输出说明

- 输出表

word1	word2	word1_count	word2_count	co_occurrences_count	pmi
w0	w0	2	2	1	2.0794415416798357
w1	w1	10	10	1	-1.1394342831883648
w1	w2	10	3	1	0.06453852113757116
w1	w3	10	7	2	-0.08961215868968704
w1	w5	10	8	1	-0.916290731874155
w1	w9	10	12	4	0.06453852113757116

w2	w3	3	7	1	0.4212134650763035
w2	w4	3	4	1	0.9808292530117262
w3	w4	7	4	1	0.13353139262452257
w3	w5	7	8	2	0.13353139262452257
w3	w6	7	7	1	-0.42608439531090014
w4	w5	4	8	1	0
w4	w6	4	7	1	0.13353139262452257
w5	w6	8	7	2	0.13353139262452257
w5	w7	8	4	1	0
w5	w9	8	12	1	-1.0986122886681098
w6	w7	7	4	1	0.13353139262452257
w6	w8	7	7	1	-0.42608439531090014
w6	w9	7	12	1	-0.9650808960435872
w7	w8	4	7	2	0.8266785731844679
w8	w8	7	7	1	-0.42608439531090014
w8	w9	7	12	2	-0.2719337154836418
w9	w9	12	12	2	-0.8109302162163288

目录

k-Core

单源最短路径

PageRank

标签传播聚类

标签传播分类

Modularity

最大联通子图

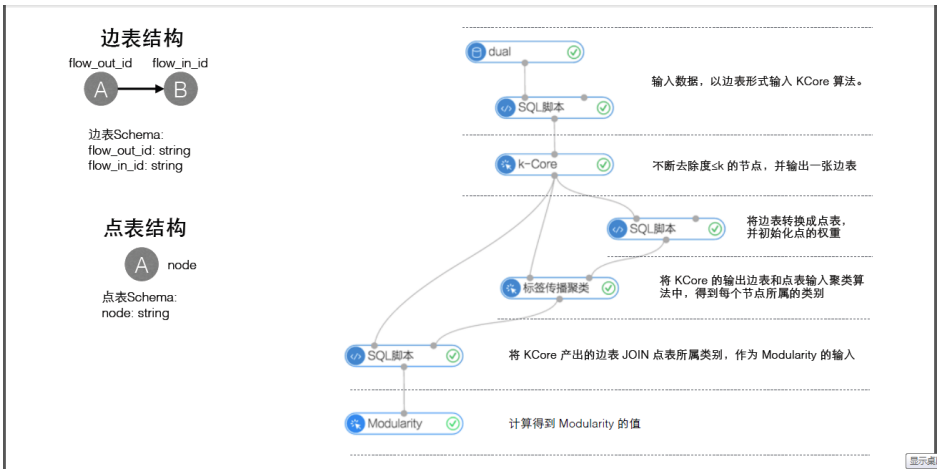
点聚类系数

边聚类系数

计数三角形

树深度

网络分析栏提供的都是基于Graph数据结构的分析算法；下图是使用平台网络分析组件构建的一个分析流程实例：



网络分析栏的算法组件都需要设置运行参数，参数说明如下：进程数：参数代号workerNum，用于设置作业并行执行的节点数；数字越大并行度越高，但框架通讯开销会增大。进程内存：参数代号workerMem，用于设置单个 worker可使用的最大内存量，默认每个worker分配4096内存；实际使用内

存超过该值，会抛出OutOfMemory异常。

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

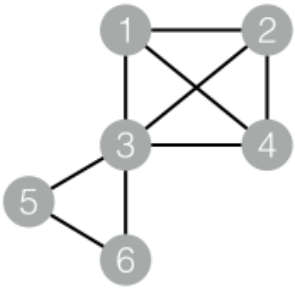
实例

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

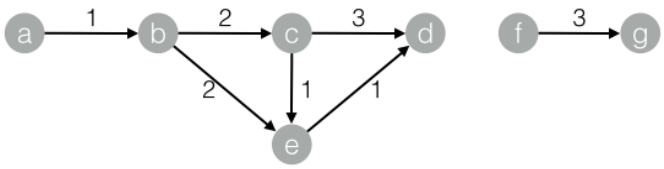
实例

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：

$$W(A) = (1-d) + d * (\sum_{i} W(i) * C(Ai))$$

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

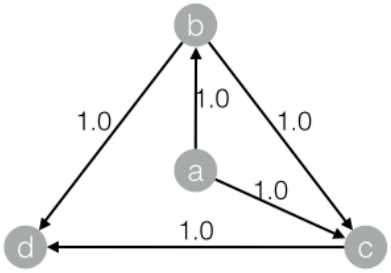
实例

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

标签传播聚类

功能介绍

图聚类是根据图的拓扑结构，进行子图的划分，使得子图内部节点的链接较多，子图之间的连接较少。标签传播算法（Label Propagation Algorithm, LPA）是基于图的半监督学习方法，其基本思路是节点的标签（community）依赖其邻居节点的标签信息，影响程度由节点相似度决定，并通过传播迭代更新达到稳定。

参数介绍

最大迭代次数：选填，默认30

实例

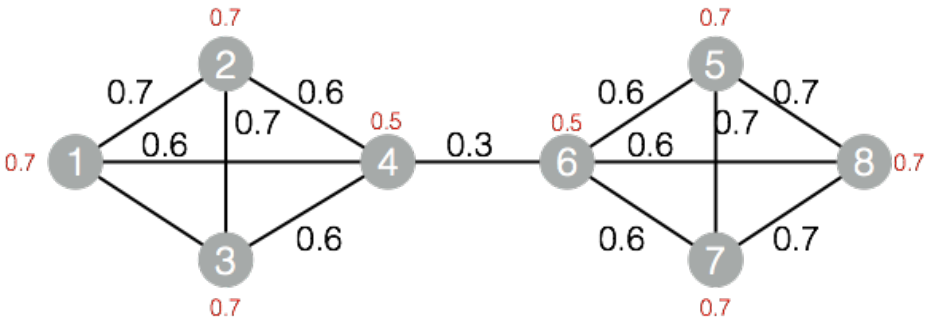
测试数据

数据生成SQL:

```
drop table if exists LabelPropagationClustering_func_test_edge;
create table LabelPropagationClustering_func_test_edge as
select * from
(
```

```
select '1' as flow_out_id,'2' as flow_in_id,0.7 as edge_weight from dual
union all
select '1' as flow_out_id,'3' as flow_in_id,0.7 as edge_weight from dual
union all
select '1' as flow_out_id,'4' as flow_in_id,0.6 as edge_weight from dual
union all
select '2' as flow_out_id,'3' as flow_in_id,0.7 as edge_weight from dual
union all
select '2' as flow_out_id,'4' as flow_in_id,0.6 as edge_weight from dual
union all
select '3' as flow_out_id,'4' as flow_in_id,0.6 as edge_weight from dual
union all
select '4' as flow_out_id,'6' as flow_in_id,0.3 as edge_weight from dual
union all
select '5' as flow_out_id,'6' as flow_in_id,0.6 as edge_weight from dual
union all
select '5' as flow_out_id,'7' as flow_in_id,0.7 as edge_weight from dual
union all
select '5' as flow_out_id,'8' as flow_in_id,0.7 as edge_weight from dual
union all
select '6' as flow_out_id,'7' as flow_in_id,0.6 as edge_weight from dual
union all
select '6' as flow_out_id,'8' as flow_in_id,0.6 as edge_weight from dual
union all
select '7' as flow_out_id,'8' as flow_in_id,0.7 as edge_weight from dual
)tmp
;

drop table if exists LabelPropagationClustering_func_test_node;
create table LabelPropagationClustering_func_test_node as
select * from
(
select '1' as node,0.7 as node_weight from dual
union all
select '2' as node,0.7 as node_weight from dual
union all
select '3' as node,0.7 as node_weight from dual
union all
select '4' as node,0.5 as node_weight from dual
union all
select '5' as node,0.7 as node_weight from dual
union all
select '6' as node,0.5 as node_weight from dual
union all
select '7' as node,0.7 as node_weight from dual
union all
select '8' as node,0.7 as node_weight from dual
)tmp
;
```



