Machine Learning Platform for AI

User Guide

User Guide

Machine learning modules

Resource

Contents

Read MaxCompute Table

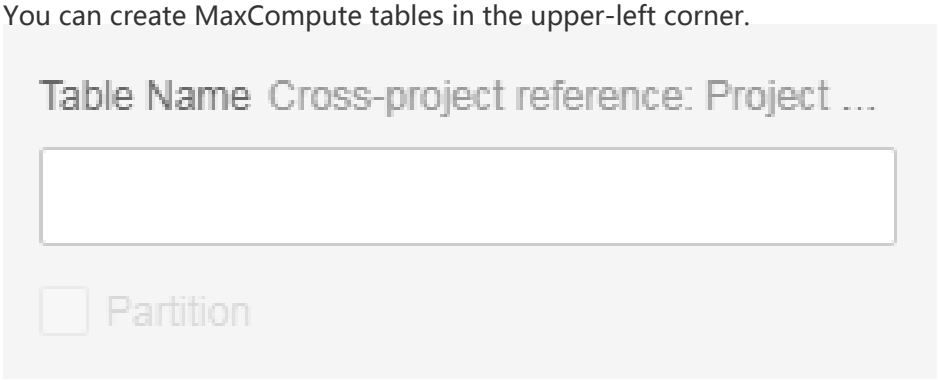
The Read MaxCompute Table component enables you to read data from MaxCompute tables. By default, the component only reads tables in the current project. To read a table from another project that you are authorized to access, join the project name and table name in the format of **Project Name.Table Name**, for example, tianchi_project.weibo_data.

After specifying a table, the system automatically reads structural information of the table. To view the structural information, click the **Column Information** tab. The component is unaware of any modifications (such as add or remove a column) made to a table that is already loaded to the component. To resolve this issue, reload the MaxCompute table by respecifying the table source.

If a partitioned table is input, the system automatically selects the Partition option for the table. You can then select or enter the partition parameters. Currently only one partition can be specified. If the Partition option is not selected or no partition name is specified, the system determines that a full table is input. The Partition option is unavailable if an unpartitioned table is input.

Read MaxCompute Table Input Field

You can create MaxCompute tables in the upper-left corner.



Partition

The Read MaxCompute Table component supports partitioned tables. However, the date format defined by this component is different than DataWorks.

To read partitioned tables, you must specify the date in the dt=@@{yyyyMMdd} format, where @@{yyyyMMdd} represents the date of the current day and @@{yyyyMMdd-1d} represents the day before the current day.

Write MaxCompute Table

The Write MaxCompute Table component enables you to write data to a MaxCompute table in current project or across projects.

However, the Write MaxCompute Table component does not support partitioned tables.

Data preprocessing

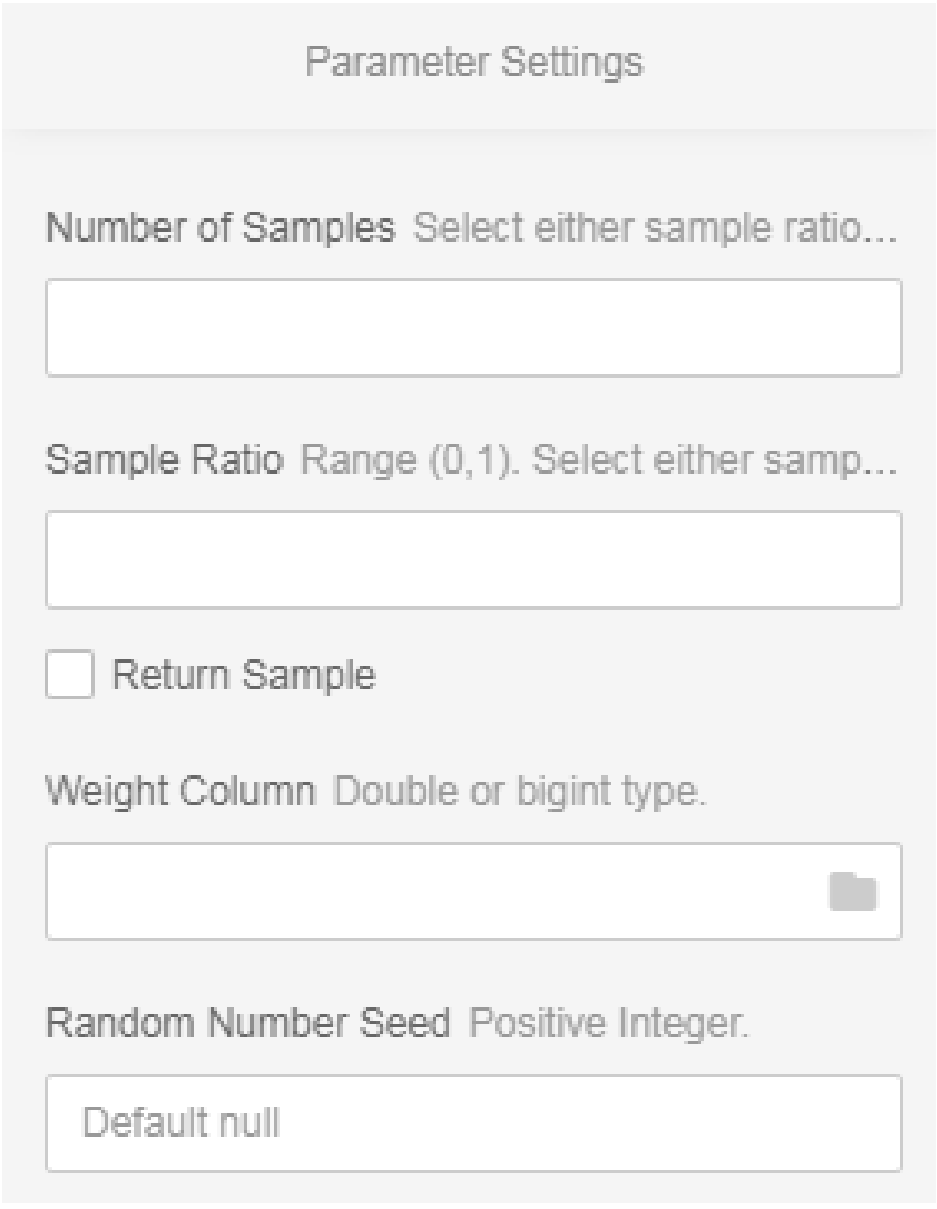
Contents

Weighted sampling

Sampling data is generated in the weighted mode. The weight column must be of Double or Int type. Sampling is performed based on the value of the weight column. For example, data with a col value of 1.2 has a higher probability to be sampled than data with a col value of 1.0.

Parameter settings

Parameter box



- You can manually enter the number of samples (or the sampling ratio).
- You can choose whether to enable sampling with replacement, which is disabled by default. To enable it, select the checkbox.
- Select the weighted column from the drop-down list. The weighted column supports the

double and bigint types.
- The random seed can be configured. By default, it is automatically assigned.

PAI command

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

Algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	NA	NA
inputTablePartitions	(Optional) Partitions used for training in the input table, in the format of Partition_name=value. The multilevel format is name1=value1/name2=value2. If multiple partitions are specified, use commas (,) to separate them.	NA	All partitions in the input table
outputTableName	(Required) Name of the output table	NA	NA
sampleSize	(Optional) Number of samples	Positive integer	Null by default
sampleRatio	(Optional) sampling ratio	Floating-point number in the range of (0,1)	Null by default
probCol	(Required) Columns to be weighted. Each value indicates the weight of a record. You do not need to normalize them.	NA	NA
replace	(Optional) Indicates whether to enable sampling with replacement. The type is boolean.	true / false	false (disabled by default)
randomSeed	(Optional) Random	Positive integer	Automatically

	seed		generated by default
lifecycle	(Optional) Life cycle of the output table	Positive integer in the range of [1,3650]	No life cycle for the output table
coreNum	(Optional) Number of computing cores	Positive integer	Automatically assigned
memSizePerCore	(Optional) Memory size of each core, in MB	Positive integer in the range of (1, 65536)	Automatically assigned

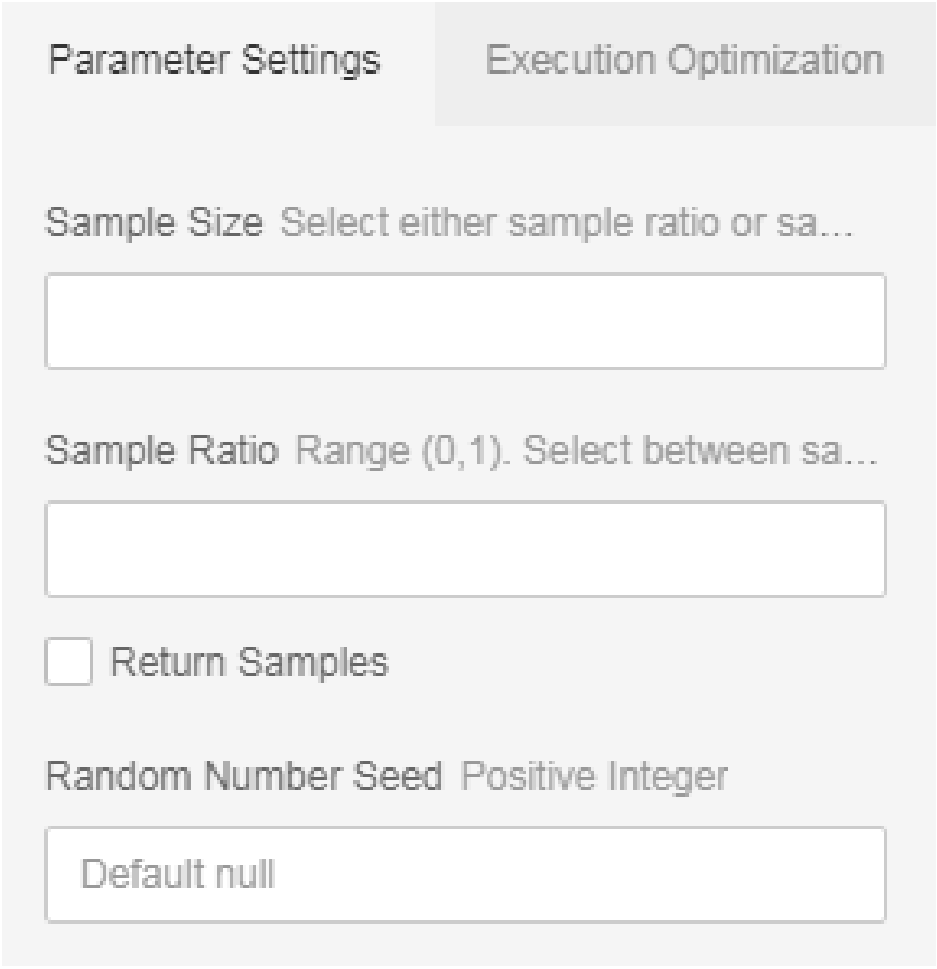
Note:

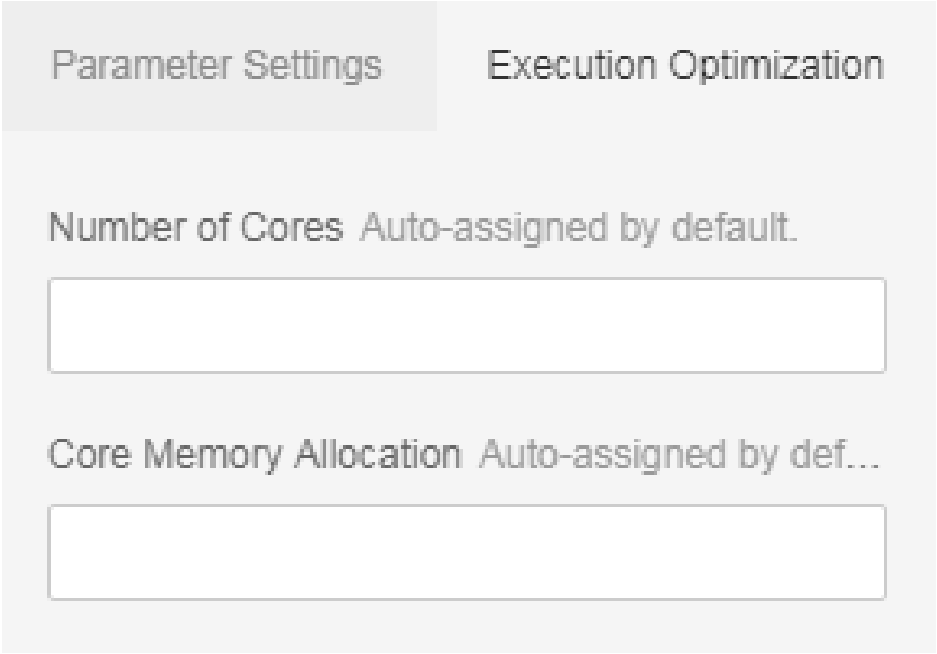
- An error is reported if sampleSize and sampleRatio are both left blank.
- If sampleSize and sampleRatio are both set, sampleSize prevails.

Random sampling

Data is randomly sampled. Each sampling is independent.

Parameter settings





- You can manually enter the number of samples (or the sampling ratio).
- You can choose whether to enable sampling with replacement, which is disabled by default. To enable it, select the checkbox.
- The random seed can be configured. By default, it is automatically assigned.
- You can configure the number of concurrent computing cores and memory size. By default, they are automatically assigned.

PAI command

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

Algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	NA	NA
inputTablePartitions	(Optional) Partitions used for training in the input table, in the format of Partition_name=value.The multilevel format is name1=value1/name2=value2. If multiple partitions	NA	All partitions in the input table

	are specified, use commas (,) to separate them.		
outputTableName	(Required) Name of the output table	NA	NA
sampleSize	(Optional) Number of samples	NA	Null by default
sampleRatio	(Optional) Sampling ratio in the range of (0, 1)	NA	Null by default
replace	(Optional) Indicates whether to enable sampling with replacement. The type is boolean.	true / false	false (disabled by default)
randomSeed	(Optional) Random seed	Positive integer	Automatically generated by default
lifecycle	(Optional) Life cycle of the output table	Positive integer in the range of [1,3650]	No life cycle for the output table
coreNum	(Optional) Number of computing cores	Positive integer	Automatically assigned
memSizePerCore	(Optional) Memory size of each core, in MB	Positive integer in the range of (1, 65536)	Automatically assigned

Note:

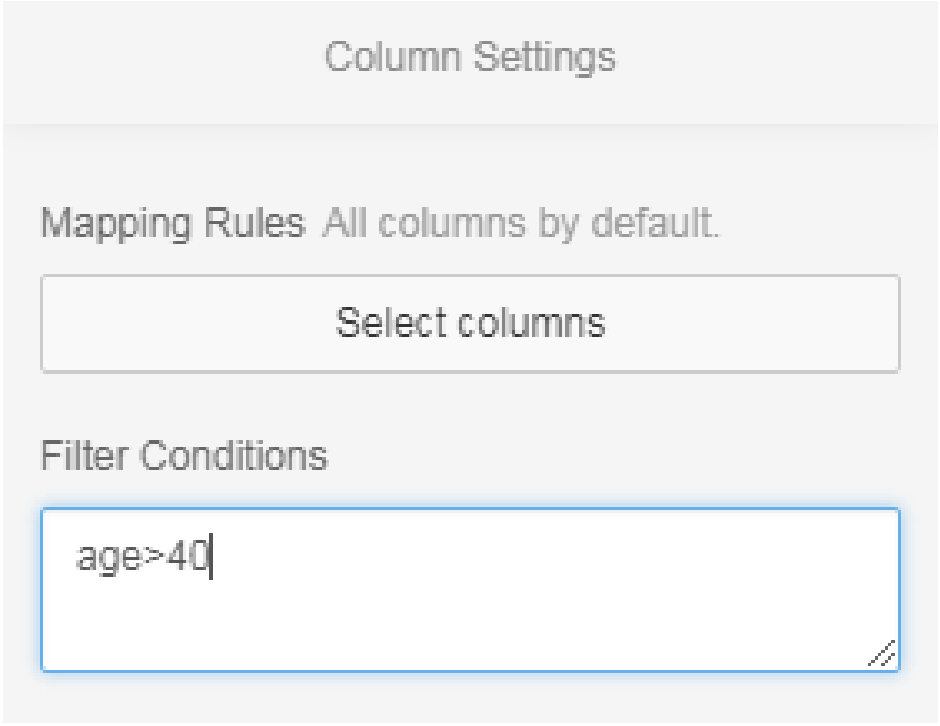
- An error is reported if sampleSize and sampleRatio are both left blank.
- If sampleSize and sampleRatio are both set, sampleSize prevails.

Filtering and mapping

Data can be filtered according to the filtering expression. Fields can be renamed.

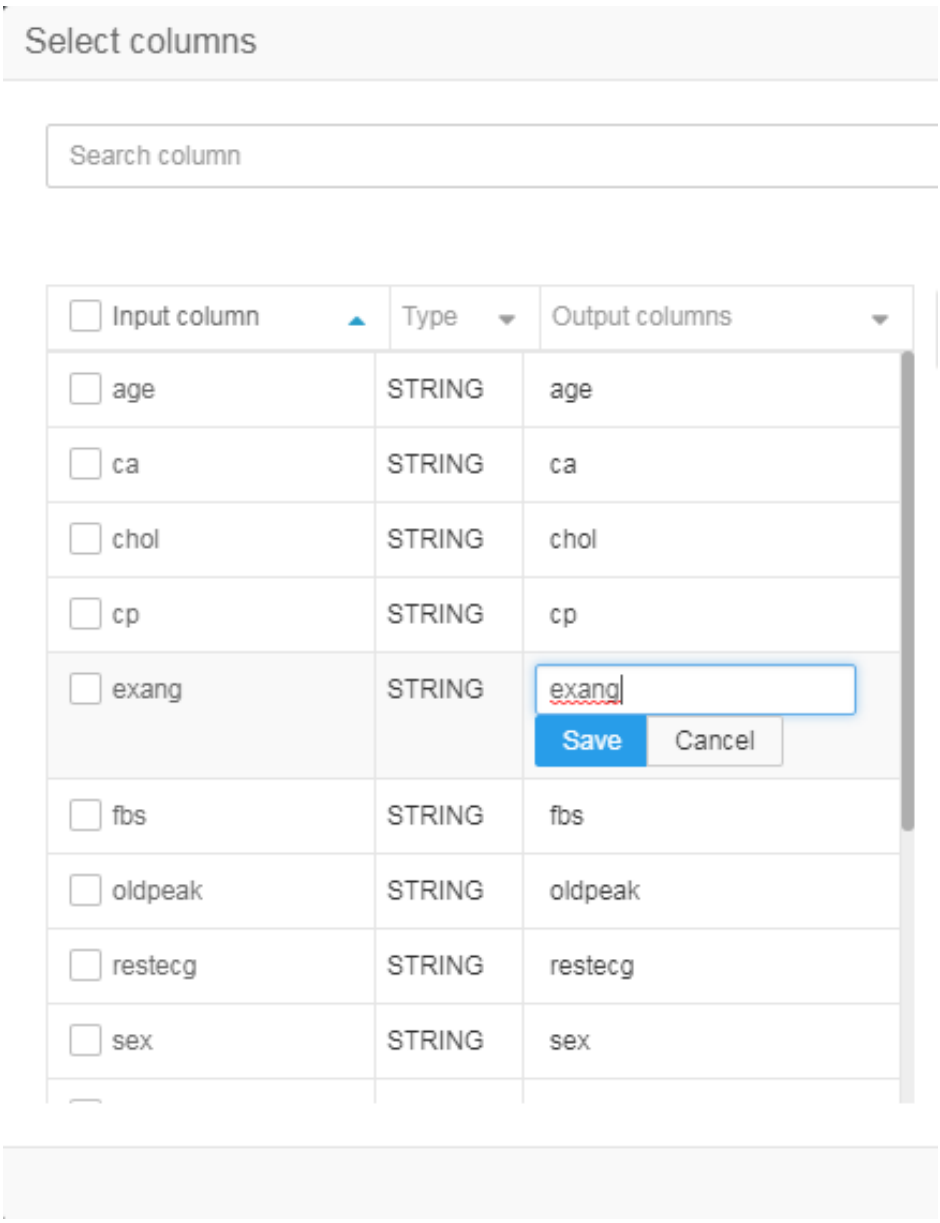
Parameter settings

Use the WHERE conditions to filter data, which is similar to SQL statements, for example:



Filtering conditions: The following operators are supported “=”, “!=”, “>”, “<”, “>=”, “<=”, “like”, and “rlike”.

Rename fields.



PAI command

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

- name: Component name.
- project: Project name used to specify the space of an algorithm. The default value is algo_public. If you change the name, the system reports an error.
- outTableName: Name of the output table.
- inputPartitions: (optional) Partitions in the training input table Input partitions corresponding

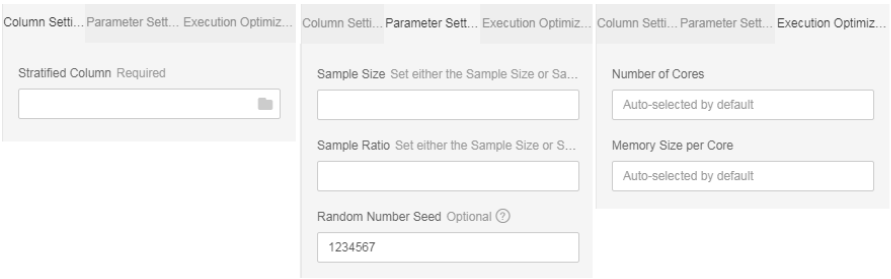
to the input table. If the entire table is selected, the value is None.
- inputTableName: Name of the input table.
- filter: WHERE filtering conditions. The following operators are supported “=”, “!=”, “>”, “<”, “>=”, “<=”, “like”, and “rlike”.

Stratified sampling

Random samples of certain data or a certain proportion of data are collected from the data set.

Parameter settings

Parameter box



- Select the group column from the drop-down list (a maximum of 100 groups are supported).
- You can manually enter the number of samples of the group (or the sampling ratio).
- You can configure the random seed. The default value is 1234567.
- You can configure the number of concurrent computing cores and memory size. By default, they are automatically assigned.

PAI command

```
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
```

Algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	NA	NA
inputTablePartitions	(Optional) Partitions in the training input table.	NA	The entire table is selected by default.
outputTableName	(Required) Name of	NA	NA

	the output table		
strataColName	(Required) Stratifying column. Used as the key for stratifying.	NA	NA
sampleSize	(Optional) Sample size. If the value is an integer, it indicates the number of samples of each stratum. If the value is a string, the format is strata0:n0,strata1:n1, which indicates the number of samples needs to be configured for each stratum.	NA	NA
sampleRatio	(Optional) Sampling ratio. If the value is a number, the value range is (0,1), and the value indicates the sampling ratio of each stratum. If the value is a string, the format is strata0:r0,strata1:r1, which indicates the sampling ratio needs to be configured for each stratum.	NA	NA
randomSeed	(Optional) Random seed	NA	Default value: 123456
lifecycle	(Optional) Life cycle of the output table	NA	This parameter is not set by default.
coreNum	(Optional). Number of cores	NA	By default, the value is automatically assigned.
memSizePerCore	(Optional) Memory size used by each core	NA	By default, the value is automatically assigned.

Note:

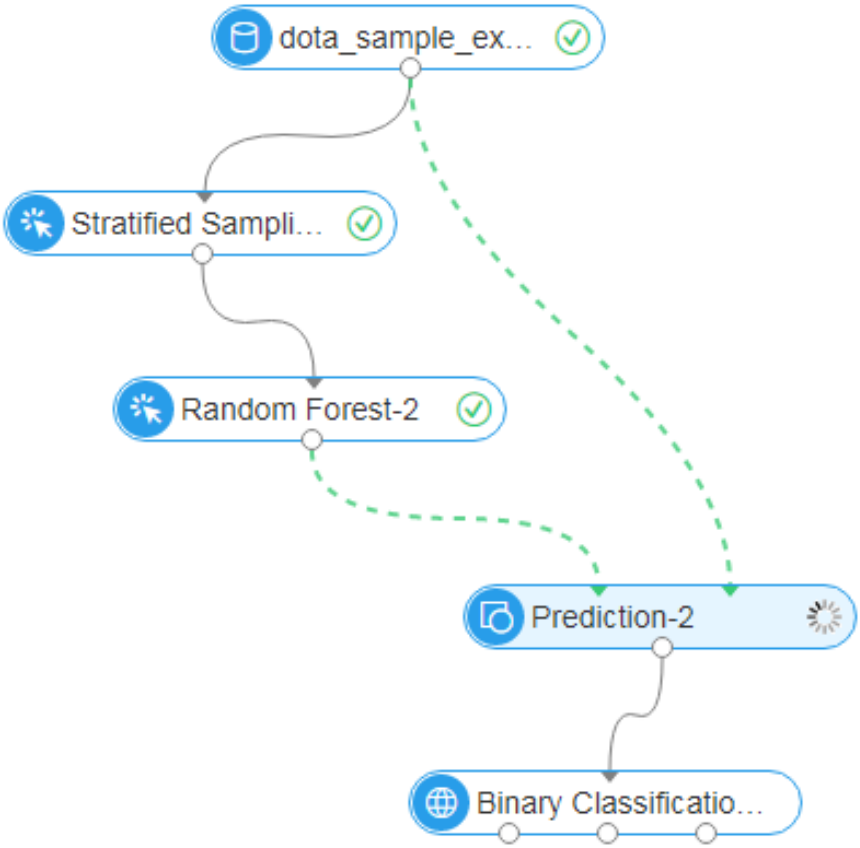
- An error is reported if sampleSize and sampleRatio are both left blank.
- If sampleSize and sampleRatio are both set, sampleSize prevails.

Example

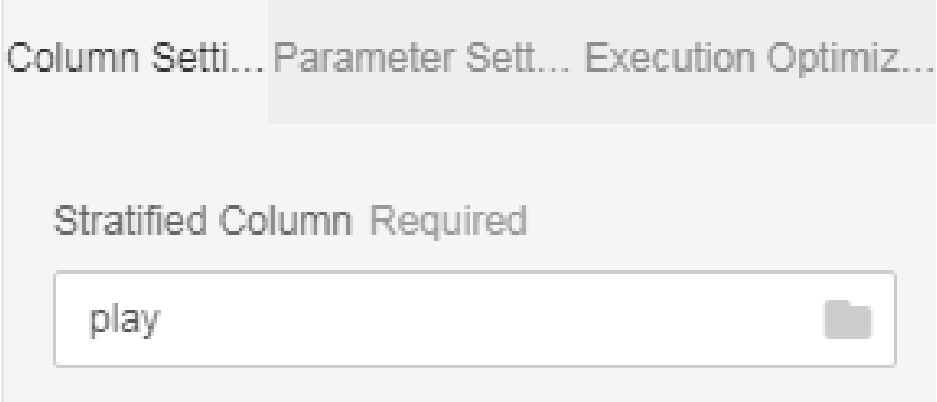
Source data

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

Create an experiment.

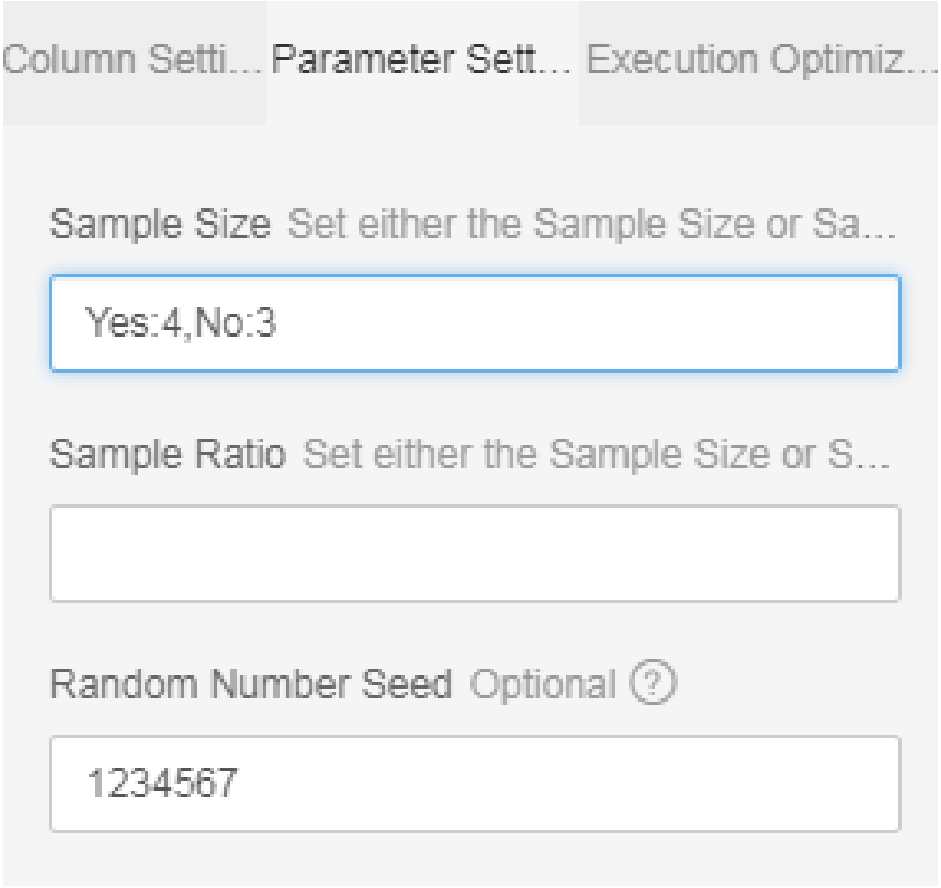


Select the group column.



Select the play column as the group column.

Configure the number of samples.



Collect four samples from the group of “play=Yes” and three samples from the group of “play=No”.

Sampling result.

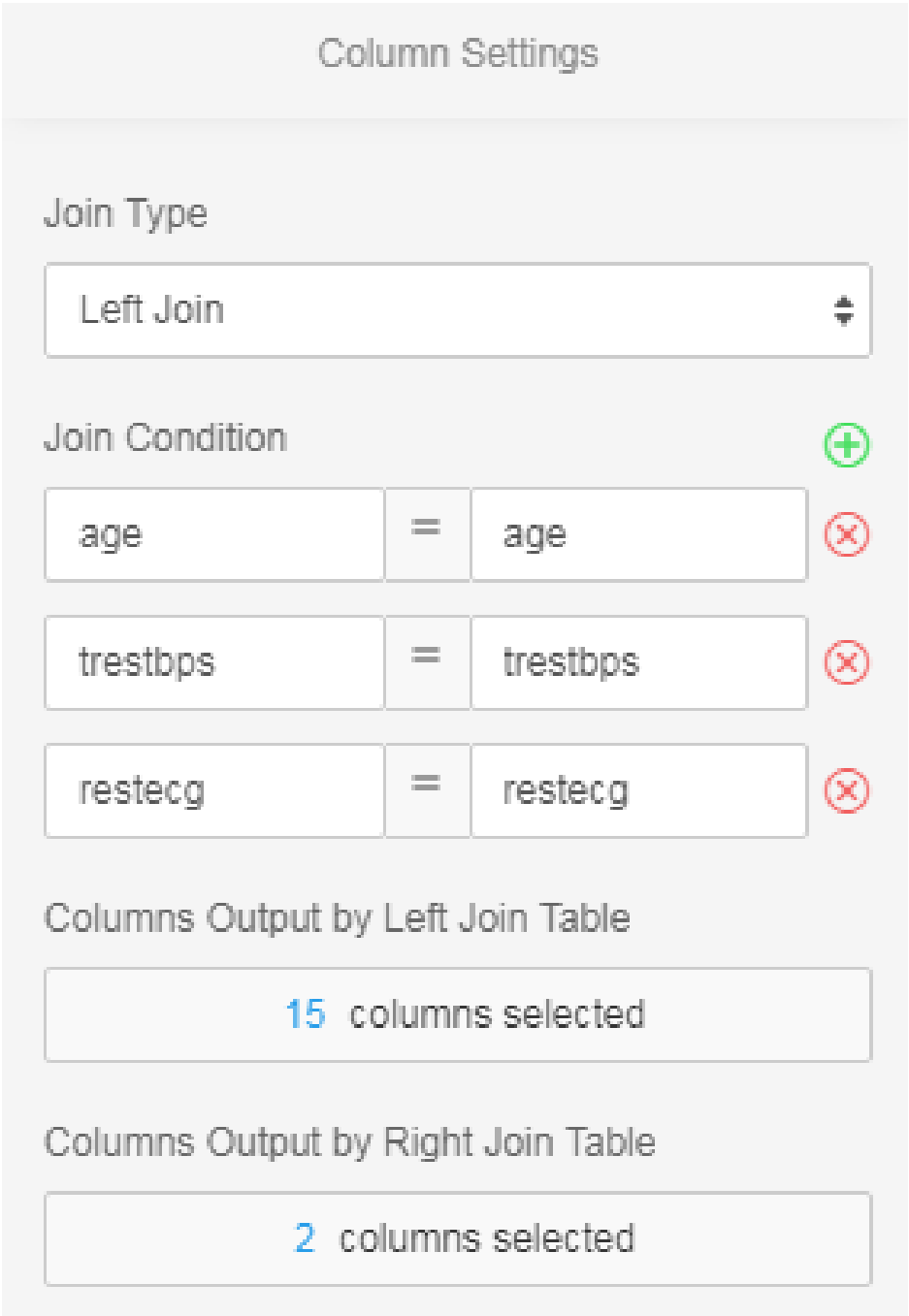
id	outlook	temperature	humidity	windy	play
1	Sunny	80	90	true	No
5	Rainy	65	70	true	No
9	Rainy	75	80	false	Yes
10	Sunny	75	70	true	Yes
11	Overcast	72	90	true	Yes
12	Overcast	81	75	false	Yes
13	Rainy	71	91	true	No

join

Through association information, two tables are merged into one table, and the output fields are determined. This is similar to the join statement of SQL.

Parameter settings

Set the parameters



- Supported connection types: left connection, internal connection, right connection, and full connection.
- The association condition must be equations.
- You can manually add or delete association conditions.

PAI command

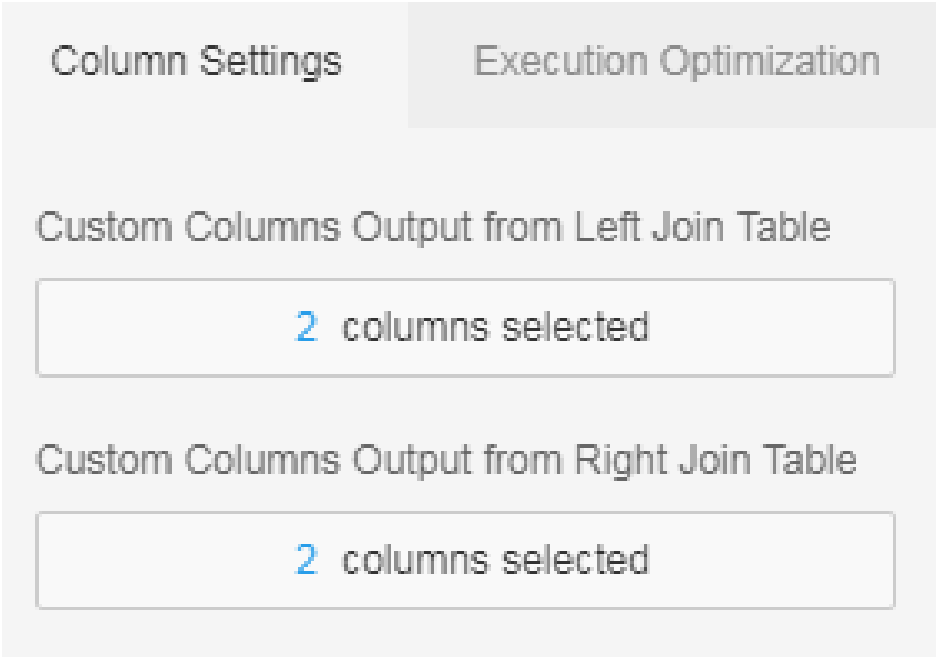
No PAI command is provided.

Merge columns

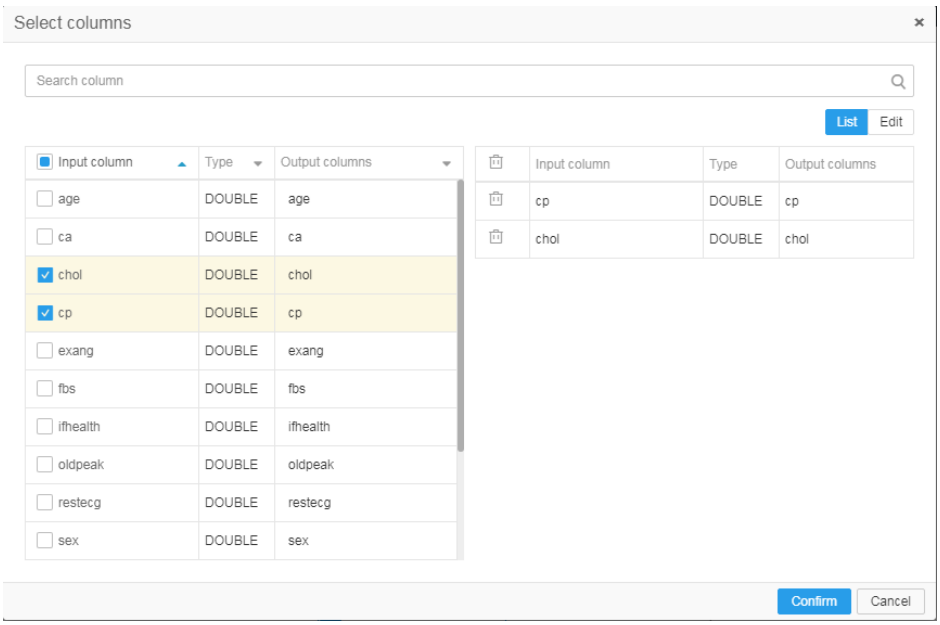
Merge the data in two tables by column. The two tables must have the same number of rows.

Parameter settings

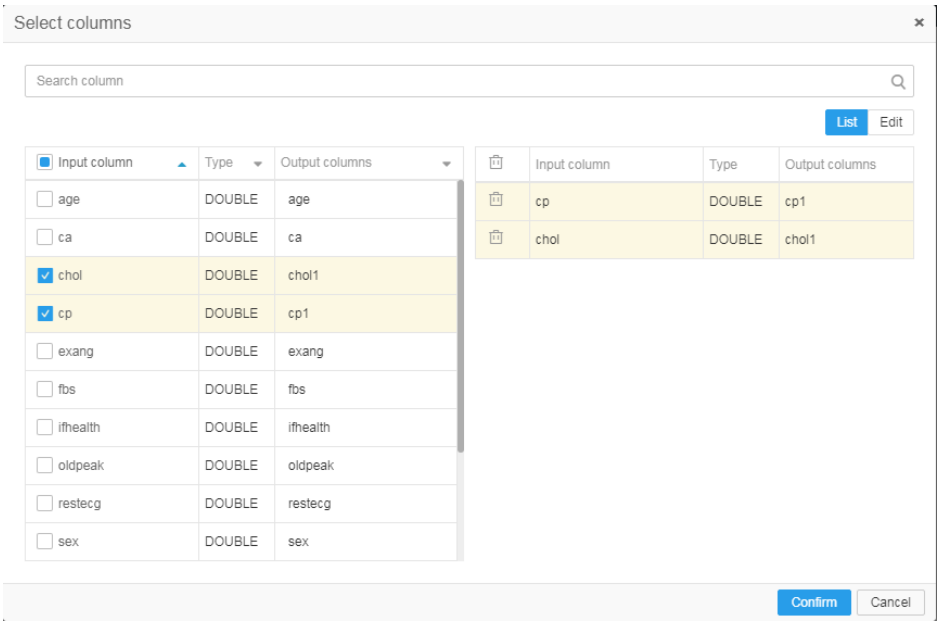
See in the following figure:



Select input columns from the left table.



Select input columns from the right table.



The two tables selected must have the same number of rows.

The names of output columns selected from the left and right tables cannot be the same.

When selecting an output field column, you can change the output field name.

If no output columns are selected from the left or right table, the full table is output by default. If **Whether to Automatically Rename the Output Column** is selected, the duplicate column is renamed and then output.