数据对应的group结构：

运行结果

结果如下：

```
+------+------------+
| node | group_id |
+------+------------+
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |
| 4 | 1 |
| 5 | 5 |
| 6 | 5 |
| 7 | 5 |
| 8 | 5 |
+------+------------+
```

pai命令示例

```
pai -name LabelPropagationClustering
-project algo_public
-DinputEdgeTableName=LabelPropagationClustering_func_test_edge
-DfromVertexCol=flow_out_id
-DtoVertexCol=flow_in_id
-DinputVertexTableName=LabelPropagationClustering_func_test_node
-DvertexCol=node
-DoutputTableName=LabelPropagationClustering_func_test_result
-DhasEdgeWeight=true
-DedgeWeightCol=edge_weight
-DhasVertexWeight=true
-DvertexWeightCol=node_weight
-DrandSelect=true
-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	-
inputEdgeTableParti	输入边表的分区	选填	全表读入

tions			
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	-
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	-
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	-
randSelect	是否随机选择最大标签	选填	false
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

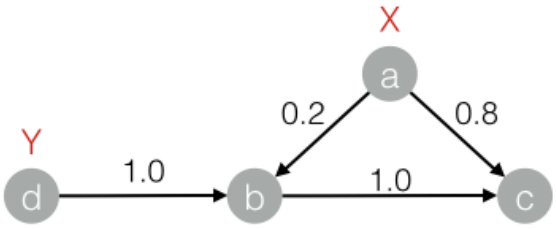
实例

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
-Depsilon=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	-
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	-
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（与标签传播聚类算法的数据相同）

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种存在若干子图，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

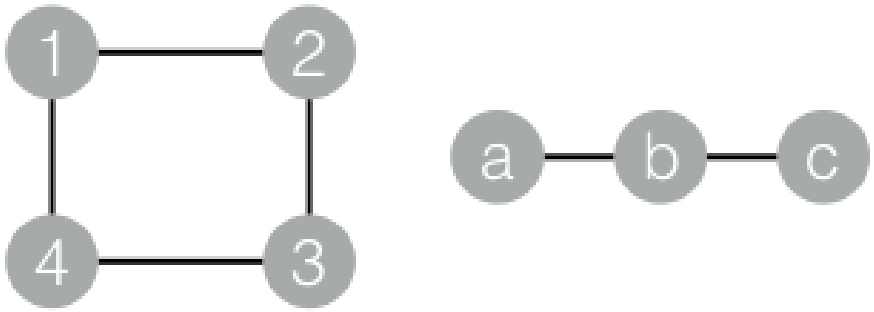
测试数据

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

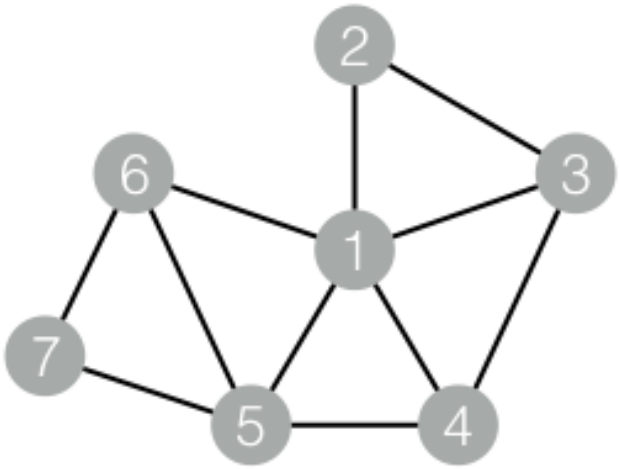
测试数据

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

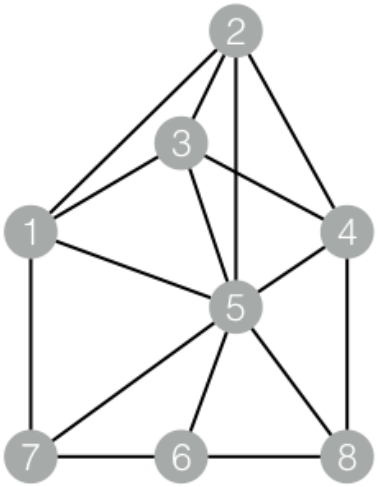
实例

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
5,6,7,3,2,0.66667
5,8,7,3,2,0.66667
6,7,3,3,1,0.33333
7,1,3,4,1,0.33333
7,5,3,7,2,0.66667
8,4,3,4,1,0.33333
8,6,3,3,1,0.33333
```

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	-
inputEdgeTableParti	输入边表的分区	选填	全表读入

tions			
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

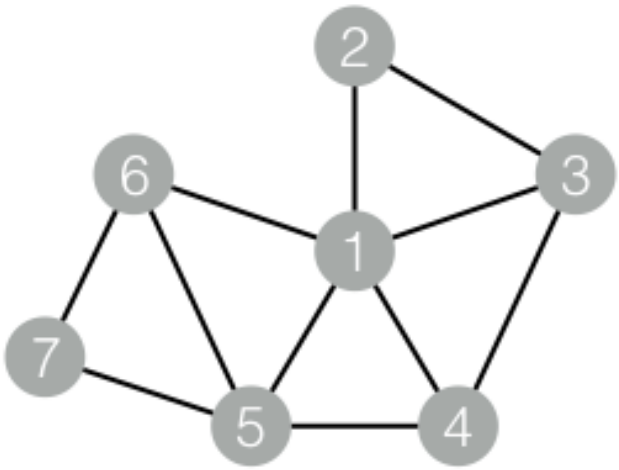
测试数据

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	-
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

树深度

功能介绍

对于众多树状网络，输出每个节点的所处深度和树ID。

参数设置

无

实例

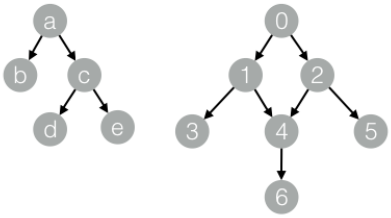
测试数据

生成数据的SQL:

```
drop table if exists TreeDepth_func_test_edge;
create table TreeDepth_func_test_edge as
select * from
(
select '0' as flow_out_id, '1' as flow_in_id from dual
union all
select '0' 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, '4' as flow_in_id from dual
union all
select '2' as flow_out_id, '5' as flow_in_id from dual
union all
select '4' as flow_out_id, '6' as flow_in_id from dual
union all
select 'a' as flow_out_id, 'b' as flow_in_id from dual
union all
select 'a' as flow_out_id, 'c' as flow_in_id from dual
union all
select 'c' as flow_out_id, 'd' as flow_in_id from dual
union all
select 'c' as flow_out_id, 'e' as flow_in_id from dual
)tmp;

drop table if exists TreeDepth_func_test_result;
create table TreeDepth_func_test_result
(
node string,
root string,
depth bigint
);
```



对应的图结构：

运行结果

```
结果如下：
0,0,0
1,0,1
2,0,1
3,0,2
4,0,2
5,0,2
6,0,3
a,a,0
b,a,1
c,a,1
d,a,2
e,a,2
```

pai命令示例

```
pai -name TreeDepth
-project algo_public
-DinputEdgeTableName=TreeDepth_func_test_edge
-DfromVertexCol=flow_out_id
-DtoVertexCol=flow_in_id
-DoutputTableName=TreeDepth_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

目录

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：

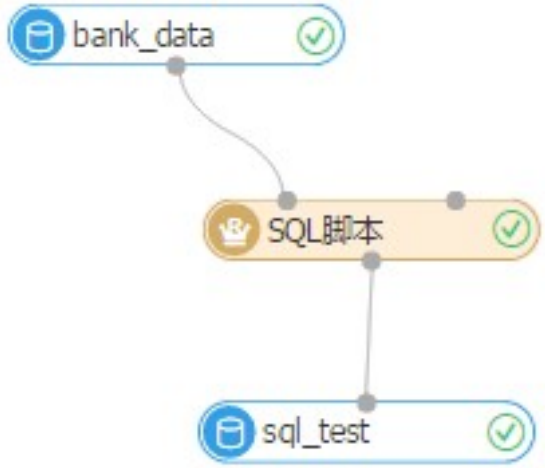
在相应位置写入想要实现的功能的sql脚本，具体介绍如下图：

- 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：

- 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

PAI 命令

不提供pai命令

金融板块

目录

线性回归

PSI

分箱

以等频或者等宽的方式进行分箱

```
pai -name binning
-project algo_public
-DinputTableName=input
-DoutputTableName=output
```

参数说明" class="reference-link">参数说明

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	-	-
selectedColNames	可选，输入表选择分箱的列	-	默认除label外的其他列，无label则选择全部
labelColumn	可选，label所在的列	-	默认无label
validTableName	可选，表示binningMethod为auto时输入的验证表名，是auto模式下，这个参数为必选	-	默认为空
validTablePartitions	可选，验证表选择的分区	-	默认选择全表
inputTablePartitions	可选，输入表选择的分区	-	默认选择全表
inputBinTableName	可选，输入的分箱表	-	默认无分箱表
selectedBinColNames	可选，分箱表选择的列	-	默认为空
positiveLabel	可选，输出的正样本的分类	-	默认为”1”
nDivide	可选，分箱的个数	-	默认为10
colsNDivide	可选，自定义列的分箱个数，例如：col0:3,col2:5。这里	-	默认为空

	有个规则是在colsNDivide中选中的列不在selectedColNames中时，多出来的列也会进行计算。例如：selectedColNames为col0,col1，colsNDivide为col0:3,col2:5，nDivide为10时，那么按照col0:3,col1:10,col2:5进行计算		
isLeftOpen	可选，选择区间为左开右闭或左闭右开	值为”true”或者”false”	默认为”true”
stringThreshold	可选，离散值为其他分箱的阈值	-	默认为无其他分箱
colsStringThreshold	可选，自定义列的阈值，同colsNDivide	-	默认为空
binningMethod	可选，分箱类型，quantile为等频，bucket为等宽，auto为quantile模式下自动选择单调性的分箱	可以为quantile，bucket或者auto	默认为quantile
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

数据转换模块

参数说明

参数名	含义
inputFeatureTableName	输入特征数据表（必选）
inputBinTableName	输入分箱结果表（必选）
inputFeatureTablePartitions	输入特征表选择的分区（可选，默认选择全表）
outputTableName	输出表（必选）
featureColNames	输入表选择的特征列（可选，默认选择全部）
metaColNames	不进行转换的数据列，选中的列会原样输出（可选，默认无meta列，可在此列中指定label,sample_id等）

transformType	数据转换的类型，支持normalize,dummy,woe，分别是“归一化，离散化，转换为WOE值”(可选，默认是dummy)
itemDelimiter	特征分隔符（可选，默认是半角逗号，目前仅对离散化有效）
kvDelimiter	KV分隔符（可选，默认是半角冒号，目前仅对离散化有效）
lifecycle	输出表的生命周期（可选，默认无生命周期）
coreNum	使用的CPU Core的个数（可选，默认会自动计算合适的CPU Core个数）
memSizePerCore	每个CPU Core所使用的内存大小，单位是M，(可选，默认会自动计算合适的内存大小)

使用示例

```
PAI -name data_transform
-project algo_public
-DinputFeatureTableName=feature_table
-DinputBinTableName=bin_table
-DoutputTableName=output_table
-DmetaColNames=label
-DfeatureColNames=feaname1,feaname2
```

归一化算法

归一化根据输入的分箱信息将变量值转换为0-1之间，缺失值填充为0，使用如下算法：

```
if feature_raw_value == null or feature_raw_value == 0 then
feature_norm_value = 0.0
else
bin_index = FindBin(bin_table, feature_raw_value)
bin_width = round(1.0 / bin_count * 1000) / 1000.0
feature_norm_value = 1.0 - (bin_count - bin_index - 1) * bin_width
```

输出格式

进行归一化和WOE转换时输出为普通表。

进行离散化转换成dummy变量时，输出为KV格式的表，生成的变量格式为：[${feaname}]_bin_${bin_id}，如sns这个变量落入第2个桶中，则生成的变量为[sns]_bin_2，若变量为空则落到空桶，生成的变量为[sns]_bin_null，若变量不为空，且不落入任何一个已定义的桶中，则落入else桶，生成的变量为[sns]_bin_else

评分卡训练

评分卡是在信用风险评估领域常用的建模工具，本实现的原理是通过分箱输入将原始变量离散化后再使用线性模型（逻辑回归，线性回归等）进行模型训练，其中包含特征选择功能，分数转换功能等等，同时也支持训练过程中的给变量添加约束条件。

注：*若未指定分箱输入，则评分卡训练过程完全等价于一般的逻辑回归/线性回归。*

特征工程

评分卡区别于普通的线性模型的最大的地方在于，评分卡在使用线性模型进行训练之前会对数据进行一定的特征工程处理，本评分卡中提供了两种特征工程方法，都是需要先经过分箱将特征离散化，一种方法是将每个变量根据分箱结果进行One-Hot编码分别生成N个dummy变量(N为变量的分箱的个数)，另一种方法是WOE转换，即将变量的原始值使用变量落入的分箱所对应的WOE值进行替换。

注：使用dummy变量变换时，每个原始变量的dummy变量之间可以设置相关的约束，具体参见后续章节。

分数转换

评分卡的信用评分等场景中，需要通过线性变换将预测得到的样本的odds转换成分数，通常通过如下的线性变换来进行：

$$log(odds) = \sum(w*x) = a * scaled\_score + b$$

用户通过如下三个参数来指定这个线性变换关系：

- scaledValue 给出一个分数的基准点
- odds 在给定的分数基准点处的odds值
- pdo(Point Double Odds) 分数增长多分odds值加倍

如scaledValue=800, odds=50, pdo=25, 则表示指定了上述直线中的两个点：

$$log(40) = a * 800 + b$$

$$log(80) = a * 825 + b$$

解出a和b，对模型中的分数做线性变换即可得到转换后的变量分。

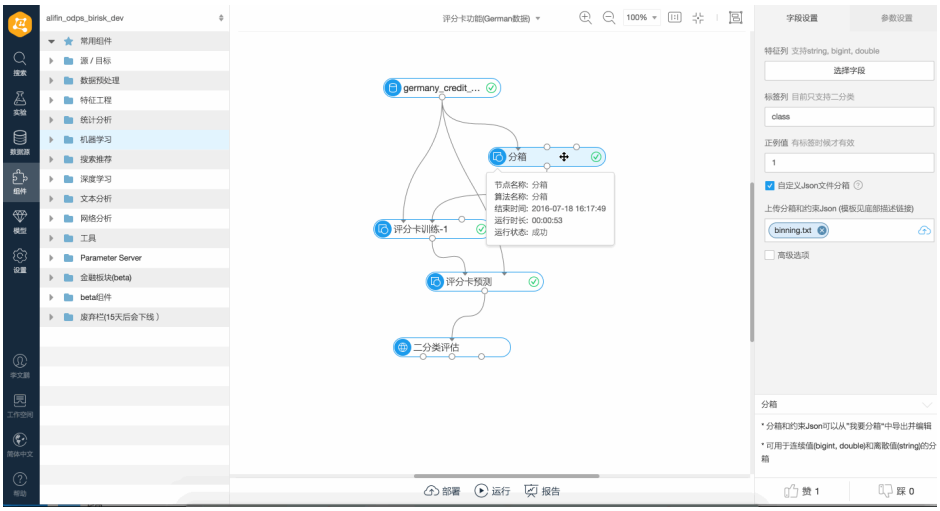
scaling信息由参数-Dscale指定，格式为json，如下：

```
{"scaledValue":800,"odds":50,"pdo":25}
```

当该参数不为空时，上述三个参数的值需同时被设置。

训练过程支持约束

评分卡训练过程支持对变量添加约束，如可指定某个bin所对应的分数为固定值，或两个bin的分数满足一定的比例，再或者bin之间的分数有大小的限制，如设置bin的分数按bin的woe值排序等等，约束的实现依赖于底层的带约束的优化算法，约束可以在分箱的UI中进行设置，设置完成后分箱会生一个json格式的约束条件，并自动传递给后面连接的训练组件，下面是一个分箱组件操作的演示：



目前支持如下几种json约束：>

- “<”：变量的权重按顺序满足升序的约束
- “>”：变量的权重按顺序满足降序的约束
- “=”：变量的权重等于固定值
- “%”：变量之间的权重符定一定的比例关系
- “UP”：变量的权重约束上限
- “LO”：变量的权重约束下限

json约束以字符串的形式存储在表中，表为单行单列（字符串类型）的表，存储如下的Json字符串：