PAI command

```
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: Component name.

project: Project name, used to specify the space of an algorithmThe default value is algo_public. If you change the name, the system reports an error.

outputTableColNames: Names of the columns in the new table Names are separated by commas (,). If autoRenameCol is set to true, this parameter is invalid.

autoRenameCol: (optional) Indicates whether to automatically rename the columns in the output table. If the value is true, the columns are renamed. If the value is false, the columns are not renamed. The default value is false.

outputTableName: Name of the output table.

inputTableNames: Name of the input table. If multiple input tables are provided, use commas (,) to separate the names.

inputPartitionsInfoList: (optional) List of the selected partitions corresponding to the input table. Partitions of the same table are separated by commas (,) and partitions of different tables are separated by semicolons (;).

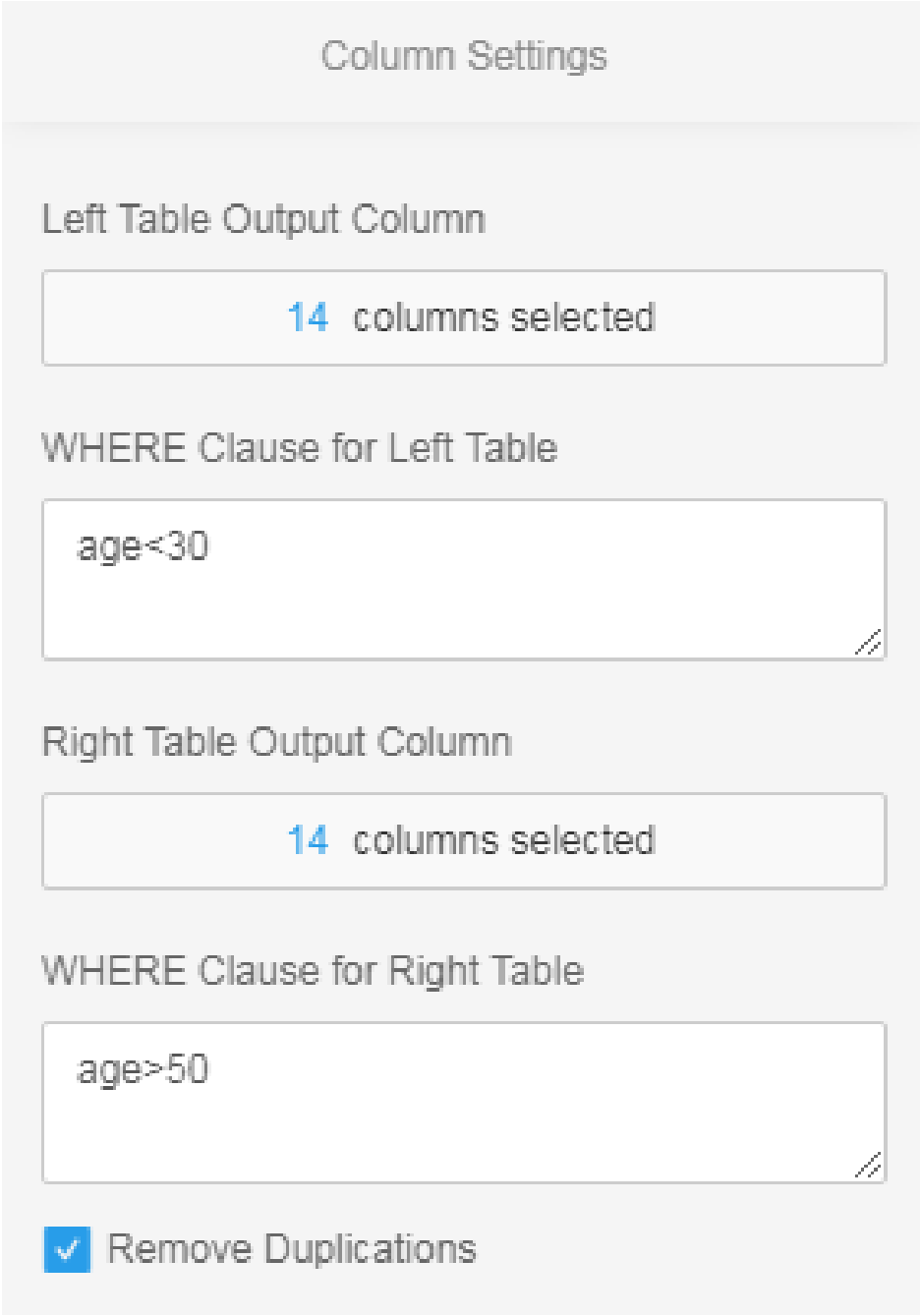
selectedColNamesList: Names of columns selected for input. Names of columns in the same table are separated by commas (,) and names of columns in different tables are separated by semicolons (;).

UNION

To merge the data in two tables by row, the numbers and types of the fields selected for output from the left and right tables must be the same. The functions of union and union all are integrated.

Parameter settings

Adjust the parameters as follows:



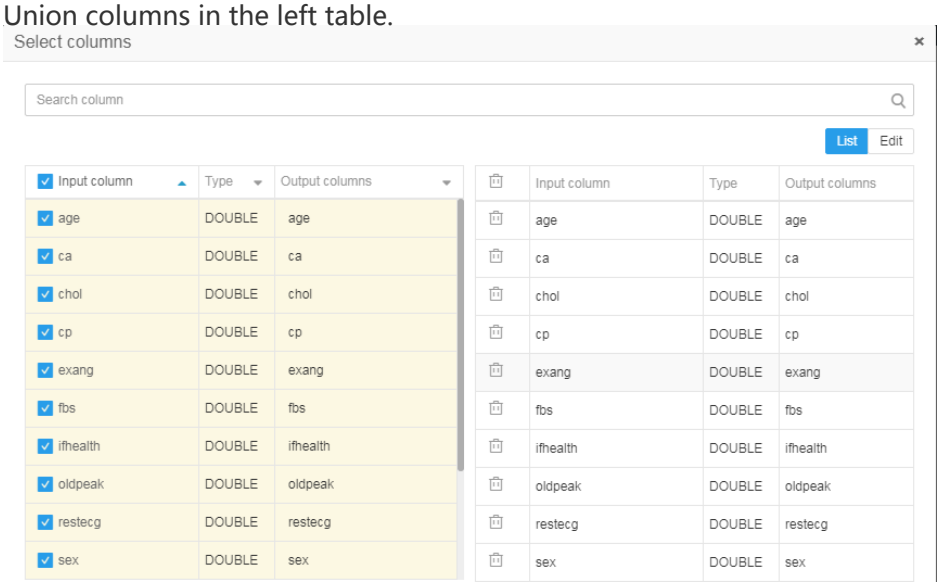
During the union operation, the numbers of rows selected from the left and right tables must be the same, and the types of corresponding columns must be the same.

You can manually enter the filtering condition for the selected fields in the condition box based on requirements (full table by default). The following operators are supported: “=”, “!=”, “>”, “<”, “>=”, “<=”, “like”, and “rlike”.

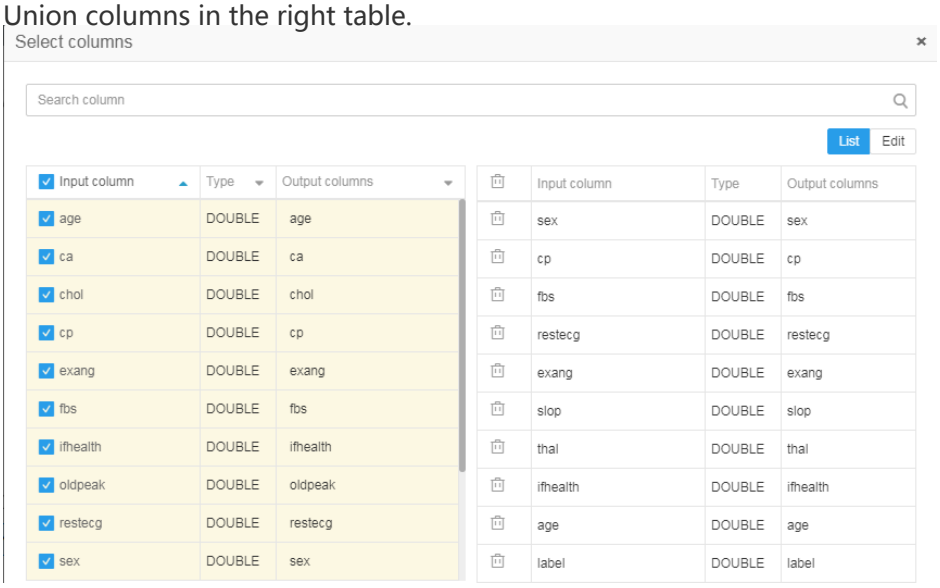
Distinct is selected by default. After this parameter is selected, duplicate rows in the

generated data table are eliminated.

Union columns in the left table.



Union columns in the right table.



PAI command

No PAI command is provided.

Append ID columns

Append an ID column at the first column of the data table and save the content as a new table. The type of the appended ID column is bigint.

Parameter settings

Skipped

PAI command

```
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: Component name.

project: Project name, used to specify the space of an algorithmThe default value is algo_public. If you change the name, the system reports an error.

IDColName: Name of the appended ID column, which is numbered from 0 and 1, 2, 3, and so on.

outputTableNames: Names of the output tables.

inputTableName: Name of the input table.

selectedColNames: Names of the fields to be retained. If multiple fields are selected, separate them with commas (,).

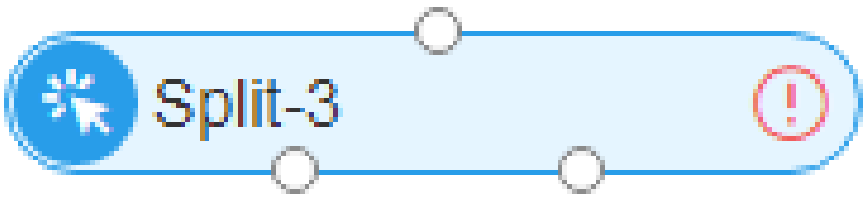
Split

Background

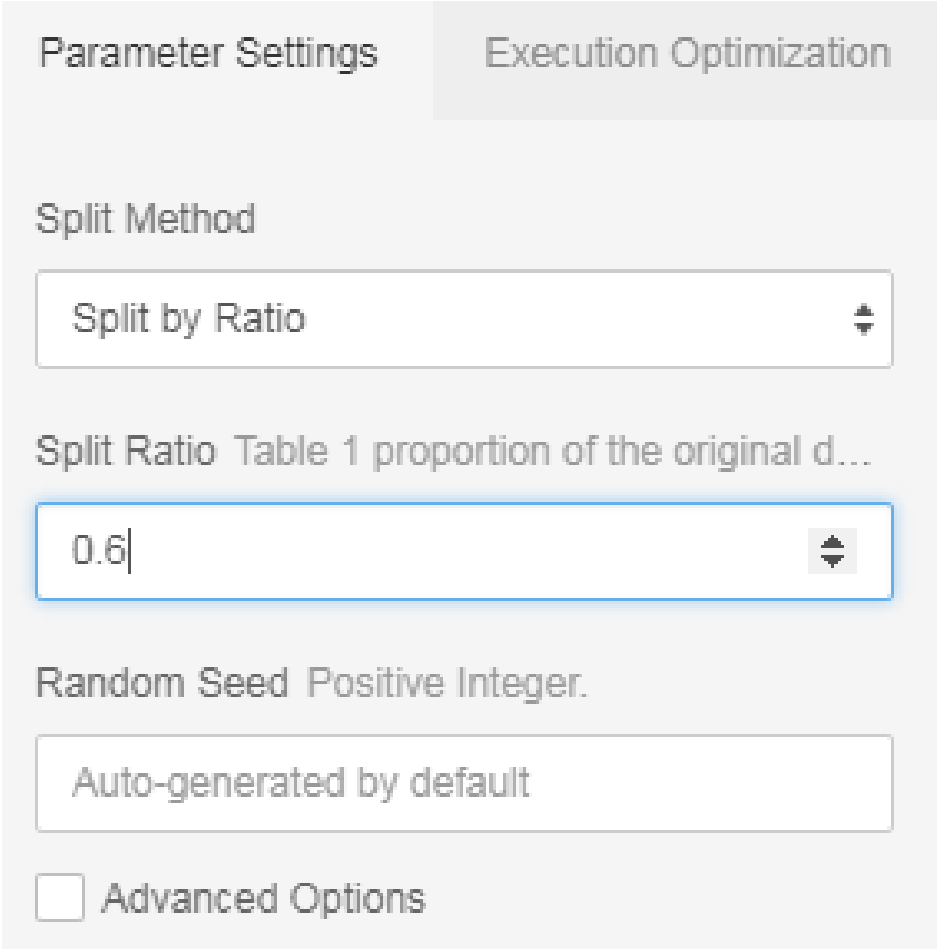
Split an input table or a partition by a certain rate, and write into two output tables.

Algorithm component

Split the component, corresponding to two output columns.



As shown in the figure, if the rate is 0.6, the output column on the left accounts for 60% of data, and the output column on the right accounts for 40% of data.



PAI command

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

Parameter settings

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	NA	NA
inputTablePartitions	(Optional) Partitions used for training in the input table, in the format of Partition_name=value. The multilevel	NA	All partitions in the input table

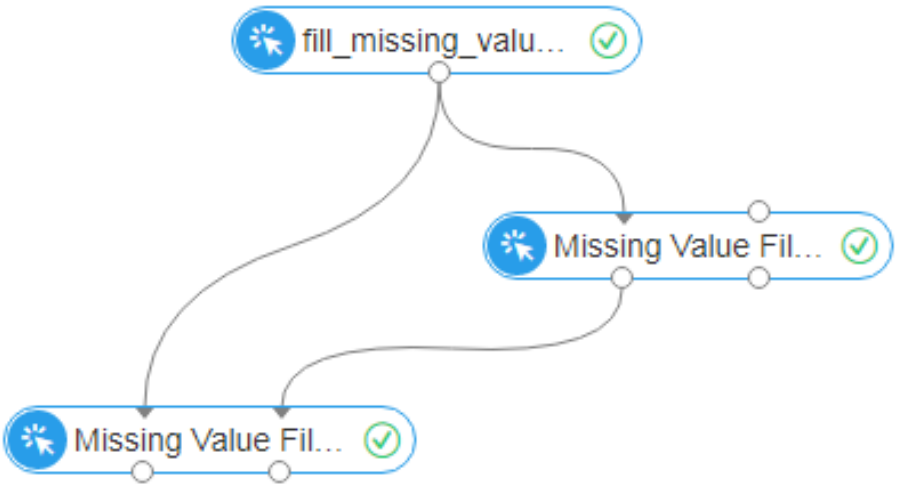
	format is name1=value1/name2=value2. If multiple partitions are specified, use commas (,) to separate them.		
output1TableName	(Required) Name of output table 1	NA	NA
output1TablePartition	(Optional) Partitions in output table 1	NA	Output table 1 is not partitioned.
output2TableName	(Required) Name of output table 2	NA	NA
output2TablePartition	(Optional) Partitions in output table 2	NA	Output table 2 is not partitioned.
fraction	(Required) Proportion of data allocated to output table 1 after splitting	(0,1)	NA
lifecycle	(Optional) Life cycle of the output table	Positive integer in the range of [1,3650]	No life cycle for the output table

Fill in missing values

Replace a null or specified value with the maximum, minimum, average, or custom value. A configuration list for missing values is defined to fill the input table with missing values that are specified.

- Replace a numeric null with the maximum, minimum, average, or custom value.
- Replace a null or empty string or a character-type null or empty string with a custom value.
- The missing values to be filled in can be null, empty characters, or custom values.
- If you set the missing values to be empty characters, the type of the target column to be filled must be string.
- You can define the numeric replacement, or directly choose to replace it with the maximum, minimum, or average value.

UI for filling in missing values



The two input columns correspond to the following parameters respectively:

inputTableName: Name of the input table to be filled.

inputParaTableName: Name of the configuration input table, that is, the parameter list generated by the missing value filling node. Based on this parameter, configuration parameters in one table can be applied to a new table.

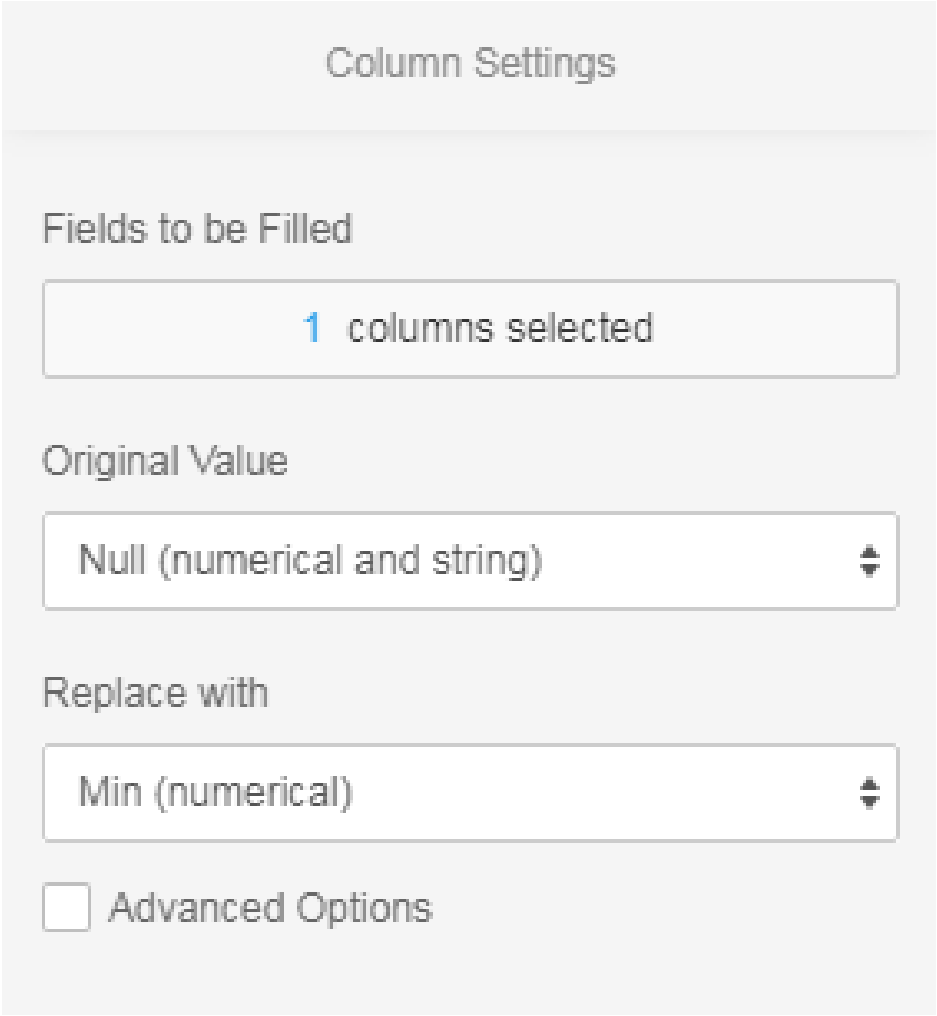
The two output columns correspond to the following parameters respectively:

outputTableName: Name of the output table after being filled.

outputParaTableName: Name of the output parameter table, which is applied to other datasets.

UI of missing value filling parameters

The filled field, original value, and new value form the config parameter and correspond to the column name, original value, and new value parts of the config parameter.



PAI command

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

Algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	NA	NA
inputTablePartitions	(Optional) Partitions in the training input table.	NA	The entire table is selected by default.
outputTableName	(Required) Name of the output table	NA	NA
configs	(Required)	NA	NA

	Configuration of missing value filling Example format: “col1, null, 3.14; col2, empty, hello; col3, empty-null, world”, of which, null indicates a null value, and empty indicates empty characters. If you set the missing values to be empty characters, the type of the target column to be filled must be string. If the maximum, minimum, or average value is used, a variable can be used. The naming format is min, max, and mean. If the replaced value is customized, use a custom value in the format of “col4,user-defined,str,str123”.		
outputParaTableName	(Required) Configuration output table	NA	NA
inputParaTableName	(Optional) Configuration input table	NA	No input by default
lifecycle	(Optional) Life cycle of the output table	NA	This parameter is not set by default.
coreNum	(Optional). Number of cores	NA	By default, the value is automatically assigned.
memSizePerCore	(Optional) Memory size used by each core	NA	By default, the value is automatically assigned.

Example

Test data

SQL statement for data generation

```
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;
```

Input data description

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

Running command

```
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";
```

Running result

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

Normalization

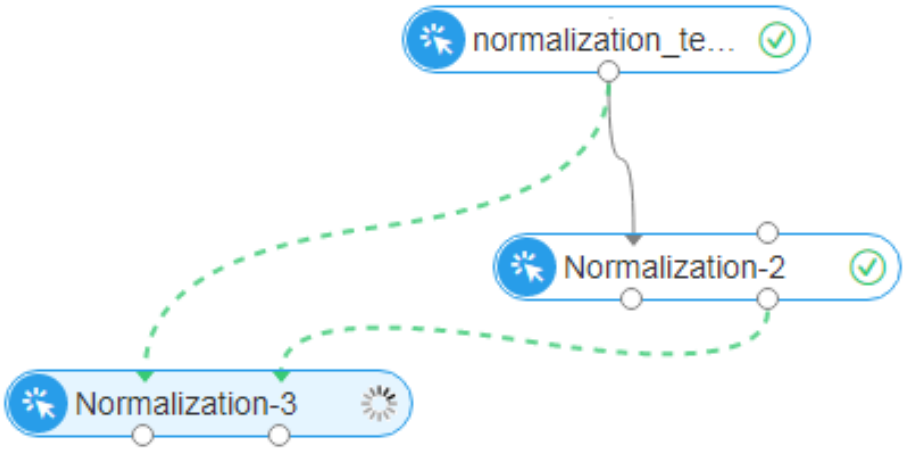
Normalize one or multiple columns in a table and save the generated data in a new table.

Linear function transformation is supported, and the expression is y=(x-MinValue)/(MaxValue-MinValue), where MaxValue and MinValue respectively indicate the maximum value and minimum value of the samples.

You can choose whether to retain the original columns. If you select the corresponding check box, the original columns are retained, and processed columns are renamed.

Click the **Select Field** and select the columns to be normalized. The double and bigint types are supported.

Normalization UI



The two input columns correspond to the following parameters respectively:

inputTableName: Name of the input table to be normalized.

inputParaTableName: Name of the configuration input table, that is, the parameter list generated by the normalization node. This parameter can be used to apply configurations of one table to a new table.

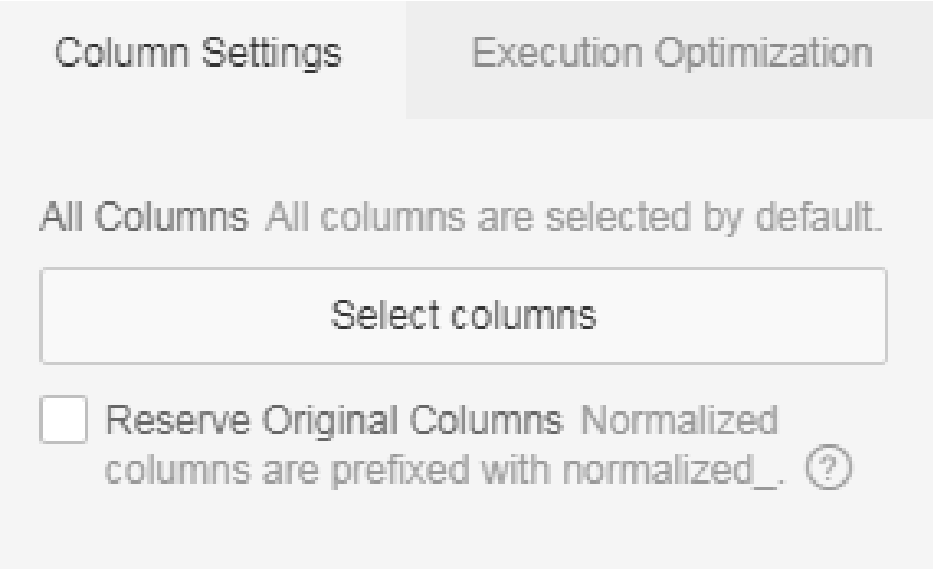
The two output columns correspond to the following parameters respectively:

outputTableName: Name of the output table after normalization.

outputParaTableName: Name of the output parameter table, which is applied to other datasets.

Normalization parameter UI

The keepOriginal parameter specifies whether to retain the original columns.



PAI command

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

Algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	NA	NA
selectedColNames	(Optional) Names of the columns selected from the input table	NA	All columns are selected by default.
inputTablePartitions	(Optional) Partitions in the training input table.	NA	The entire table is selected by default.
outputTableName	(Required) Name of	NA	NA

	the output table		
outputParaTableName	(Required) Configuration output table	NA	NA
inputParaTableName	(Optional) Configuration input table	NA	No input by default
keepOriginal	(Optional) Indicates whether to retain original columns. If keepOriginal=true, processed columns are renamed with the normalized_prefix, and the original columns are retained. If keepOriginal=false, all columns are retained but not renamed.	NA	Default value: false
lifecycle	(Optional) Life cycle of the output table	NA	This parameter is not set by default.
coreNum	(Optional). Number of cores	NA	By default, the value is automatically assigned.
memSizePerCore	(Optional) Memory size used by each core	NA	By default, the value is automatically assigned.

Example

Test data

SQL statement for data generation

```
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;
```

Input data description

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

Running command

```
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";
```

Running result

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

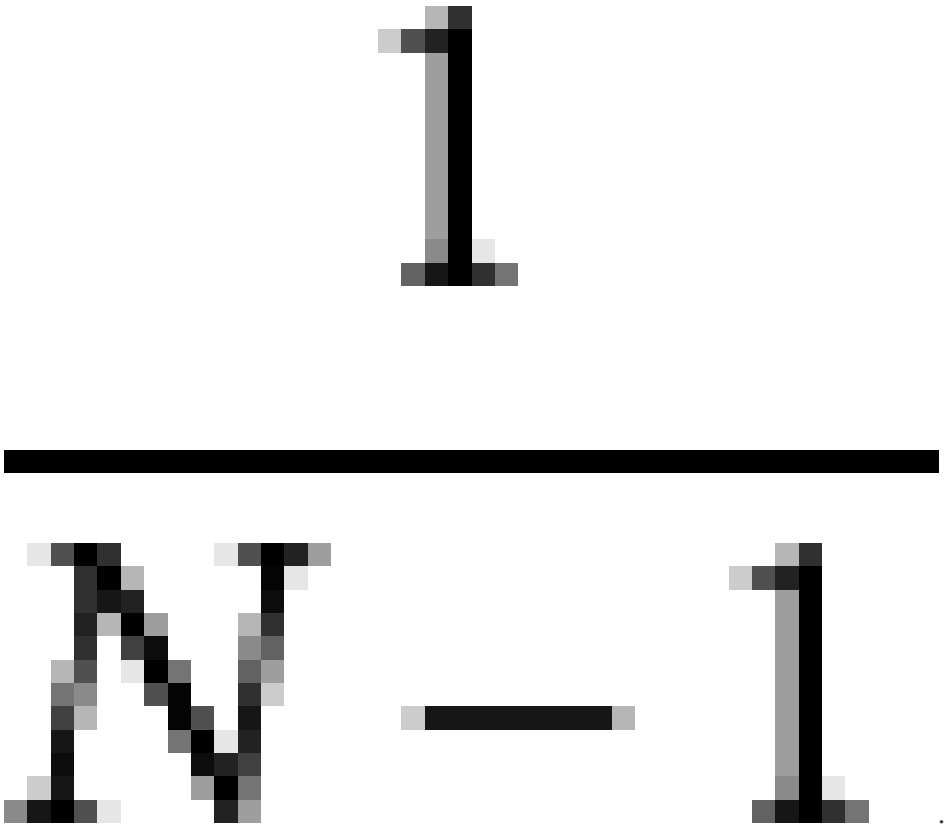
Standardization

Standardize one or multiple columns in a table and save the generated data in a new table.

The formula used for standardization is (X - Mean)/(Standard deviation).

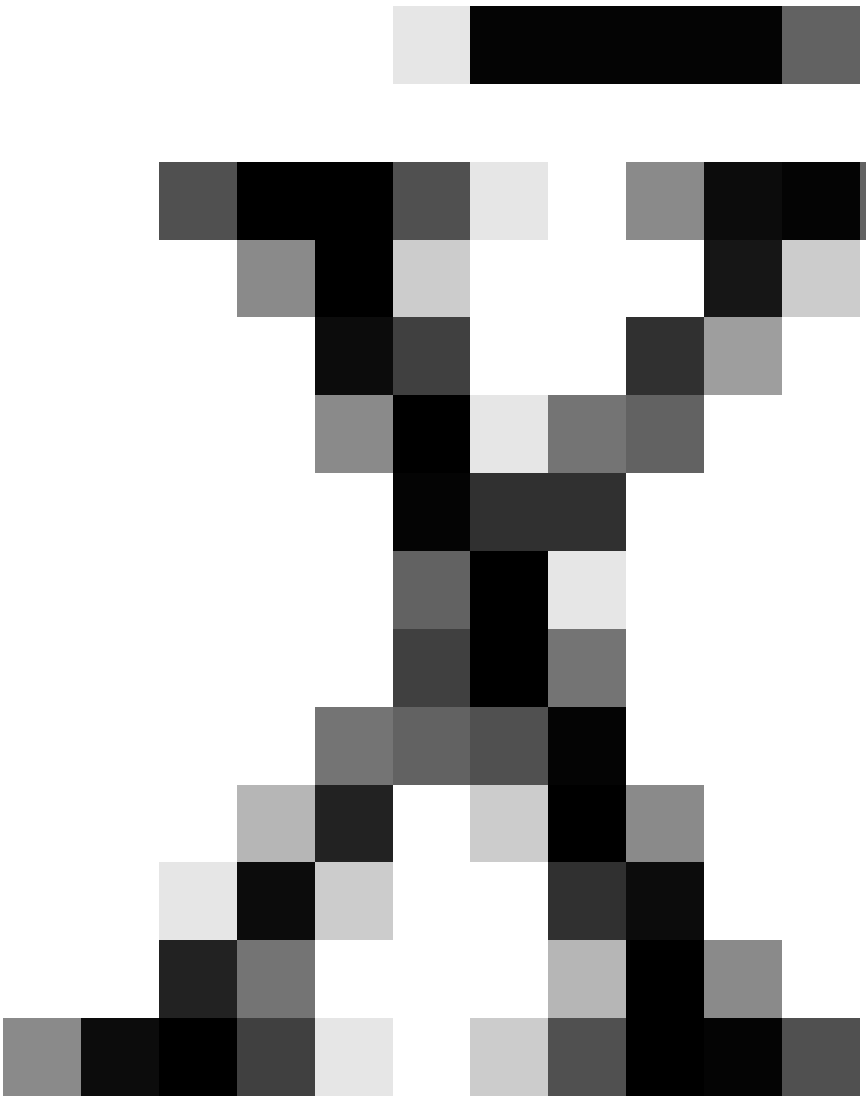
Mean: Average value of samples.

Standard deviation: Standard sample deviation. This field is used when samples are selected for a whole, and the samples are used to calculate the total deviation. To make the calculated value closer to the overall level, you must moderately increase the calculated value of standard deviation, that is,



Formula of standard sample deviation:

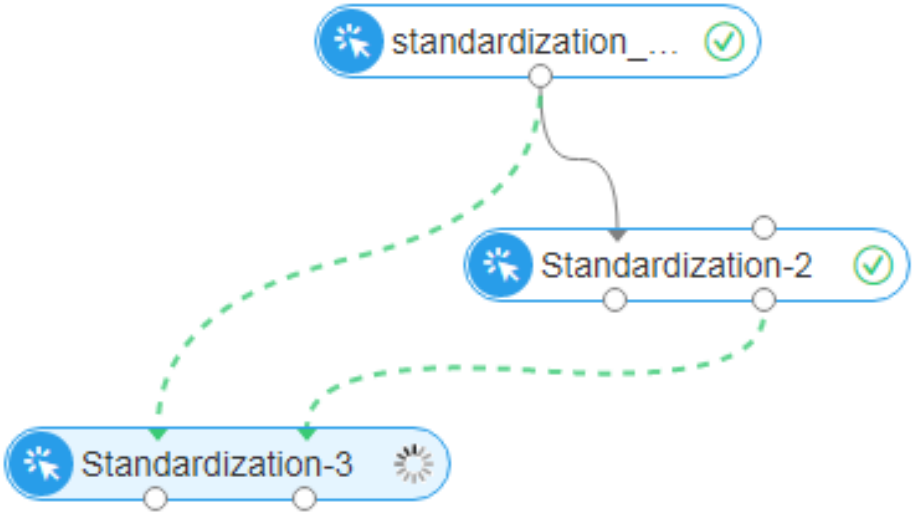
$$S = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(X_i - \bar{X})^2}$$

 indicates the average value of the used samples X1, X2,..., Xn.

You can choose whether to retain the original columns. If you select the corresponding check box, the original columns are retained, and processed columns are renamed.

Click the **Select Field** and select the columns to be standardized. The double and bigint types are supported.

Standardization UI



The two input columns correspond to the following parameters respectively:

inputTableName: Name of the input table to be standardized.

inputParaTableName: Name of the input parameter table, that is, the parameter list generated by the standardization node. Configuration parameters in the table can be applied to a new table based on this parameter.

The two output columns correspond to the following parameters respectively:

outputTableName: Name of the output table after standardization.

outputParaTableName: Name of the output parameter table, which is applied to other datasets.

UI of standardization parameters

The keepOriginal parameter specifies whether to retain the original columns.

PAI command

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

Algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	NA	NA
selectedColNames	(Optional) Names of the columns selected from the input table	NA	All columns are selected by default.
inputTablePartitions	(Optional) Partitions in the training input table.	NA	The entire table is selected by default.
outputTableName	(Required) Name of the output table	NA	NA
outputParaTableName	(Required) Configuration output table	NA	NA
inputParaTableName	(Optional) Configuration input table	NA	No input by default
keepOriginal	(Optional) Indicates whether to retain	NA	Default value: false

	original columns. If keepOriginal=true, processed columns are renamed with the stdized_prefix, and the original columns are retained. If keepOriginal=false, all columns are retained but not renamed.		
lifecycle	(Optional) Life cycle of the output table	NA	This parameter is not set by default.
coreNum	(Optional). Number of cores	NA	By default, the value is automatically assigned.
memSizePerCore	(Optional) Memory size used by each core	NA	By default, the value is automatically assigned.

Example

Test data

SQL statement for data generation

```
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;
```

Input data description

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

Running command

```
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";
```

Running result

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

Type conversion

Convert the types of table fields.

PAI command

```
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"
```

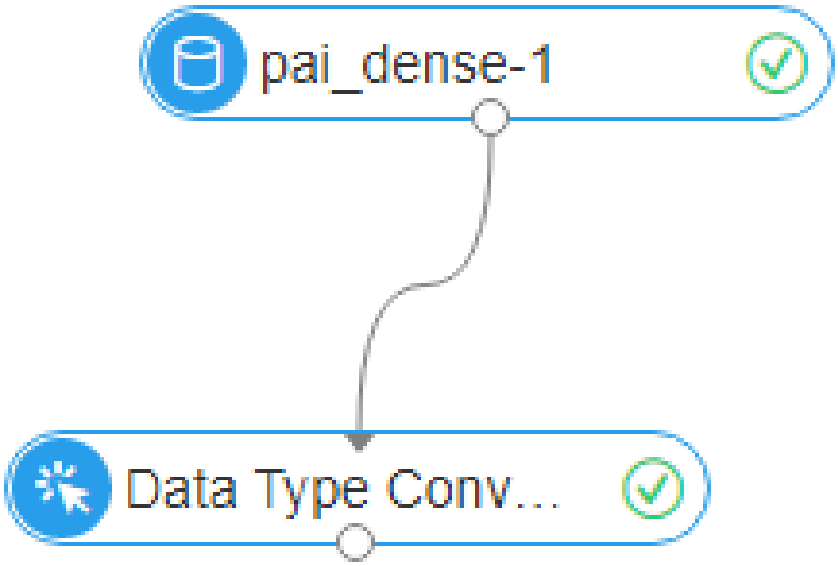
Algorithm parameters

Parameter	Description	Option	Default value
inputTable	(Required) Name of the input table	NA	NA
inputTablePartitions	(Optional) Partitions used for training in the input table, in the format of Partition_name=value.The multilevel format is name1=value1/name2=value2. If multiple partitions are specified, use	NA	All partitions in the input table

	commas (,) to separate them.		
outputTable	(Required) Name of the type conversion result table	NA	NA
selectedCols	(Required) Feature columns that require type conversion. The type must be the same.	NA	NA
pre_type	Original field type. The type must be the same as the field type selected in selectedCols; otherwise, an error of field type inconsistency is reported.	NA	NA
new_type	New field type set by the user	NA	100
lifecycle	(Optional) Life cycle of the outputTable. Default value: 7	NA	7

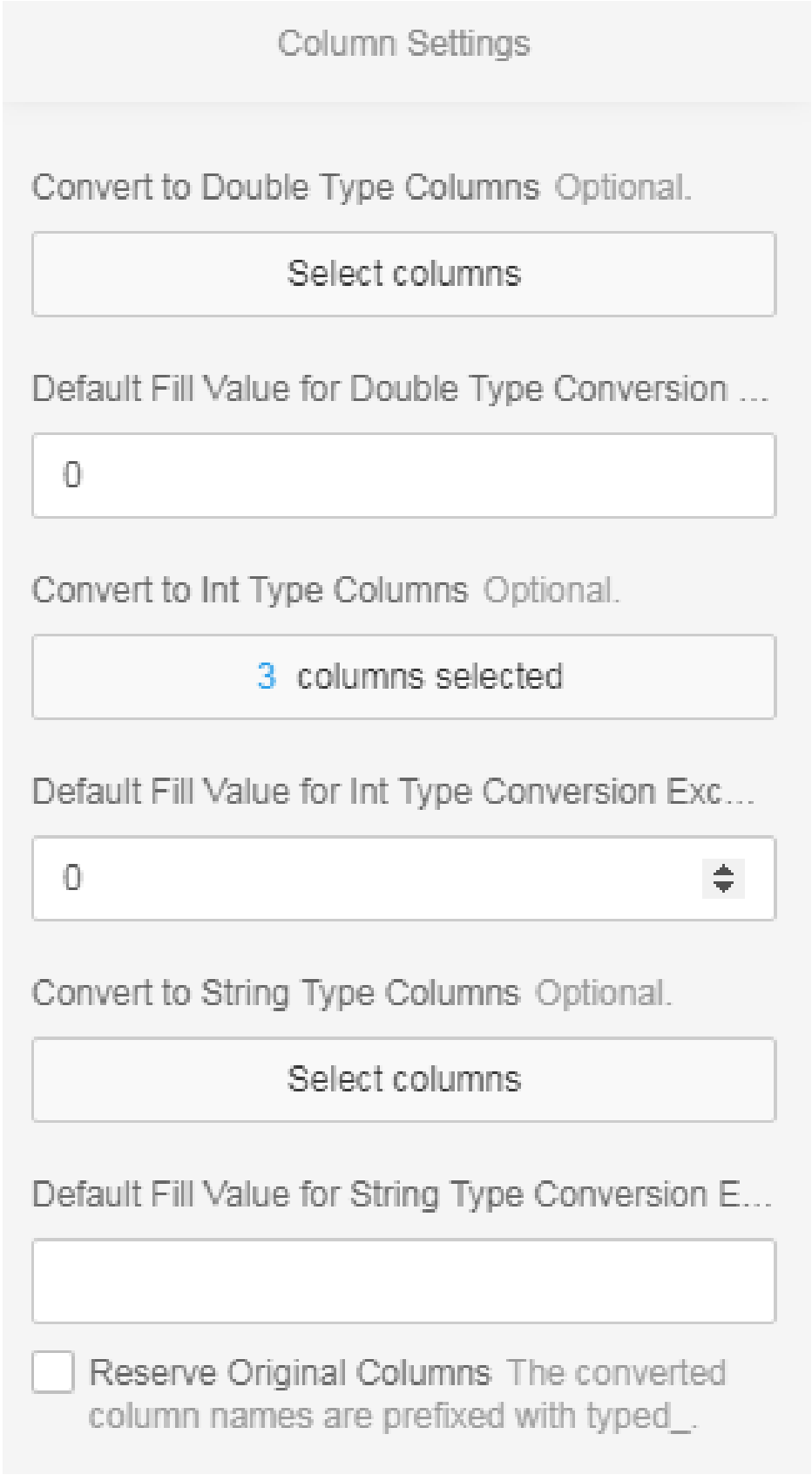
Use demo

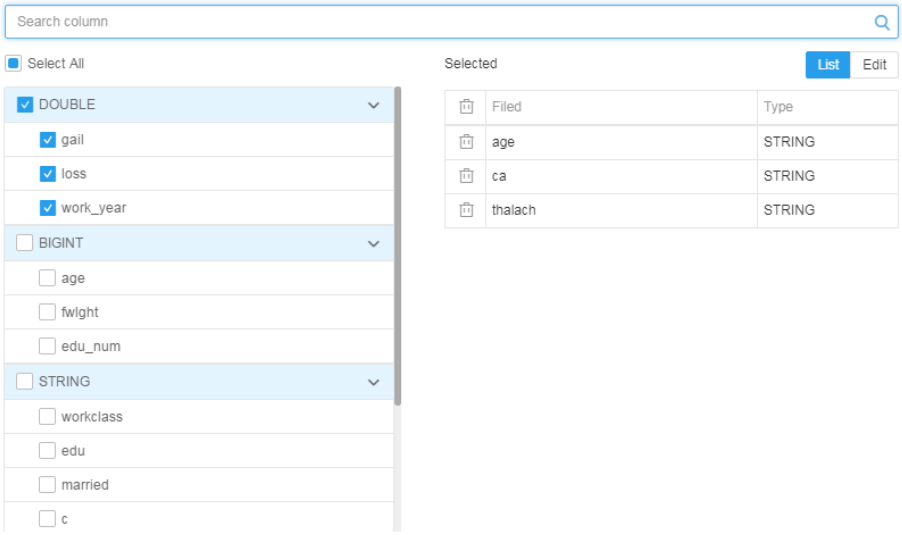
Drag a data table reading component and configure the data table pai_dense as follows:



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		

Drag a type conversion component and select the feature to be converted in the parameter configuration column on the right. For example, in the following figure, select three columns whose original data type is double (the double type must be the same as the selected field type) and convert them into the bigint type.


Column Settings
Convert to Double Type Columns Optional.
Select columns
Default Fill Value for Double Type Conversion ...
0
Convert to Int Type Columns Optional.
3 columns selected
Default Fill Value for Int Type Conversion Exc...
0
Convert to String Type Columns Optional.
Select columns
Default Fill Value for String Type Conversion E...
Reserve Original Columns The converted column names are prefixed with typed_.



Right-click the **Run**. The field type is changed in the output table.

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

Convert the specified kv (key:value) format into the common table format. The key is converted into a column name of the table, and the value is converted into the value of the column in the corresponding row.

KV table format definition: Key is the column name index, and the value type can be bigint or double. This sparse format allows input of algorithm components such as logical regression and linear regression. The key type can also be string. The custom key_map table (mapping of column names and keys) can be imported into this component. Regardless of whether this table is imported, this component outputs the mapping of column names and keys after the key_map table is converted.

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

KeyMap table format definition: A triple table that includes mapping of column names and indexes

and type information. The type of col_name, col_index, and col_datatype must be string, and the default value of col_datatype is double if it is not provided.

col_name	col_index	col_datatype
col1	1	bigint
col2	2	double

PAI command

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

Parameter description

Parameter	Description	Value range	Required/Optional, default value
inputTableName	Name of the input table	NA	(Required) The table cannot be empty.
kvColName	kv column name	Select only one column	Required
outputTableName	Name of the output table	NA	Required
outputKeyMapTableName	Name of the output index table	NA	Required
inputKeyMapTableName	Name of the input index table	NA	Optional, “”
appendColName	Name of the appended column	Multiple columns can be selected	Optional, “”
inputTablePartitions	Input table partitions	NA	Optional, “”
kvDelimiter	Delimiter between the key and the value	NA	(Optional) Default value: “:”
itemDelimiter	Delimiter between key-value pairs	NA	(Optional) Default value: “,”
top1200	Indicates whether to truncate the first 1200 columns	NA	(Optional) The default value is true. If the value is false, an error is reported when the number of columns exceeds the

			maximum value.
lifecycle	Life cycle	An integer equal to or more than -1	(Optional) Default value: -1, which indicates that life cycle is not set.
coreNum	Number of cores	An integer more than 0	(Optional) Default value: -1, which indicates that the number of started instances is determined based on the input data volume.
memSizePerCore	Memory size	(100,64*1024)	(Optional) Default value: -1, which indicates that the memory size is determined based on the input data volume.

Example

Data generation

```
    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;
```

Running command

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

Output description

Output table

kv_1	kv_2	kv_3
1.0	2.0	-3.3

10.0	20.0	-33.3

Output mapping table

col_name	col_index	col_type
kv_1	1	double
kv_2	2	double
kv_3	3	double

Algorithm scale

The converted columns include those converted from the append and kv columns. The kv column is output before the append column. When the number of columns exceeds the maximum ODPS column number, the maximum number of columns are output if top1200 is set to True; otherwise, an error is reported. The maximum ODPS column number is 1200.

The number of data records cannot exceed 100 million.

Important notes

- If the key_map table is input, the converted columns are the keys that exist in both the key_map table and the key-value table.
- The converted column type must be numeric.
- If the key_map table is input, the type of the converted key column must be the same as that in the key_map table. If the key_map table is not input, the type of the converted key column is double.
- If the key_map table is not input, the name format of the converted key column is kv column name+" "+key. If the key contains any of "%,&()*+-./;<>=?", an error is reported.
- Column name conflict: an error is reported if the append column is specified, and the append column name is the same as the converted key column name.
- If a row contains duplicate keys, the values are added.
- If the column name contains more than 128 characters, only the first 128 characters are kept.

Table2KV

Convert common tables (table) into tables of the kv format. Null values in the original table are not reflected in the new table. You can specify columns that need to be retained in the new table. The specified columns are retained in their original formats.

PAI command

```
PAI -name TableToKV
-project algo_public
```

```
-DinputTableName=maple_tabletokv_basic_input
-DoutputTableName=maple_tabletokv_basic_output
-DselectedColNames=col0,col1,col2
-DappendColNames=rowid;
```

Parameter description

Parameter	Description	Value range	Required/Optional, default value
inputTableName	Input table	Table name	Required
inputTablePartitions	Partitions used for training in the input table, in the format of partition_name=value. The multilevel partition name format is name1=value1/name2=value2. If you specify multiple partitions, separate them with a comma (,).	NA	(Optional) Default: all partitions
selectedColNames	Selected column names. The type must be bigint or double.	Column name	(Optional) Select the entire table.
appendColNames	Column to be retained. It is written in the output table in its original format.	Column name	(Optional) Default value: null
outputTableName	Name of the output KV table	Table name	Required
kvDelimiter	Delimiter between the key and the value	Character	(Optional) Default value: ":"
itemDelimiter	Delimiter between key-value pairs	Character	(Optional) Default value: ","
convertColToIndexId	Indicates whether to convert columns into numbers. 1: yes; 0: no	0 or 1	(Optional) Default value: 0
inputKeyMapTableName	Input index table name. This parameter is valid only when convertColToIndexId=1. If this parameter is not set, a set of	Table name	(Optional) Default value: ""

	IDs are automatically generated.		
outputKeyMapTableName	Output index table name. This parameter is required only when convertColToIndexId=1.	Table name	Specified by convertColToIndexId
lifecycle	(Optional) Life cycle of the output table	Positive integer	No life cycle
coreNum	Number of nodes	Used together with the memSizePerCore parameter, positive integer in the range of [1, 9999]	(Optional) Automatically calculated by default
memSizePerCore	Memory size of each node, in MB	Positive integer in the range of [1024, 64*1024]	(Optional) Automatically calculated by default

Example

Data generation

rowid	kv
0	col0:1,col1:1.1,col2:2
1	col0:0,col1:1.2,col2:3
2	col0:1,col1:2.3
3	col0:1,col1:0.0,col2:4

Running command

```
PAI -name TableToKV
-project algo_public
-DinputTableName=maple_tabletokv_basic_input
-DoutputTableName=maple_tabletokv_basic_output
-DselectedColNames=col0,col1,col2
-DappendColNames=rowid;
```

Output description

Output table:
maple_tabletokv_basic_output

rowid:bigint	kv:string
0	1:1.1,2:2
1	1:1.2,2:3

2	1:2.3
3	1:0.0,2:4

Example 2

Running command

```
PAI -name TableToKV
-project projectxlib4 -DinputTableName=maple_tabletokv_basic_input
-DoutputTableName=maple_tabletokv_basic_output
-DselectedColNames=col0,col1,col2 -DappendColNames=rowid
-DconvertColToIndexId=1
-DinputKeyMapTableName=maple_test_tabletokv_basic_map_input
-DoutputKeyMapTableName=maple_test_tabletokv_basic_map_output;
```

Output description

Output table:
maple_test_tabletokv_basic_map_output

col_name:string	col_index:string	col_datatype:string
col1	1	bigint
col2	2	double

Important notes

If the key_map table is input, the converted columns are the keys that exist in both the key_map table and the key-value table.

If the key_map table is input, and the type is different from the input table, the output key_map table uses the type specified by the user.

The type of the columns that need to be converted into kv in the input table must be bigint or double.

Feature engineering

Contents

PCA

Feature scaling

Feature discretization

Feature exception smoothing

Random forest feature importance

GBDT feature importance

Linear model feature importance

Preference calculation

Filter-based feature selection

RFM

Feature encoding

One-hot encoding

Exception detection

Feature importance filtering

PCA

The principal component analysis (PCA) method is used to reduce dimensionality. For more information about the PCA algorithm, see wiki.

The algorithm supports the dense data format.

PAI command

```
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;
```

Algorithm parameter description

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the PCA input table	NA	NA
eigOutputTableName	(Required) Name of the output table of the feature vectors and feature values	NA	NA
princompOutputTableName	(Required) Name of the output table after dimensionality reduction of the principal component	NA	NA
selectedColNames	(Required) Feature columns involved in PCA operation	NA	NA
transType	(Optional) Mode of transforming the original table to the principal component table.	Simple、Sub-Mean、Normalization	Simple
calcuType	(Optional) Mode of feature breakdown for the original table.	CORR、COVAR_SAMP、COVAR_POP	CORR
contriRate	(Optional) Information retaining ratio after dimensionality reduction	(0,1)	0.9
remainColumns	(Optional) Fields retained from the original table after dimensionality reduction	NA	NA

PCA output example

Data table after dimensionality reduction.

Data exploration - pai_temp_119984_1312630_1 - (Show top one hundred rows.)

Index	prin0	prin1	prin2	prin3	prin4	prin5	prin6	prin7	prin8
1	3.9502...	8.0921...	-4.006...	3.0526...	3.9853...	3.4786...	-3.595...	-0.963...	-1.283...
2	6.5803...	7.4339...	0.6765...	4.8516...	3.2312...	2.9784...	-4.404...	-2.738...	-2.610...
3	6.5495...	5.4907...	0.1534...	4.3590...	3.4399...	2.5504...	-4.404...	-1.791...	-2.067...
4	2.3568...	5.3355...	-2.161...	1.7458...	4.8333...	3.9009...	-5.633...	0.4030...	-3.108...
5	1.1662...	6.1372...	0.2903...	3.6492...	4.6080...	3.5394...	-3.997...	-1.712...	-3.955...
6	1.2732...	5.3936...	0.0268...	4.5742...	3.1969...	3.6985...	-5.123...	-1.601...	-2.316...
7	5.3451...	7.9003...	0.2053...	2.6088...	4.0000...	2.6223...	-4.960...	-0.348...	-4.467...
8	2.0404...	6.8155...	2.2117...	4.0433...	3.7733...	5.2542...	-4.842...	-0.952...	-1.093...
9	4.9300...	5.9636...	-0.240...	4.7164...	4.0782...	2.6441...	-5.131...	-1.787...	-2.405...
10	6.1176...	5.5337...	-2.283...	3.6747...	4.7869...	5.0206...	-2.716...	-0.836...	-2.861...

Feature value and feature vector table.

Data exploration - pai_temp_119984_1312630_2 - (Show top one hundred rows.)

Index	prin_name	ifhealth	age	sex	cp	fbs	restecg	exang	slop	thal	trestbps	chol	thalach
1	prin0	0.4286188...	0.25...	0.13...	0.1...	0.06...	0.141500...	0.3130...	0.319...	0.336...	0.1513181...	0.085...	-0.35131...
2	prin1	-0.154695...	0.43...	-0.4...	-0....	0.23...	0.260407...	-0.194...	0.019...	-0.24...	0.3952703...	0.379...	-0.00041...
3	prin2	0.0551163...	0.08...	-0.3...	0.5...	-0.4...	0.010280...	0.1757...	-0.18...	-0.16...	-0.267927...	0.374...	-0.15711...
4	prin3	0.1650162...	0.06...	0.34...	0.1...	0.34...	0.110508...	-0.040...	-0.53...	0.198...	0.0838695...	0.164...	0.234805...
5	prin4	0.0490915...	-0.3...	0.13...	0.0...	-0.1...	0.627666...	0.1407...	0.104...	0.091...	0.1336500...	0.367...	0.335242...
6	prin5	-0.041022...	-0.1...	-0.2...	0.2...	0.33...	-0.48151...	0.3881...	0.006...	0.164...	0.4441276...	0.155...	0.094564...
7	prin6	-0.079481...	-0.0...	-0.0...	0.1...	0.58...	0.409439...	0.2820...	0.111...	-0.30...	-0.257934...	-0.29...	-0.23676...
8	prin7	0.0203088...	-0.2...	-0.1...	-0....	0.37...	-0.21906...	-0.159...	0.249...	0.036...	-0.493746...	0.404...	0.247308...
9	prin8	-0.125609...	0.25...	0.25...	-0....	0.03...	-0.08457...	0.2550...	-0.07...	0.165...	-0.352987...	0.424...	-0.31212...
10	prin9	0.0748853...	-0.2...	-0.1...	-0....	-0.1...	-0.05144...	0.6158...	-0.27...	-0.32...	0.0324055...	-0.05...	0.111965...
11	prin10	-0.120658...	0.12...	0.59...	0.2...	0.01...	-0.18063...	0.0080...	0.090...	-0.62...	0.1277918...	0.221...	0.020841...

Feature scaling

Supports common **scale functions** such as log2, log10, ln, abs, and sqrt. Supports dense and sparse data formats.

PAI command

```
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;
```

Algorithm parameters

Parameter	Description	Option	Default value
inputTable	(Required) Name of the input table	NA	NA

inputTablePartitions	(Optional) Partitions used for training in the input table, in the format of Partition_name=value.The multilevel format is name1=value1/name2=value2. If multiple partitions are specified, use commas (,) to separate them.	NA	All partitions in the input table
outputTable	(Required) Name of the scaling result table	NA	NA
scaleCols	(Required) Select the features that require scaling. Sparse features are automatically filtered. Only numeric features can be selected.	NA	NA
labelCol	(Optional) Label field. If this column is set, the x-y distribution histogram from the feature to the target variable will be visualized.	NA	NA
categoryCols	(Optional) Process the selected fields as enumeration features. Scaling is not supported.	NA	“”
scaleMethod	(Optional) Scaling method. Default value: log2. The following methods are supported: log2, log10, ln, abs, and sqrt.	NA	SameDistance
scaleTopN	Top N scaling features automatically selected when scaleCols is not selected. Default value: 10	NA	10
isSparse	(Optional) Indicates whether it is a	NA	100

	sparse feature in the key:value format. Default: dense data		
itemSpliter	(Optional) Delimiter between sparse feature items. Default value: “,”	NA	“,”
kvSpliter	(Optional) Delimiter between sparse feature items. Default value: “:”	NA	“:”
lifecycle	(Optional) Life cycle of the output table. Default value: 7	NA	7

Example

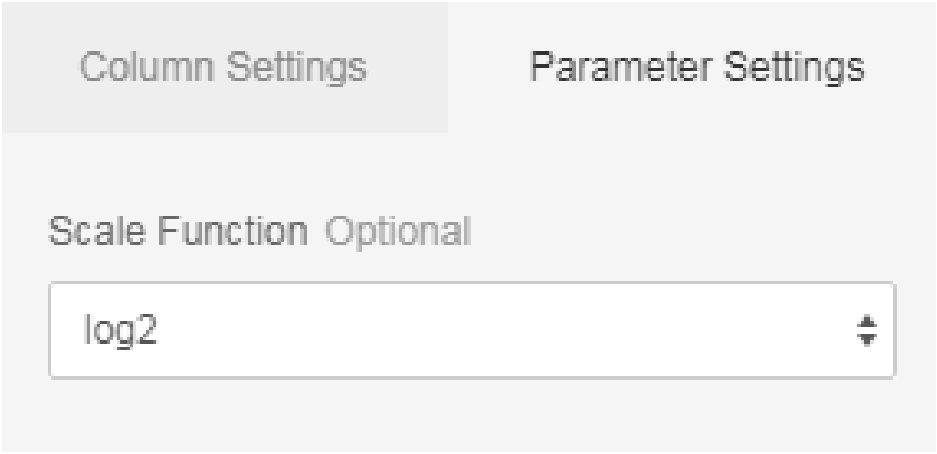
Input data

SQL statement for data generation:

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

Parameter settings

Select nr_employed for feature scaling. Only numeric features are supported.Select log2 as the scale function.



Running result

nr_employed
12.352071021075528
12.34313018339218

12.285286613666395
12.316026916036957
12.309533196497519
12.352071021075528
12.316026916036957
12.316026916036957
12.309533196497519
12.316026916036957

Feature discretization

- Supports **dense and sparse numeric feature** discretization.
- Supports same-frequency discretization and same-distance discretization (default).

PAI command

```
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;
```

Algorithm parameters

Parameter	Description	Option	Default value
inputTable	(Required) Name of the input table	NA	NA
inputTablePartitions	(Optional) Partitions used for training in the input table, in the format of Partition_name=value.The multilevel format is name1=value1/name2=value2. If multiple partitions are specified, use commas (,) to separate them.	NA	All partitions in the input table
outputTable	(Required) Name of the discretization output table	NA	NA

discreteCols	(Required) Selected features that require discretization. Sparse features are automatically filtered.	NA	“”
labelCol	(Optional) Label field. If this column is set, the x-y distribution histogram from the feature to the target variable will be visualized.	NA	NA
categoryCols	(Optional) Process the selected features as enumeration features. Discretization is not supported.	NA	“”
discreteMethod	(Optional) Discretization method. Default value: SameDistance. The options are SameDistance and SameFrequency.	NA	SameDistance
discreteTopN	Top N features that require discretization are automatically selected when discreteCols is not selected. Default value: 10	NA	10
maxBins	Discrete interval. Default value: 100	NA	100
isSparse	(Optional) Indicates whether it is a sparse feature in the key:value format. Default: dense data	NA	100
itemSpliter	(Optional) Delimiter between sparse feature items. Default value: “,”	NA	“,”
kvSpliter	(Optional) Delimiter between sparse feature items. Default value: “:”	NA	“:”

lifecycle	(Optional) Life cycle of the output table. Default value: 7	NA	7

Modeling example

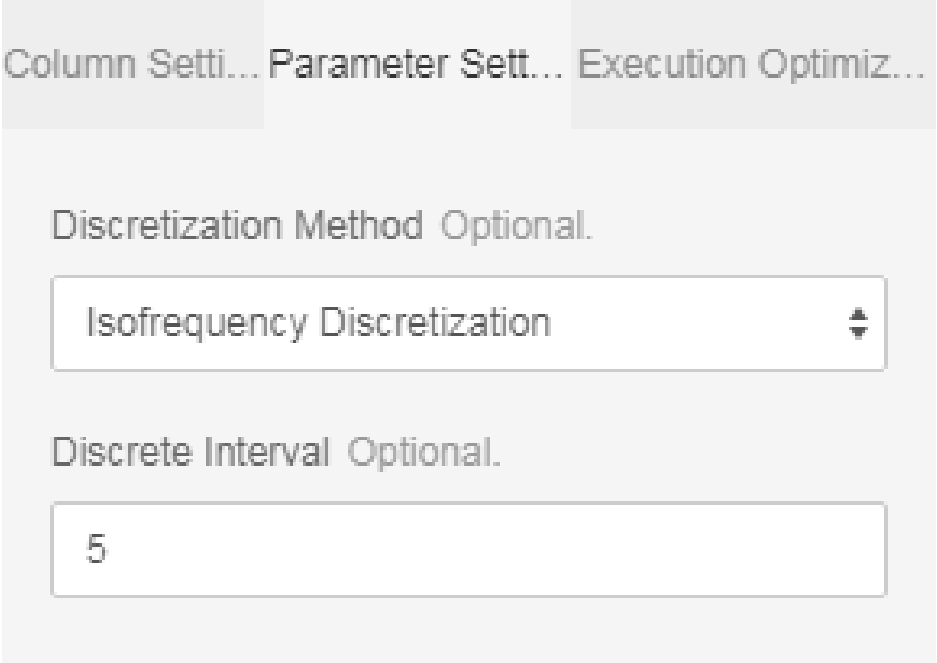
Input data

SQL statement for data generation:

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

Parameter settings

The input data is pai_dense_10_1. If nr_employed is selected for the discrete feature, use the **same-distance method** to discretize it into five intervals. The result is as follows:



Running result

nr_employed
4.0
3.0
1.0
3.0
2.0
4.0

3.0
3.0
2.0
3.0

Feature exception smoothing

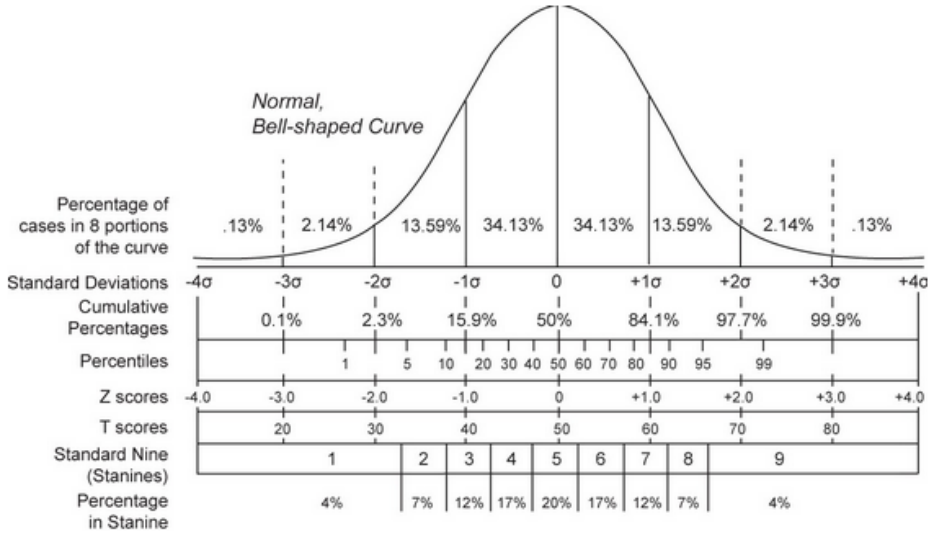
Function: Smooths exceptional data in the input feature to a specific interval. **Sparse and dense features are supported.**

Note: The feature smoothing component only rectifies abnormal values but does not filter or delete any records. Therefore, the dimensions and number of input data records remain unchanged.

Smoothing methods

- Zscore: If a feature follows normal distribution, Zscore smooths data in the rage to [-3xalpha,3xalpha] because the noise is normally out of the range of [-3xalpha,3xalpha].

Example: If a feature follows normal distribution, the mean is 0, and the standard deviation is 3, feature value -10 is determined to be abnormal and changed to -3x3+0=-9 and feature value 10 is changed to 3x3+0=9.



Percentile: smooths data distributed out of [minPer, maxPer] to the minPer and maxPer quantiles.
Example: The age feature value is in the range of 0-200. Set minPer to 0 and maxPer to 50%. Feature values out of 0-100 are changed to 0 or 100.

Threshold: smooths data distributed out of [minThresh, maxThresh] to the minThresh and

maxThresh data points.
Example: The age feature value is in the range of 0-200. Set minThresh to 10 and maxThresh to 80. Feature values out of 0-80 are changed to 0 or 80.

PAI command

```
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;
```

Algorithm parameters

Parameter	Description	Option	Default value
inputTable	(Required) Name of the input table	NA	NA
inputTablePartitions	(Optional) Partitions used for training in the input table, in the format of Partition_name=value. The multilevel format is name1=value1/name2=value2. If multiple partitions are specified, use commas (,) to separate them.	NA	All partitions in the input table
outputTable	(Required) Name of the smoothing output table	NA	NA
labelCol	(Optional) Label field. If this column is set, the x-y distribution histogram from the feature to the target variable will be visualized.	NA	NA
categoryCols	(Optional) Process the selected fields as enumeration features	NA	“”
softenCols	(Required) Selected features that require	NA	NA

	smoothing. Sparse features are automatically filtered.		
softenMethod	(Optional) Smoothing method. Default value: zscore. The options are min-max-thresh and min-max-per.	NA	zscore
softenTopN	Top N features that require smoothing are automatically selected when softenCols is not selected. Default value: 10	NA	10
cl	Confidence level. This parameter is valid when the smoothing method is zscore.	NA	10
minPer	Minimum percentile. This parameter is valid when the smoothing method is min-max-per.	NA	0.0
maxPer	Maximum percentile. This parameter is valid when the smoothing method is min-max-per.	NA	1.0
minThresh	Minimum threshold value. Default value: -9999, which indicates the minimum threshold is not set. This parameter is valid when the smoothing method is min-max-thresh.	NA	-9999
maxThresh	Maximum threshold value. Default value: -9999, which indicates the maximum threshold is not set. This parameter is valid when the smoothing method is min-max-thresh.	NA	-9999

isSparse	(Optional) Indicates whether it is a sparse feature in the key:value format. Default: dense data	NA	100
itemSpliter	(Optional) Delimiter between sparse feature items. Default value: “,”	NA	“,”
kvSpliter	(Optional) Delimiter between sparse feature items. Default value: “:”	NA	“:”
lifecycle	(Optional) Life cycle of the output table. Default value: 7	NA	7

Example

Input data

SQL statement for data generation:

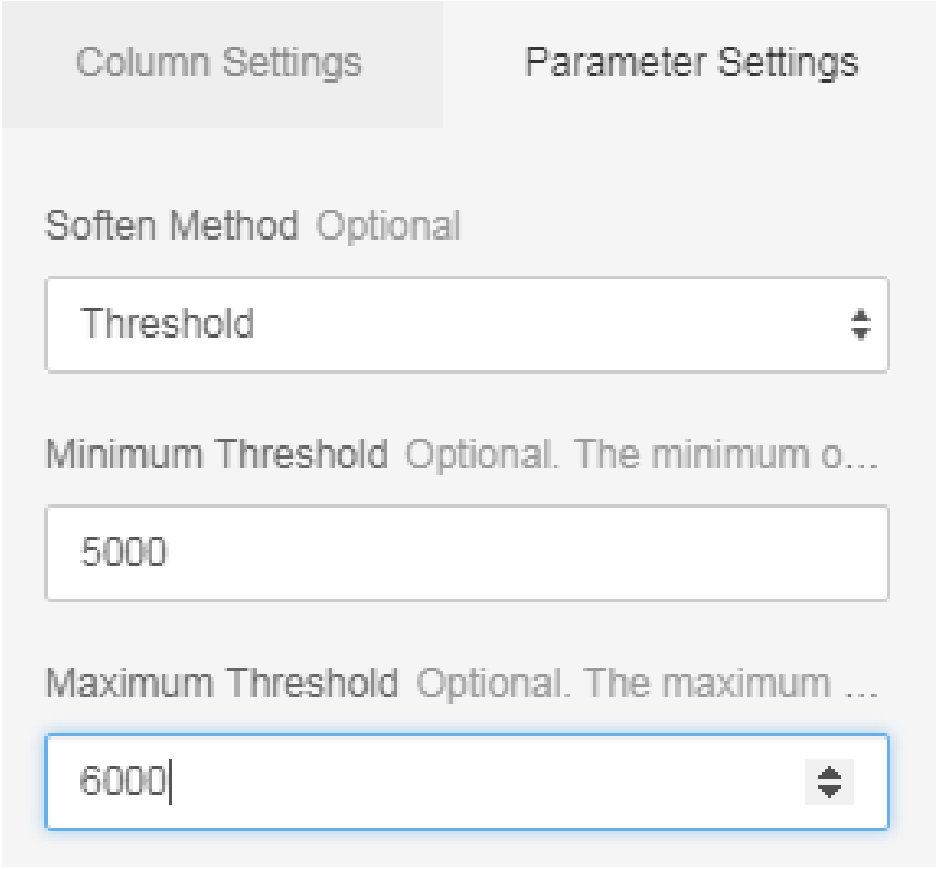
```
create table if not exists pai_dense_10_1 as
select
nr_employed
from bank_data limit 10;
```

nr_employed
5228.1
5195.8
4991.6
5099.1
5076.2
5228.1
5099.1
5099.1
5076.2
5099.1

Parameter settings

Select nr_employed for the smoothing feature column, select min-max-thresh in parameter settings,

and set the upper limit to 5000 and lower limit to 6000.



Running result

nr_employed
5228.1
5195.8
5000.0
5099.1
5076.2
5228.1
5099.1
5099.1
5076.2
5099.1

Random forest feature importance

The function of the component is to use the original data and random forest model to calculate

feature importance.

PAI command

```
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 ;
```

Algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	NA	NA
outputTableName	(Required) Name of the output table	NA	NA
labelColName	(Required) Name of the label column	NA	NA
modelName	(Required) Name of the input model	NA	NA
featureColNames	(Optional) Selected feature columns in the input table	NA	All columns are selected by default except the label column.
inputTablePartitions	(Optional) Names of the selected partitions in the input table	NA	All partitions are selected by default.
lifecycle	(Optional) Life cycle of the output table	NA	Unspecified by default
coreNum	(Optional) Number of cores	NA	Automatically calculated by default
memSizePerCore	(Optional) Size of memory	NA	Automatically calculated by default

Example

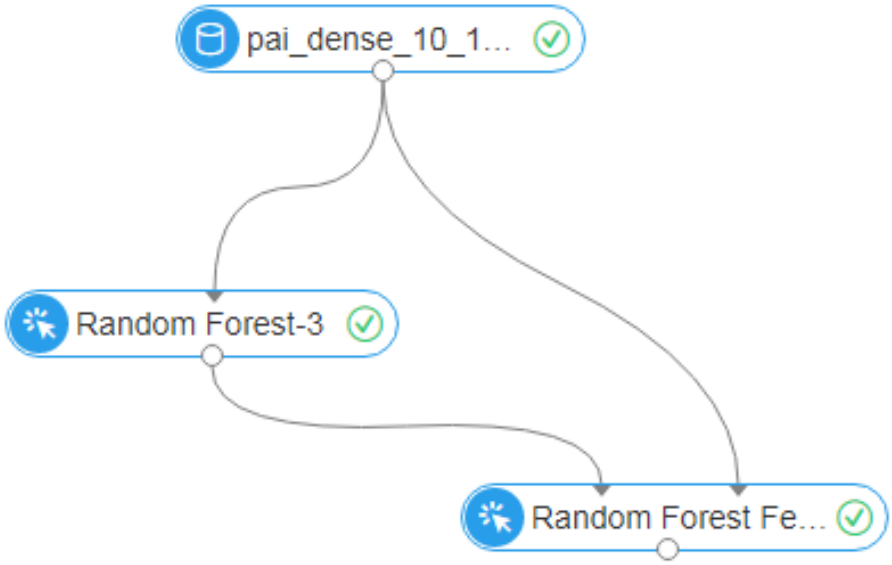
Input data

SQL statement for data generation:

```
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;
```

Parameter settings

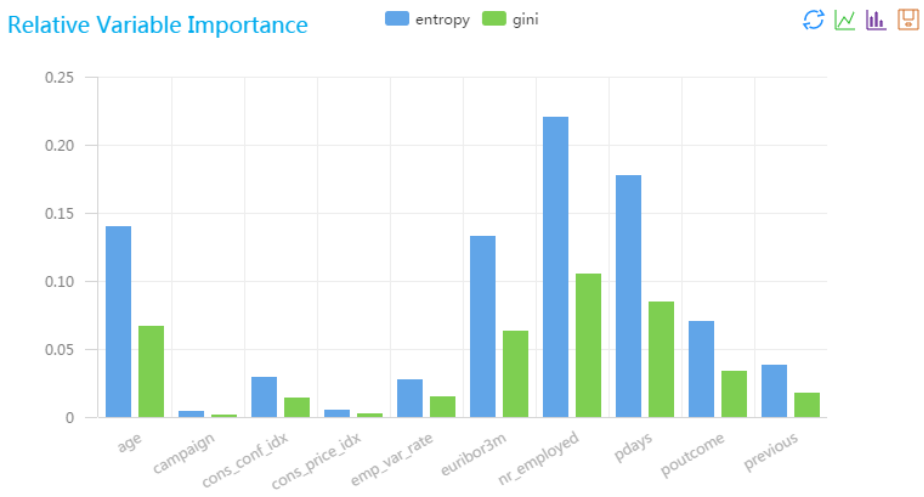
The example flowchart is as follows. The data source is pai_dense_10_10. Column y is the label column of the random forest, and other columns are feature columns. Select age and campaign for the required conversion, which indicates that the two features are processed as enumeration features. Use the default values for other parameters. The result is as follows after successful running:



Running result

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

Right-click the random forest feature importance component and choose **View Visualized Analysis**. The result is as follows:



GBDT feature importance

The function of the component is to calculate the GBDT feature importance.

PAI command

```
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;
```

Algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	NA	NA
outputTableName	(Required) Name of the output table	NA	NA
labelColName	(Required) Name of the label column	NA	NA
modelName	(Required) Name of the input model	NA	NA
featureColNames	(Optional) Selected feature columns in the input table	NA	All columns are selected by default except the label column.

inputTablePartitions	(Optional) Names of the selected partitions in the input table	NA	All partitions are selected by default.
lifecycle	(Optional) Life cycle of the output table	NA	Unspecified by default
coreNum	(Optional) Number of cores	NA	Automatically calculated by default
memSizePerCore	(Optional) Size of memory	NA	Automatically calculated by default

Example

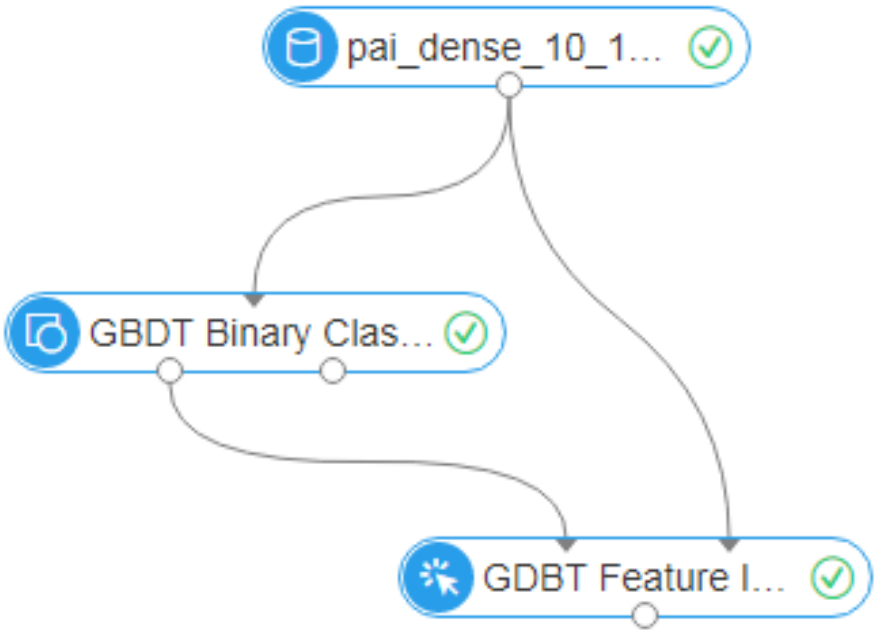
Input data

SQL statement for data generation:

```
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;
```

Parameter settings

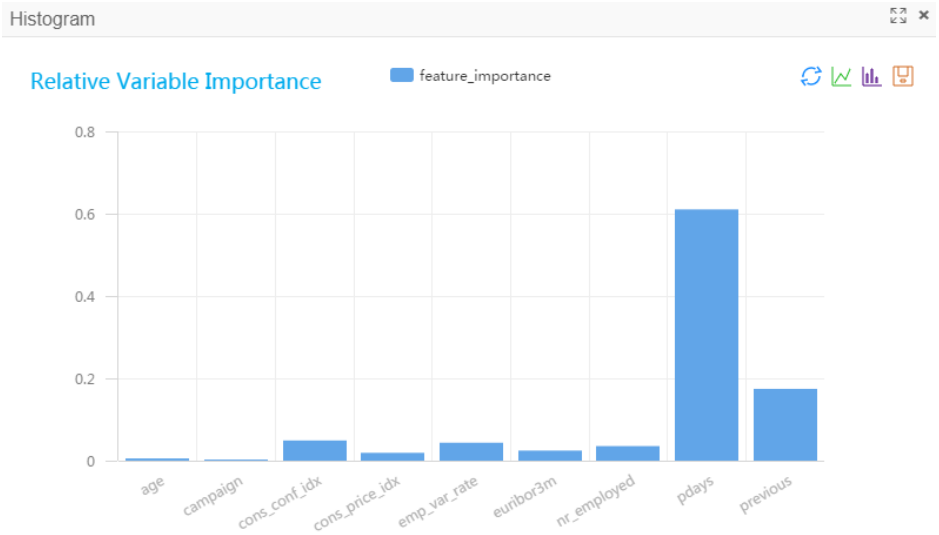
The example flowchart is as follows. The input data is pai_dense_10_9. Select label column y for the GBDT binary classification component, and use other columns as the feature columns. Set the **minimum number of leaf node samples** to 1 and then run the command.



Running result

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

Right-click the GBDT feature importance component and choose **View Visualized Analysis**. The result is as follows:



Linear model feature importance

The function of this component is to calculate the importance of the linear model features, including linear regression models and binary class logical regression models. Sparse and dense features are supported.

PAI command

```
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;
```

Algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	NA	NA
outputTableName	(Required) Name of the output table	NA	NA
modelName	(Required) Name of the input model	NA	NA
labelColName	(Required) Name of the label column	NA	NA
featureColNames	(Optional) Selected features in the input table	NA	All columns are selected by default except the label column.
inputTablePartitions	(Optional) Selected partitions in the input table	NA	All partitions are selected by default.
enableSparse	(Optional) Indicates whether the input table data is in sparse format	true, false	false
itemDelimiter	(Optional) Indicates the delimiter between key-value pairs when the input table data is in sparse format.	NA	Space
kvDelimiter	(Optional) Indicates the delimiter between keys and values when the input table data is in sparse format.	NA	Colon
lifecycle	(Optional) Life cycle of the output table	NA	Unspecified by default
coreNum	(Optional) Number of cores	NA	Automatically calculated by default
memSizePerCore	(Optional) Size of memory	NA	Automatically calculated by default

Example

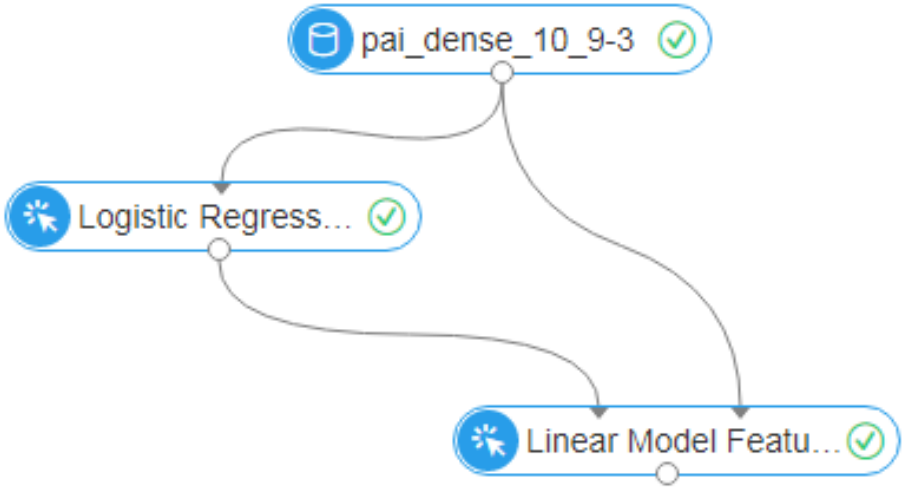
Input data

SQL statement for data generation:

```
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;
```

Parameter settings

The modeling process is shown in the following figure. Select the y label column for the logical regression multiclass classification, use other columns as the feature columns, use the default values for other parameters, and then run the command.