```
{
"name": "feature0",
"<": [
[0,1,2,3]
],
">": [
[4,5,6]
],
"=": [
"3:0","4:0.25"
],
"%": [
["6:1.0","7:1.0"]
]
}
```

内置约束

每个原始变量会有一个隐含的约束，不需要用户指定，即单个变量的人群的分数平均值为0，通过增加这个约束，模型截距项的scaled_weight即为整个人群的平均分。

优化算法

在高级选项中可以选择训练过程中使用的优化算法，目前支持下面四种优化算法:

- L-BFGS
- Newton’ s Method
- Barrier Method
- SQP

其中L-BFGS是一阶的优化算法支持较大规模的特征数据级，牛顿法是经典的二阶算法，收敛速度快，准确度高，但由于要计算二阶的Hessian Matrix，因此不适用于较大特征规模的问题，这两种算法均为无约束的优化算法，当选择这两种优化算法时会自动忽略约束条件。

当训练过程中有约束条件时，可以选择Barrier Method和SQP，这两种算法都是二阶的优化算法，在没有约束条件的情况下完全等价于牛顿法，这两种算法的计算性能和准确性差别不大，我们默认建议选择SQP。

如果用户对优化算法不太了解，建议默认选择”自动选择 “，会自动根据用户任务的数据规模和约束情况来选择最合适的优化算法。

特征选择

训练模块支持stepwise特征选择功能，stepwise是一种前向选择和后向选择的融合，在每进行一次前向特征选择选出一个新变量进入模型后，需要对已进入模型中的变量再进行一次后向选择，移除显著性不满足需求的变量。由于同时支持多种目标函数，也支持多种特征变换方法，因此stepwise特征选择过程也支持不同的选择标准，目前有如下三种选择标准：

- 边缘贡献(Marginal Contribution)，对于所有的目标函数和特征工程方法均适用。
- 评分检验(Score Test)，仅支持WOE转换或无特征工程的逻辑回归选择。
- F检验(F Test)，仅支持WOE转换或无特征工程的线性回归选择。

边缘贡献

边缘贡献度的定义为是训练两个模型，模型A中不包含变量X，模型B在包含所有A的变量的基础上还包含变量X，两个模型最终收敛时所对应的目标函数的差值，即为变量X在模型B中所有变量之间的边缘贡献度。而在特征工程为dummy变换的场景中，原始变量的X的边缘贡献度定义为两个模型分别包含和不包含该变量的所有dummy变量的目标函数的差值，因此使用边缘贡献度的方法进行特征选择支持所有的特征工程方法。

该方法的优点是比较灵活，不局限于某一种模型，直接选择使目标函数更优的变量进入模型。缺点是边缘贡献度不同于统计显著性，统计显著性一般会选择0.05为阈值，而边缘贡献度新用户没有一个绝对的概念阈值选在在多少比较合适，我们给出的建议是可以先设置为10E-5。

评分检验

评分检验仅适应于逻辑回归的特征选择；在前向选择的过程中，首先训练一个仅有截距项的模型，在接下来每一步迭代中，分别对未进入模型的变量计算其评分卡方统计量(Score Chi-Square)，将评分卡方统计量最大的

变量选入模型，同时根据卡方分布计算该统计量所对应的显著性P Value，若评分卡方统计量最大的变量的P Value大于用户指定的进入模型的最大显著性阈值(slentry)，则不会将该变量纳入模型，并停止选择过程。

在完成一轮前向选择之后，会对已选中进入模型的变量进行一轮后向选择，后向选择过程中，对于已经进入模型中的变量分别计算其对应的沃尔德卡方统计量(Wald Chi-Square)，对计算其对应的显著性P Value，若P Value大于用户指定的移除模型的最大显著性阈值(slstay)，则将变量从模型中移除，继续下一轮迭代选择过程。

F检验

F检验仅适应于线性回归的特征选择；在前向选择的过程中，首先训练一个仅有截距项的变量，在接下来的每一步迭代中，分别对未进入模型的变量计算其F Value，F Value的计算与边缘贡献度的计算有些类似，同样需要训练两个模型以计算一个变量的F Value，F Value符合F分布，可根据其F分布的概率密度函数求得其对应的显著性P Value，若P Value大于用户指定的进入模型的最大显著性阈值(slentry)，则不会将变量纳入模型，并停止选择过程。

后向选择过程同样是使用F Value来计算显著性，过程与评分检验类似。

强制选中的变量

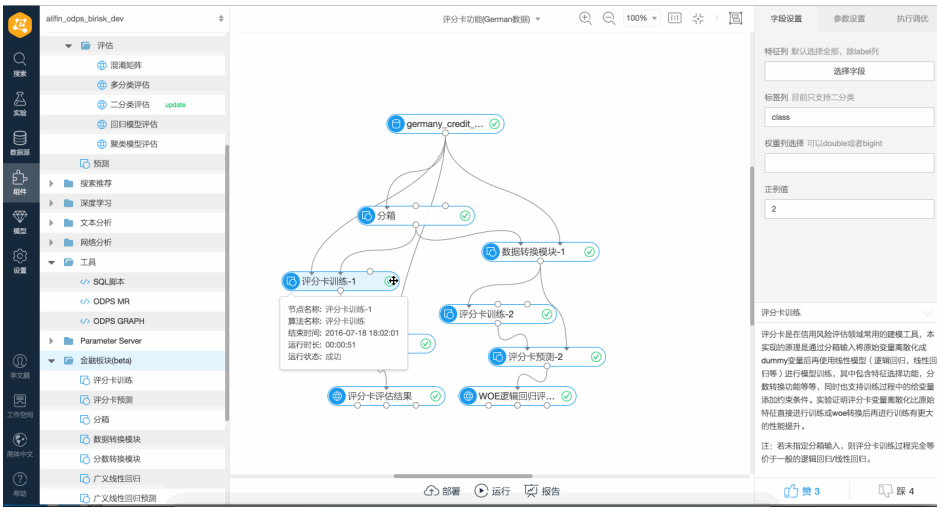
在进行特征选择之前可以设置强制进入模型的变量，被选中的变量不论其显著性为多少，默认都会直接进入模型中，不会参与前向和后向的特征选择过程。

迭代次数和显著性阈值由用户在命令行中通过-Dselected参数指定，格式为json，如下：