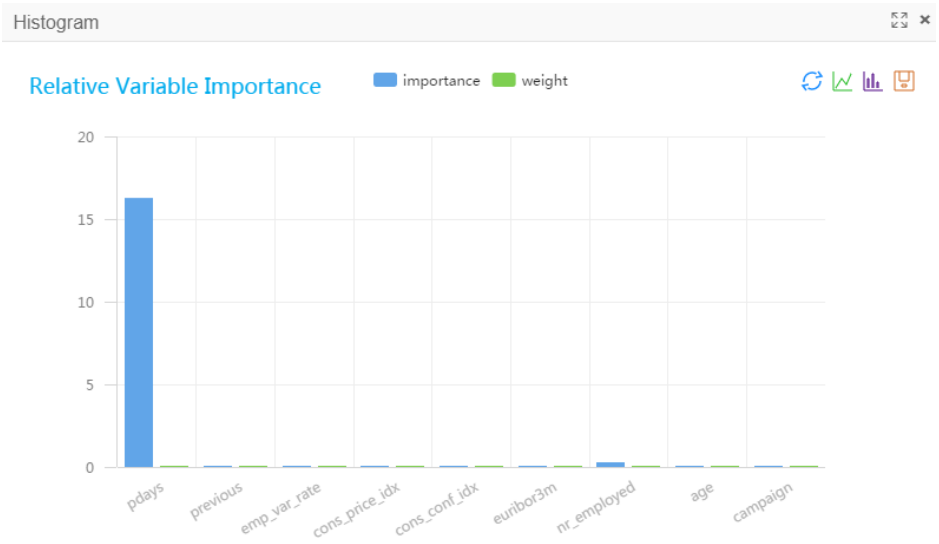
Running result

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	0.00004179335366052985 5

Metric calculation formula

Column	Formula
weight	abs(w_)
importance	abs(w_j) * STD(f_i)

Right-click the linear model feature importance component and choose **View Visualized Analysis**. The result is as follows:



Preference calculation

The function of this component is:

Specify the detailed behavior feature data of users and automatically calculate users' preference scores for feature values.

The input table includes the user ID and detailed behavior features of the user. For example, if user 2088xxx1 eats Sichuan food twice and western fast food once within three months, the input table is as follows.

user_id	cate
2088xxx1	Sichuan food
2088xxx1	Sichuan food
2088xxx1	Western fast food

- The output table shows the user' s preference scores for Sichuan food and western fast food.

user_id	cate
2088xxx1	Sichuan food:0.0544694,western fast food:0.0272347

PAI command

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

Algorithm parameters

Parameter key	Description	Required/Optional	Default value
InputTableName	Name of the input table	Required	NA
IdColumnName	User ID column	Required	NA
FeatureColNames	User feature column	Required	NA
OutputTableName	Name of the output table	Required	NA
OutputTablePartitions	Partitions in the output table	Optional	NA
mapInstanceNum	Number of mappers	Optional	2
reduceInstanceNum	Number of reducers	Optional	1

Example

Input data

SQL statement for data generation:

```
drop table if exists preference_input_table;
create table preference_input_table as
select
*
from
(
select '2088xxx1' as user_id, 'Sichuan food' as cate from alipaydw.dual
union all
select '2088xxx1' as user_id, 'Sichuan food' as cate from alipaydw.dual
union all
```

```
select '2088xxx1' as user_id, 'western fast food' cate from alipaydw.dual
union all
select '2088xxx3' as user_id, 'Sichuan food' as cate from alipaydw.dual
union all
select '2088xxx3' as user_id, 'Sichuan food' as cate from alipaydw.dual
union all
select '2088xxx3' as user_id, 'western fast food' as cate from alipaydw.dual
) tmp;
```

Running result

```
+------------+------------+
| user_id | cate |
+------------+------------+
| 2088xxx1 | Sichuan food:0.0544694,western fast food:0.0272347 |
| 2088xxx3 | Sichuan food:0.0544694,western fast food:0.0272347 |
+------------+------------+
```

Filter-based feature selection

Component functions

Select and filter Top N feature data based on different feature selection modes of users and save all feature importance tables (right output). Sparse and dense data is supported.

PAI command

```
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;
```

Algorithm parameters

Parameter	Description	Option	Default value
inputTable	(Required) Name of the input table	NA	NA
inputTablePartitions	(Optional) Partitions used for training in the input table, in the format of Partition_name=value.The multilevel format is	NA	All partitions in the input table

	name1=value1/name2=value2. If multiple partitions are specified, use commas (,) to separate them.		
outputTable	(Required) Name of the output table after filtering	NA	NA
featImportanceTable	(Required) Importance weight values of all input features	NA	NA
selectedCols	(Required) Selected feature columns	NA	NA
labelCol	(Required) Label column/target column	NA	NA
categoryCols	(Optional) Enumeration features that contain Int or Double characters	NA	""
maxBins	(Optional) Maximum number of intervals for division of consecutive features	NA	100
selectMethod	(Optional) Feature selection method. Default value: iv. The options are iv (Information Value), GiniGain, InfoGain, and Lasso.	iv,GiniGain,InfoGain, Lasso	iv
topN	(Optional) Top N features selected. If N is greater than the number of input features, the output includes all the features. Default value: 10	NA	10
isSparse	(Optional) Indicates whether it is a sparse feature in the key:value format. Default: dense data	NA	100
itemSpliter	(Optional) Delimiter between sparse feature items. Default value: ","	NA	","

kvSpliter	(Optional) Delimiter between sparse feature items. Default value: ":"	NA	":"
lifecycle	(Optional) Life cycle of the output table. Default value: 7	NA	7

Example

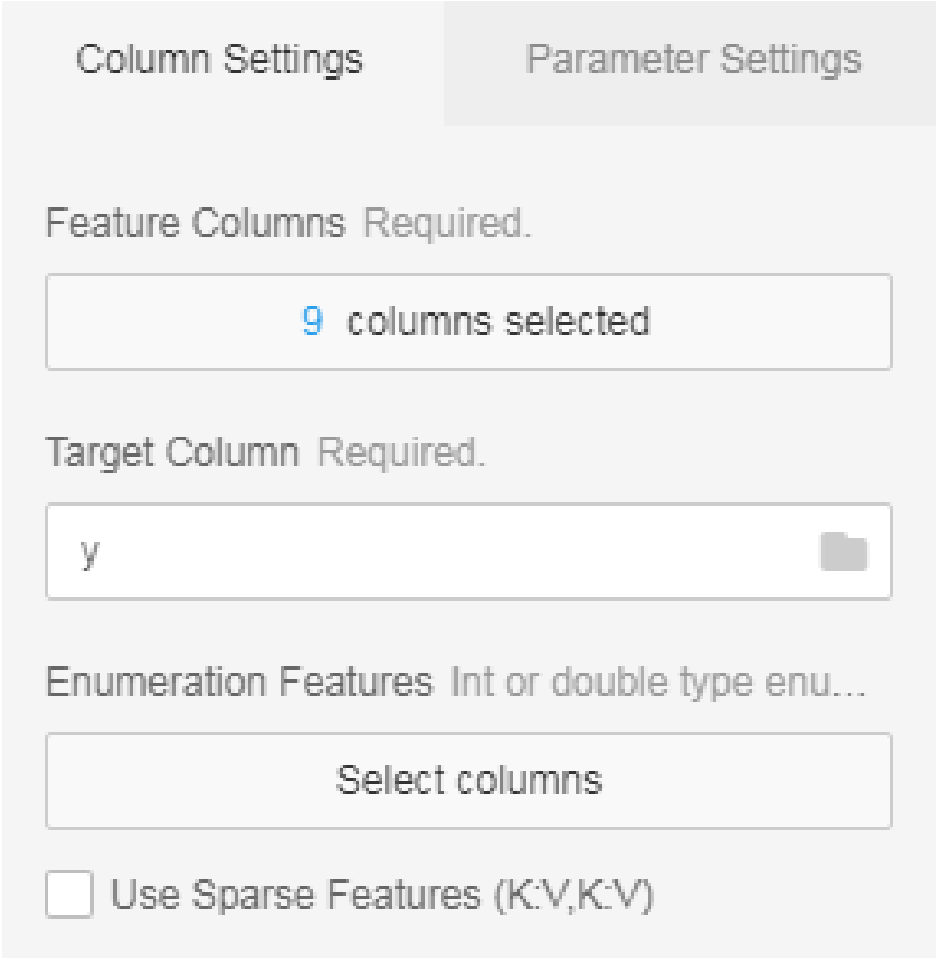
Input data

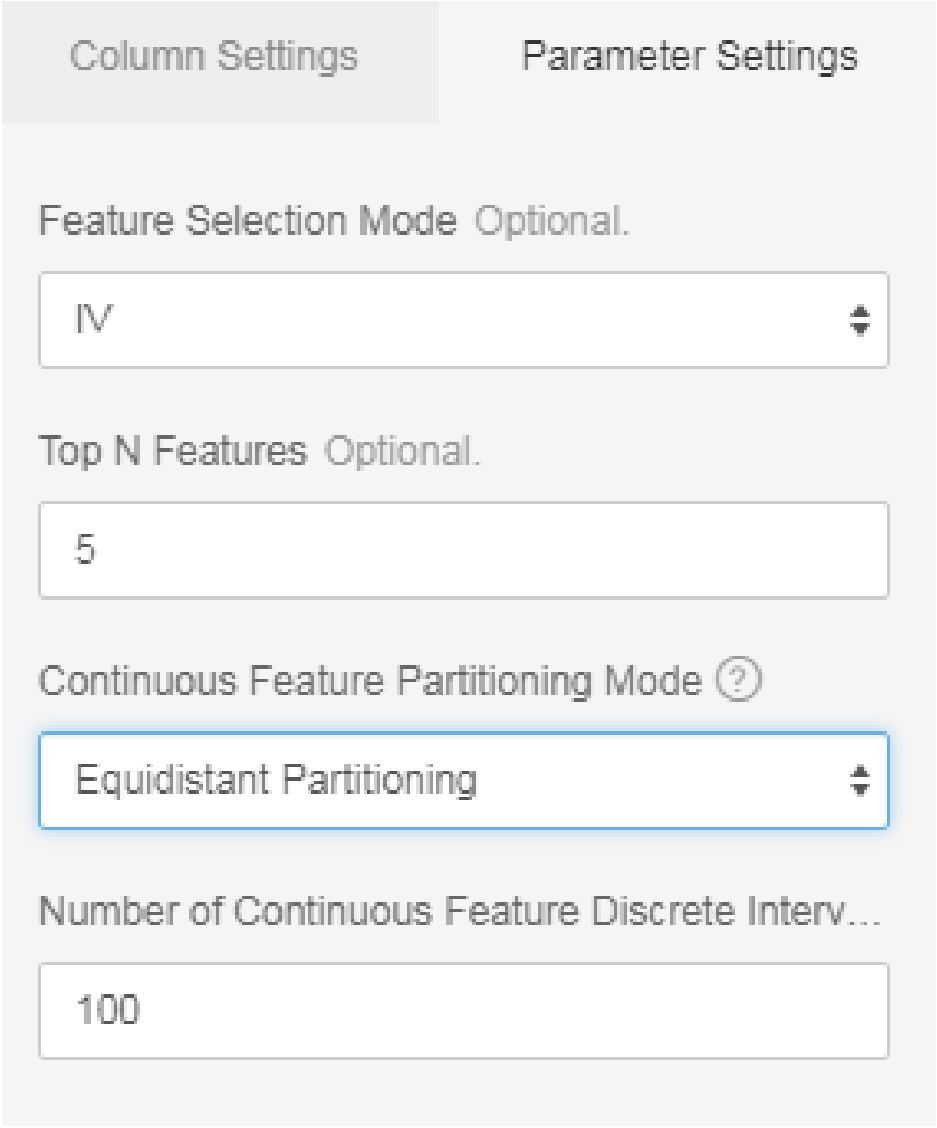
SQL statement for data generation:

```
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;
```

Parameter settings

Select the y field as the label column, select other fields as the feature columns, set the feature selection method to IV, and set topN to 5, which indicates top 5 features are filtered.


Column Settings
Parameter Settings
Feature Columns Required.
9 columns selected
Target Column Required.
y
Enumeration Features Int or double type enu...
Select columns
Use Sparse Features (K:V,K:V)



Running result

The left output is the filtered data.

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

The right output is the feature importance table.

The field structure of the feature importance table is as follows.

The featureName field indicates feature names, and weight indicates the weight calculated based on the feature selection method.

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 calculation formula

Selection method	weight contains
IV	Information value
GiniGain	Gini gain
InfoGain	Information entropy gain
Lasso	Absolute value of the linear model weight

RFM

RFM calculates the purchase frequencies and total monetary value for a user in a specific time window. For example, if the time windows are set to 1,7,30,90,180, RFM calculates the purchase frequencies and total monetary value in 1 day, 7 days, 30 days, 90 days, and 180 days.

PAI command

```
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;
```

Algorithm parameters

Parameter key	Description	Required/Optional	Default value
inputTableName	Name of the input table	Required	NA
userName	User ID column	Required	NA
dtName	Time (format: 20160101)	Required	NA
cntName	Frequency	Required	NA
amtName	Monetary value	Required	NA
window	Time window (format: 1,7,30,90...)	Required	NA
outputTableName	Name of the output table	Required	NA
outputTablePartitions	Partitions in the output table	Optional	NA
mapInstanceNum	Number of mappers	Optional	2
reduceInstanceNum	Number of reducers	Optional	2

Example

Input data

SQL statement for data generation

```
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
```

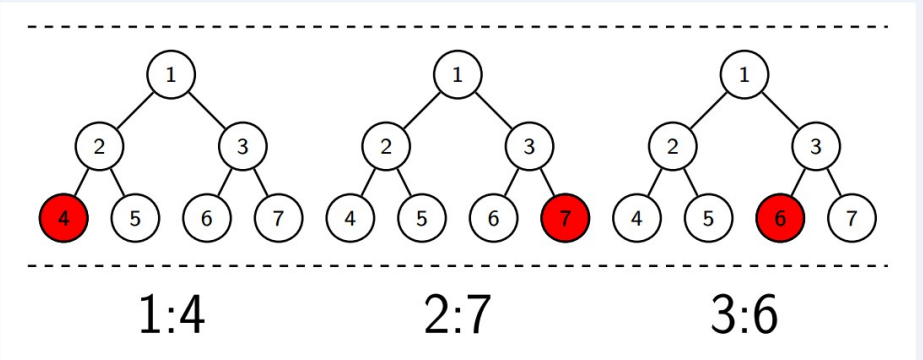
```
union all
select 'b' as user_id, '20151212' as dt, 1 as cnt, 12.0 as amt from dual
union all
select 'b' as user_id, '20160110' as dt, 2 as cnt, 30.0 as amt from dual
union all
select 'c' as user_id, '20151001' as dt, 3 as cnt, 60.0 as amt from dual
union all
select 'c' as user_id, '20151201' as dt, 2 as cnt, 39.0 as amt from dual
) tmp;
```

Running result

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

Feature encoding

Feature encoding is a method of using the decision tree and Ensemble algorithm for **new feature** mining. Features are derived from the decision tree branches formed by one or multiple features. For example, in the tree on the left of the following figure, node 1 -> node 2 -> node 4 is a feature. Apparently, this coding policy effectively transforms non-linear features into linear features.



PAI command

```
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";
```

Parameter description

Parameter	Description	Option	Default value
inputTable	(Required) Name of the input table	NA	NA
inputTablePartitions	(Optional) Partitions used for training in the input table, in the format of Partition_name=value.The multilevel format is name1=value1/name2=value2. If multiple partitions are specified, use commas (,) to separate them.	NA	All partitions in the input table
encodeModel	(Required) GBDT binary classification model for encoding	NA	NA
outputTable	(Required) Name of the scaling result table	NA	NA
selectedCols	(Required) Selected GBDT features involved in encoding (mainly training features of the GBDT component)	NA	NA
labelCol	(Required) Label field	NA	NA
lifecycle	(Optional) Life cycle of the output table. Default value: 7	NA	7

Example

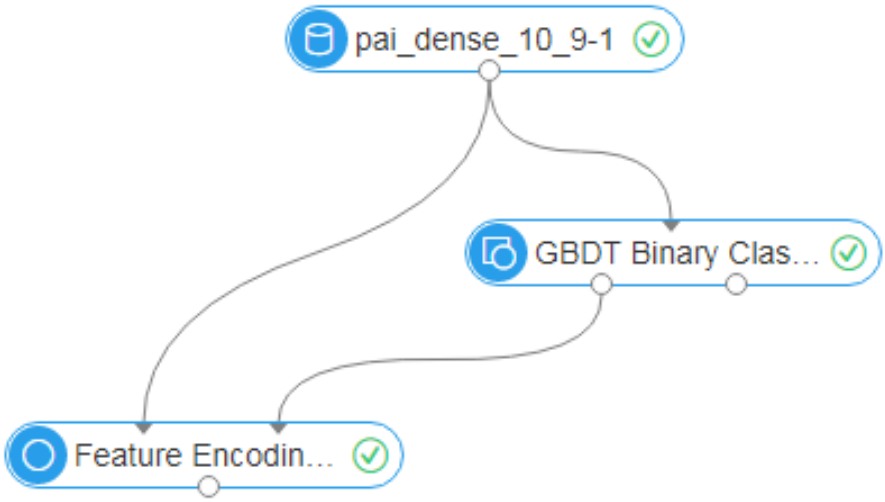
Input data

SQL statement for data generation:

```
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;
```

Parameter settings

The modeling process is as follows. It is generally used with the GBDT binary classification component.



To simplify the demonstration, set the number of GBDT binary classification trees to 5, set the depth to 3, set field y as the label column, and set other fields as the feature columns. The running result is as follows.

Running result

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:	0.0

1,42:1,43:1,46:1,46:1,63:1,64:1,67:1,68:1,68:1	

One-hot encoding

One-hot encoding converts a feature with m possible values into m binary features. In addition, these features are mutually exclusive, that is, only one feature can be activated at a time. As a result, the data is sparse, and the output is also in the sparse key:value structure.

PAI command

```
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;
```

Parameter description

Parameter	Description	Option	Default value
inputTable	(Required) Name of the input table	NA	NA
inputTablePartitions	(Optional) Partitions used for training in the input table, in the format of Partition_name=value.The multilevel format is name1=value1/name2=value2. If multiple partitions are specified, use commas (,) to separate them.	NA	All partitions in the input table
binaryCols	(Required) One-hot encoding field. The features must be enumeration features, and the field type is not required.	NA	NA
binStrategy	(Required) Encoding policy. This component provides	NA	NA

	two encoding policies: noDealStrategy and autoStrategy. The classification label must be set.		
labelCol	Classification label type. **Required when the purity-based binarization policy is used**, and optional when other policies are used	NA	NA
impurityMergeThresh	(Optional) Purity increase threshold value when the binarization features are merged when the purity-based binarization policy is used	NA	0.1
densityMergeThresh	(Optional) Binarization feature density (percentage) merging threshold value when the density-based binarization policy is used	NA	0.1
binaryReserve	(Optional) Indicates whether the output table contains the original one-hot encoding features.	NA	NA
outputTable	(Required) Output table of one-hot encoding. The kv field contains the encoding result.	NA	NA
binaryIndexTable	(Required) KV serialization table, which contains the mapping between encoded feature names and keys in key-value pairs.	NA	NA
modelTable	(Required) Mapping logic of one-hot encoding, stored in the JSON format	NA	NA
lifecycle	(Optional) Life cycle of the output table.	NA	7

	Default value: 7		

Encoding policy

- noDealStrategy: Simple binarization, for example, the feature values of sex are Female|Male|Unknown. The features are sex_female, sex_male, and sex_unknown after binarization.
- autoStrategy: Automatic binarization Based on the simple binarization policy, this policy uses the logical regression algorithm to calculate each one-hot feature and merges the features, whose one-hot feature weight is 0, into another item (named other).

Example

Input data

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

Parameter settings

Select edu_num as the feature to be binarized and use the default parameters for others, that is, the simple binarization policy is used.

Rnuning result

Encoding result (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

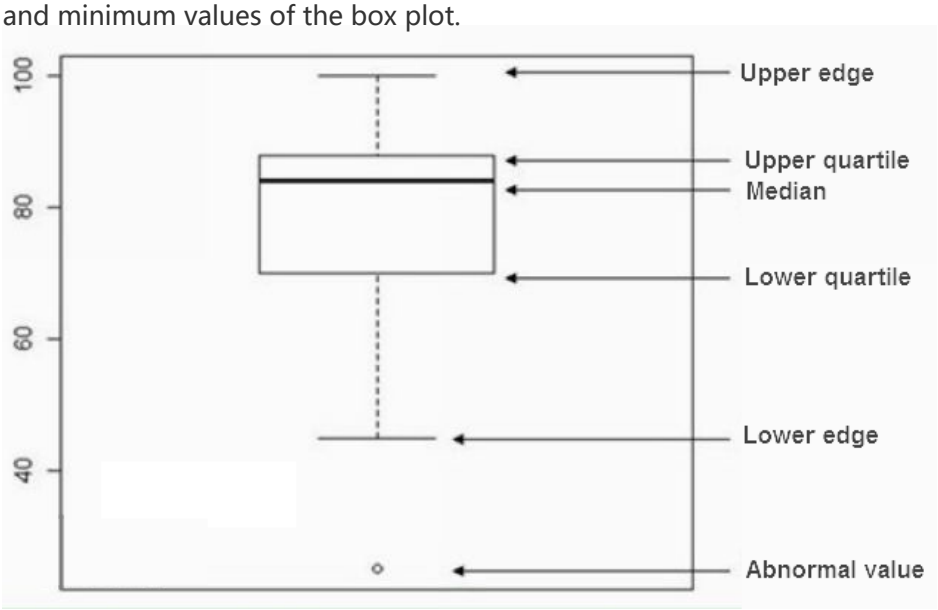
KV sequence table (binaryIndexTable)

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

Exception detection

The component supports the following functions:

For continuous-value features, perform abnormal feature detection based on the maximum and minimum values of the box plot.



For enumeration-value features, perform abnormal feature detection based on the enumeration frequency threshold value of the features.

PAI command

```
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";
```

Parameter settings

Parameter	Description	Option	Default value
inputTable	(Required) Name of the input table	NA	NA
inputTablePartitions	(Optional) Partitions used for training in the input table, in the format of Partition_name=value.The multilevel format is name1=value1/name2=value2. If multiple partitions are specified, use commas (,) to separate them.	NA	All partitions in the input table
selectedCols	(Required) Input features. The field type is not required.	NA	NA
detectStrategy	(Required) The options are boxPlot (for continuous-value features) and avf (for enumeration-value features).	NA	boxPlot
outputTable	(Required) Data set after detected abnormal features are filtered	NA	NA
modelTable	(Required) Exception detection model	NA	NA
lifecycle	(Optional) Life cycle of the output table. Default value: 7	NA	7

Feature importance filtering

The component provides the filter function for importance evaluation components such as **linear feature importance, GBDT feature importance, and random forest feature importance** and supports top N feature filtering.

PAI command

```
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,po
utcome
-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;
```

Component parameters

Parameter	Description	Option	Default value
inputTable	(Required) Name of the input table	NA	NA
inputTablePartitions	(Optional) Partitions used for training in the input table, in the format of Partition_name=value.The multilevel format is name1=value1/name2=value2. If multiple partitions are specified, use commas (,) to separate them.	NA	All partitions in the input table
weightTable	(Required) Feature importance weight table (output table of linear feature importance/GBDT feature importance/random forest feature importance)	NA	NA
outputTable	(Required) Output table after top N features are filtered	NA	NA
modelTable	(Required) Model file generated by feature filtering	NA	NA

selectedCols	(Optional) All the fields in the input table are selected by default.	NA	NA
topN	Top N features selected. Default value: 10	NA	10
lifecycle	(Optional) Life cycle of the output table. Default value: 7	NA	7

Modeling example

Input data

The input on the left of the feature importance filtering component is the original data table, and the output on the right is the feature importance table.The output is the top N feature data after filtering.

Note: The feature importance table uses a specified format. The first field indicates the feature name, and the second field indicates the weight value of the feature, whose data type is generally double. For example, the output table of random forest feature importance is as follows:

Field	Type	Label	Comment
colname	string	NA	NA
gini	double	NA	NA
entropy	double	NA	NA

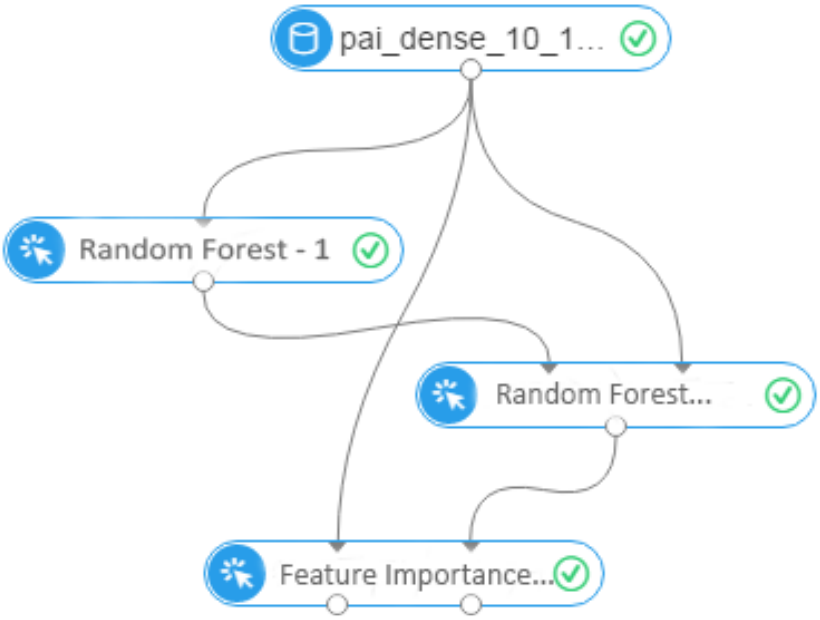
SQL statement for data generation

```
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;
```

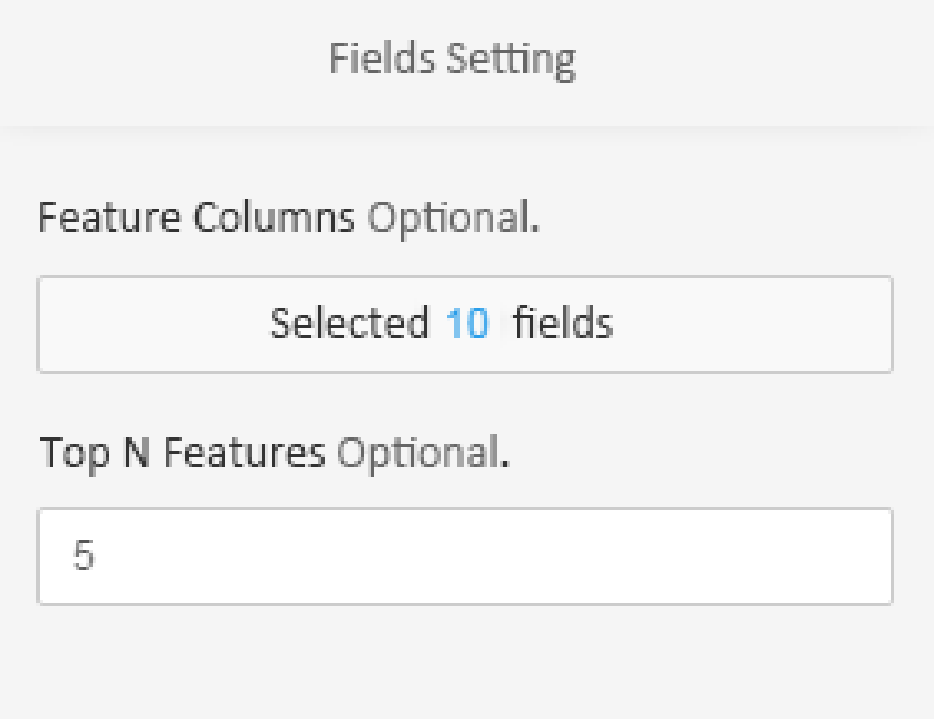
The feature importance table is the output table of the random forest feature importance component.

Parameter settings

Modeling flowchart (**This component is generally used with a feature importance component**, for example, linear feature importance, GBDT feature importance, and random forest feature importance.)



The input data is pai_dense_10_10.
For random forest feature importance, select field y as the label column and select other fields as the feature columns.
For feature importance filtering components, set topN to 5, which indicates only the top 5 features are filtered.



Running result

The output on the left is the result.

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

The output on the right is the filtering model, which is empty currently.

Statistical analysis

Contents

Data view

Covariance

Empirical probability density chart

Box plot

Scatter plot

Correlation coefficient matrix

Normality test

Lorenz curve

Percentile

Calculate the percentile for a single column of data in an existing table.

Parameter settings

Select the fields to be analyzed. Only the double and bigint types are supported.

The execution result is as follows:

PAI command

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

name: Component name.

project: Project name, used to specify the space of an algorithmThe default value is algo_public. If you change the name, the system reports an error.

outputTableName: Result table automatically allocated after the system executes the percentile operation.

colName: Column for percentile calculation. Only the number type is supported.

- inputTableName: Name of the input table.

Full table statistics

For an existing table, conduct basic statistics for the full table or conduct statistics only for selected columns.

PAI command

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

Parameter description

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	NA	NA
outputTableName	(Required) Name of the recommendation result output table	NA	NA
inputTablePartitions	(Optional) Partitions in the input table	NA	“”
selectColNames	(Optional) Names of	NA	“”

	columns for statistics		
lifecycle	(Optional) Life cycle of the output table	Positive integer	No life cycle
coreNum	(Optional)Number of nodes, Used together with the memSizePerCore parameter	Positive integer	Automatically calculated by default
memSizePerCore	(Optional)Memory size of each node, in MB	Positive integer in the range of [100, 64*1024]	Automatically calculated by default

Input format

In the input column box, select the columns that require statistics. By default, the statistics are conducted for all columns.

Output format

The fields of statistics output are as follows:

Column name	Description
colname	Column name
datatype	Type
totalcount	Total number
count	Number of Non-NULL values
missingcount	Number of NULL values
nancount	Number of NAN values
positiveinfinitycount	Number of positive infinity values
negativeinfinitycount	Number of negative infinity values
min	Minimum value
max	Maximum value
mean	Mean value
variance	Variance
standarddeviation	Standard deviation
standarderror	Standard error
skewness	Skewness
kurtosis	Kurtosis
moment2	Second moment

moment3	Third moment
moment4	Fourth moment
centralmoment2	Second central moment
centralmoment3	Third central moment
centralmoment4	Fourth central moment
sum	Sum
sum2	Sum of squares
sum3	Sum of cubes
sum4	Sum of biquadrates

Example

Test data

SQL statement for data generation:

```
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;
```

Running command

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

Running result

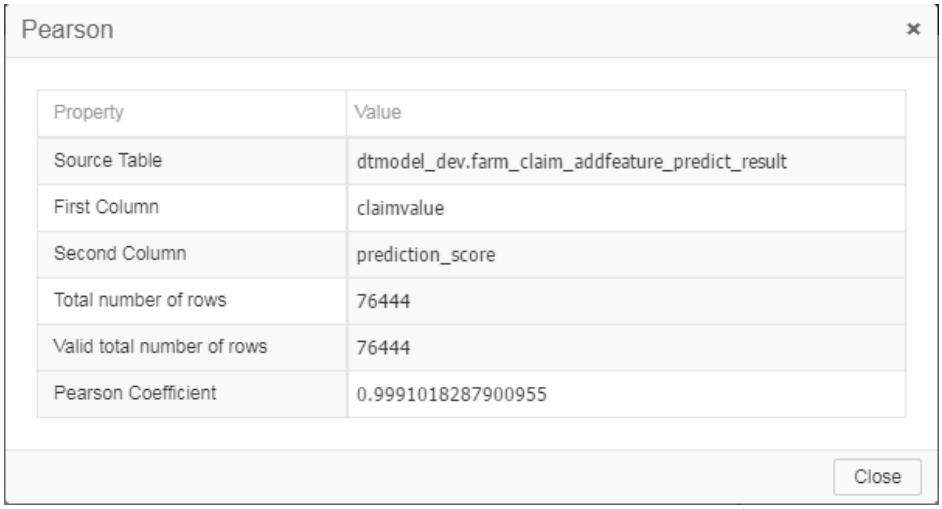
```
    | colname    | datatype   | totalcount | count      | missingcount | nancount   | positiveinfinitycount |
negativeinfinitycount | min        | max        | mean       | variance   | standarddeviation | standarderror | skewness   |
kurtosis   | moment2    | moment3    | moment4    | centralmoment2 | 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 |

Pearson coefficient

Calculate the Pearson correlation coefficient of two numeric columns in an input table or a partition and save the result to the output table.

> The component has only two parameters: input column 1 and input column 2. Enter the names of two columns for which the Pearson correlation coefficient is calculated.
>
> After running, right-click the component and choose **Menu > View Analysis Report**. The result is as follows:



The Pearson correlation coefficient is listed in the last column.

PAI command

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

Parameter description

Parameter key	Description	Value range	Required/Optional, default value/act
inputTableName	Name of the input	Table name	Required

	table		
inputTablePartitions	Partitions used for calculation in the input table	Format: partition_name=value. The multilevel format is name1=value1/name2=value2. Multiple partitions are separated by commas (,).	All partitions in the input table
col1Name	Input column 1	Column name	Required
col2Name	Input column 2	Column name	Required
outputTableName	Name of the output table	Table name	Required

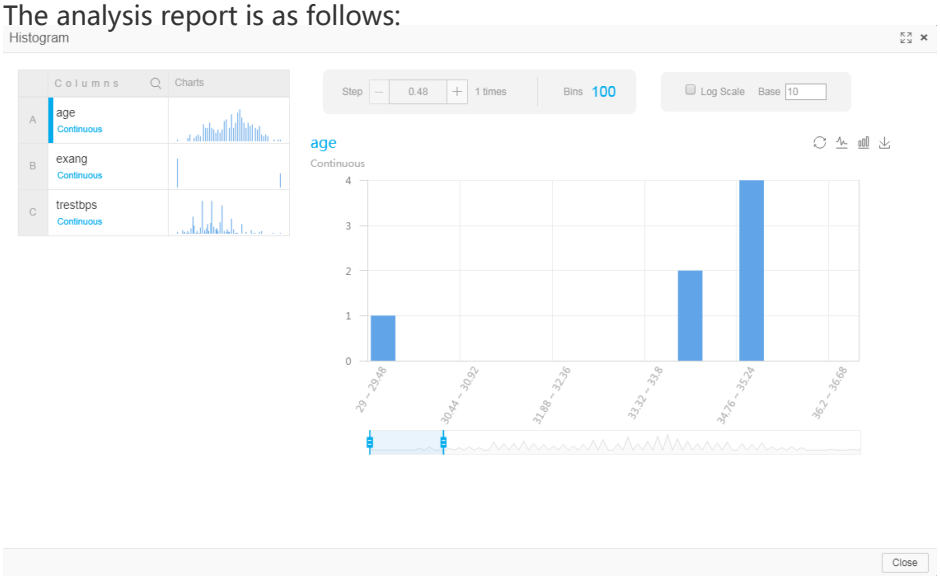
Histogram

Calculate the histogram for a single column of data in an existing table.

Parameter settings

Select the fields to be analyzed. Only the double and bigint types are supported.

The analysis report is as follows:



You can adjust the step and move the slider to view the histogram.

Discrete value feature analysis

Discrete value feature analysis shows the indicators such as discrete feature distribution, gini, entropy,

gini gain, information gain, and information gain ratio. Calculate gini and entropy for each discrete value and calculate gini gain, information gain, and information gain ratio for each column.

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

gini index:

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

entropy:

PAI command

```
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;
```

Parameter description

Parameter key	Description	Value range	Default value
inputTableName	(Required) Name of the input table	NA	NA
inputTablePartitions	(Optional) Names of the selected partitions in the input table	NA	All partitions are selected by default.
featureColNames	(Optional) Selected	NA	By default, all

	column names in the input table		columns are selected except the label column. If the input table format is KV, all the string-type columns are selected by default.
labelColName	(Required) Name of the label column	NA	NA
enableSparse	(Optional) Indicates whether the input table is in the KV format.	NA	Table by default
kvFeatureColNames	(Optional) Features in the KV format	NA	The entire table is selected by default.
kvDelimiter	(Optional) Delimiter used between key-value pairs	NA	Default value: ":"
itemDelimiter	(Optional) Delimiter used between keys and values	NA	Default value: ","
outputCntTableName	(Required) Output table of enumeration value distribution number of discrete values	NA	NA
outputValueTableName	(Required) Output table of gini and entropy of discrete features	NA	NA
outputEnumValueTableName	(Required) Output table of gini and entropy of discrete feature enumeration values	NA	NA
lifecycle	(Optional) Life cycle of the output table	Positive integer	No life cycle
coreNum	(Optional)Number of nodes, Used together with the memSizePerCore parameter	Positive integer	Automatically calculated by default
memSizePerCore	(Optional)Memory size of each node, in MB	Positive integer	Automatically calculated by default

Example

Test data

SQL statement for data generation

```
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;
```

Input data description

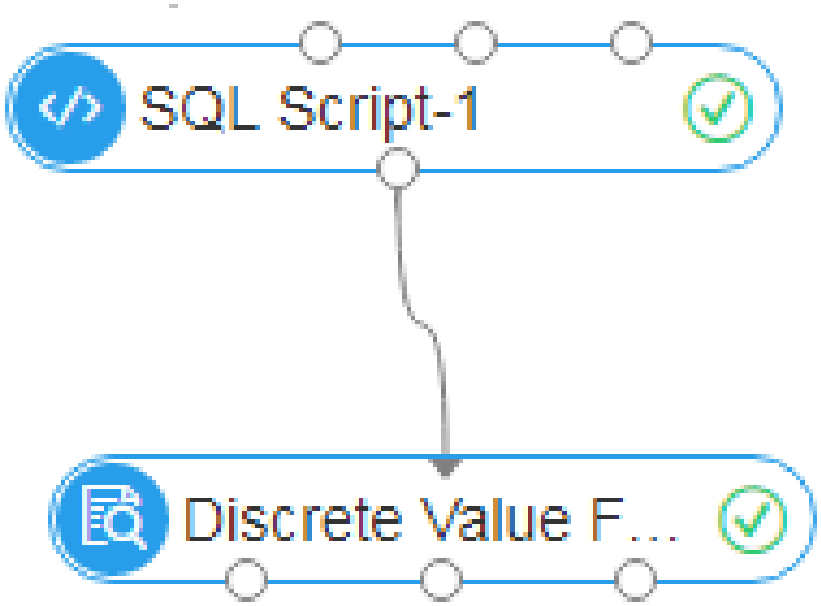
```
+------------+------------+------------+
| col_string | col_bigint | col_double |
+------------+------------+------------+
| 01 | 1 | 1.0 |
```

```
| 01 | 0 | 1.0 |
| 01 | 1 | NULL |
| NULL | 0 | 0.0 |
| 00 | 1 | 0.0 |
| 00 | 0 | 0.0 |
+------------+------------+------------+
```

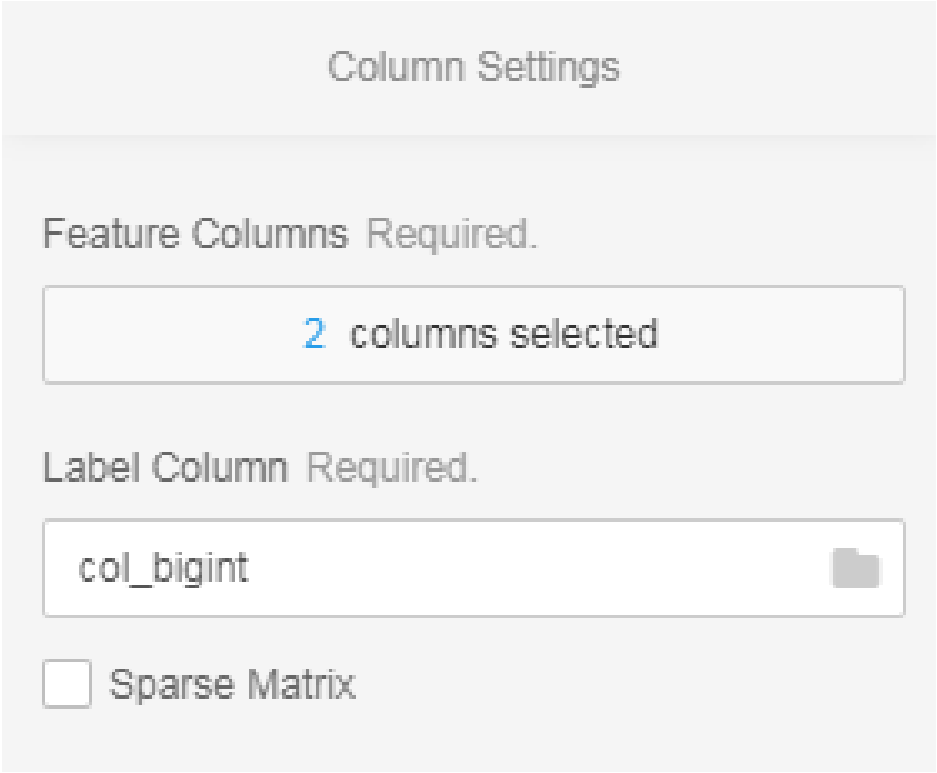
Running command

```
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";
```

UI



Parameter UI



Running results

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-test

A one-sample T-test tests whether the mean of a normally distributed population differs significantly from a target value. The premise of a T-test is that the sample population follows normal distribution.

PAI command

```
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
```

Parameter description

Parameter	Description	Value range	Required/Optional,

			default value/act
xTableName	Input table x	Table name	Required
xColName	Column that requires a T-test	Column name. The type must be double or bigint.	Required
outputTableName	Output table	Non-existent table name	Required
xTablePartitions	List of partitions in input table x	Partition list	(Optional) Default value: “”
alternative	Alternative hypothesis	two.sided, less, greater”	(Optional) Default value: two.sided
mu	Hypothesized mean	double	(Optional) Default value: 0
confidenceLevel	Confidence level	0.8,0.9,0.95,0.99,0.995,0.999	(Optional) Default value: 0.95

Output description

The output is a table with only one row and one column in the JSON format.