```
{"max_step":2, "slentry": 0.0001, "slstay": 0.0001}
```

该参数为空或max_step为0表示正常的训练流程，不进行特征选择。

我们推荐用户使用PAI web来使用评分卡，下面是一个简单的评分卡STEPWISE特征选择和特征WOE变换 + 逻辑回归STEPWISE特征选择的对比演示：



模型报告

评分卡模型的输出为一个Model Report，其中包含了变量的分箱信息，分箱的约束信息，及WOE, Marginal Contribution等一些基本的统计指标，PAI web端展示的评分卡模型评估报告

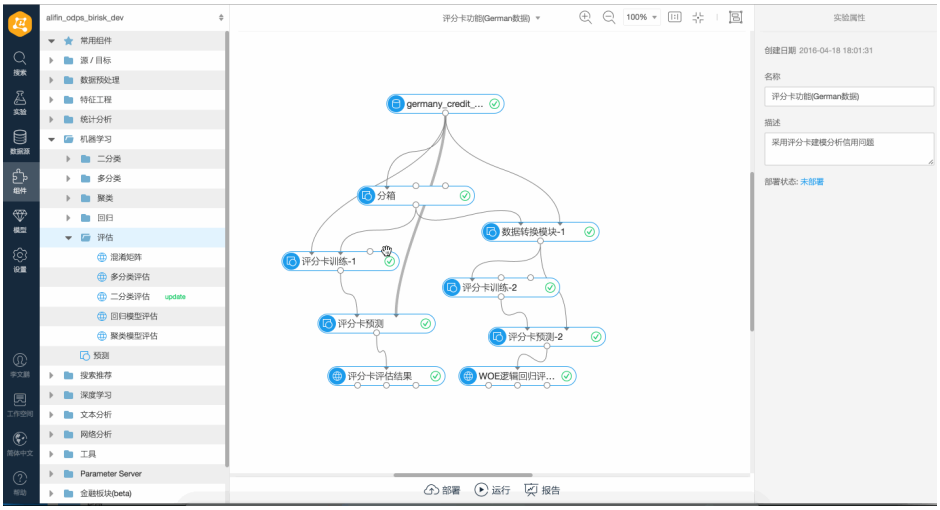
相关列的描述如下:

列名	列类型	列描述
feaname	string	特征名
binid	bigint	分箱ID
bin	string	分箱描述，用于表明该分箱的值域
constraint	string	训练时增加到该分箱上面的约束条件
weight	double	训练完成后所对应的分箱变量权重，或未指定分箱输入的非评分卡模型，该项直接对应到模型变量权重
scaled_weight	double	评分卡模型训练过程中指定分数转换信息后，将分箱变量权重经过线性变换后得到的分数值
woe	double	统计指标：训练集上该分箱的woe值
contribution	double	统计指标：训练集上该分箱的marginal contribution值
total	bigint	统计指标：训练集上该分箱的总的样本数
positive	bigint	统计指标：训练集上该分箱的正样本数
negative	bigint	统计指标：训练集上该分箱的负样本数
percentage_pos	double	统计指标：训练集上该分箱的正样本数点总正样本的比例
percentage_neg	double	统计指标：训练集上该分箱的负样本数点总负样本的比例
test_woe	double	统计指标：测试集上该分箱的woe值
test_contribution	double	统计指标：测试集上该分箱的marginal contribution值
test_total	bigint	统计指标：测试集上该分箱的总的样本数
test_positive	bigint	统计指标：测试集上该分箱的正样本数
test_negative	bigint	统计指标：测试集上该分箱的负样本数

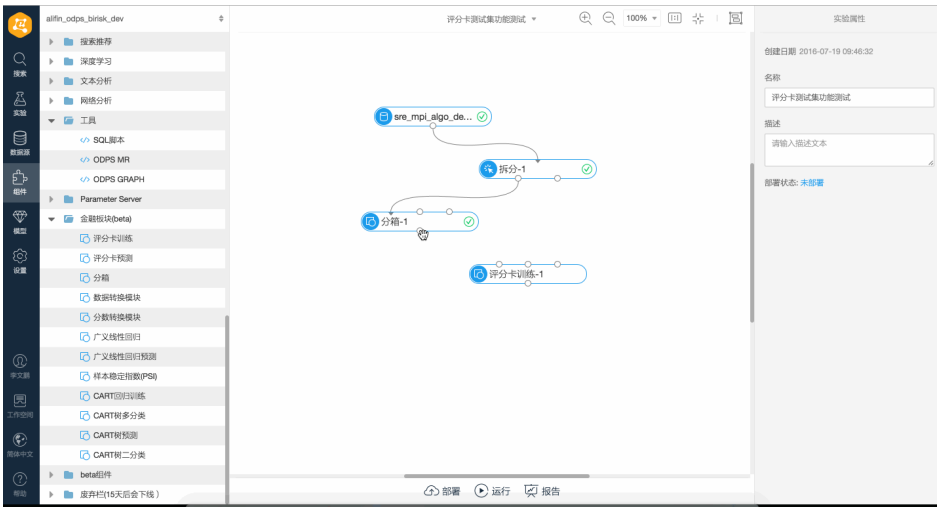
test_percentage_pos	double	统计指标：测试集上该分箱的正样本数点总正样本的比例
test_percentage_neg	double	统计指标：测试集上该分箱的负样本数点总负样本的比例

使用演示

下面是一个简单的评分卡训练和特征WOE变换 + 逻辑回归的对比演示：



当输入训练组件中连接测试集时，在输出的模型报告中会同时输出模型在测试集上的一些统计指标，如WOE,MC等，下面是一个简单的带测试集的训练演示：



PAI命令

```
pai -name=linear_model -project=algo_public
-DinputTableName=input_data_table
-DinputBinTableName=input_bin_table
-DinputConstraintTableName=input_constraint_table
-DoutputTableName=output_model_table
-DlabelColName=label
```

```
-DfeatureColNames=feaname1,feaname2
-Doptimization=barrier_method
-Dloss=logistic_regression
-Dlifecycle=8
```

算法参数

参数名称	参数描述	参数值可选项	默认值
inputTableName	必选，输入特征数据表	-	-
inputTablePartitions	可选，输入特征表选择的分区	-	默认选择全表
inputBinTableName	可选，输入分箱结果表，若该表指定，则自动根据该表的分箱规则对原始特征进行离散化后再进行训练	-	-
featureColNames	可选，输入表选择的特征列	-	默认选择全部，自动排除label列
labelColName	必选，目标列	-	-
outputTableName	必选，输出模型表	-	-
inputConstraintTableName	可选，输入的json格式约束条件，存储在表的一个单元中	-	-
optimization	可选，优化类型	支持lbfgs,newton,barrier_method,sqp,auto，目前仅sqp,barrier_method支持约束，auto即为根据用户数据和相关参数自动选择合适的优化算法，我们建议用户对上述几种优化算法不太了解的情况下选择使用auto	默认为auto
loss	可选，loss类型	支持logistic_regression和least_square	默认为logistic_regression
iterations	可选，优化的最大迭代次数	-	默认为100
l1Weight	可选，L1正则的参数权重，目前仅lbfgs支持l1weight	-	默认为0
l2Weight	可选，L2正则的参数权重	-	默认为0
m	可选，lbfgs优化过程中的历史长度，仅对	-	默认为10

	lbfgs有效		
scale	可选，评分卡对weight进行scale的信息	-	默认为空
selected	可选，评分卡特征选择功能	-	默认为空
convergenceTolerance	可选，收敛条件	-	默认为1e-6
positiveLabel	可选，正样本的分类	-	默认为”1”
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

评分卡预测

评分卡预测组件对原始数据根据训练组件产出的模型结果进行预测打分，支持的训练组件包括：评分卡训练，逻辑回归二分类(金融版块)，线性回归(金融版块)。

输入参数

预测组件支持如下参数：

- **特征列**：选择用于预测的原始特征列，默认选择全部。
- **原样填加到结果表**：选择不进行任何处理，直接附加到预测结果表中的列，如常用的ID列，目标列等等。
- **输出变量分**：是否输出每个特征变量所对应的分数，最终的预测总得分为截距项的得分加所有的变量分。

输出打分表

输出打分表的示例如下：

churn ▲	prediction_score ▲	prediction_prob ▲	prediction_detail ▲
1	-2.152581613419945	0.10409022740823...	{"0":0.8959097726,"1":0.1040902274}
1	0.40321295914989297	0.599459362718963	{"0":0.4005406373,"1":0.5994593627}
0	-5.9448781609701316	0.00261237905306...	{"0":0.9973876209,"1":0.0026123791}
1	-1.4235254136279643	0.19410950481015...	{"0":0.8058904952,"1":0.1941095048}
0	-0.4354127766052662	0.3928345539282477	{"0":0.6071654461,"1":0.3928345539}
0	-2.322642905577369	0.08926496638122...	{"0":0.9107350336,"1":0.0892649664}
1	-1.8095060182152187	0.14069783849895...	{"0":0.8593021615,"1":0.1406978385}
0	0.09435077042706097	0.5235702098997645	{"0":0.4764297901,"1":0.5235702099}
0	-2.460605112010114	0.0786664686456529	{"0":0.9213335314,"1":0.0786664686}

其中第一列churn为用户选择的原样添加到结果表中的列，与预测结果无关，其它三列为预测结果列，其意义分别如下：

列名	列类型	列描述
prediction_score	double	预测分数列，线性模型中特征值和模型权重值直接相乘相加的结果，对应到评分卡模型中，若模型进行了分数转换，则该分数输出的是转换后的得分
prediction_prob	double	二分类的场景中预测得到的正例的概率值，原始得分(未经分数转换)经过sigmoid变换后所得
prediction_detail	string	用json格式描述的各类别概率值，其中0代表负类，1代表正类，如：{“0”:0.1813110520,”1”:0.8186889480}

PAI 命令

```
pai -name=lm_predict
-project=algo_public
-DinputFeatureTableName=input_data_table
-DinputModelTableName=input_model_table
-DmetaColNames=sample_key,label
-DfeatureColNames=fea1,fea2
-DoutputTableName=output_score_table
```

算法参数

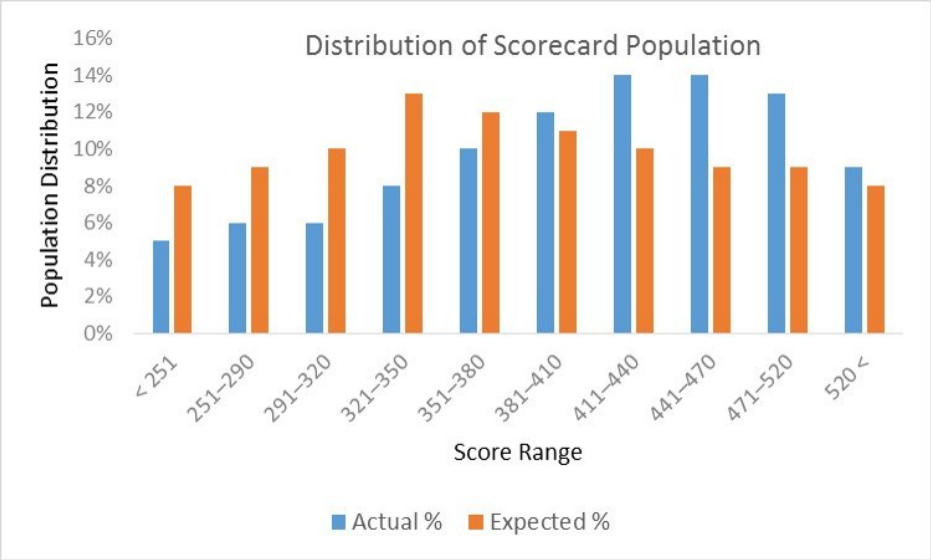
参数名称	参数描述	参数值可选项	默认值
inputFeatureTableName	必选，输入特征数据表	-	-
inputFeatureTablePartitions	可选，输入特征表选择的分区	-	默认选择全表
inputModelTableName	必选，输入的模型表	-	-
featureColNames	可选，输入表选择的特征列	-	默认选择全部
metaColNames	可选，不进行转换的数据列，选中的列会原样输出	-	默认无meta列，可在此列中指定label,sample_id等
outputFeatureScore	可选，预测结果中是否输出变量分	true/false	默认值为false
outputTableName	必选，输出预测结果表	-	-
lifecycle	可选，输出表的生命周	-	默认不设置

	期		
coreNum	可选，核心数	-	默认自动计算
memSizePerCore	可选，内存数	-	默认自动计算

PSI

样本稳定指数是衡量样本变化所产生的偏移量的一种重要指标，通常用来衡量样本的稳定程度，比如样本在两个月份之间的变化是否稳定。通常变量的PSI值在0.1以下表示变化不太显著，在0.1到0.25之间表示有比较显著的变化，大于0.25表示变量变化比较剧烈，需要特殊关注。

衡量样本在不是时刻的稳定性时，可以使用画图的方法，即将要比较的变量离散化成N个分箱，然后计算样本分别在各个分箱中的数量及比例，并以柱状图的形式表示出来，如下图:

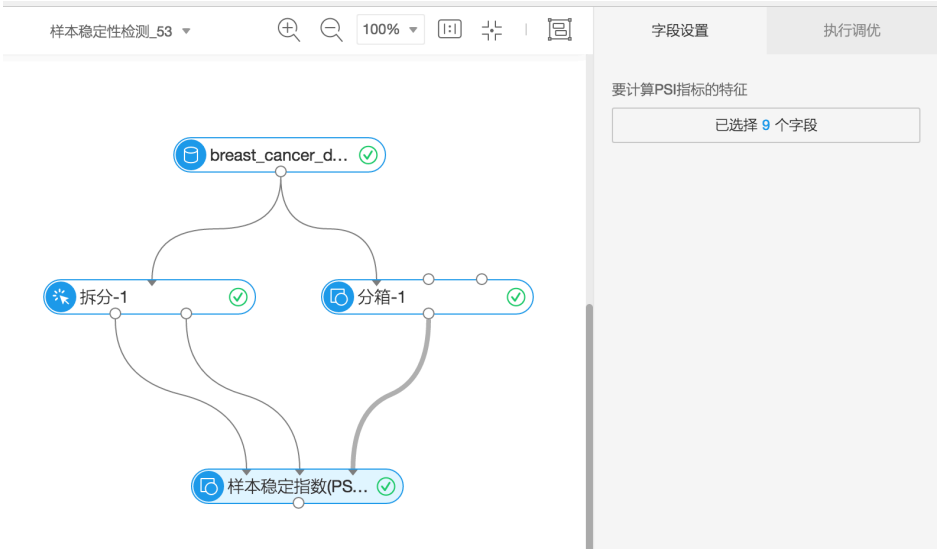


这种方法可以比较直观地看出某个变量在两批样本上是否有剧烈的变化，但该方法的缺点是无法量化，无法实现对样本稳定性的自动监控，因此PSI就显得尤为重要，在使用PSI之前也需要对特征数据进行分箱，因此需要一个分箱组件，PSI的计算公式如下：

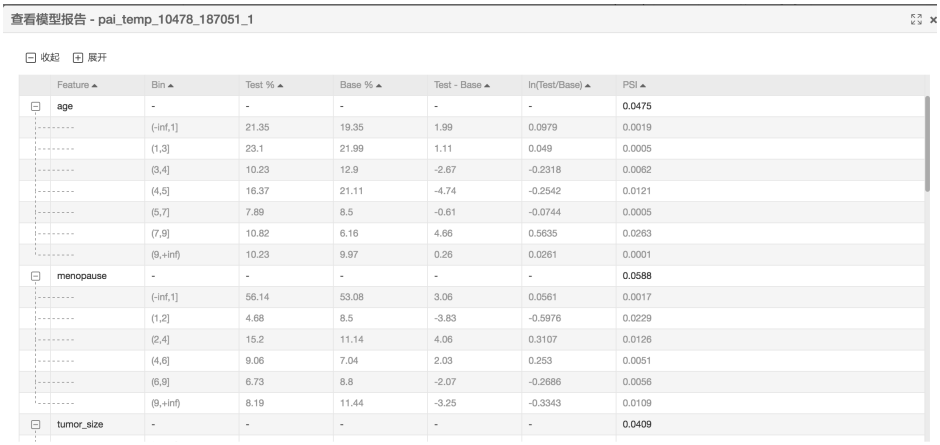
$$PSI = \sum((Actual\ \% - Expected\ \%) \times (ln(\frac{Actual\ \%}{Expected\ \%}))$$

使用示例

下图是一个使用的示例，PSI组件分别连接待比较的两个样本数据集，再连接一个分箱组件，只有一个参数可供选择，即选择要进行PSI计算的特征。



结果示例



Feature	Bin	Test %	Base %	Test - Base	ln(Test/Base)	PSI
age	-	-	-	-	-	0.0475
	(-inf,1]	21.35	19.35	1.99	0.0979	0.0019
	(1,3]	23.1	21.99	1.11	0.049	0.0005
	(3,4]	10.23	12.9	-2.67	-0.2318	0.0062
	(4,5]	16.37	21.11	-4.74	-0.2542	0.0121
	(5,7]	7.89	8.5	-0.61	-0.0744	0.0005
	(7,9]	10.82	6.16	4.66	0.5635	0.0263
	(9,+inf)	10.23	9.97	0.26	0.0261	0.0001
menopause	-	-	-	-	-	0.0588
	(-inf,1]	56.14	53.08	3.06	0.0561	0.0017
	(1,2]	4.68	8.5	-3.83	-0.5976	0.0229
	(2,4]	15.2	11.14	4.06	0.3107	0.0126
	(4,6]	9.06	7.04	2.03	0.253	0.0051
	(6,9]	6.73	8.8	-2.07	-0.2686	0.0056
	(9,+inf)	8.19	11.44	-3.25	-0.3343	0.0109
tumor_size	-	-	-	-	-	0.0409

PAI命令

```
PAI -name psi
-project algo_public
-DinputBaseTableName=psi_base_table
-DinputTestTableName=psi_test_table
-DoutputTableName=psi_bin_table
-DinputBinTableName=pai_index_table
-DfeatureColNames=fea1,fea2,fea3
-Dlifecycle=7
```

算法参数

参数名称	参数描述	参数值可选项	默认值
inputBaseTableNam e	必选，输入基础表表名，计算测试表在基础表的基础上产生的偏移量	-	-
inputBaseTablePartit	可选，输入基础表分区	-	默认选择全表

ions			
inputTestTableName	必选，输入测试表表名，计算测试表在基础表的基础上产生的偏移量	-	-
inputTestTablePartitions	可选，输入测试表分区	-	默认选择全表
inputBinTableName	必选，输入分箱结果表表名	-	
featureColNames	可选，要计算PSI指标的特征	-	默认选择所有特征
outputTableName	必选，输出的指标表	-	-
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

时间序列

目录

x13_arima

Arima全称为自回归积分滑动平均模型(Autoregressive Integrated Moving Average Model,简记ARIMA)，是由博克思(Box)和詹金斯(Jenkins)于70年代初提出一著名时间序列预测方法，所以又称为box-jenkins模型、博克思-詹金斯法.

x13-arima是基于开源X-13ARIMA-SEATS封装的针对季节性调整的arima算法。

X-13ARIMA-SEATS Seasonal Adjustment Program 详细介绍请见链接 wiki

Arima 详细介绍请见链接 wiki

pai命令行