```
{
"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
}
```

Chi-square test

Chi-square Goodness of Fit Test is used to determine the differences between the observed frequencies and the expected frequencies for each classification of a single multiclass classification nominal variable. The null hypothesis assumes that the observed frequencies and the expected frequencies are consistent.

PAI command

```
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
```

Parameter description

Parameter	Description	Value range	Required/Optional, default value/act
inputTableName	Input table	Table name	Required
colName	Column requiring a chi-square test	Column name	Required
outputTableName	Output table	Non-existent table name	Required
outputDetailTableName	Output detail table	Non-existent table name	Required
inputTablePartitions	List of partitions in the input table	Partition list	(Optional) Default value: “”
probConfig	Class probability configurations	key-value pair; format: class:probability,class:probability...The sum of all probabilities is 1.	(Optional) All classes have the same probability by default.

Example

Test data

```
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 command

```
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
```

Output description

Output table outputTableName has only one row and one column in the JSON format.

```
{
"Chi-Square": {
"comment": "Pearson's chi-square test".
"df": 1,
"p-value": 0.75,
"value": 0.2380952380952381
}
}
```

Output table outputDetailTableName: Corresponding columns are class, observed frequency (observed), expected frequency (expected), standard error (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

Data view

The data view component provides the data pivot function. It visualizes feature value distribution and feature and label column distribution for users. Facilitates data analysis after the features and characteristics are clear and supports dense or sparse numeric features.

PAI command

```
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" ;
```

Parameter description

Parameter key	Description	Required/Optional	Default value
inputTable	Name of the input data table	Required	NA
inputTablePartitions	Partitions in the input table	Optional	NA
outputTable	Name of the output table	Required	NA
mapTable	Output mapping table. The data view maps string-type character strings into numbers (converts to Int to allow machine learning, recognition, and training)	Required	NA
selectedCols	Type of the selected column names in the input table	Required	NA
categoryCols	(Optional) Process Int- or double-type fields as enumeration features	NA	“”
maxBins	(Optional) Maximum number of intervals for same-distance division of consecutive features	NA	100
isSparse	(Optional) Indicates whether it is a sparse feature in the key:value format. Default: dense data	NA	100
itemSpliter	(Optional) Delimiter between sparse feature items. Default value: “,”	NA	“,”
kvSpliter	(Optional) Delimiter between sparse feature items.	NA	“:”

	Default value: ":"		
lifecycle	Life cycle of the output table, in days	Optional	28

Example

Input data

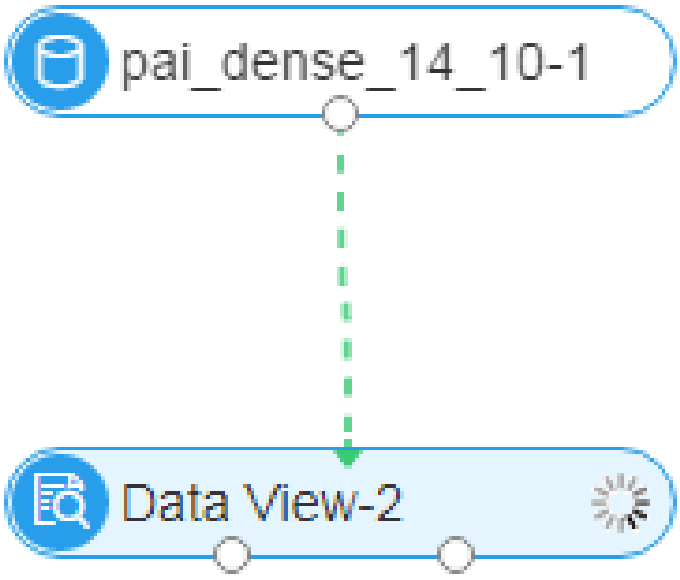
age	workclass	fwlght	edu	edu_num	married	c	family	race	sex	gail	loss	work_year	country	income
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174.0	0.0	40.0	United-States	<=50K
50	Self-emp-not-inc	833111	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White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

					v-spouse	clean ers						es	
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	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0.0	0.0	16.0	Jamaica	<=50K
52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spous	Exec-managerial	Husband	White	Male	0.0	0.0	45.0	United-States	>50K

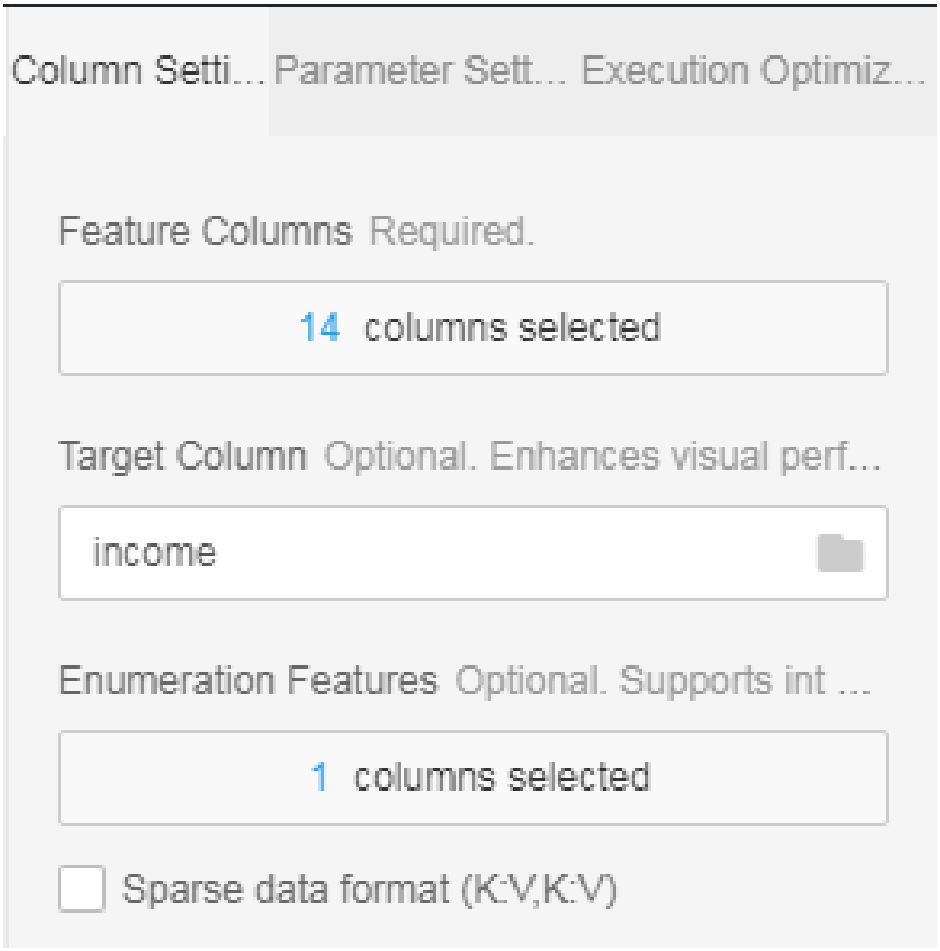
					e								
31	Private	457811	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084.0	0.0	50.0	United-States	>50K
42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	51780.0	0.0	40.0	United-States	>50K

Modeling DAG

Modeling DAG



Data view configuration: select **income** as the target column and the other 14 fields as the feature columns. Perform enumeration value processing for the edu_num field of the bigint type.



Modeling effect

- Enter data on the left. The family, race, sex, and income fields previously in the format of string are converted into numeric values. This **allows data to be learnt by machines for algorithm training (provides the data format conversion function in some extent)**.

gail	loss	work_year	age	fwlght	edu_num	workclass	edu	married	c	family	race	sex	country	income
21740.0	0.0	40.0	39.0	775160.0	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	833110.0	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	21564 6.0	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	23472 1.0	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	33840 09.0	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	28458 2.0	2.0	1.0	5.0	2.0	2.0	3.0	2.0	1.0	3.0	0.0
0.0	0.0	16.0	49.0	16018 7.0	3.0	1.0	2.0	3.0	4.0	2.0	1.0	1.0	2.0	0.0
0.0	0.0	45.0	52.0	20964 2.0	5.0	2.0	4.0	2.0	2.0	1.0	2.0	2.0	3.0	1.0
14084.0	0.0	50.0	31.0	45781 0	2.0	1.0	5.0	4.0	5.0	2.0	2.0	1.0	3.0	1.0
5178.0	0.0	40.0	42.0	15944	1.0	1.0	3.0	2.0	2.0	1.0	2.0	2.0	3.0	1.0

				9. 0										

- The mapping table is generated on the right.

feature_name	feature_value	map_id
income	<=50	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

Covariance

In the probability theory and statistics, covariance reflects the overall error between two variables. Variance is a special covariance when the two variables are the same.
Definition of the covariance between real-number random variable X and Y whose expected values are E(X) = μ and E(Y) = ν respectively:
cov(X, Y) = E((X - μ) (Y - ν))

PAI command

```
PAI -name cov
-project algo_public
-DinputTableName=maple_test_cov_basic12x10_input
-DoutputTableName=maple_test_cov_basic12x10_output
-DcoreNum=6
-DmemSizePerCore=110;
```

Parameter description

Parameter	Description	Value range	Required/Optional, default value
inputTableName	Input table	Table name	Required
inputTablePartitions	Partitions used for training in the input table, in the format of partition_name=value. The multilevel partition name format is name1=value1/name2=value2. If you specify multiple partitions, separate them with a comma (,).		(Optional) Default: all partitions
outputTableName	List of the output table names	Table name	Required

selectedColNames	Selected column name type in the input table	Column name	(Optional) All columns are selected by default
lifecycle	(Optional) Life cycle of the output table	Positive integer	No life cycle
coreNum	Number of nodes	Used together with the memSizePerCore parameter, positive integer in the range of [1, 9999]	(Optional) Automatically calculated by default
memSizePerCore	Memory size of each node, in MB	Positive integer in the range of [1024, 64*1024]	(Optional) Automatically calculated by default

Empirical probability density chart

An empirical distribution indicates the nonparametric distribution obtained based on probability distribution estimation, when accurate parametric distribution is unavailable. The algorithm uses kernel distribution to estimate the probability density of sample data. Similar to the histogram, the generating function describes sample data distribution. The difference is that kernel distribution overlays contribution of all parts to generate a continuous smooth distribution curve while the histogram provides discrete description. When kernel distribution is used, the probability density of non-sample data points is not 0 but weighted overlay of probability density of sampling points in certain kernel distribution. In this implementation, kernel distribution uses Gaussian distribution.

> For details about kernel distribution, see wiki.
>
> For details about empirical distribution, see wiki.

PAI command

```
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
```

Parameter description

Parameter	Description	Value range	Required/Optional, default value
inputTableName	Name of the input	NA	Required

	table		
outputTableName	Name of the output table	NA	Required
featureColNames	Input column	Multiple double or bigint columns can be selected	Required
labelColName	Input label column. Calculate the feature column by group based all the label values in this column	Select one bigint or string column. The number of label values cannot exceed 100	Optional, ""
inputTablePartitions	Partitions in the input table	NA	Optional, ""
intervalNum	Number of calculation frequency intervals. The larger number, the higher accuracy	[1,1E14)	(Optional) Default value: -1, which indicates that intervals are divided based on the value range of data in each column and the number of intervals is automatically calculated.
lifecycle	Life cycle of the output table	Positive integer	(Optional) Default value: -1, which indicates that life cycle is not set.
coreNum	Number of cores, used together with the memSizePerCore parameter	Positive integer	(Optional) Default value: -1, which indicates that the number of started instances is determined based on the input data amount.
memSizePerCore	Memory size of each node, in MB	Positive integer in the range of [1024, 64*1024]	(Optional) Default value: -1, which indicates that the memory size is determined based on the input data amount.

Example

Data generation

```
    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 command

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

Input description

You can select multiple columns that need to be calculated. You can select the label column and divide the selected columns into groups based on each label value. For example, if the label column contains value 0 and 1, the columns that need to be calculated are divided into a group where the label is 0 and another group where the label is 1. Then, the probability density can be drawn separately for the two groups. If the label column is not selected, the feature column is calculated as a whole.

Output description

Diagram and result table. The result table fields are as follows. If the label column is not selected, the label output is NULL.

Column name	Data type
colName	string
label	string
x	double
pdf	double

Output table

```
+------------+------------+------------+------------+
```

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

Algorithm scale

If the label column is selected, the number of labels cannot exceed 100.

Box plot

Visualize the box plot and disturbance of a number of continuous features and a specific

enumeration feature.

PAI command

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

Parameter description

Parameter key	Description	Required/Optional	Default value
inputTable	Name of the input data table	Required	NA
inputTablePartitions	Partitions in the input table	Optional	NA
outputTable	Name of the output table that contains the box plot and samples	Required	NA
continueCols	(Required) Continuous value feature. You can select multiple columns.	Required	NA
categoryCol	(Required) Enumeration feature column. Select one column.	Required	NA
sampleSize	Number of samples used for drawing disturbance of each feature	NA	1000
lifecycle	Life cycle of the output table, in days	Optional	28

Example

Input data

```
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

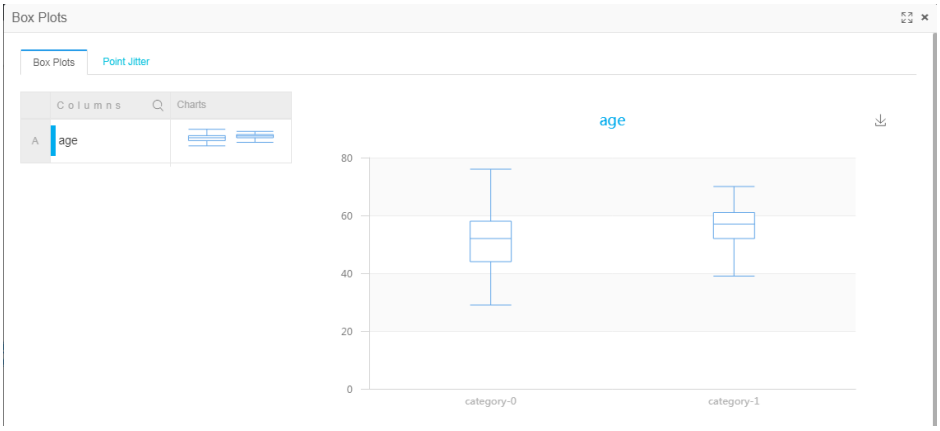
44	0

Parameter settings

Select age as the continuous feature and y as the enumeration feature and use the default value for other parameters.

Running effect

Box plot distribution:



Disturbance point:

Scatter plot

The Scatter plot component represents a chart where data points are distributed on the Cartesian coordinate plane in regression analysis.

PAI command

```
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;
```

Parameter description

Parameter key	Description	Required/Optional	Default value
inputTable	Name of the input data table	Required	NA
inputTablePartitions	Partitions in the input table	Optional	NA
outputTable	Name of the output table that contains the samples	Required	NA
mapTable	Output table that contains the maximum value, minimum value, and enumeration values of each feature	Required	NA
selectedCols	Columns selected in the input table, used to draw the scatter chart between every two features. A maximum of five features can be selected.	Required	NA
labelCol	(Optional) Use the Int or String field as the enumeration label column	NA	“”
sampleSize	(Optional) Perform sampling for input data	NA	1000
lifecycle	Life cycle of the output table, in days	Optional	28

Example

Input data

```
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

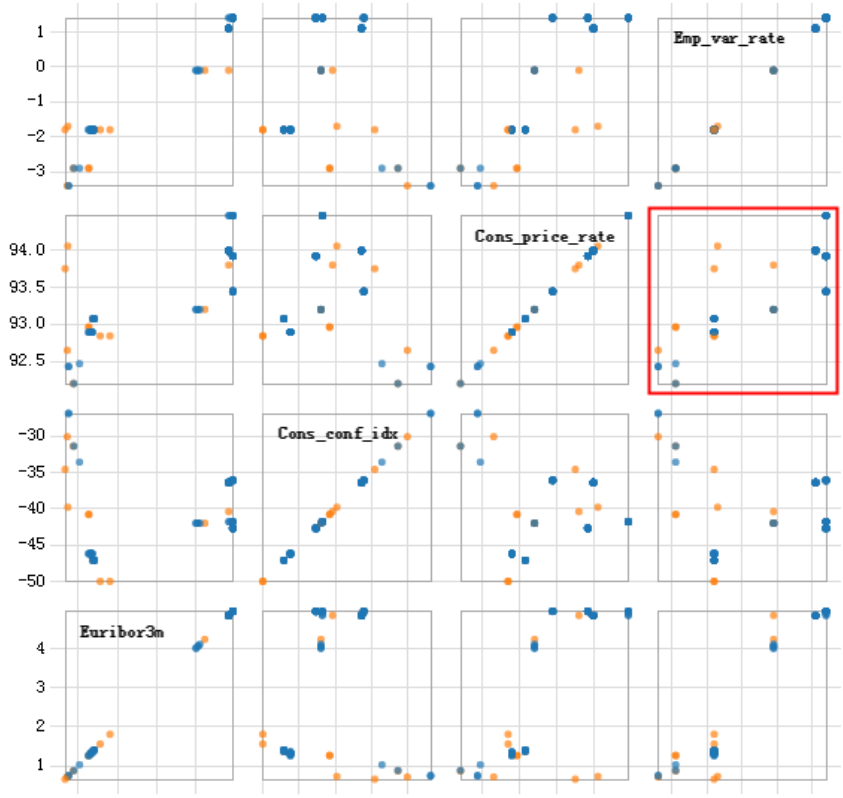
1.1	93.994	-36.4	4.857	0
-1.8	92.893	-46.2	1.259	0
1.1	93.994	-36.4	4.857	0
1.4	94.465	-41.8	4.866	0
-2.9	92.201	-31.4	0.883	0
-0.1	93.2	-42.0	4.076	0
1.1	93.994	-36.4	4.857	0
1.4	93.918	-42.7	4.96	0
1.4	93.444	-36.1	4.962	0
1.1	93.994	-36.4	4.858	0
1.1	93.994	-36.4	4.857	0
1.1	93.994	-36.4	4.856	0
1.4	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

Parameter settings

Select **select emp_var_rate**, **cons_price_rate**, **cons_conf_idx**, and **euribor3m** as the feature columns, and select **y** for the optional label columns for the scatter plot.

Running effect

Distribution of classification tags between features is directly displayed.



Correlation matrix

The correlation coefficient is used to measure the correlation between columns in a matrix. The value range of the correlation coefficient is [-1,1]. The value of count is the number of non-zero elements in both successive columns. This value may vary with columns.

PAI command

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

Parameter description

Parameter	Description	Value range	Required/Optional, default value
inputTableName	Input table	Table name	Required
inputTablePartitions	Partitions used for training in the input table, in the format of partition_name=value. The multilevel partition name format is name1=value1/name2=value2. If you specify multiple partitions, separate them with a comma (,).		(Optional) Default: all partitions
outputTableName	List of the output table names	Table name	Required
selectedColNames	Selected column name type in the input table	Column name	(Optional) All columns are selected by default
lifecycle	(Optional) Life cycle of the output table	Positive integer	No life cycle
coreNum	Number of nodes	Used together with the memSizePerCore parameter. Positive integer in the range of [1, 9999]. Detailed description	(Optional) Automatically calculated by default
memSizePerCore	Memory size of each node, in MB	Positive integer in the range of [1024, 64*1024] Detailed description	(Optional) Automatically calculated by default

Example

Data generation

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 command

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

Output table

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	0.25999035706	-0.17507636	0.18518346647	1	0.03988018649	-0.43737887	-0.05381829	0.29008564415	-0.36075479	0.49120190749

	84693	221594533	293102		854009	418329147	6425267184	86986	10075688	30449
col4	-0.34832491882255586	0.40943384150571377	-0.20934839228057014	0.03988018649854009	1	0.14656052092468675	-0.50160303643479555	0.54960243257111117	0.01374325611539412 2	0.07497231559184887
col5	-0.28716254396809926	0.09135976026101403	-0.18964175123896 59	-0.43737887418329147	0.14656052092468 75	1	0.16729809310873522	-0.29890655828796964	0.36185181010146 17	-0.17139609572868 85
col6	0.47880162127435116	-0.30185063746265 74	0.17993774988632 13	-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

Normality test

Normality tests are used to determine whether a data set is well-modeled by a normal distribution. This component consists of three test methods: Anderson-Darling Test (for details, see wiki), Kolmogorov-Smirnov Test (for details, see wiki), and QQ Plot (for details, see wiki). Use one or more methods based on requirements.

Algorithm description

Original hypothesis H0: The observed values are in a normal distribution; H1: The observed values are not in a normal distribution.

KS p-value calculation method progressively calculates the CDF of KS distribution regardless of the sample size. For details, see wiki.

If the sample size is greater than 1,000, the QQ Plot method samples to calculate and output plots, which means the data points in the plot do not necessarily cover all the samples.

PAI command

```
PAI -name normality_test
-project algo_public
-DinputTableName=test
-DoutputTableName=test_out
-DselectedColNames=col1,col2
-Dlifecycle=1;
```

Parameter description

Parameter	Description	Value range	Required/Optional, default value
inputTableName	Input table	NA	Required
outputTableName	Name of the output table	NA	Required
selectedColNames	Selected field columns	You can select multiple double or bigint columns.	Optional
inputTablePartitions	Input table partitions	NA	Optional, “”
enableQQplot	Use the QQ plot	NA	(Optional) Default value: true
enableADtest	Use the Anderson-Darling test	NA	(Optional) Default value: true
enableKStest	Use the Kolmogorov-Smirnov test	NA	(Optional) Default value: true

lifecycle	Life cycle	An integer equal to or more than -1	(Optional) Default value: -1, which indicates that life cycle is not set.
coreNum	Number of cores	An integer more than 0	(Optional) Default value: -1, which indicates that the number of started instances is determined based on the input data volume.
memSizePerCore	Memory size of each node, in MB	[100,64*1024]	(Optional) Default value: -1, which indicates that the memory size is determined based on the input data volume.

Example

Data generation

```
    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 command

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

Input description

Input format: Select the columns that need to be calculated. You can select multiple double or bigint columns.

Output description

Output format: Diagram and result table. The result table fields are as follows: If the table has two partitions, the p=’ test’ partition provides the results of the AD or KS test only when the enableADtest or enableKStest parameter is set to true.

The p=’ plot’ partition provides the QQ plot data only when the enableQQplot parameter is set to true, and the p=’ test’ column is reused. That is, when p=’ plot’, the testvalue column records the original observed data (X-axis of the QQ plot), and the pvalue column records the expected data (Y-axis of the QQ plot) when the data follows normal distribution.

Column name	Data type	Description
colName	String	Column name
testname	String	Test name
testvalue	double	Test value/X-axis of the QQ plot
pvalue	double	Test p value/Y-axis of the QQ plot
p	double	Partition name

Output table

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

Lorenz curve

The Lorenz curve studies how income is distributed among a population. To study the distribution of income among a population, the American statistician (or Austrian statistician) M.O.Lorenz (Max Otto Lorenz, 1903 -) proposed the famous Lorenz curve in 1907 (or 1905). Later, the Italian economist Gini defined the Gini Coefficient based on Lorenz curve.

The Gini Coefficient is represented by a rectangle. The height of the rectangle measures the percentage of social wealth and is divided into N equal fractions, each of which equals to 1/N of total social wealth. On the length of the rectangle, all families are arranged from left to right in the poorest to the richest order and are also divided into N equal fractions. The first fraction represents the 1/N of the families with lowest income. In this rectangle, cumulating the proportions of the wealth owned by every 1/N families and illustrating them as points in the plot produces a curve, namely, the Lorenz curve.

PAI command

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

Parameter description

Parameter	Description	Value range	Required/Optional, default value
inputTableName	Input table	Table name	Required
outputTableName	Name of the output table	Required	NA
colName	Column names separated by commas (,)	NA	(Optional) By default, the entire table is selected.
N	Quantile count	(Optional) Default value: 100	(Optional) Default value: 100
inputPartitions	Partitions used for training in the input table, in the format of Partition_name=valu	NA	(Optional) Default: all partitions

	e The multilevel partition name format is name1=value1/name2=value2. If you specify multiple partitions, separate them with a comma (,).		
lifecycle	(Optional) Life cycle of the output table	Positive integer	No life cycle
coreNum	Number of nodes	Used together with the memSizePerCore parameter, positive integer in the range of [1, 9999]. [Detailed description]	(Optional) Automatically calculated by default
memSizePerCore	Memory size of each node, in MB	Positive integer in the range of [1024, 64*1024]	(Optional) Automatically calculated by default

Example

Data generation

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

PAI command

```
PAI -name LorenzCurve
-project algo_public
```

```
-DinputTableName=maple_test_lorenz_basic10_input
-DcolName=col0
-DoutputTableName=maple_test_lorenz_basic10_output
-DcoreNum=20
-DmemSizePerCore=110;
```

Output description

Output table

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

Machine learning

Contents

Multiclass classification evaluation

Binary classification evaluation

Regression model evaluation

Clustering model evaluation

Prediction

PS-SMART binary classification

PS-SMART multiclass classification

PS-SMART regression

PS linear regression

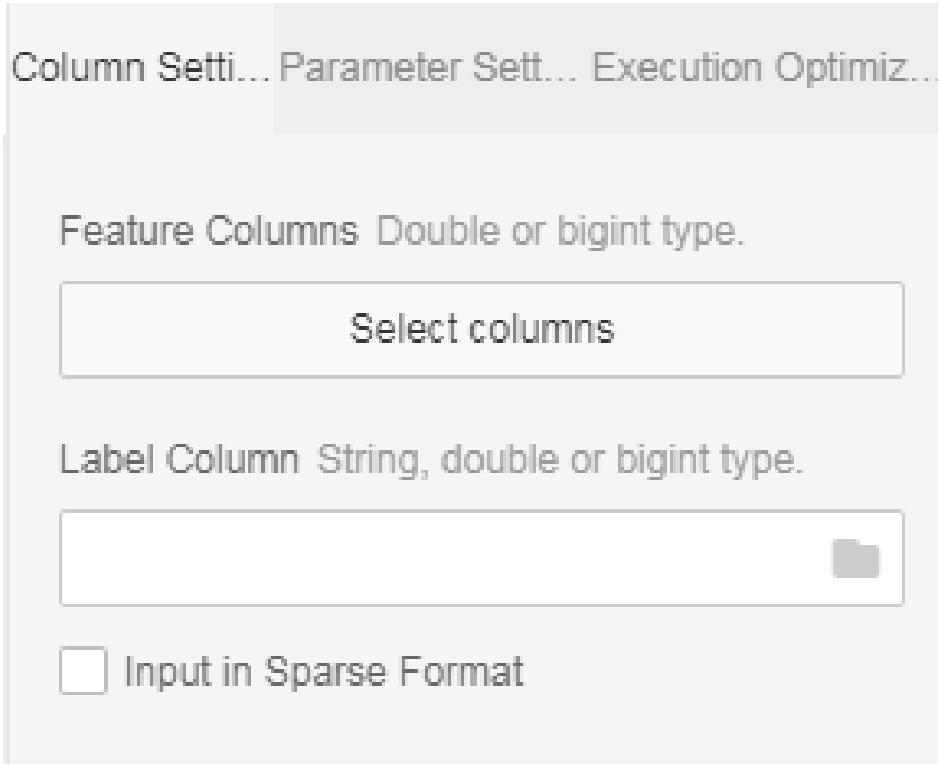
Linear SVM

Developed in the mid 90’ s, Support Vector Machine (SVM) is a statistical learning theory based machine learning method. It seeks to improve the learning machine’ s generalization ability through structural risk minimization, so as to minimize the empirical risk and confidence range. Therefore, good statistics can be obtained from small sample sizes. For more information about SVM, see wiki.

This linear SVM is not implemented using the kernel function. For details about the implementation, see section “6 Trust Region Method for L2-SVM” in http://www.csie.ntu.edu.tw/~cjlin/papers/logistic.pdf. This algorithm only supports binary classification models.

Algorithm component

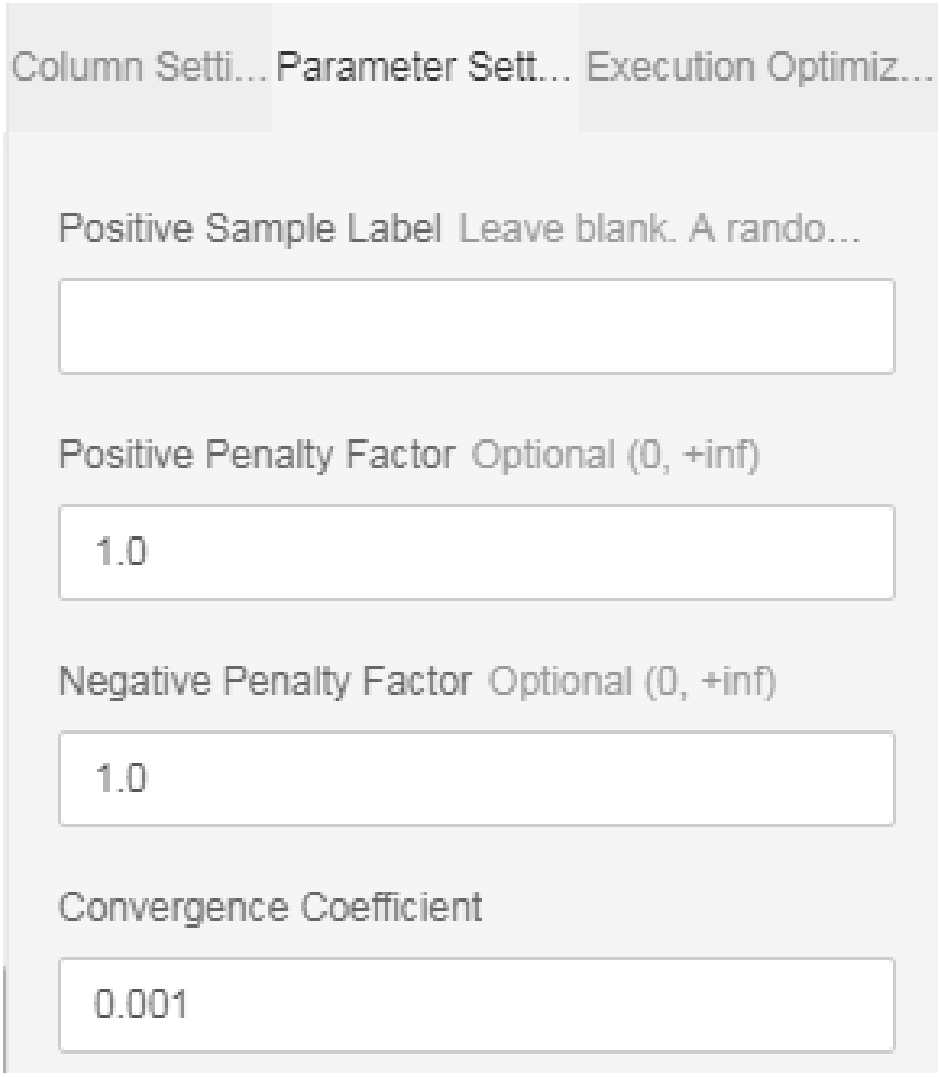
Set field parameters of the component.



Input column: You can select only a column of bigint or double type.

Label column: The data type of the label column can be bigint, double, or string. This component supports only binary classification models.

Set algorithm parameters.



Penalty factor: Defaults to 1.

Base value: (Optional) Positive value. If this parameter is not specified, the system selects a random value. We recommend that you specify this parameter when the positive and negative samples are significantly different.

Positive weight: (Optional) Positive penalty factor. The default value is 1.0, and the value range is (0, ~).

Negative weight: (Optional) Negative penalty factor. The default value is 1.0, and the value range is (0, ~).

Convergence coefficient: (Optional) Convergence deviation. The default value is 0.001, and the value range is (0, 1).

NOTE: If the base value is not specified, the positive weight and negative weight must be the same.

PAI command

```
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";
```

Parameter settings

Parameter	Description	Option	Default value
inputTableName	Name of the input table	NA	NA
inputTableParitions	(Optional) Partitions used for training in the input table, in the format of Partition_name=value. The multilevel format is name1=value1/name2=value2, multiple partitions are separated by commas	NA	All partitions in the input table
modelName	(Required) Name of the output model	NA	NA
featureColNames	(Required) Names of the feature columns used for training in the input table	NA	NA
labelColName	(Required) Name of the label column in the input table	NA	NA
positiveLabel	(Optional) Positive value	NA	Random value selected among label values
negativeCost	(Optional) Negative weight (negative penalty factor)	(0, ∞)	1.0
positiveCost	(Optional) Positive	(0, ∞)	1.0

	weight (positive penalty factor)		
∞	(Optional) Convergence coefficient	(0, 1)	0.001

Example

Training data

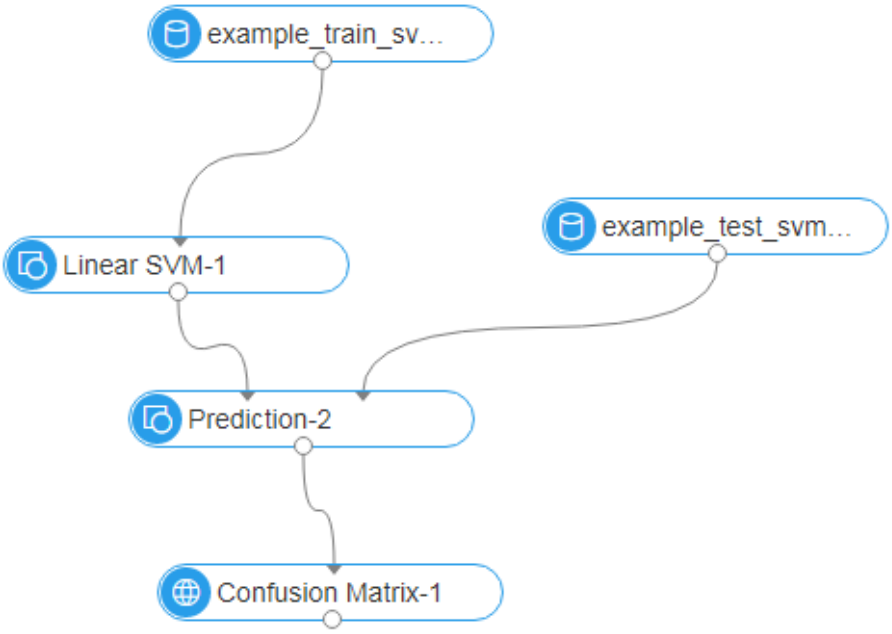
id	y	f0	f1	f2	f3	f4	f5	f6	f7
1	-1	-0.294118	0.487437	0.180328	-0.292929	-1	0.0014902	-0.53117	-0.0333333
2	+1	-0.882353	-0.145729	0.0819672	-0.414141	-1	-0.207153	-0.766866	-0.666667
3	-1	-0.0588235	0.839196	0.0491803	-1	-1	-0.305514	-0.492741	-0.633333
4	+1	-0.882353	-0.105528	0.0819672	-0.535354	-0.777778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

Test data

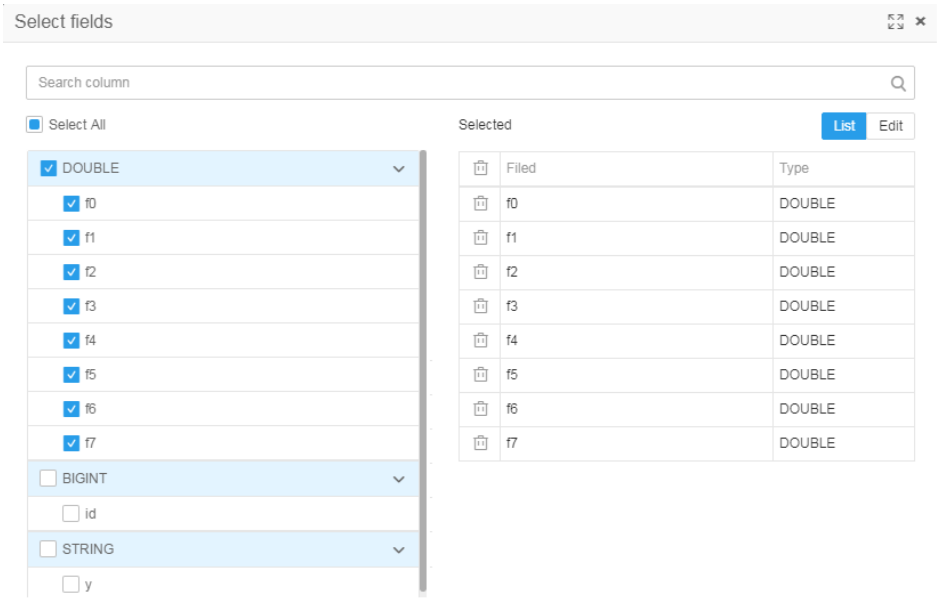
id	y	f0	f1	f2	f3	f4	f5	f6	f7
1	+1	-0.882	0.0854271	0.442623	-0.616	-1	-0.192	-0.725	-0.9

		353			162		25	021	
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	0.0076857 4	-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

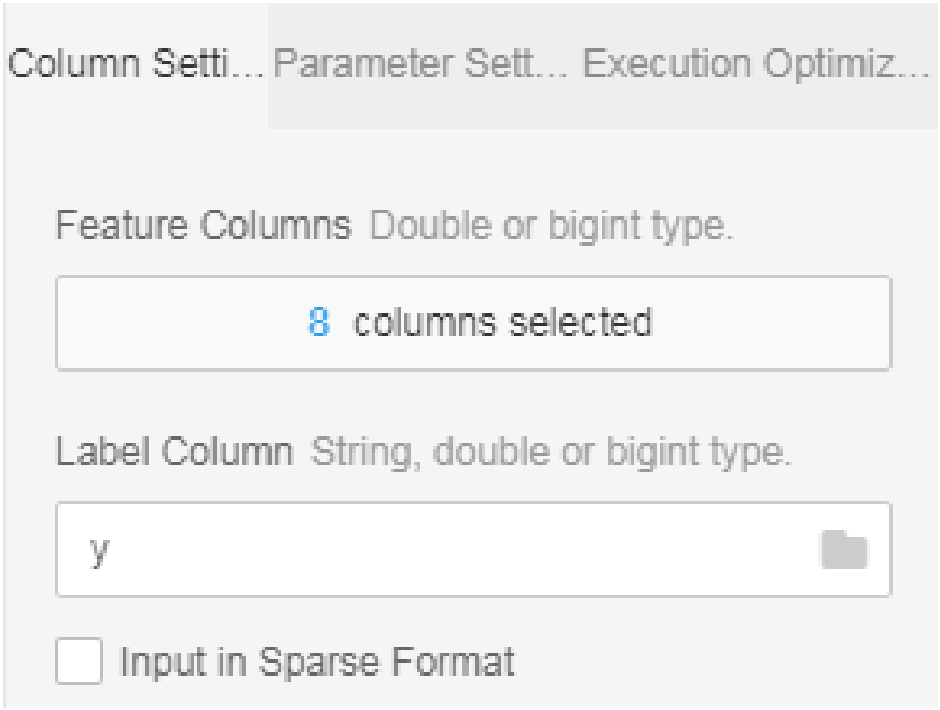
1. Create an experiment svm_example.