```
pai -name x13_arima
-project algo_public
-DinputTableName=pai_ft_x13_arima_input
-DseqColName=id
-DvalueColName=number
-Dorder=3,1,1
-Dstart=1949.1
-Dfrequency=12
-Dseasonal=0,1,1
-Dperiod=12
-DpredictStep=12
-DoutputPredictTableName=pai_ft_x13_arima_out_predict
-DoutputDetailTableName=pai_ft_x13_arima_out_detail
```

参数说明

参数名称	参数描述	取值范围	是否必选，默认值/行为
inputTableName	输入表	表名	必选
inputTablePartitions	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		可选， 默认值选择所有分区
seqColName	时序列	列名	必选，仅用来对valueColName 排序，具体数值与计算无关
valueColName	数值列	列名	必选
groupColNames	分组列，多列已逗号分隔，如col0,col1; 每个分组会构建一个时间序列	列名	可选
order	p,d,q 分别表示自回归系数、差分、滑动回归系数	p,d,q均为非负整数，范围[0, 36]	必选
start	时序开始日期	字符串，格式year.seasonal， 如1986.1 时序格式介绍	可选，默认为1.1
frequency	时序频率	正整数，范围(0, 12] 时序格式介绍	可选，默认为12, 表示12月/年
seasonal	sp,sd,sq 分别表示季节性自回归系数、季节性差分、季节性滑动回归系数	sp,sd,sq均为非负整数，范围[0, 36]	可选，默认无seasonal

period	seasonal周期	数字，范围(0, 100]	可选，默认为frequency
maxiter	最大迭代次数	正整数	可选，默认为 1500
tol	tolerance，容忍度	double类型	可选，默认为 1e-5
predictStep	预测条数	数字，范围(0, 365]	可选，默认为12
confidenceLevel	预测置信水平	数字，范围(0, 1)开区间	可选，默认为 0.95
outputPredictTableName	预测输出表	表名	必选
outputDetailTableName	详细信息表	表名	必选
outputTablePartition	输出分区	分区名	可选，默认不输出到分区
coreNum	节点个数	与参数memSizePerCore配对使用，正整数	可选， 默认自动计算
memSizePerCore	单个节点内存大小，单位M	正整数，范围[1024, 64 *1024]	可选， 默认自动计算
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期

时序格式介绍

参数start和frequency规定了数据（valueColName）的两个时间维度ts1、ts2。frequency表示单位周期内数据的频率，既单位ts1中ts2的频率。 start格式是n1.n2， 既表示开始日期是第n1个ts1中的第n2个ts2

单位时间	ts1	ts2	frequency	start
12月/年	年	月	12	1949.2 表示第1949年中的第2个月
4季/年	年	季	4	1949.2 表示第1949年中的第2个季度
7天/周	天	周	7	1949.2 表示第1949周中的第2天
1	任何时间单位	1	1	1949.1 表示第1949（年、天、时等）

例如. value=[1,2,3,5,6,7,8,9,10,11,12,13,14,15]

1) start=1949.3, frequency=12表示数据是monthly per year, 预测开始的日期是1950.06

year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1949			1	2	3	4	5	6	7	8	9	10
1950	11	12	13	14	15							

2) start=1949.3, frequency=4表示数据是quarterly per year, 预测开始的日期是1953.02

year	Qtr1	Qtr2	Qtr3	Qtr4
1949			1	2
1950	3	4	5	6
1951	7	8	9	10
1952	11	12	13	14
1953	14			

3) start=1949.3, frequency=7表示数据是daily per week, 预测开始的日期是1951.04

week	Sun	Mon	Tue	Wed	Thu	Fri	Sat
1949			1	2	3	4	5
1950	6	7	8	9	10	11	12
1951	13	14	15				

4) start=1949.1, frequency=1 可以表示任何时间单位， 预测开始日期是1963.00

cycle	p1
1949	1
1950	2
1951	3
1951	4
1952	5
1953	6
1954	7
1955	8
1956	9
1957	10
1958	11

1959	12
1960	13
1961	14
1962	15

具体示例

- 测试数据使用的数据：AirPassengers。这个数据集是1949-1960年每个月国际航空的乘客数量的数据。```

create table pai_ft_x13_arima_input(id bigint,number bigint);tunnel upload data/airpassengers.csv pai_ft_x13_arima_input -h true;

```
* pai命令
```bash
pai -name x13_arima
-project algo_public
-DinputTableName=pai_ft_x13_arima_input
-DseqColName=id
-DvalueColName=number
-Dorder=3,1,1
-Dseasonal=0,1,1
-Dstart=1949.1
-Dfrequency=12
-Dperiod=12
-DpredictStep=12
-DoutputPredictTableName=pai_ft_x13_arima_out_predict
-DoutputDetailTableName=pai_ft_x13_arima_out_detail
```

- 输出说明

输出表：outputPredictTableName，字段分别为：

column name	comment
pdate	预测日期
forecast	预测结论
lower	置信度为confidenceLevel(默认0.95)时，预测结论下界
upper	置信度为confidenceLevel(默认0.95)时，预测结论上界

数据展示
```

序号	pdate	forecast	lower	upper
1	196101	444.445401291661	422.354385657496	466.536416925825
2	196102	420.971087699795	394.745551231805	447.196624167785
3	196103	453.432019696644	423.435661316528	483.428378076759
4	196104	490.922534668881	458.870488854835	522.974580482926
5	196105	503.877174753018	470.308964411549	537.445385094488
6	196106	566.536076521906	531.945171132091	601.126981911721
7	196107	652.606993945368	617.2596149799	687.954372910837
8	196108	639.841497141155	603.933582941945	675.749411340364
9	196109	542.341866147189	506.000630530659	578.683101763719
10	196110	494.745102803541	458.0614903363	531.428715270782
11	196111	426.635134341211	389.672783323704	463.597485358717
12	196112	468.722280768837	431.527372120416	505.917189417259

输出表：outputDetailTableName，字段分别为：

column name	comment
key	model表示模型 evaluation表示评估结果 parameters表示训练参数 log表示训练日志
summary	存储具体信息

数据展示

序号	key	summary
1	model	{ "comment": { "ma": "arima estimate", "mr": "regress...
2	evaluation	{ "comment": { "aic": "AIC", "aicc": "AICC (F-correcte...
3	paramters	{ "arima": { "d": 1, "isSeasonal": true, "p": 3, "period":...
4	log	1 Log for X-13ARIMA-SEATS program (Version 1.1...

PaiWeb展示-模型系数(key=model)

operator	factor	period	lag	estimate	standard error
AR	Nonseasonal	1	1	0.6135	0.0928
AR	Nonseasonal	1	2	0.2403	0.1035
AR	Nonseasonal	1	3	-0.0732	0.0906
MA	Nonseasonal	1	1	0.9737	0.0376
MA	Seasonal	12	12	0.1051	0.1031

PaiWeb展示-评估指标(key=evaluation)

Name	Indicator
AIC	1019.6973
BIC	1036.9485
Hannan Quinn	1026.7072
Log likelihood	-503.8487
Effective number of observations	131
Number of observations	144
variance	127.0384

算法规模

- 支持规模

行：单group数据最大1200条

列：1数值列

- 资源计算方式

不设置groupColNames, 默认计算方式

coreNum = 1

memSizePerCore = 4096

设置groupColNames，默认计算方式

coreNum = floor(总数据行数 / 12万)

memSizePerCore = 4096

训练数据集

id	number
1	112
2	118
3	132
4	129
5	121
...	...

x13_auto_arima

x13-auto-arima includes an automatic ARIMA model selection procedure based largely on the procedure of Gomez and Maravall (1998) as implemented in TRAMO (1996) and subsequent revisions.

x13-auto-arima选择过程如下

default model esitmation.

当frequency = 1，默认模型是(0,1,1)当frequency > 1，默认模型是(0,1,1)(0,1,1)

ident cation of dierencing orders

如果设置了diff和seasonalDiff，则跳过此步使用 Unit root test (wiki)确定差分d，和季节性差分D

ident cation of ARMA model orders

根据BIC(wiki)准则选择最合适的模型， 其参数maxOrder、maxSeasonalOrder在此步骤其作用

comparison of identif ed model with default model

使用Ljung-Box Q statistic(wiki)比较模型，如果两个模型均是不可接受的，则使用(3,d,1)(0,D,1)模型

final model checks

pai命令行