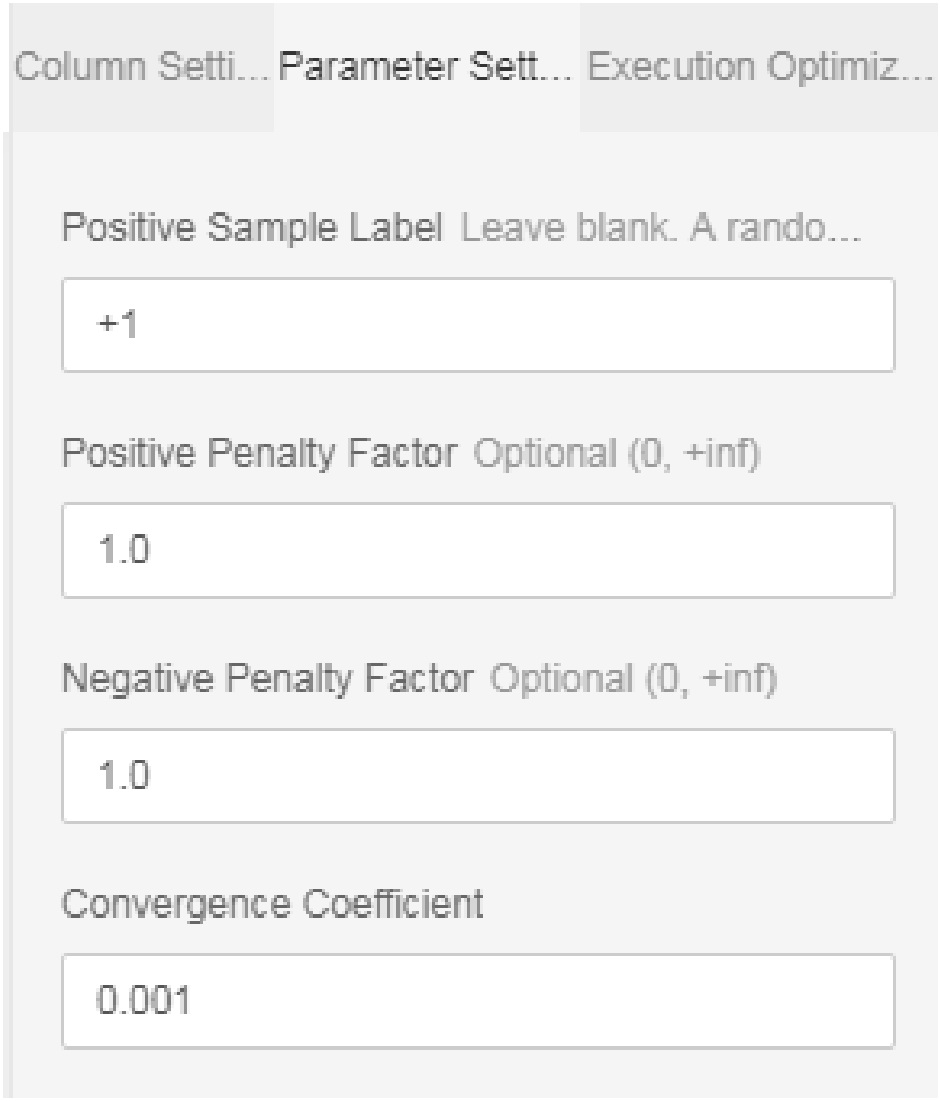
Select feature columns.



Select the label column.

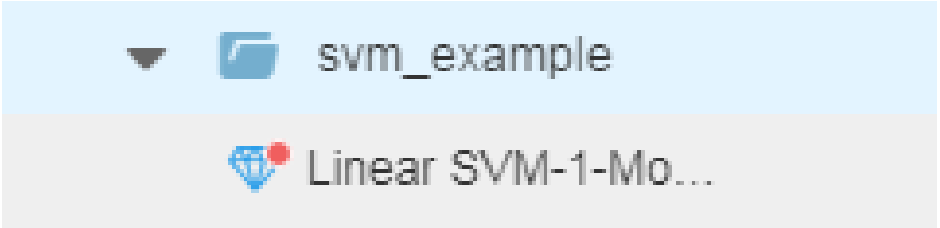


Set SVM parameters.



Run the experiment.

The following model is generated:



The following figure shows the predicted result.

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	0.09966248806840 9...	{"+1": 0.09966248806840972, "-1": -0.09966248806840972}
7	-1	-1	-0.7519810414882623	{"+1": -0.7519810414882623, "-1": 0.7519810414882623}
8	+1	+1	0.1802939705739862...	{"+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}

Logistic regression

Classic logistic regression is a binary classification algorithm. Logistic regression on the algorithm platform supports multiclass classification.

The logistic regression component supports two data formats: sparse and dense.

Logical regression for multiclass classification supports a maximum of 100 classes.

Parameter settings

Parameters of the logistic regression component:

Support for sparse matrix: The component supports the sparse matrix format.

Base value: (Optional) Specify the label value of the training coefficient in the case of binary classification. If this parameter is left blank, the system selects a random value.

Maximum iterations: (Optional) Maximum number of L-BFGS iterations. The default value is 100.

Convergence deviation: (Optional) Condition for L-BFGS termination, that is, the log-likelihood deviation between two iterations. The default value is 1.0e-06.

Regularization type: (Optional) Options are l1’, ‘l2’, and ‘None’. The default value is ‘l1’.

Regularization coefficient: (Optional) The default value is 1.0. This parameter is ignored when regularizedType is set to None.

PAI command (Not inheriting the type setting node)

```
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: component name.

project: Name of the project. This parameter specifies the space where an algorithm is located. The default value is algo_public. If you change the default value, the system returns an error.

modelName: Name of the output model.

regularizedLevel: (Optional) Regularization coefficient. The default value is 1.0. This parameter is ignored if regularizedType is set to None.

maxIter: (Optional) Maximum iterations. It specifies the maximum number of L-BFGS iterations. The default value is 100.

regularizedType: (Optional) Regularization type. Options are l1’, ‘l2’, and ‘None’. The default value is ‘l1’.

epsilon: (Optional) Convergence deviation. It is the condition for L-BFGS termination, that is, the log-likelihood deviation between two iterations. The default value is 1.0e-06.

labelColName: Name of the label column in the input table.

featureColNames: Names of the feature columns used for training in the input table.

goodValue: (Optional) Base value. For binary classification, specify the label value of the training coefficient. If this parameter is left blank, the system selects a random value.

inputTableName: Name of the input table for training.

Example

Binary classification

Test data

SQL statement for data generation

```
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;
```

Input data description

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

Running command

```
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";
```

Running result

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

Multiclass classification

Test data

SQL statement for data generation

```
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;
```

Input data description

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

Running command

```
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";
```

Running result

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": 2.018833979850994e-07,
"1": 2.324679066261573e-07,
"2": 0.9999999155958832} |
| 1 | 1 | 0.9999999155958832 | {
"0": 2.018833979850994e-07,
"1": 0.9999999155958832,
"2": 2.324679066261569e-07} |
+------------+-------------------+------------------+-------------------+
```

GBDT binary classification

This component is used to resolve binary classification issues based on GBDT regression and sorting. Values greater than the preset threshold are positive values and those smaller than the threshold are negative values.

PAI command

```
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"
```

Parameter description

Parameter	Description	Value range	Required/Optional, default value
inputTableName	Input table	Table name	Required
featureColNames	Names of the feature columns used for training in the input table	Column names	Optional, default: all columns with values
labelColName	Name of the label column in the input table	Column name	Required
inputTablePartitions	Partitions used for training in the input table, in the format of partition_name=value. The multilevel format is name1=value1/name2=value2. Multiple partitions are separated by commas (,).	NA	(Optional) All partitions are selected by default.

modelName	Name of the output model	NA	Required
outputImportanceTableName	Name of the output feature importance table	NA	Optional
groupIDColName	Name of a grouping column	Column name	Optional, default: full table
lossType	Loss function type, 0: GBRANK, 1: LAMBDAMART_DCG, 2: LAMBDAMART_NDCG, 3: LEAST_SQUARE, 4: LOG_LIKELIHOOD	0, 1, 2, 3, 4	Optional, default: 0
~~metricType~~	Metric type, 0(NDCG)-: normalized discounted cumulative gain; 1(DCG): discounted cumulative gain; 2 (AUC) adaptive only to 0/1 label	0, 1, 2	Optional, default: 2
treeCount	Number of trees	[1, 10,000]	Optional, default: 500
shrinkage	Learning rate	(0, 1]	Optional, default: 0.05
maxLeafCount	Maximum number of leaves, which must be an integer	[2, 1000]	Optional, default: 32
~~maxDepth~~	Maximum depth of a tree, which must be an integer	[1, 11]	Optional, default: 11
minLeafSampleCount	Minimum number of samples on a leaf node, which must be an integer	[100, 1000]	Optional, default: 500
sampleRatio	Ratio of samples collected during the training	(0, 1]	Optional, default: 0.6
featureRatio	Ratio of features collected during the training	(0, 1]	Optional, default: 0.6
tau	Parameter Tau in gbrank loss	[0, 1]	Optional, default: 0.6
p	Parameter p in gbrank loss	[1, 10]	Optional, default: 1

randSeed	Random seed	[0, 10]	Optional, default: 0
newtonStep	Whether to use the newton method	0,1	Optional, default: 1
featureSplitValueMaxSize	Number of records that a feature can split into	[1, 1000]	Optional, default: 500
lifecycle	Lifecycle of the output table	NA	Optional, default: not set

Example

Data generation

```
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;
```

PAI command

Training

```
drop offlinemodel if exists gbdt_lr_test_model;
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"
```

Prediction

```
drop table if exists gbdt_lr_test_prediction_result;
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";
```

Input description

gbdt_lr_test_input

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

Output description

gbdt_lr_test_prediction_result

label	prediction_result	prediction_score	prediction_detail
0	0	0.9984308925552831	{“0”: 0.9984308925552831, “1”: 0.001569107444716943}
0	0	0.9984308925552831	{“0”: 0.9984308925552831, “1”: 0.001569107444716943}
1	1	0.9982721832240973	{“0”: 0.001727816775902724, “1”: 0.9982721832240973}
1	1	0.9982721832240973	{“0”: 0.001727816775902724, “1”: 0.9982721832240973}
0	0	0.9984308925552831	{“0”: 0.9984308925552831, “1”: 0.001569107444716943}
0	0	0.9984308925552831	{“0”: 0.9984308925552831, “1”:

			0.001569107444716943}

Important notes

GBDT and GBDT_LR have different default types of loss functions. The default loss function of GBDT is regression loss:mean squared error loss, and that of GBDT_LR is logistic regression loss. The system automatically writes the default loss function for GBDT_LR, and you do not need to set the loss function type for it.

For GBDT binary classification, the label column can only be a binary classification column and does not support data of the string type.

When connecting the ROC curve, select the custom mode for the prediction component and select a base value.

K-NN

This component selects from the training table K records nearest to each row of the prediction table, and takes the class with the maximum number of records among the K records as the class of the specific row.

The K-NN algorithm solves classification issues.

PAI command

```
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;
```

Parameter description

Parameter	Description	Option	Default value
trainTableName	(Required) Name of the input table	NA	NA
trainFeatureColNames	(Required) Names of the feature columns in the training table	NA	NA
trainLabelColName	(Required) Name of	NA	NA

	the label column in the training table		
trainTablePartitions	(Optional) Partitions used for training in the training table	NA	All partitions
predictTableName	(Required) Name of the prediction table	NA	NA
outputTableName	(Required) Name of the output table	NA	NA
predictFeatureColNames	(Optional) Names of feature columns in the prediction table	NA	Same as trainFeatureColNames
predictTablePartitions	(Optional) Partitions used for prediction in the prediction table	NA	All partitions
appendColNames	(Optional) Name of the prediction table appended to the output table	NA	Same as predictFeatureColNames
outputTablePartition	(Optional) Partitions in the output table	NA	Output table not partitioned
k	(Optional) Number of nearest neighbors	Positive integer in the range of [1, 1000]	100
enableSparse	Whether data in the input table is in sparse format	true, false	Optional, default: false
itemDelimiter	Delimiter used between key-value pairs when data in the input table is in sparse format		Optional, default: space
kvDelimiter	Delimiter used between keys and values when data in the input table is in sparse format		Optional, default: colon
coreNum	Number of nodes	Used together with the memSizePerCore parameter, positive integer in the range of [1, 20000]	(Optional) Automatically calculated by default
memSizePerCore	Memory size of each node, in MB	Positive integer in the range of [1024, 64*1024]	(Optional) Automatically calculated by default
lifecycle	(Optional) Lifecycle of the output table	Positive integer	No lifecycle

Example

Test data

```
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 command

```
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;
```

Output description

f0 and f1 are appended columns in the output table.

- prediction_result: classification result.
- prediction_score: probability of the classification result.

  prediction_detail: latest K conclusions and their probabilities.

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}

Random forest

A random forest is a classifier that contains multiple decision trees. The output class is the mode of classes of the individual trees.

ID3, C4.5, or CART can be used as the single decision tree algorithm.
For more information about the random forest, see wiki.

PAI command

```
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";
```

Parameter description

Parameter	Description	Value range	Required/Optional, default value/act
inputTableName	Input table	Table name	Required
inputTablePartitions	Partitions used for training in the input table, in the format of partition_name=value. The multilevel format is name1=value1/name2=value2. Multiple partitions are separated by commas (,).	NA	(Optional) All partitions are selected by default.
labelColName	Name of the label column in the input table	Column name	Required
modelName	Name of the output model	NA	Required
treeNum	Number of trees in the random forest	Positive integer in the range of (0, 1000]	Required
weightColName	Name of the weight column in the input table	NA	Optional, default: no weight column
featureColNames	Names of feature	NA	Optional, default: all

	columns used for training in the input table		columns except labelColName and weightColName
excludedColNames	Names of feature columns excluded from training in the input table, mutually exclusive with featureColNames	NA	Optional, default: empty
forceCategorical	The default feature parsing rule is: Parse columns of string, boolean, and datetime types as discrete columns, and parse columns of double and bigint types as contiguous columns. You can specify the forceCategorical parameter to parse bigint columns as categorical columns.		Optional, default: int columns are parsed as contiguous columns
algorithmTypes	Location of the single decision tree algorithm in the forest	The value contains two characters. If the forest has n trees and algorithmTypes is set to [a,b], then [0,a) indicates ID3, [a,b) indicates CART, and [b,n) indicates C4.5. If this parameter is set to [2, 4] for a forest with five trees, [0, 1) indicates the ID3 algorithm, [2, 3) indicates the CART algorithm, and 4 indicates the C4.5 algorithm. If the value is None, the algorithms are evenly allocated in the forest.	Optional, default: algorithms evenly allocated in the forest
randomColNum	Number of random features selected in each split during generation of a single decision tree	[1-N], where N is the number of features	Optional, default: log2N
minNumObj	Minimum number of leaf nodes	Positive number	Optional, default: 2
minNumPer	Minimum ratio of	[0,1]	Optional, default: 0.0

	leaf nodes to parent nodes		
maxTreeDeep	Maximum depth of a tree	[1, ∞)	Optional, default: ∞
maxRecordSize	Number of input random values on each tree in the forest	(1000, 1,000,000]	Optional, default: 100,000

Example

Test data

```
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 command

```
 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";
```

Output description

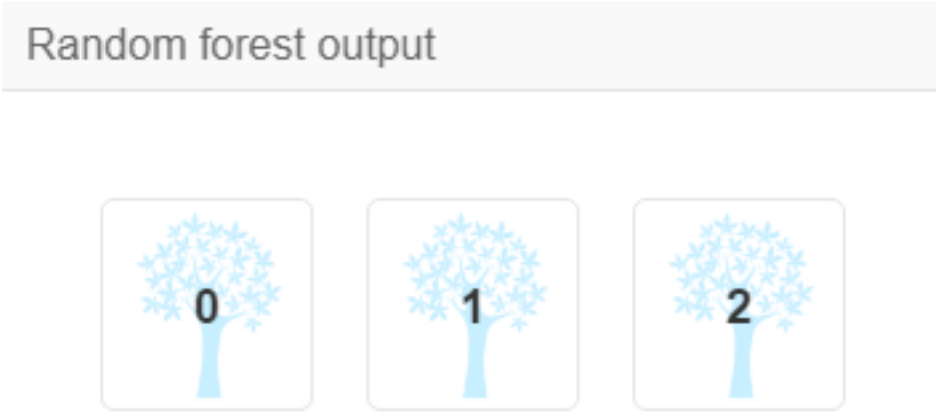
Model PMML

```
<?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
```

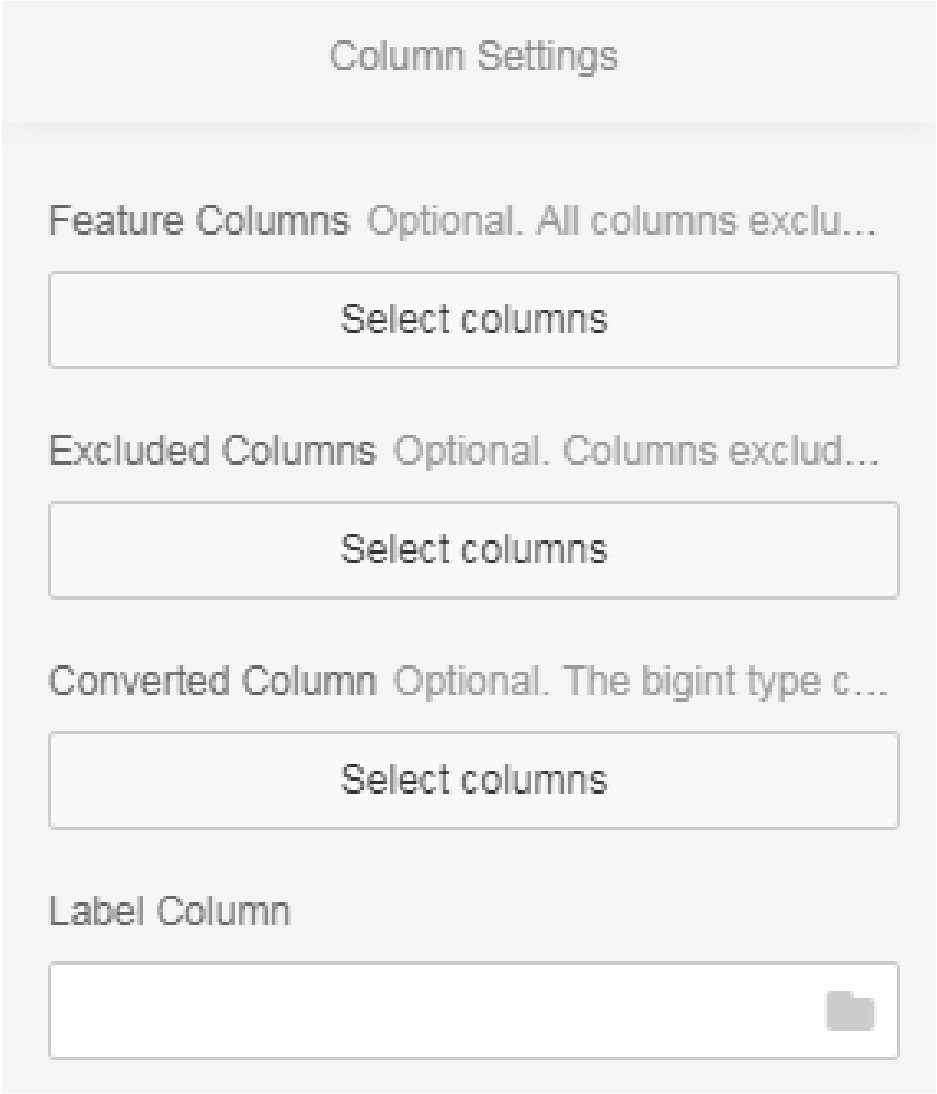
Model visualization



Naive Bayes

Naive Bayes classifier is a simple probabilistic classifier applying Bayes' theorem with strong (naive) independence assumptions between the features. A probabilistic model that can more accurately describe this potential is called an independent feature model. For details about this algorithm, see Naive Bayes classifier.

Algorithm component



Feature column: Data type of feature columns can be string, double, or bigint.

Label column: You can select only a column other than feature columns. Its data type can be double, string, or bigint.

PAI command

```
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";
```

Parameter description

Parameter	Description	Option	Default value

inputTableName	(Required) Name of the input table	NA	NA
inputTablePartitions	(Optional) Partitions used for training in the input table	Format: Partition_name=value. The multilevel format is name1=value1/name2=value2; multiple partitions are separated by commas	All partitions in the input table
modelName	(Required) Name of the output model	NA	NA
labelColName	(Required) Name of the label column in the input table	NA	NA
featureColNames	(Optional) Names of feature columns used for training in the input table	NA	All columns except the label column
excludedColNames	Names of feature columns excluded from training in the input table, mutually exclusive with featureColNames	NA	Empty
forceCategorical	(Optional) The default feature parsing rule is: Parse columns of string, boolean, and datetime types as discrete columns, and parse columns of double and bigint types as contiguous columns. You can specify the forceCategorical parameter to parse bigint columns as categorical columns.	NA	int columns are parsed as contiguous columns

Example

Training data

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

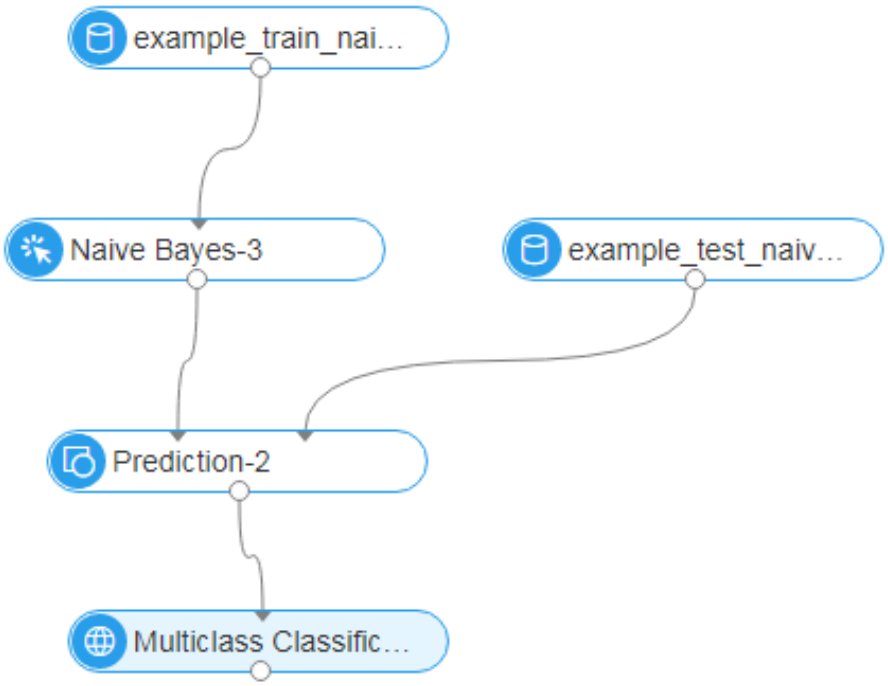
2	+1	-0.882353	-0.145729	0.0819672	-0.414141	-1	-0.207153	-0.766866	-0.666667
3	-1	-0.0588235	0.839196	0.0491803	-1	-1	-0.305514	-0.492741	-0.633333
4	+1	-0.882353	-0.105528	0.0819672	-0.535354	-0.777778	-0.162444	-0.923997	-1
5	-1	-1	0.376884	-0.344262	-0.292929	-0.602837	0.28465	0.887276	-0.6
6	+1	-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

Test data

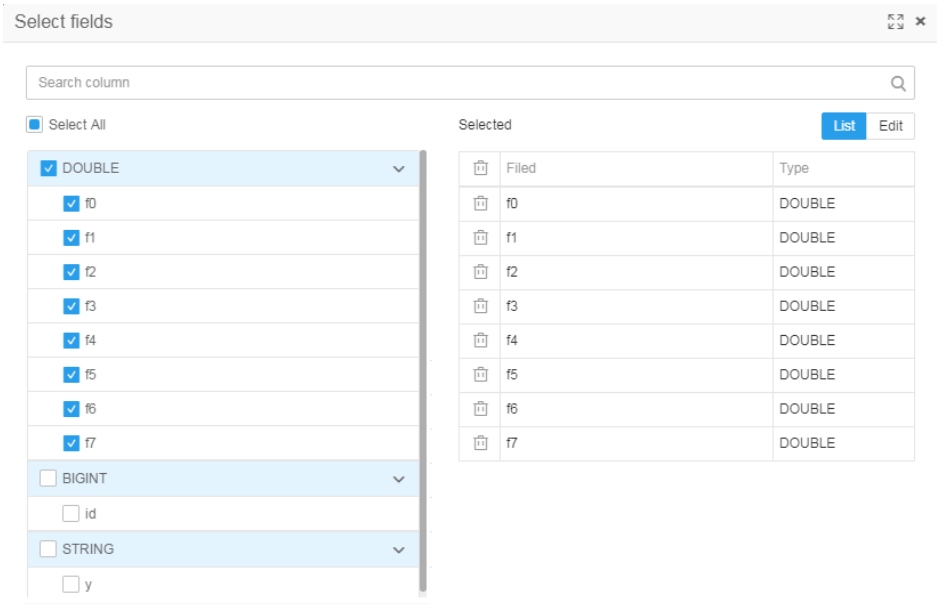
id	y	f0	f1	f2	f3	f4	f5	f6	f7
1	+1	-0.882353	0.0854271	0.442623	-0.616162	-1	-0.19225	-0.725021	-0.9
2	+1	-0.294118	-0.0351759	-1	-1	-1	-0.293592	-0.904355	-0.766667
3	+1	-0.882353	0.246231	0.213115	-0.272727	-1	-0.171386	-0.981213	-0.7
4	-1	-0.176471	0.507538	0.278689	-0.414141	-0.702128	0.0491804	-0.475662	0.1
5	-1	-0.529412	0.839196	-1	-1	-1	-0.153502	-0.885568	-0.5
6	+1	-	0.246	-	-	-1	0.067	-	-1

		0.882353	231	0.0163934	0.353535		0641	0.627669	
7	-1	-0.882353	0.819095	0.278689	-0.151515	-0.307329	0.19225	0.0076857 4	-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

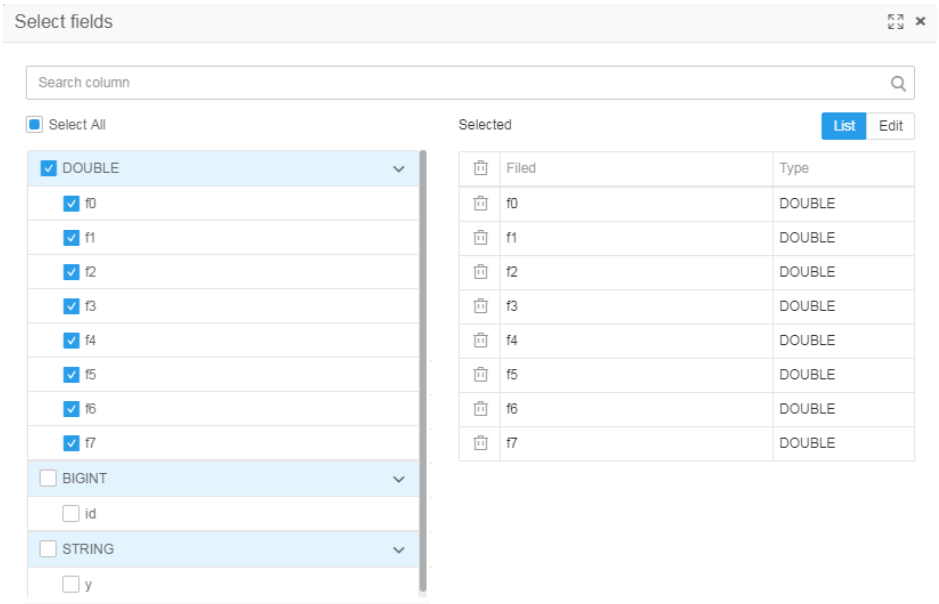
Create an experiment.



Select feature columns.

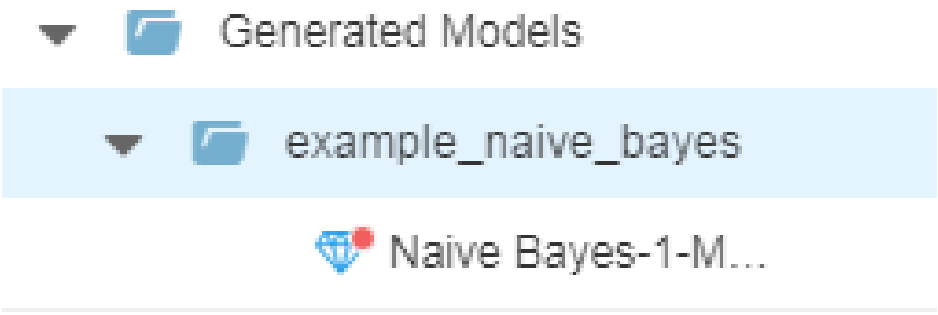


Select the label column.



Run the experiment.

The following model is generated.



The following figure shows the predicted result.

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-means clustering

K-means clustering is the most widely used clustering algorithm, which divides n objects into k clusters to maintain high similarity in each cluster. The similarity is calculated based on the average value of objects in a cluster.

This algorithm randomly selects k objects, each of which originally represents the average value or center of a cluster. Then, the algorithm assigns the remaining objects to the nearest clusters based on their distances from the center of each cluster, and re-calculates the average value of each cluster. This process repeats until the criterion function converges.

The K-means clustering algorithm assumes that object attributes are obtained from the spatial vector, and its objective is to ensure the minimum mean square error sum inside each group.
For more information about K-means clustering, see wiki.

PAI command

```
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
```

Parameter description

Parameter	Description	Value range	Required/Optional, default value/act
inputTableName	Input table	Table name	Required
selectedColNames	Names of the columns used for training in the input table, which are separated by commas, columns of int and double types supported	NA	Optional, default: all columns in the input table
inputTablePartitions	Partitions used for training in the input table, in the format of partition_name=value. The multilevel format is name1=value1/name2=value2. Multiple partitions are separated by commas (,).	NA	(Optional) All partitions are selected by default.
centerCount	Number of clusters	Positive integer in the range of [1, 1000]	Required
loop	Maximum number of iterations	Positive integer in the range of [1, 1000]	Optional, default: 100
accuracy	Algorithm terminal condition. The algorithm is terminated if the variation between two iterations is smaller than this value.	NA	Optional, default: 0.0
distanceType	Distance measuring method	euclidean (Euclidean distance), cosine (included angle cosine), cityblock (Manhattan	Optional, default: euclidean

		distance)	
initCenterMethod	Centroid initialization method	random (random sampling), topk (first k rows of the input table), uniform (even distribution), kmpp (k-means++), external (specifying the initial centroid table)	Optional, default: random
initCenterTableName	Name of the initial centroid table	Table name	Valid when initCenterMethod is set to external
seed	Initial random seed	Positive integer	Optional, default: current time. If the seed is set to a fixed value, clustering results do not fluctuate greatly.
enableSparse	Whether data in the input table is in sparse format	true, false	Optional, default: false
itemDelimiter	Delimiter used between key-value pairs when data in the input table is in sparse format	NA	Optional, default: space
kvDelimiter	Delimiter used between keys and values when data in the input table is in sparse format	NA	Optional, default: colon
appendColNames	Names of appended columns in the inputTableName to be exported to the idxTableName table, which are separated by commas		Optional, default: no appended column
modelName	Name of the output model	Model name	Required
idxTableName	Clustering result output table corresponding to the input table, specifying class number of each record after clustering	Table name	Required
idxTablePartition	Partition of the output cluster table	Table name	Optional, default: no partition

clusterCountTableName	Output cluster count table, showing the number of nodes in each cluster		Optional, no output
centerTableName	Output cluster center table	NA	Optional, about to be deprecated, the modelName parameter recommended

Distance measurement method

Parameter	Description
euclidean	$d(x-c)=(x-c)(x$
cosine	$d(x-c)=0.5-0.5*\frac{x}{\sqrt{xx}}$
cityblock	$d(x-c)=\lvert x-$

Centroid initialization method

Parameter	Description
random	Sample K initial centers randomly from the input table. The initial random seed can be specified using the seed parameter.
topk	Read the first K rows of the input table as the initial centers
uniform	Calculate K evenly distributed initial centers in the input table in ascending order of values
kmpp	Use the k-means++ algorithm to select K initial centers. For details about this algorithm, see wiki
external	Specify an external initial center table

Example

Test data

```
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 command

```
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
```

Output description

Model 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
```

Model visualization

Output cluster table: The number of rows equals the total number of rows in the input table. The values in each row represent the cluster numbers of the points in the corresponding row of the input table.

cluster_index
2
2
1
0
0

Output cluster count table: The number of rows equals the number of clusters. The values in each row represent the number of points in the corresponding clusters.

cluster_count
2
1
2

Linear regression

Linear regression is a model used to analyze the linear relationship between a dependent variable and multiple independent variables. For details, see https://en.wikipedia.org/wiki/Linear_regression.

PAI command

```
PAI -name linearregression
-project algo_public
-DinputTableName=lm_test_input
-DfeatureColNames=x
-DlabelColName=y
-DmodelName=lm_test_input_model_out;
```

Algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	NA	NA
modelName	(Required) Name of the output model	NA	NA
outputTableName	(Optional) Name of the output model evaluation table	Required when enableFitGoodness is true	""
labelColName	(Required) Dependent variable	Double or bigint type, only one column allowed	NA
featureColNames	(Required) Independent variables	double or bigint in dense format and string type in sparse format, multiple columns allowed	NA
inputTablePartitions	(Optional) Partitions in the input table	NA	""
maxIter	(Optional) Maximum number of iterations	NA	100
epsilon	(Optional) Minimum likelihood deviation	NA	0.000001
enableSparse	(Optional) Whether the input table data is in sparse format	true, false	false
enableFitGoodness	(Optional) Whether to perform model evaluation, based on metrics including R-squared, AdjustedR-Squared, AIC, degree of freedom, residual standard deviation, and deviation	[true, false]	false
enableCoefficientEstimate	(Optional) Whether to perform regression coefficient evaluation, based on metrics including value t, value p, and confidence interval [2.5%, 97.5%]. This parameter is valid only when enableFitGoodness	true, false	false

	is true and is set to false otherwise.		
itemDelimiter	(Optional) Delimiter used between key-value pairs in sparse format, valid only when enableSparse is true	-	Space on the command line interface and comma (,) on web pages
kvDelimiter	(Optional) Delimiter used between keys and values in sparse format, valid only when enableSparse is true	NA	Colon (:)
lifecycle	(Optional) Lifecycle of the model evaluation output table	0	-1
coreNum	(Optional) Total number of instances	[1, 800)	Automatically calculated
memSizePerCore	(Optional) Memory size for core	[1024, 20*1024]	Automatically calculated

Example

Test data

SQL statement for data generation:

```
  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;
```

Running command

```
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;
```

Running result

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 | 0.3537449832384626 5 | 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 regression

GBDT, short for gradient boosting decision tree, is an iterative decision tree algorithm based on multiple decision trees. The final output is the sum of conclusions of all trees. GBDT applies to almost all regression models (linear or nonlinear) and has a wider scope of application than logistic regression that only applies to linear regression. For details, see the following references (a) A Regression Framework for Learning Ranking Functions Using Relative Relevance Judgments, (b) From RankNet to LambdaRank to LambdaMART: An Overview.

PAI command

```
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"
```

Parameter description

Parameter	Description	Value range	Required/Optional, default value
inputTableName	Input table	Table name	Required
featureColNames	Names of the	Column names	Optional, default: all

	feature columns used for training in the input table		columns with values
labelColName	Name of the label column in the input table	Column name	Required
inputTablePartitions	Partitions used for training in the input table, in the format of partition_name=value. The multilevel format is name1=value1/name2=value2. Multiple partitions are separated by commas (,).	NA	(Optional) All partitions are selected by default.
modelName	Name of the output model	NA	Required
outputImportanceTableName	Name of the output feature importance table	NA	Optional
groupIDColName	Name of a grouping column	Column name	Optional, default: full table
lossType	Loss function type, 0: GBRANK, 1: LAMBDAMART_DCG, 2: LAMBDAMART_NDCG, 3: LEAST_SQUARE, 4: LOG_LIKELIHOOD	0, 1, 2, 3, 4	Optional, default: 0
metricType	Metric type, 0(NDCG)-: normalized discounted cumulative gain; 1(DCG): discounted cumulative gain; 2 (AUC) adaptive only to 0/1 label	0, 1, 2	Optional, default: 2
treeCount	Number of trees	[1, 10,000]	Optional, default: 500
shrinkage	Learning rate	(0, 1]	Optional, default: 0.05
maxLeafCount	Maximum number of leaves, which must be an integer	[2, 1000]	Optional, default: 32
maxDepth	Maximum depth of	[1, 11]	Optional, default: 11

	a tree, which must be an integer		
minLeafSampleCount	Minimum number of samples on a leaf node, which must be an integer	[100, 1000]	Optional, default: 500
sampleRatio	Ratio of samples collected during the training	(0, 1]	Optional, default: 0.6
featureRatio	Ratio of features collected during the training	(0, 1]	Optional, default: 0.6
tau	Parameter Tau in gbrank loss	[0, 1]	Optional, default: 0.6
p	Parameter p in gbrank loss	[1, 10]	Optional, default: 1
randSeed	Random seed	[0, 10]	Optional, default: 0
newtonStep	Whether to use the newton method	0,1	Optional, default: 1
featureSplitValueMaxSize	Number of records that a feature can split into	[1, 1000]	Optional, default: 500
lifecycle	Lifecycle of the output table	NA	Optional, default: not set

Example

Data generation

```
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;
```

PAI command

Training

```
drop offlinemodel if exists gbdt_ls_test_model;
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"
```

Prediction

```
drop table if exists gbdt_ls_test_prediction_result;
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"
```

Input description

gbdt_ls_test_input

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

Output description

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}

Collaborative filtering (etrec)

Etrec is an item-based collaborative filtering algorithm that uses two input columns and provides the top K items with the highest similarity as the output.

For more information about jaccard, see Jaccard_index.