```
pai -name x13_auto_arima
-project algo_public
-DinputTableName=pai_ft_x13_arima_input
-DseqColName=id
-DvalueColName=number
-Dstart=1949.1
```

```
-Dfrequency=12
-DpredictStep=12
-DoutputPredictTableName=pai_ft_x13_arima_out_predict2
-DoutputDetailTableName=pai_ft_x13_arima_out_detail2
```

参数说明

参数名称	参数描述	取值范围	是否必选，默认值/行为
inputTableName	输入表	表名	必选
inputTablePartitions	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		可选， 默认值选择所有分区
seqColName	时序列	列名	必选，仅用来对valueColName 排序，具体数值与计算无关
valueColName	数值列	列名	必选
groupColNames	分组列，多列已逗号分隔，如col0,col1; 每个分组会构建一个时间序列	列名	可选
start	时序开始日期	字符串，格式year.seasonal， 如1986.1 时序格式介绍	可选，默认为1.1
frequency	时序频率	正整数，范围(0, 12] 时序格式介绍	可选，默认为12, 表示12月/年
maxOrder	p,q最大值	正整数，范围 [0,4]	可选，默认2
maxSeasonalOrder	季节性p，q最大值	正整数，范围[0,2]	可选，默认1
maxDiff	差分d最大值	正整数，范围[0,2]	可选，默认 2
maxSeasonalDiff	季节性差分d最大值	正整数，范围[0,1]	可选，默认1
diff	差分d	正整数，范围[0,2] diff与maxDiff同时设置时，maxDiff被忽略 diff与seasonalDiff要同时设置	可选，默认-1， 不指定diff
seasonalDiff	季节性差分d	正整数，范围[0,1] seasonalDiff与maxSeasonalDiff同时设置时maxSeasonalDiff被忽略	可选，默认-1，不指定seasonalDiff

maxiter	最大迭代次数	正整数	可选，默认为 1500
tol	tolerance，容忍度	double类型	可选，默认为 1e-5
predictStep	预测条数	数字，范围(0, 365]	可选，默认为12
confidenceLevel	预测置信水平	数字，范围(0, 1)开区间	可选，默认为 0.95
outputPredictTableName	预测输出表	表名	必选
outputDetailTableName	详细信息表	表名	必选
outputTablePartition	输出分区	分区名	可选，默认不输出到分区
coreNum	节点个数	与参数memSizePerCore配对使用，正整数	可选， 默认自动计算
memSizePerCore	单个节点内存大小，单位M	正整数，范围[1024, 64 *1024]	可选， 默认自动计算
lifecycle	可选，指定输出表的生命周期	正整数	没有生命周期

时序格式介绍

参数start和frequency规定了数据（valueColName）的两个时间维度ts1、ts2。frequency表示单位周期内数据的频率，既单位ts1中ts2的频率。 start格式是n1.n2， 既表示开始日期是第n1个ts1中的第n2个ts2

单位时间	ts1	ts2	frequency	start
12月/年	年	月	12	1949.2 表示第1949年中的第2个月
4季/年	年	季	4	1949.2 表示第1949年中的第2个季度
7天/周	天	周	7	1949.2 表示第1949周中的第2天
1	任何时间单位	1	1	1949.1 表示第1949（年、天、时等）

例如. value=[1,2,3,5,6,7,8,9,10,11,12,13,14,15]

1) start=1949.3, frequency=12表示数据是monthly per year, 预测开始的日期是1950.06

ye	Ja	Fe	M	Ap	M	Ju	Jul	Au	Se	Oc	N	De

ar	n	b	ar	r	ay	n		g	p	t	ov	c
1949			1	2	3	4	5	6	7	8	9	10
1950	11	12	13	14	15							

2) start=1949.3, frequency=4表示数据是quarterly per year, 预测开始的日期是1953.02

year	Qtr1	Qtr2	Qtr3	Qtr4
1949			1	2
1950	3	4	5	6
1951	7	8	9	10
1952	11	12	13	14
1953	14			

3) start=1949.3, frequency=7表示数据是daily per week, 预测开始的日期是1951.04

week	Sun	Mon	Tue	Wed	Thu	Fri	Sat
1949			1	2	3	4	5
1950	6	7	8	9	10	11	12
1951	13	14	15				

4) start=1949.1, frequency=1 可以表示任何时间单位， 预测开始日期是1963.00

cycle	p1
1949	1
1950	2
1951	3
1951	4
1952	5
1953	6
1954	7
1955	8
1956	9
1957	10
1958	11
1959	12
1960	13

1961	14
1962	15

具体示例

- 测试数据使用的数据：AirPassengers。这个数据集是1949-1960年每个月国际航空的乘客数量的数据。查看数据```

create table pai_ft_x13_arima_input(id bigint,number bigint);tunnel upload data/airpassengers.csv pai_ft_x13_arima_input -h true;

```
* pai命令
```

pai -name x13_auto_arima -project algo_public -DinputTableName=pai_ft_x13_arima_input -DseqColName=id -DvalueColName=number -Dstart=1949.1 -Dfrequency=12 -DmaxOrder=4 -DmaxSeasonalOrder=2 -DmaxDiff=2 -DmaxSeasonalDiff=1 -DpredictStep=12 -DoutputPredictTableName=pai_ft_x13_arima_auto_out_predict -DoutputDetailTableName=pai_ft_x13_arima_auto_out_detail```

- 输出说明

输出表：outputPredictTableName，字段分别为：

column name	comment
pdate	预测日期
forecast	预测结论
lower	置信度为confidenceLevel(默认0.95)时，预测结论下界
upper	置信度为confidenceLevel(默认0.95)时，预测结论上界

数据展示

序号	pdate	forecast	lower	upper
1	196101	444.445401291661	422.354385657496	466.536416925825
2	196102	420.971087699795	394.745551231805	447.196624167785
3	196103	453.432019696644	423.435661316528	483.428378076759
4	196104	490.922534668881	458.870488854835	522.974580482926
5	196105	503.877174753018	470.308964411549	537.445385094488
6	196106	566.536076521906	531.945171132091	601.126981911721
7	196107	652.606993945368	617.2596149799	687.954372910837
8	196108	639.841497141155	603.933582941945	675.749411340364
9	196109	542.341866147189	506.000630530659	578.683101763719
10	196110	494.745102803541	458.0614903363	531.428715270782
11	196111	426.635134341211	389.672783323704	463.597485358717
12	196112	468.722280768837	431.527372120416	505.917189417259

输出表：outputDetailTableName，字段分别为：

column name	comment
key	model表示模型 evaluation表示评估结果 parameters表示训练参数 log表示训练日志
summary	存储具体信息

数据展示

序号	key	summary
1	model	{ "comment": { "ma": "arima estimate", "mr": "regress...
2	evaluation	{ "comment": { "aic": "AIC", "aicc": "AICC (F-correcte...
3	paramters	{ "arima": { "d": 1, "isSeasonal": true, "p": 3, "period":...
4	log	1 Log for X-13ARIMA-SEATS program (Version 1.1...

PaiWeb展示-模型系数(key=model)

operator	factor	period	lag	estimate	standard error
AR	Nonseasonal	1	1	0.6135	0.0928
AR	Nonseasonal	1	2	0.2403	0.1035
AR	Nonseasonal	1	3	-0.0732	0.0906
MA	Nonseasonal	1	1	0.9737	0.0376
MA	Seasonal	12	12	0.1051	0.1031

PaiWeb展示-评估指标(key=evaluation)

Name	Indicator
AIC	1019.6973
BIC	1036.9485
Hannan Quinn	1026.7072
Log likelihood	-503.8487
Effective number of observations	131
Number of observations	144
variance	127.0384

算法规模

- 支持规模

行：单group数据最大1200条

列：1数值列

- 资源计算方式

不设置groupColNames, 默认计算方式

coreNum = 1

memSizePerCore = 4096

设置groupColNames，默认计算方式

coreNum = floor(总数据行数 / 12万)

memSizePerCore = 4096

常见问题

- 为什么预测结果都一样？

在模型训练异常时，会调用均值模型，则所有预测结果是训练数据的均值。常见的异常如： 时序差分diff后不stationary；训练没有收敛；方差为0等。 可以在logview中，查看单独节点的stderr文件，获取具体的异常信息

训练数据集

id	number
1	112
2	118

3	132
4	129
5	121
…	…