PAI command

```
PAI -name pai_etrec
-project algo_public
-DsimilarityType="wbcosine"
-Dweight="1"
-DminUserBehavior="2"
-Dlifecycle="28"
-DtopN="2000"
-Dalpha="0.5"
-DoutputTableName="etrec_test_result"
-DmaxUserBehavior="500"
-DinputTableName="etrec_test_input"
-Doperator="add"
-DuserColName="user"
-DitemColName="item"
```

Parameter description

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	NA	NA
userColName	(Required) Name of the selected user column in the input table	NA	NA
itemColName	(Required) Name of the selected item column in the input table	NA	NA
~~payloadColName~~	(Optional) Name of the selected payload column in the input	NA	No payload column

	table		
inputTablePartitions	(Optional) Names of the selected partitions in the input table	NA	All partitions in the input table
outputTableName	(Required) Name of the output table	NA	NA
outputTablePartition	(Optional) Partition of the output table	NA	No partition
similarityType	(Optional) Type of similarity	wbcosine, asymcosine, jaccard	wbcosine
topN	(Optional) N items with the highest similarity	[1, 10,000]	2000
minUserBehavior	(Optional) Minimum user behavior	[2,)	2
maxUserBehavior	(Optional) Maximum user behavior	[2, 100,000]	500
itemDelimiter	(Optional) Delimiter used between items in the output table	NA	" "
kvDelimiter	(Optional) Delimiter used between keys and values in the output table	NA	":"
alpha	(Optional) Value of the smoothing factor for asymcosine	NA	0.5
weight	(Optional) Weighting exponent for asymcosine	NA	1.0
operator	(Optional) Action taken when the same items exist for one user	add, mul, min, max	add
lifecycle	(Optional) Lifecycle of the output table	NA	1

Example

Data generation

```
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;
```

PAI command

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

Input description*etrec_test_input*

user	item
0	0
0	1
1	0
1	1

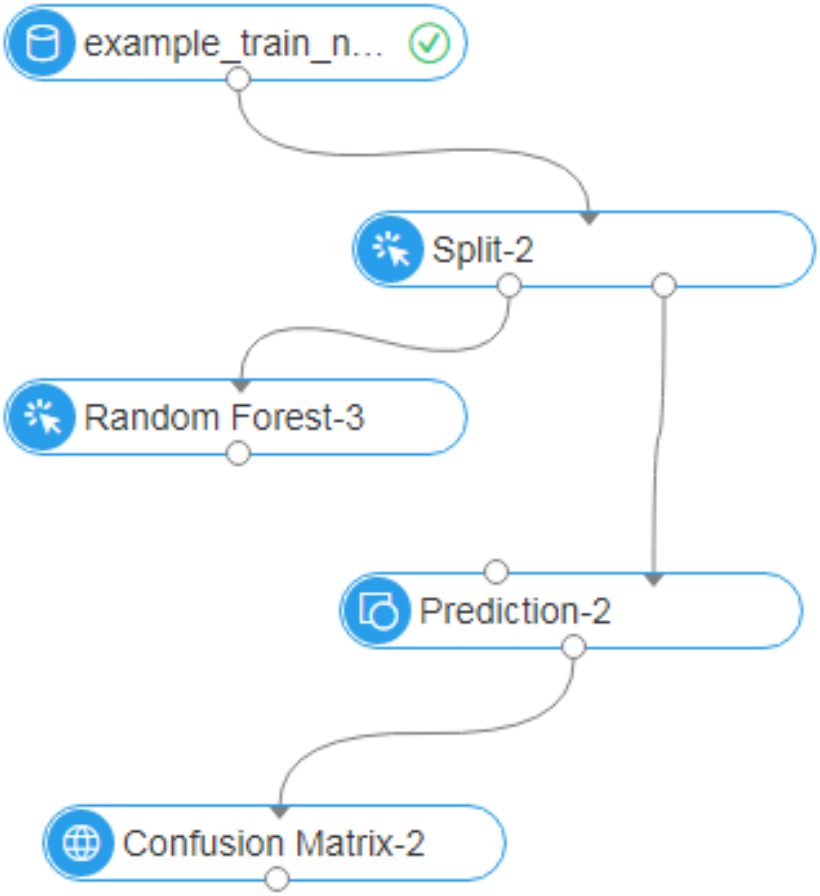
Output description*etrec_test_result*

itemid	similarity
0	1:1
1	0:1

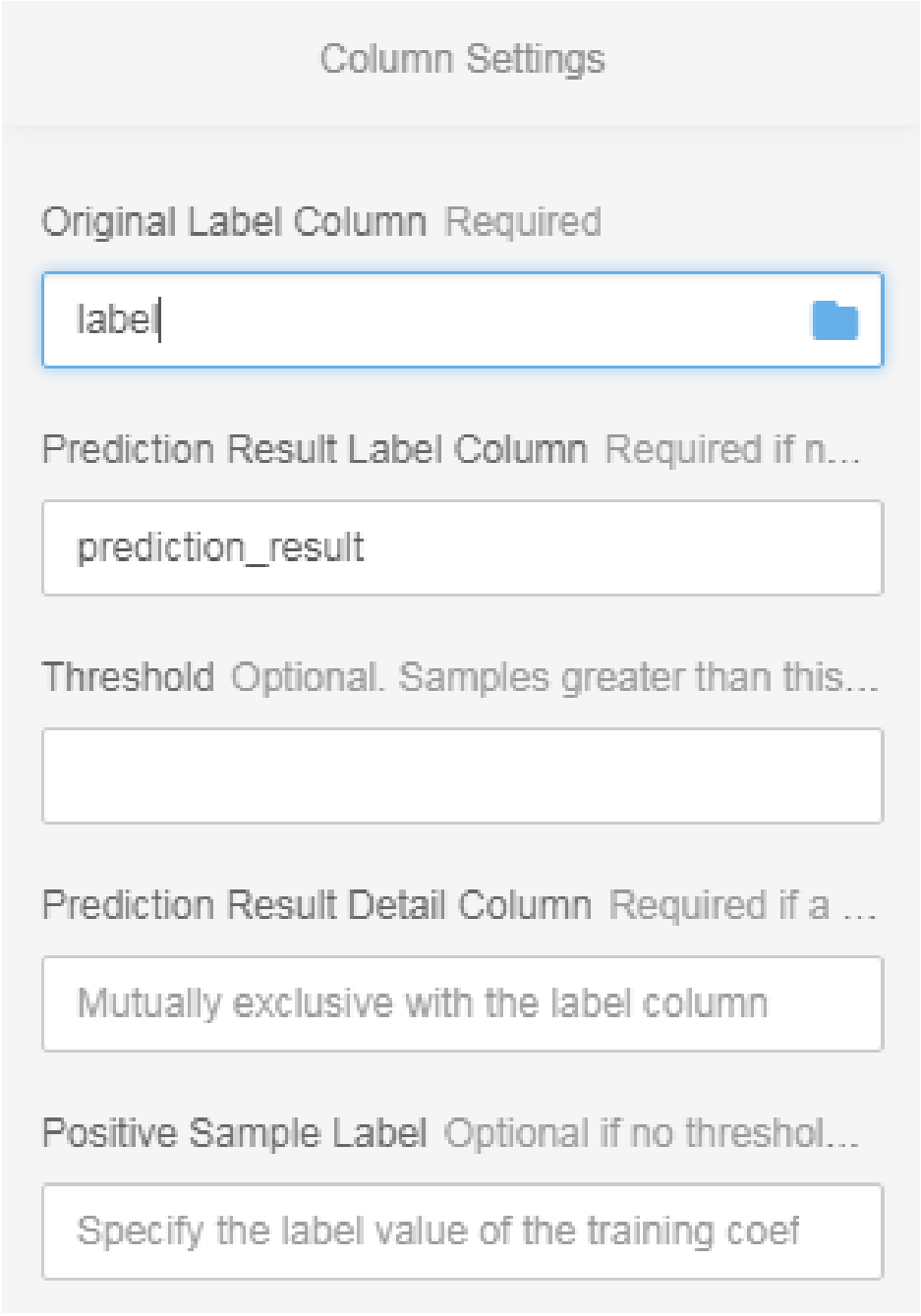
Confusion matrix

The confusion matrix is a visualization tool typically used in supervised learning. (In unsupervised learning, it is called a matching matrix). This tool is used to compare classification results with actual measured values and display the accuracy of classification results in a confusion matrix. For more information about the confusion matrix, see Confusion matrix.

Component connection: Generally, the classification prediction component is on the upper layer. The following figure shows an example.



Parameter settings

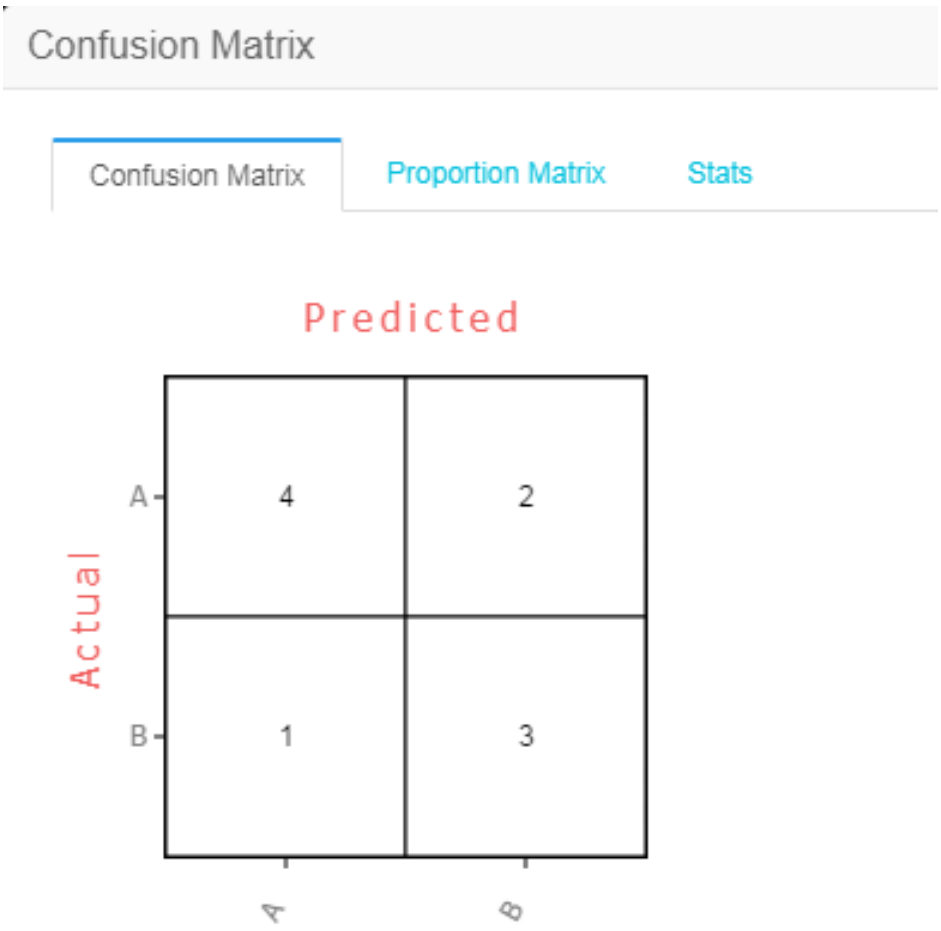


NOTE: Only one of the label column and detail column can be selected in the predicted result table.

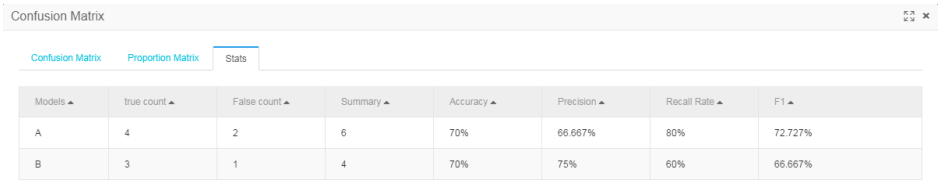
Evaluation report

Right-click the confusion matrix to view the evaluation report.

Confusion matrix



The following figure shows information about each label, including the metrics, accuracy, and recall rate.



PAI command

- Sample of a PAI command without a threshold specified

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

- Sample of a PAI command with a threshold specified

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

Parameter description

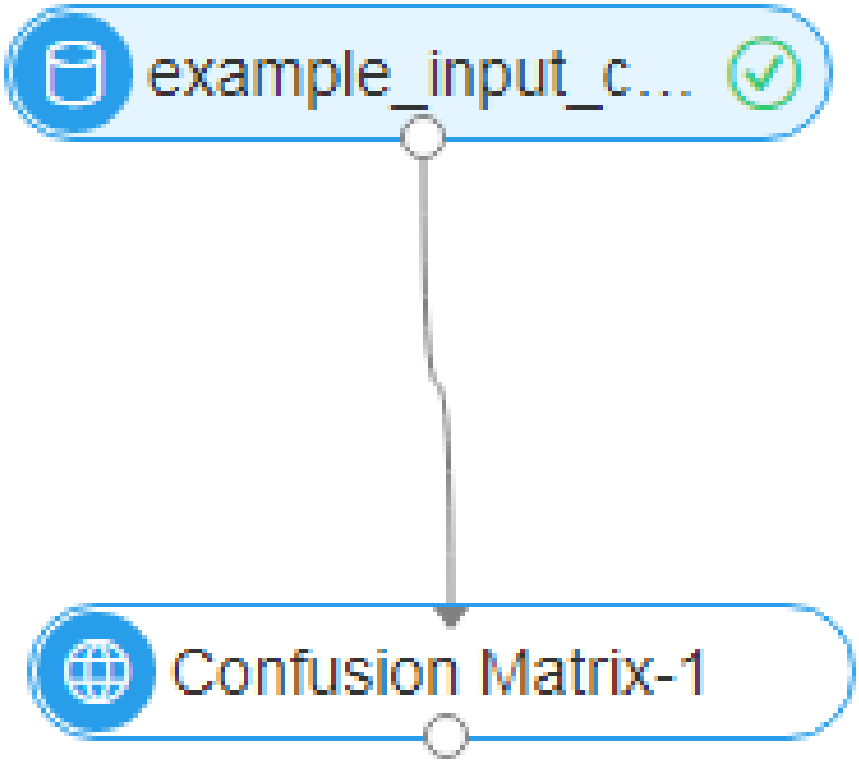
Parameter key	Description	Option	Default value
inputTableName	(Required) Name of the input table (prediction output table)	NA	NA
labelColName	(Required) Name of the original label column	NA	NA
outputTableName	(Required) Name of the output table that stores the confusion matrix	NA	NA
inputTablePartition	(Optional) Partition of the input table	NA	All partitions in the input table
predictionColName	(Optional) Name of the label column of the predicted result table, required when no threshold is specified	NA	NA
predictionDetailColName	(Optional) Name of the detail column of the predicted result table, required when a threshold is specified	NA	NA
threshold	(Optional) Threshold for a good value	NA	0.5
goodValue	(Optional) Label value of a training coefficient during binary classification, required when a threshold is specified	NA	NA
lifecycle	(Optional) Lifecycle of the output table	Positive integer	No lifecycle

Example

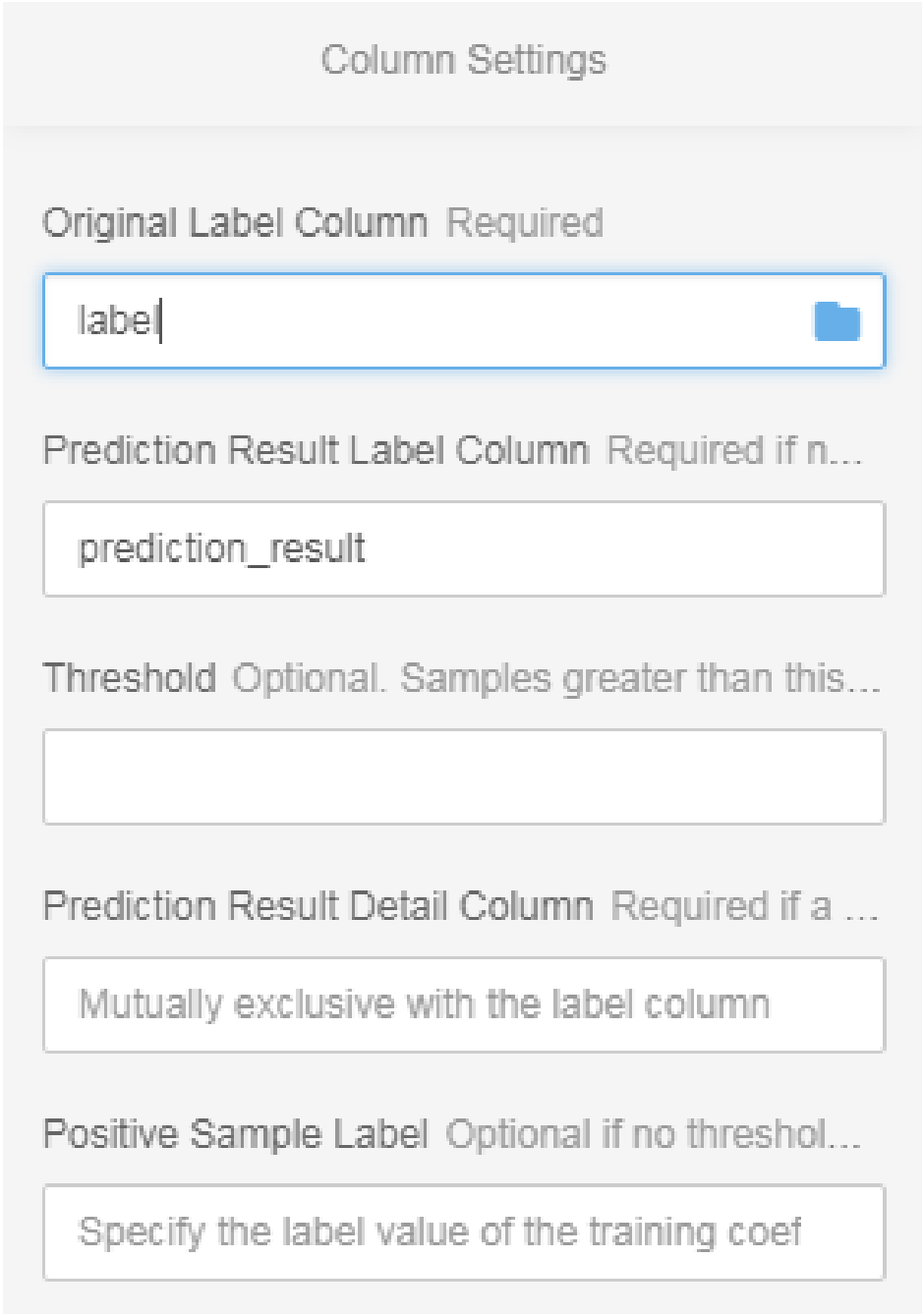
Test data

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

Create an experiment.

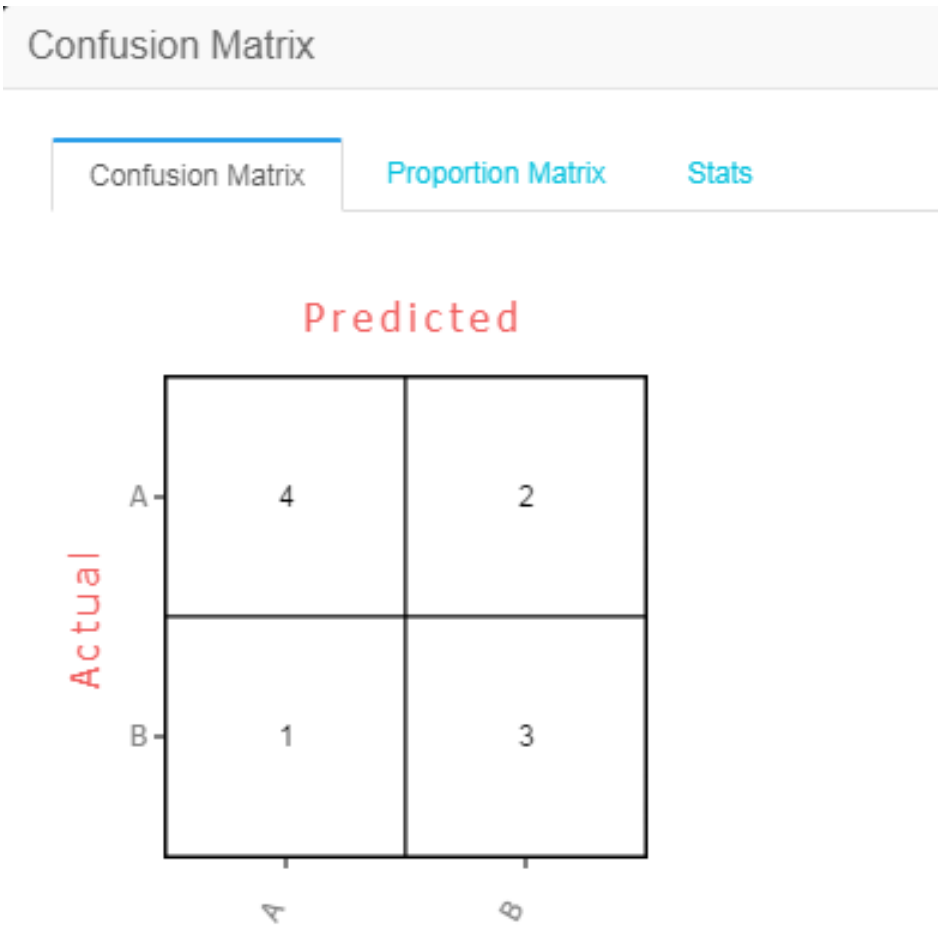


Configuration parameters.

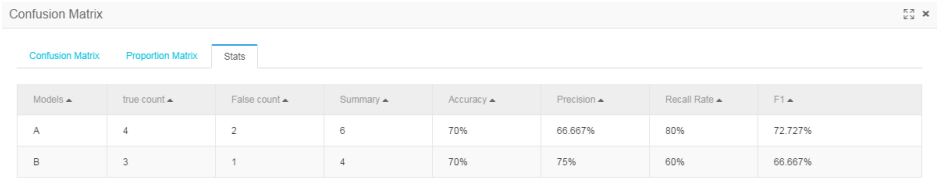


Run the experiment.

The confusion matrix is as follows:



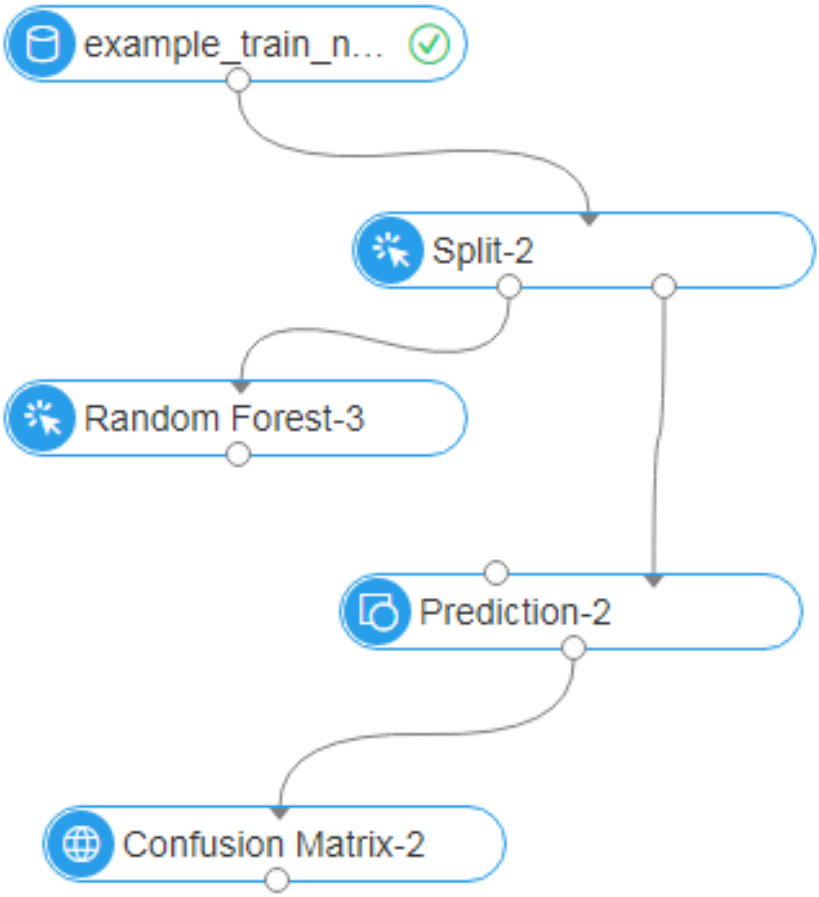
The following figure shows statistics about each label.



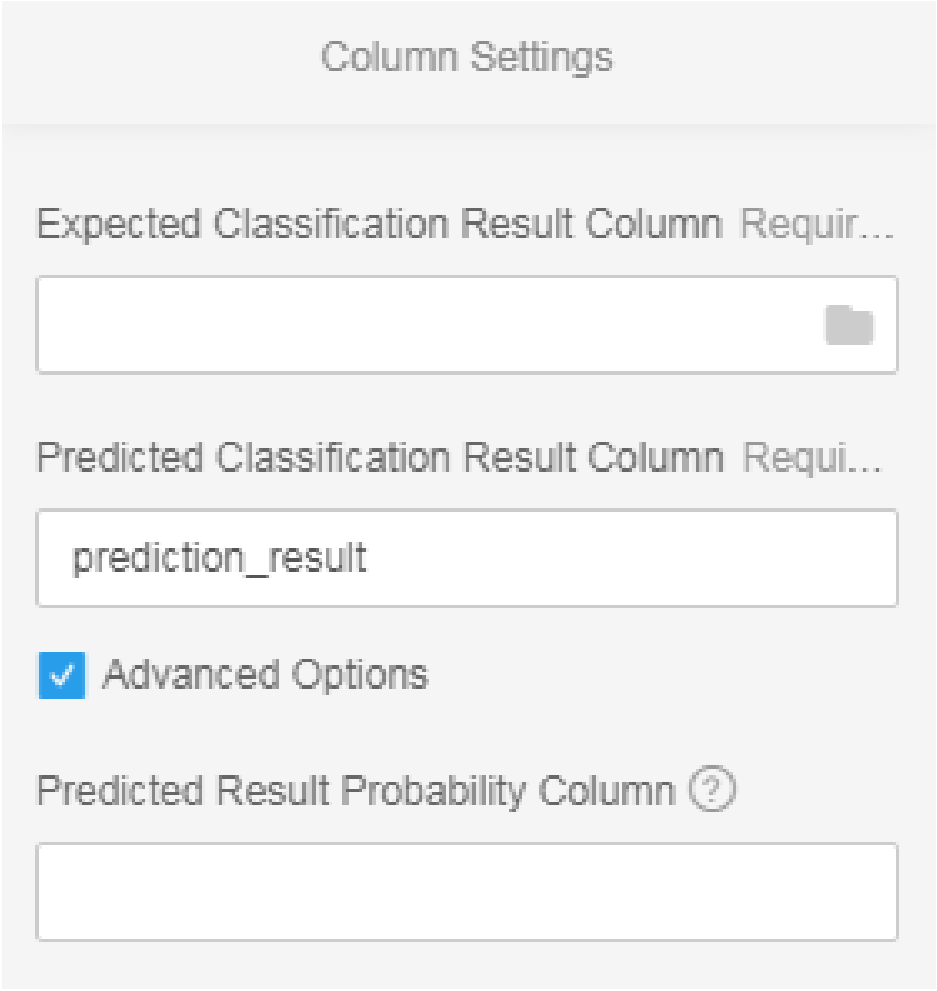
Multiclass classification evaluation

This component evaluates a multiclass classification algorithm model based on the accuracy, kappa, F1 score, and other metrics in the predicted result and expected result of the classification model.

Component connection: The multiclass classification evaluation component needs to be connected to the prediction component and does not support regression models. The following figure shows an example of connection between components.



Parameter settings



- In the Expected Classification Result Column drop-down list, you can select the Original Label column.

NOTE: A maximum of 1000 classes are supported.

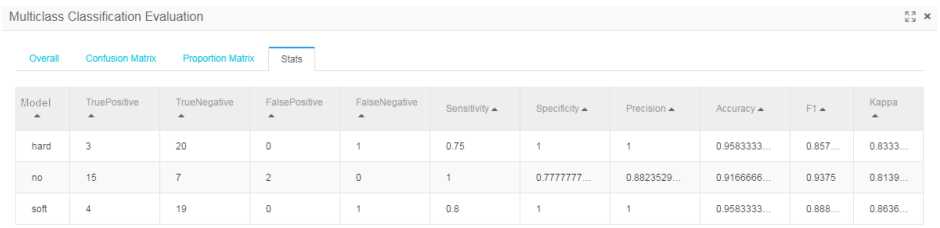
In the Predicted Classification Result Column drop-down list, you can select the Predicted Label column. The default value is prediction_result.

In the Predicted Result Probability Column drop-down list, you can select the Predicted Label Probability column. Generally, the name of this field is prediction_detail.

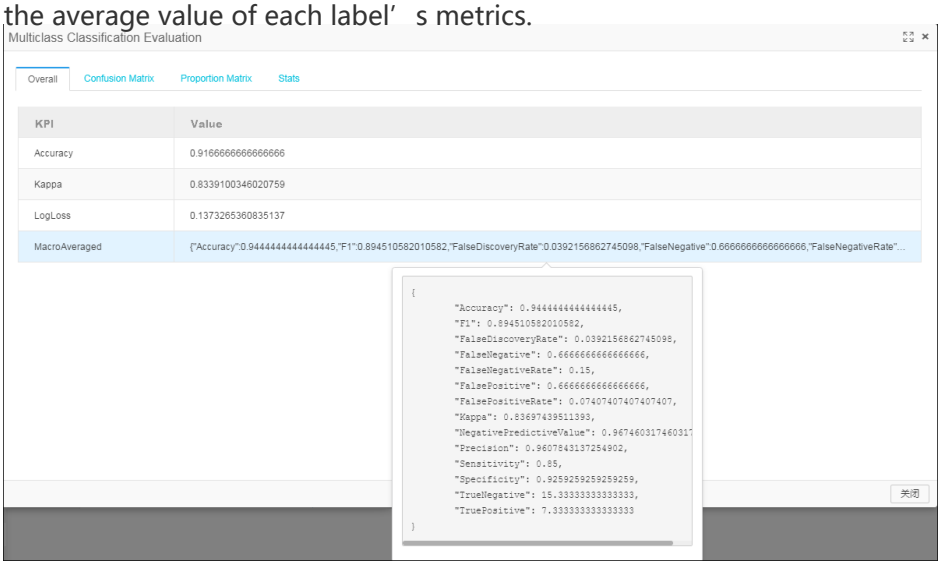
NOTE: This parameter applies only to random forest prediction.

Evaluation report description

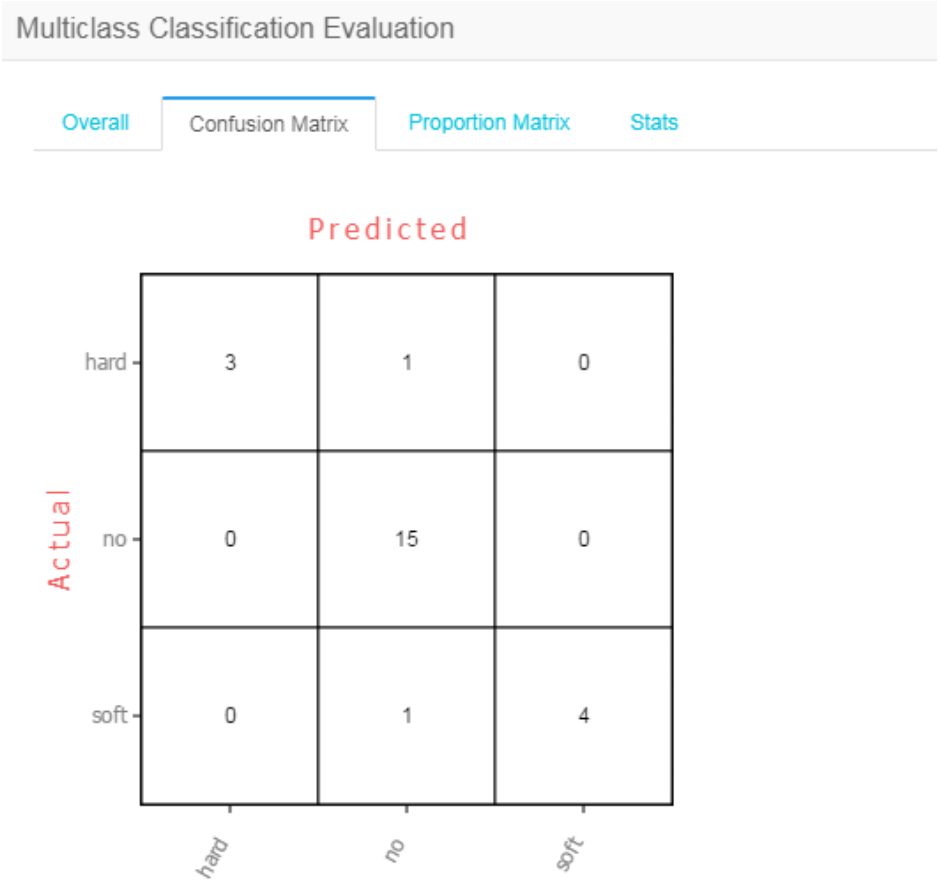
Metrics of each label are calculated using the one-vs-all method, as shown in the following figure.



The following figure shows the summary of metrics, among which MacroAveraged indicates the average value of each label’ s metrics.



Confusion matrix.



PAI command

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

Parameter settings

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	NA	NA
inputTablePartitions	(Optional) Partitions used for calculation in the input table	NA	All partitions in the input table
outputTableName	(Required) Name of the output table	NA	NA
labelColName	(Required) Name of the original label column in the input	NA	NA

	table		
predictionColName	(Required) Name of the label column of the predicted result table	NA	NA
predictionDetailColName	(Optional) Name of the predicted result probability column, in the format of { "A ":0.2, "B ":0.3, "C ", 0.5}	NA	Empty
lifecycle	(Optional) Lifecycle of the output table	Positive integer	No lifecycle
coreNum	(Optional) Number of cores for calculation	Positive integer	Automatically assigned
memSizePerCore	(Optional) Memory size for each core, in MB	Positive integer in the range of (0, 65536)	Automatically assigned

Description of the output table in JSON format

```
{
"LabelNumber": 3,
"LabelList": ["A", "B", "C"],
"ConfusionMatrix": [ // Confusion matrix [actual][predict]
[100, 10, 20],
[30, 50, 9],
[7, 40, 90] ],
"ProportionMatrix": [ // Proportion in each row [actual][predict]
[0.6, 0.2, 0.2],
[0.3, 0.6, 0.1],
[0.1, 0.4, 0.5] ],
"ActualLabelFrequencyList": [ // Actual frequency of each label
200, 300, 600],
"ActualLabelProportionList": [ // Actual proportion of each label
0.1, 0.2, 0.7],
"PredictedLabelFrequencyList": [ // Predicted frequency of each label
300, 400, 400],
"PredictedLabelProportionList": [ // Predicted proportion of each label
0.2, 0.1, 0.7],
"OverallMeasures": { // Overall metrics
"Accuracy": 0.70,
"Kappa" : 0.3,
"MacroList": { // Average value of each label's metrics
"Sensitivity": 0.4,
"Specificity": 0.3,
},
"MicroList": { // Metrics calculated based on the sum of TP, TN, FP, and FN values of each label
"Sensitivity": 0.4,
"Specificity": 0.3,
```

```
},
"LabelFrequencyBasedMicro": { // Frequency-based weighted average value of each label's metrics
"Sensitivity": 0.4,
"Specificity": 0.3,
},
},
"LabelMeasuresList": [ // Metrics of each label
{
"Accuracy": 0.6,
"Sensitivity": 0.4,
"Specificity": 0.3,
"Kappa": 0.3
},
{
"Accuracy": 0.6,
"Sensitivity": 0.4,
"Specificity": 0.3,
"Kappa": 0.3
},
]
}
```

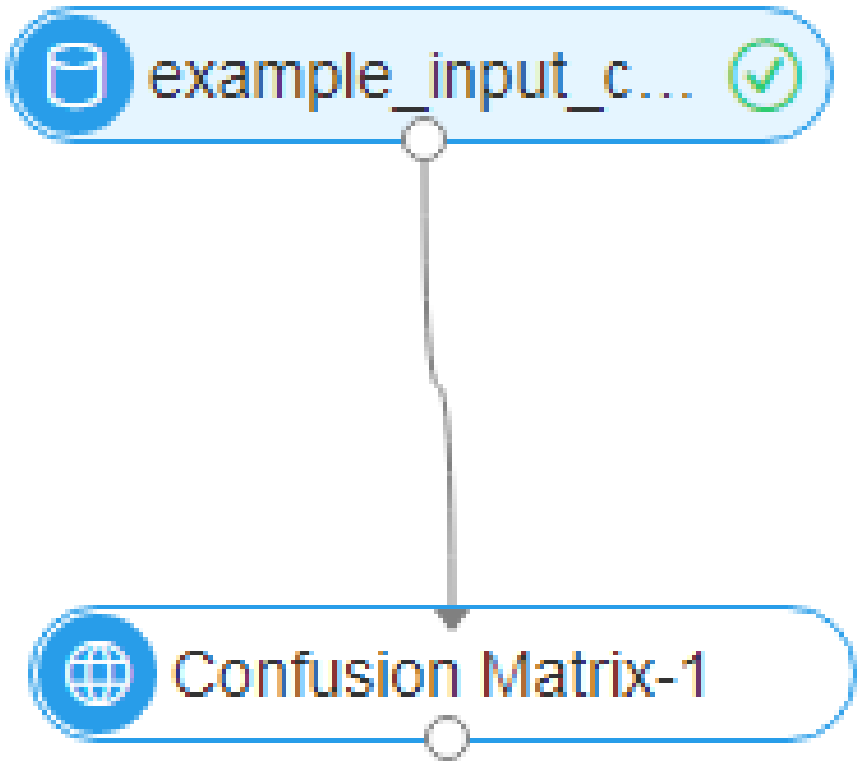
Example

Test data

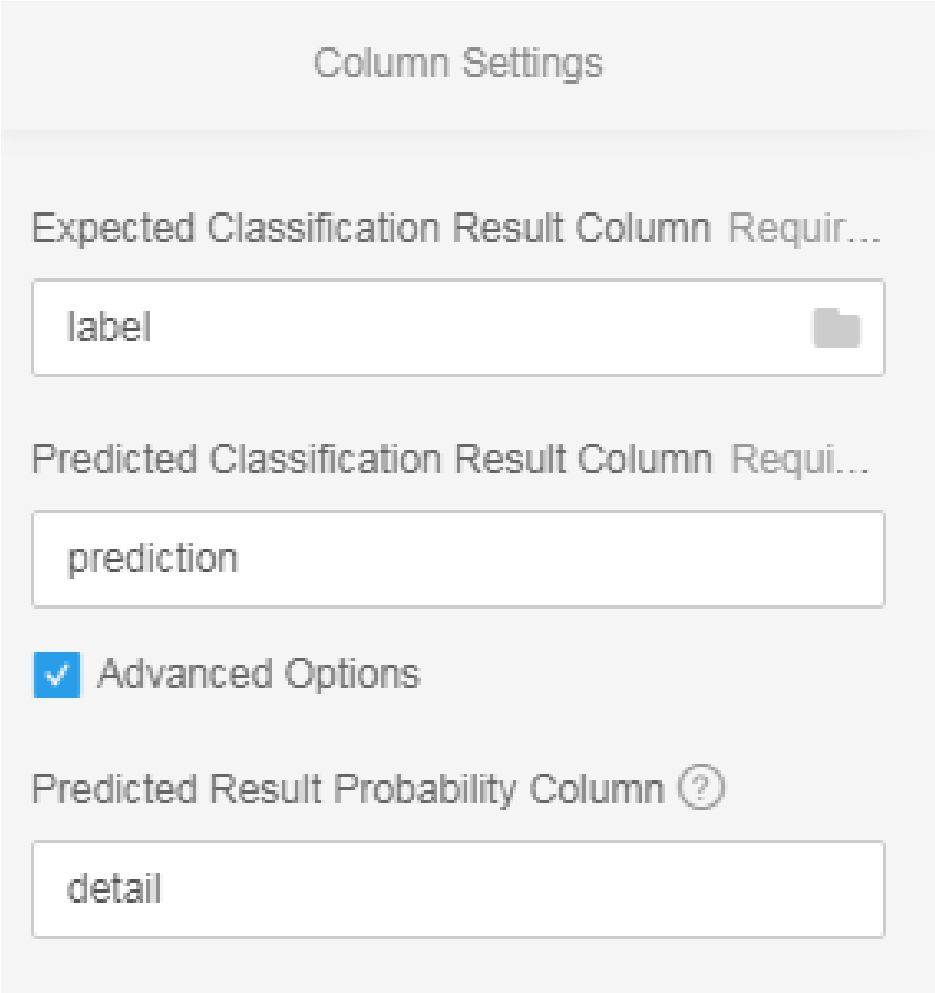
data table example_input_mc_eval

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}

Create an experiment.

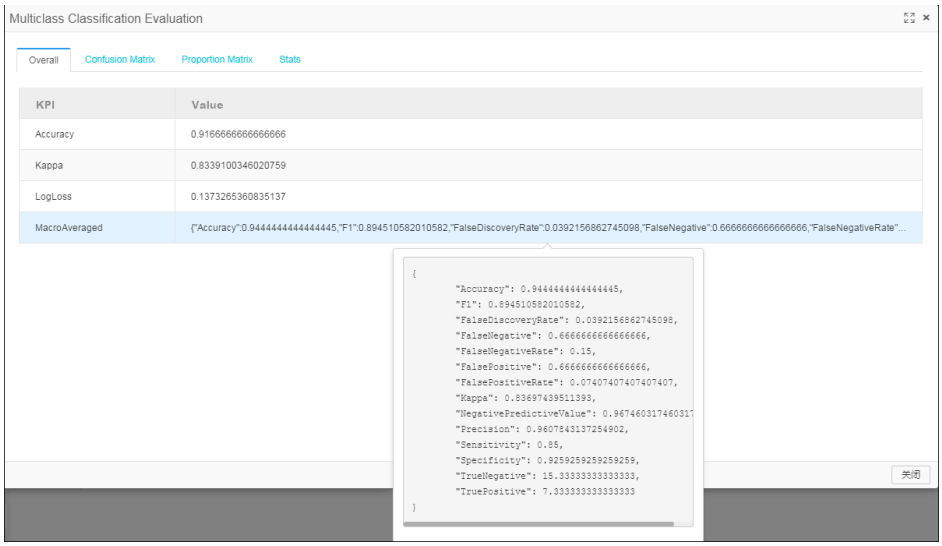


Configuration parameters.



Run the experiment.

The overall report is as follows:



Statistics about each label are as follows:

The output table in JSON format is as follows:


```
{
"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]]}
```

Binary classification evaluation

The evaluation module can calculate the AUC, KS, and F1 score, and provide output data for drawing the KS curve, PR curve, ROC curve, LIFT chart, and gain chart. The module also supports grouping evaluation.

PAI command

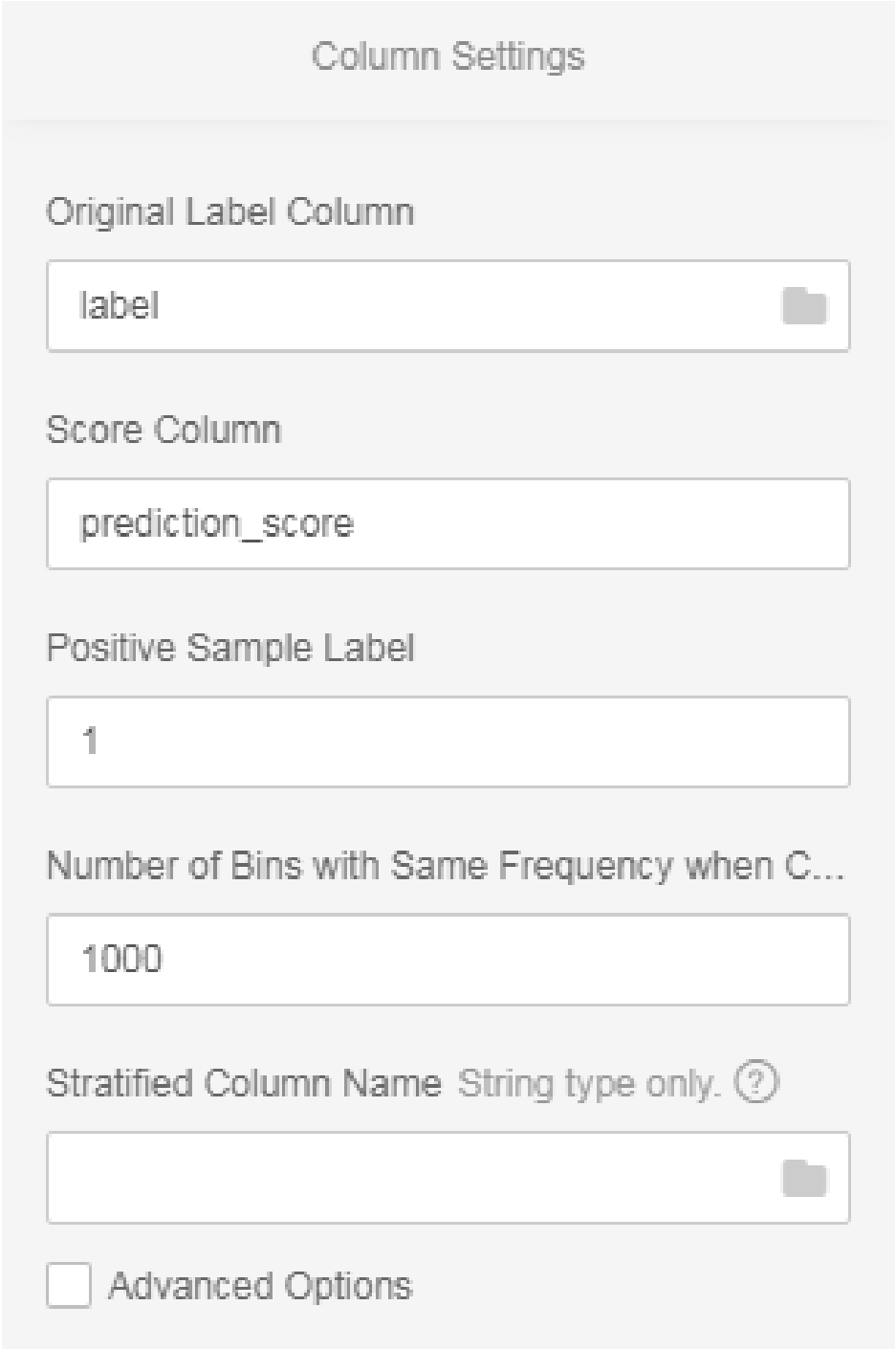
```
pai -name=evaluate -project=algo_public
-DoutputMetricTableName=output_metric_table
-DoutputDetailTableName=output_detail_table
-DinputTableName=input_data_table
-DlabelColName=label
-DscoreColName=score
```

Parameter description

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	NA	NA
inputTablePartitions	(Optional) Partitions in the input table	-	All partitions in the input table
labelColName	(Required) Name of the label column	NA	NA
scoreColName	(Required) Name of the score column	NA	NA
groupColName	(Optional) Name of the group column used in grouping evaluation.	NA	NA
binCount	(Optional) Number of equal-frequency bins used to calculate metrics such as KS and PR	NA	1000
outputMetricTableName	(Required) Name of the output metric table, consisting of two columns (Metric and Value) and three rows (AUC, KS, and F1 Score)	NA	NA
outputDetailTableNa	(Optional) Name of	NA	NA

me	the output detail table used for chart drawing		
positiveLabel	(Optional) Class of the positive sample	NA	1
lifecycle	(Optional) Life cycle of the output table	NA	Unspecified by default
coreNum	(Optional) Number of cores	NA	Automatically calculated by default
memSizePerCore	(Optional) Size of memory	NA	Automatically calculated by default

Parameter settings

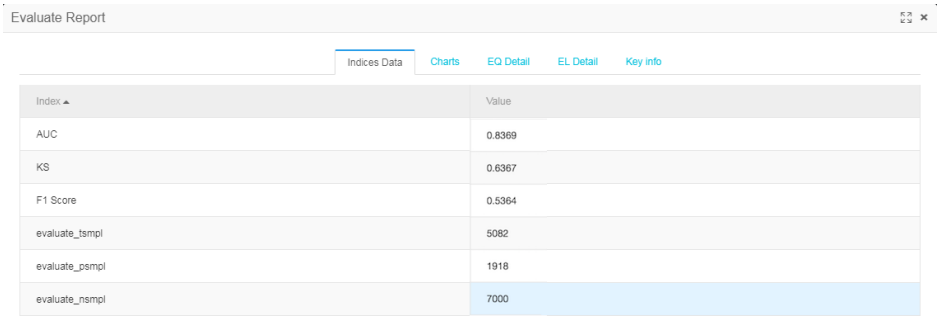


- In the Original Label Column drop-down list, you can select the Original Label column.
- In the Score Column box, you can select the Prediction Score column. The default option is prediction_score.
- In the Positive Sample Label box, you can select the label value corresponding to the positive sample.
- In the Number of Bins with Same Frequency when Calculating Indexes such as KS and PR box, you can determine how many equal-frequency bins to divide data into. The default value is 1000.
- In the Grouping Columns box, you can select the grouping columns used to divide data

during grouping evaluation.

Evaluation result display

Right-click the Binary Classification Evaluation node and select View Evaluation Report from the shortcut menu to view the evaluation result, as shown in the following figure.



Regression model evaluation

This component evaluates a regression algorithm model based on metrics and residual histograms in the predicted result and expected result. The metrics include SST, SSE, SSR, R2, R, MSE, RMSE, MAE, MAD, MAPE, count, yMean, and predictMean.

PAI command

```
PAI -name regression_evaluation -project algo_public
-DinputTableName=input_table
-DyColName=y_col
-DpredictionColName=prediction_col
-DindexOutputTableName=index_output_table
-DresidualOutputTableName=residual_output_table;
```

Parameter	Description	Option	Default value
inputTableName	(Required) Name of	NA	NA

	the input table		
inputTablePartitions	(Optional) Partitions used for calculation in the input table	NA	All partitions in the input table
yColName	(Required) Name of the original dependent variable column in the input table, numerical value	NA	NA
predictionColName	(Required) Name of the predicted dependent variable column, numerical value	NA	NA
indexOutputTableName	(Required) Name of the regression metric output table	NA	NA
residualOutputTableName	(Required) Name of the residual histogram output table	NA	NA
intervalNum	(Optional) Number of intervals in a histogram	NA	100
lifecycle	(Optional) Lifecycle of the output table	Positive integer	No lifecycle
coreNum	(Optional) Number of instances	NA	Not set
memSizePerCore	(Optional) Memory size for each core	NA	Not set

Output result

Regression metric output table

The output table is in JSON format. JSON fields are described as follows:

Field	Description
SST	Total sum of squares
SSE	Sum of squared errors
SSR	Sum of squares due to regression
R2	Coefficient of determination
R	Coefficient of multiple correlation
MSE	Mean squared error

RMSE	Root-mean-square error
MAE	Mean absolute error
MAD	Mean error
MAPE	Mean absolute percentage error
count	Number of rows
yMean	Mean of original dependent variables
predictionMean	Mean of predicted results

Prediction

The prediction component is used to make model-based predictions. The component has two inputs (training model and prediction data) and one output (predicted result).
Traditional data mining algorithms all use this component to make predictions.

PAI command

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

Parameter description

Parameter	Description	Value range	Required/Optional, default value/act
inputTableName	Name of the input table	Table name	Required
modelName	Name of a model	Model name	Required
outputTableName	Name of the output table	Table name	Required
featureColNames	Names of the feature columns used for prediction in the input table	Column names	Optional, default: all columns
appendColNames	Names of the columns in the prediction input table to be appended to the output table		Optional, default: no appended column
inputTablePartitions	Partitions used for prediction in the	NA	Optional, default: all partitions in the

	input table, in the format of partition_name=value. The multilevel format is name1=value1/name2=value2; multiple partitions are separated by commas		input table
outputTablePartition	Partition in the output table	NA	Optional, default: no partition
resultColName	Name of the result column in the output table	NA	Optional, default: prediction_result
scoreColName	Name of the score column in the output table	NA	Optional, default: prediction_score
detailColName	Name of the detail column in the output table	NA	Optional, default: prediction_detail
enableSparse	Whether data in the input table is in sparse format	true, false	Optional, default: false
itemDelimiter	Delimiter used between key-value pairs when data in the input table is in sparse format	NA	Optional, default: space
kvDelimiter	Delimiter used between keys and values when data in the input table is in sparse format	NA	Optional, default: colon
lifecycle	Lifecycle of the output table	Positive integer	Optional, default: no lifecycle

Prediction formula

Naive Bayes

Prediction_result formula:

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

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

Prediction_score formula:

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

Classification variables:

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

Continuous variables:

K-means

$$k^{'} = arg \min_{k} d(x_i - c_k)$$

Prediction_result formula:

Prediction_score formulas:

$$d(x-c) = (x-c)(x-c)^{'}$$

euclidean:

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

cosine:

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

cityblock:

Output description

Classification	Model	Prediction_resu	Prediction_scor	Prediction_deta

		lt	e	il
Binary classification	Logistic regression model	Predicted label	Predicted label probability	Each label and its probability
	Linear SVM model	Predicted label	Predicted label probability	Each label and its probability
	Random forest model	Predicted label	Predicted label probability	Each label and its probability
	GBDT_LR model	Predicted label	Predicted label probability	Each label and its probability
	Naive Bayes model	Predicted label	Log (predicted label probability)	Each label and its log (probability)
	XGboost model	Predicted label	Predicted label probability	Each label and its probability
Multiclass classification	Logistic regression model	Predicted label	Predicted label probability	Each label and its probability
	Random forest model	Predicted label	Predicted label probability	Label of each leaf node and its probability
	Naive Bayes model	Predicted label	Log (predicted label probability)	Each label and its log (probability)
Regression	Linear regression model	Empty	Regression value	Label column name: regression value
	GBDT model	Empty	Regression value	Label column name: regression value
	XGboost model	Empty	Regression value	Label column name: regression value
Cluster	K-means model	Predicted center sequence number	Distance to the predicted center	Distance to each center

Example

Test data

```
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;
```

Modeling

```
 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";
```

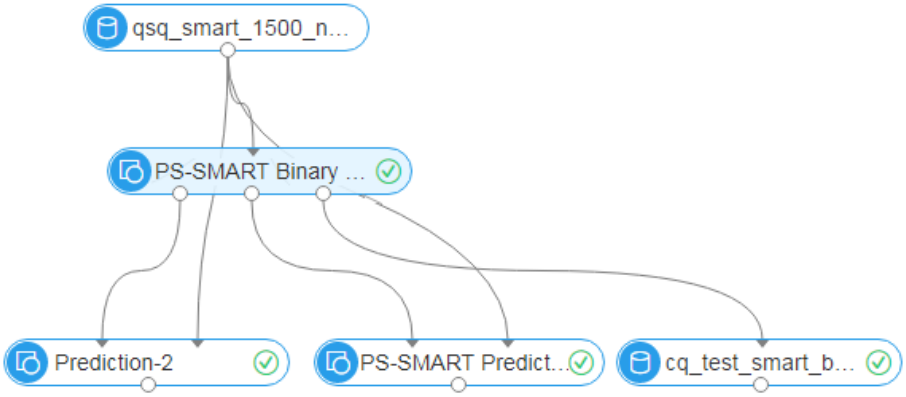
PAI prediction

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

PS-SMART binary classification

PS stands for parameter server, which is used for online and offline training tasks of large-scale models. Scalable Multiple Additive Regression Tree (SMART) is an implementation of Gradient boosting decision tree (GBDT) on PS. PS-SMART can run training tasks containing up to tens of billions of samples and hundreds of thousands of features on thousands of nodes. It also supports failover to maintain a high stability. Additionally, PS-SMART supports various data formats, training targets, evaluation targets, output feature importance, and training acceleration (such as histogram similarity).

Quick Start



As shown in the figure, a PS-SMART binary classification model is learned based on training data. The model has three output studs:

Output model: offline model, which is connected to the unified prediction component. This model does not support output of leaf node numbers.

Output model table: a binary table that is not readable. To ensure compatibility with the PS-SMART prediction component, the table provides outputs such as leaf node numbers and evaluation metrics. However, the output table has strict requirements on data formats, resulting in poor user experiences. It will be improved gradually or be replaced by another component.

Output feature importance table: lists importance of each feature. Three importance types are supported (see parameter description).

PAI command

Training

```
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"
```

Prediction

```
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"
```

Parameter description

Data parameters

Command option	Parameter	Description	Value range	Required/Optional, default value
featureColNames	Feature Column	Name of the feature column selected from the input table for training	The column name must be bigint or double type in dense format or string type in sparse KV format. If the sparse KV format is used, the keys and values must be numerical type data.	Required
labelColName	Label Column	Name of the label column in the input table	The column name can be either a string or a numerical value, but it can only be saved as a numerical value. For example, the value can be 0 or 1 for binary classification.	Required

weightCol	Weight Column	You can specify a weight for each row of samples.	Column name, numerical type	(Optional) It is left blank by default.
enableSparse	Sparse Format	Whether data is in sparse format, in which key-value pairs are separated by spaces whereas keys and values are separated by colons, for example, 1:0.3 3:0.9	[true, false]	(Optional) The default value is false.
inputTableName	Name of the input table	NA	NA	Required
modelName	Name of the output model	NA	NA	Required
outputImportanceTableName	Output Feature Importance Table Name	NA	NA	(Optional) It is left blank by default.
inputTablePartitions	Input Table Partitions	NA	NA	Optional, in the format of ds=1/pt=1
outputTableName	Output Model Table Name	The output table is an ODPS table that uses the binary format and is not readable. The self-contained prediction component of SMART can be used to provide output of leaf node numbers.	true, false	(Optional) The default value is false.
lifecycle	Output Table Lifecycle	NA	Positive integer	(Optional) The default value is 3.

Algorithm parameters

Command option	Parameter	Description	Value range	Required/Optional, default value
objective	Objective Function Type	The objective function type		Required

		affects learning directly and must be selected correctly. Select “binary:logistic” for binary classification.		
metric	Evaluation Metric Type	Evaluation metrics in the training set, which are output to stdout of the coordinator in a logview	logloss, error, auc	(Optional) It is left blank by default.
treeCount	Number of Decision Trees	Number of decision trees. The training time is in direct proportion to this number.	Positive integer	(Optional) The default value is 1.
maxDepth	Maximum Decision Tree Depth	Maximum depth of a decision tree, recommended value: 5 (a maximum of 32 leaf nodes)	Positive integer, [1, 20]	(Optional) The default value is 5.
sampleRatio	Data Sampling Ratio	Data sampling ratio for creating a weak learner to accelerate training when building each decision tree	(0, 1]	(Optional) The default value is 1.0, indicating that data sampling is disabled.
featureRatio	Feature Sampling Ratio	Feature sampling ratio for creating a weak learner to accelerate training when building each decision tree	(0, 1]	(Optional) The default value is 1.0, indicating that feature sampling is disabled.
l1	L1 Penalty Coefficient	This parameter controls the number of leaf nodes. The larger the value is, the fewer the leaf nodes are. Set a larger value for	Non-negative real number	(Optional) The default value is 0.

		overfitting.		
l2	L2 Penalty Coefficient	This parameter controls distribution of leaf nodes. The larger the value is, the more evenly the leaf nodes are distributed.Set a larger value for overfitting.	Non-negative real number	(Optional) The default value is 1.0.
shrinkage	Learning Rate		(0, 1]	(Optional) The default value is 0.3.
sketchEps	Sketch Precision	Threshold of the splitting point when creating a sketch. The number of bins is O (1.0/sketchEps). The smaller the value is, the more bins are divided. This value does not need to be adjusted under normal conditions.	(0, 1)	(Optional) The default value is 0.03.
minSplitLoss	Minimum Split Loss	Minimum split loss required for splitting a node. The larger the value is, the more conservatively the node splits.	Non-negative real number	(Optional) The default value is 0.
featureNum	Feature Quantity	Number of features or the largest feature ID. Specify this parameter for resource usage estimation.	Positive integer	Optional
baseScore	Global Offset	Original predicted values of all samples	Real number	(Optional) The default value is 0.5.
featureImportanceType	Feature Importance	Type of feature importance.	“weight”, “gain”,	(Optional) The default value is

	Type	“weight” indicates the number of times features are split. “gain” indicates information gain provided by the feature. “cover” indicates the number of samples that feature covers on the splitting node.	“cover”	“gain”.

NOTE:

- Specify different values for the objective parameter in different learning models. On the binary classification web GUI, the objective function is automatically specified and hidden from users. On the command line interface, set the objective parameter to “binary:logistic”.
- Mappings between metrics and objective functions are: “logloss” for “negative loglikelihood for logistic regression”, “error” for “binary classification error”, and “auc” for “Area under curve for classification”.

Execution optimization

Command option	Parameter	Description	Value range	Required/Optional, default value
coreNum	Number of Cores	Number of cores used for computing. The larger the value is, the faster the computing algorithm runs.	Positive integer	(Optional) Cores are automatically assigned by default.
memSizePerCore	Memory Size per Core (MB)	Memory sized used by each core, where 1024 represents 1 GB memory	Positive integer	(Optional) Memory is automatically assigned by default.

Example

Data generation

The following example generates data in dense format:

```
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;
```

The following figure shows the generated data.

Index	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

The table contains six feature columns.

Training

Configure training data and the training component according to Quick start. Select the label column as the target column and columns f0, f1, f2, f3, f4, f5 as feature columns. The following figure shows the algorithm parameter settings page.

Column Settings | Parameter Settings | Execution Optimizatior

Evaluation Index Type Optional

Negative Loglikelihood for Logistic Regression

Tree Quantity Optional, positive integer.

5

Max Tree Depth Optional. Positive integer.

5

Data Sampling Ratio Optional (0,1]

1.0

Feature Sampling Ratio Optional (0,1]

1.0

L1 Penalty Coefficient Optional. Non-negative real n...

0

L2 Penalty Coefficient Optional. Non-negative real n...

1.0

Learning Rate Optional (0,1]

0.3

Approximate Sketch Precision Optional (0,1]

0.03

Minimum Split Loss Optional. Non-negative r...

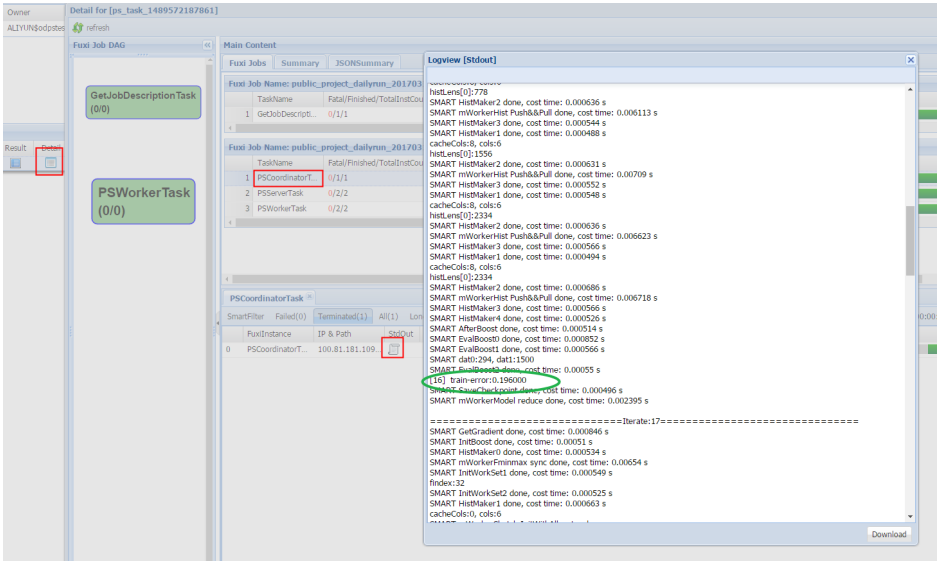
You do not need to set the number of features because this number is calculated automatically by the algorithm. If you have a large number of features and want the algorithm to accurately estimate the amount of resources required, enter the actual number of features here.

To accelerate the training, you can set the number of cores on the execution optimization page. The larger the number is, the faster the algorithm runs. Generally, you do not need to enter the memory size per core because the algorithm can accurately estimate the memory size. In addition, the PS algorithm starts to run only when all hosts obtain resources. Therefore, you may need to wait for a longer time when the cluster is busy and requires many resources.

You can view metrics output values in stdout of the coordinator in a logview (http link starting with " http://logview.odps.aliyun-inc.com:8080/logview "). One PS-SMART training job has multiple tasks; therefore, multiple logviews are available. Select the logview with logs starting with PS, as shown in the following figure.

The one in the red box is the logview of the PS job. You can identify different tasks by information in the green circle.

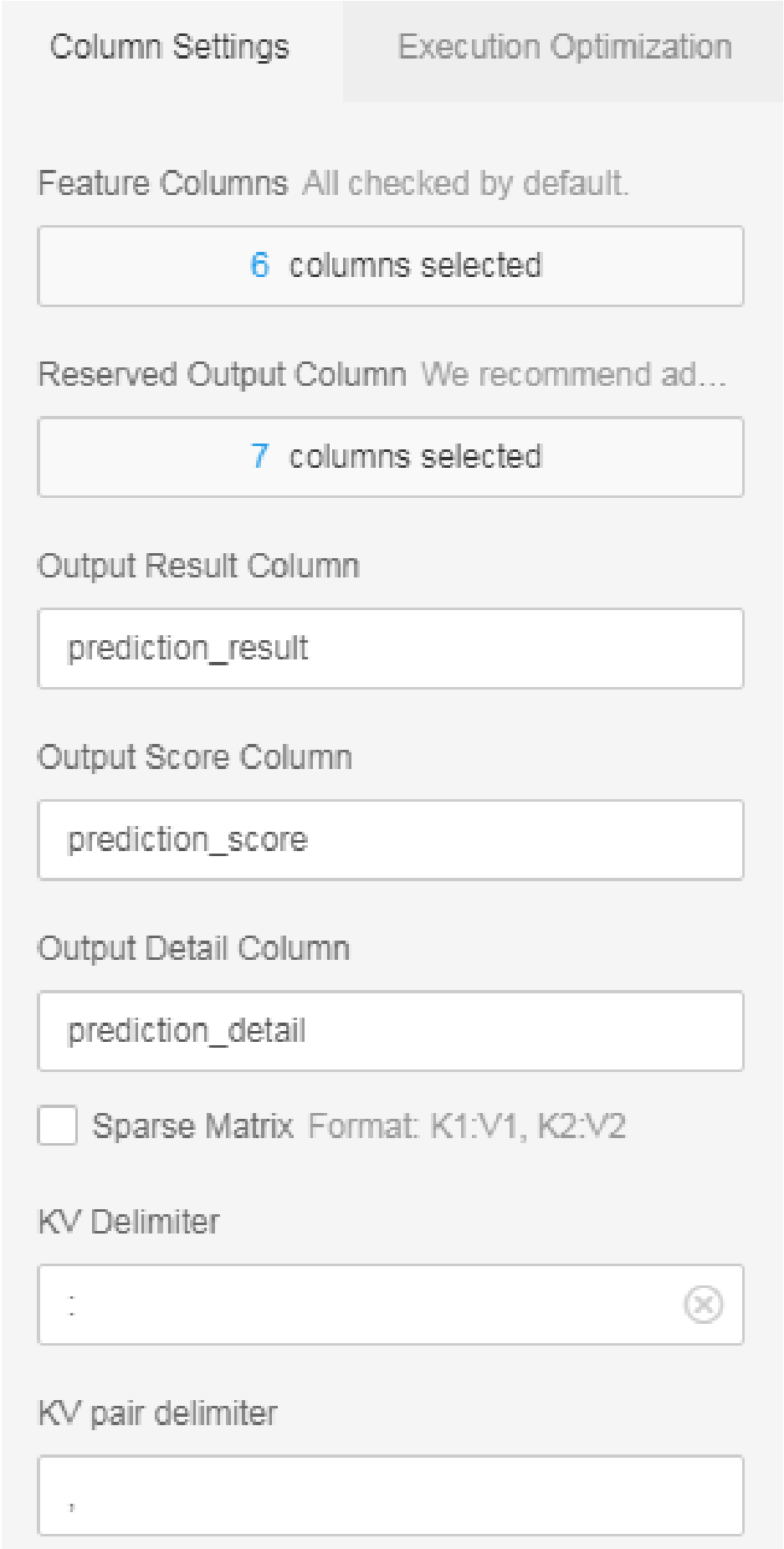
Then, perform operations in the logview according to the following figure.



Prediction

Use the unified prediction component

The output model obtained after training is saved in binary format and can be used for prediction. Configure the input model and test data for the prediction component according to Quick start and set parameters according to the following figure.


Column Settings
Execution Optimization
Feature Columns All checked by default.
6 columns selected
Reserved Output Column We recommend ad...
7 columns selected
Output Result Column
prediction_result
Output Score Column
prediction_score
Output Detail Column
prediction_detail
Sparse Matrix Format: K1:V1, K2:V2
KV Delimiter
:
KV pair delimiter
,

If the dense format is used, you only need to select feature columns. (All columns are selected by default, and extra columns do not affect the prediction.) If the KV format is used, set the data format to sparse format and select a correct delimiter. In the SMART model, key-value pairs are separated by space characters. Therefore, the delimiter must be set to space or "\u0020" (escape expression of space).

The following figure shows the prediction result.

Index	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}

In the "prediction_detail" column, the value 1 indicates a positive sample, and the value 0 indicates a negative sample. The values following 0 and 1 indicate probabilities of the corresponding classes.

Use the PS-SMART prediction component

The output model table obtained after training is saved in binary format and can be used by the PS-SMART prediction component for prediction. Configure the input model and test data for the prediction component according to Quick start and set parameters, including the data format, feature columns, target column, and number of classes. The ID column can only be a string type column other than feature columns and the target column. The loss function must be explicitly set to "binary:logistic". The following figure shows the prediction result.

Index	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

The "prediction_score" column lists probabilities of predicted positive samples. A sample is predicted as a positive sample if its score is greater than 0.5. Otherwise, it is predicted as a negative sample. The "leaf_index" column lists the predicted leaf node numbers. Each sample has N numbers, where N is the number of decision trees. Each tree maps to a number, which indicates the

number of the leaf node on this tree.

NOTE:

- The output model table is a binary table that is not readable. To ensure compatibility with the PS-SMART prediction component, the table provides outputs such as leaf node numbers and evaluation metrics.However, the output table has strict requirements on data formats, resulting in poor user experiences. It will be improved gradually or be replaced by another component.
- A string type column must be selected as the label column. You can enter character strings in the column but cannot leave it blank or enter Null in it. A feature column can be converted into the string type by using the data type conversion component and used as the label column.
- The loss function must be explicitly set to “binary:logistic”. By default, the loss function does not work.

View feature importance

To view feature importance, you can export the third output stud to an output table, or directly right-click in the PS-SMART component and choose View Data > Output Feature Importance Table. The following figure shows the output feature importance table.

Index	id	value
1	0	0.56903380155563335
2	1	0.21714292466640472
3	4	0.21382322907447815

In the table, the id column lists numbers of input features. In this example, the data is in dense format and input features are “f0, f1, f2, f3, f4, f5”. Therefore, ID 0 represents f0, and ID 4 represents f4. If the KV format is used, the IDs represent keys in key-value pairs. Each value indicates a feature importance type. The default value is “gain”, indicating the sum of information gains brought by a feature in the model. The preceding figure shows only three features because only these three features are used in split of trees. The importance of unused features is considered 0.

Important notes

The target column in a PS-SMART binary classification model supports only numerical values (0 for negative samples and 1 for positive samples). Even if values in the ODPS table are strings, they are saved as numerical values. If the classification target is a type string similar to “Good” or “Bad”, convert it into 1 or 0.

In the key-value format, feature IDs must be positive integers, and feature values must be real numbers. If feature IDs are character strings, use the serialization component to serialize them. If feature values are type character strings, perform engineering on the features, such as discretization.

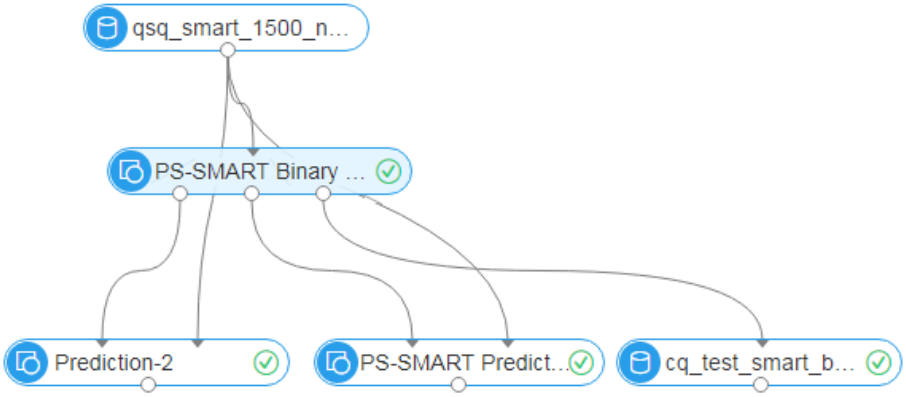
Although PS-SMART supports tasks with hundreds of thousands of features, such tasks consume many resources and run slowly. Therefore, we do not recommend such a large number of features. The GBDT algorithm is suitable for training with continuous features. Continuous type features require one-hot coding (to filter out infrequent features) before they can be used for training. Continuous numerical features can be used for training with the GBDT algorithm directly. Discretization is not recommended for numerical features.

The PS-SMART algorithm applies randomness in many scenarios. For example, the data_sample_ratio and fea_sample_ratio items require data or feature sampling. In addition, the PS-SMART algorithm uses histograms to show similarity. When multiple workers run in a cluster in distributed mode, local sketches are merged to global sketches in a random order. Although different merging orders result in different tree structures, this does not bring great variation in the output model theoretically. Therefore, it is a normal situation if different results are obtained after the algorithm runs multiple times with the same data and same parameter settings.

PS-SMART multiclass classification

PS stands for parameter server, which is used for online and offline training tasks of large-scale models. Scalable Multiple Additive Regression Tree (SMART) is an implementation of Gradient boosting decision tree (GBDT) on PS. PS-SMART can run training tasks containing up to tens of billions of samples and hundreds of thousands of features on thousands of nodes. It also supports failover to maintain a high stability. Additionally, PS-SMART supports various data formats, training targets, evaluation targets, output feature importance, and training acceleration (such as histogram similarity).

Quick Start



As shown in the figure, a PS-SMART multiclass classification model is learned based on training data.The model has three output studs:

Output model: offline model, which is connected to the unified prediction component. This model does not support output of leaf node numbers.

Output model table: a binary table that is not readable. To ensure compatibility with the PS-SMART prediction component, the table provides outputs such as leaf node numbers and evaluation metrics.However, the output table has strict requirements on data formats, resulting in poor user experiences. It will be improved gradually or be replaced by another component.

Output feature importance table: lists importance of each feature. Three importance types are supported (see parameter description).

PAI command

Training

```
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"
```

Prediction

```
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"
```

Parameter description

Data parameters

Command option	Parameter	Description	Value range	Required/Optional, default value
featureColNames	Feature Column	Name of the feature column selected from the input table for training	The column name must be bigint or double type in dense format or string type in sparse KV format. If the sparse KV format is used, the keys and values must be numerical type data.	Required
labelColName	Label Column	Name of the label column in the input table	The column name can be either a string or a numerical value, but it can only be saved as a numerical value. For multiclass classification, select the column name	Required

			among 0, 1, 2, ..., n-1, where n is the number of classes	
weightCol	Weight Column	You can specify a weight for each row of samples.	Column name, numerical type	(Optional) It is left blank by default.
enableSparse	Sparse Format	Whether data is in sparse format, in which key-value pairs are separated by spaces whereas keys and values are separated by colons, for example, 1:0.3 3:0.9	[true, false]	(Optional) The default value is false.
inputTableName	Name of the input table	NA	NA	Required
modelName	Name of the output model	NA	NA	Required
outputImportanceTableName	Output Feature Importance Table Name	NA	NA	(Optional) It is left blank by default.
inputTablePartitions	Input Table Partitions	NA	NA	Optional, in the format of ds=1/pt=1
outputTableName	Output Model Table Name	The output table is an ODPS table that uses the binary format and is not readable. The self-contained prediction component of SMART can be used to provide output of leaf node numbers.	true, false	(Optional) The default value is false.
lifecycle	Output Table Lifecycle	NA	Positive integer	(Optional) The default value is 3.

Algorithm parameters

Command	Parameter	Description	Value range	Required/Optio

option				nal, default value
classNum	Number of Classes	Number of classes in multiclass classification. If the number of classes is n, the label column name can be 0, 1, 2, ..., or n-1	Integer greater than or equal to 3	Required
objective	Objective Function Type	The objective function type affects learning directly and must be selected correctly. Set it to "multi:softprob" for multiclass classification.		Required
metric	Evaluation Metric Type	Evaluation metrics in the training set, which are output to stdout of the coordinator in a logview	"mlogloss","merror"	(Optional) It is left blank by default.
treeCount	Number of Decision Trees	Number of decision trees. The training time is in direct proportion to this number.	Positive integer	(Optional) The default value is 1.
maxDepth	Maximum Decision Tree Depth	Maximum depth of a decision tree, recommended value: 5 (a maximum of 32 leaf nodes)	Positive integer, [1, 20]	(Optional) The default value is 5.
sampleRatio	Data Sampling Ratio	Data sampling ratio for creating a weak learner to accelerate training when building each decision tree	(0, 1]	(Optional) The default value is 1.0, indicating that data sampling is disabled.
featureRatio	Feature	Feature	(0, 1]	(Optional) The

	Sampling Ratio	sampling ratio for creating a weak learner to accelerate training when building each decision tree		default value is 1.0, indicating that feature sampling is disabled.
l1	L1 Penalty Coefficient	This parameter controls the number of leaf nodes. The larger the value is, the fewer the leaf nodes are.Set a larger value for overfitting.	Non-negative real number	(Optional) The default value is 0.
l2	L2 Penalty Coefficient	This parameter controls distribution of leaf nodes. The larger the value is, the more evenly the leaf nodes are distributed.Set a larger value for overfitting.	Non-negative real number	(Optional) The default value is 1.0.
shrinkage	Learning Rate		(0, 1]	(Optional) The default value is 0.3.
sketchEps	Sketch Precision	Threshold of the splitting point when creating a sketch. The number of bins is O (1.0/sketchEps). The smaller the value is, the more bins are divided. This value does not need to be adjusted under normal conditions.	(0, 1)	(Optional) The default value is 0.03.
minSplitLoss	Minimum Split Loss	Minimum split loss required for splitting a node. The larger the value is, the more conservatively	Non-negative real number	(Optional) The default value is 0.

		the node splits.		
featureNum	Feature Quantity	Number of features or the largest feature ID.Specify this parameter for resource usage estimation.	Positive integer	Optional
baseScore	Global Offset	Original predicted values of all samples	Real number	(Optional) The default value is 0.5.
featureImportanceType	Feature Importance Type	Type of feature importance. “weight” indicates the number of times features are split. “gain” indicates information gain provided by the feature. “cover” indicates the number of samples that feature covers on the splitting node.	“weight”, “gain”, “cover”	(Optional) The default value is “gain”.

NOTE:

Specify different values for the objective parameter in different learning models. On the multiclass classification web GUI, the objective function is automatically specified and hidden from users. On the command line interface, set the objective parameter to “multi:softprob”.

Mappings between metrics and objective functions are: “mlogloss” for “multiclass negative log likelihood” and “merror” for “multiclass classification error”.

Execution optimization

Command option	Parameter	Description	Value range	Required/Optional, default value
coreNum	Number of Cores	Number of cores used for computing. The	Positive integer	(Optional) Cores are automatically

		larger the value is, the faster the computing algorithm runs.		assigned by default.
memSizePerCore	Memory Size per Core (MB)	Memory sized used by each core, where 1024 represents 1 GB memory	Positive integer	(Optional) Memory is automatically assigned by default.

Example

Data generation

The following example uses the KV data format.

```
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;
```

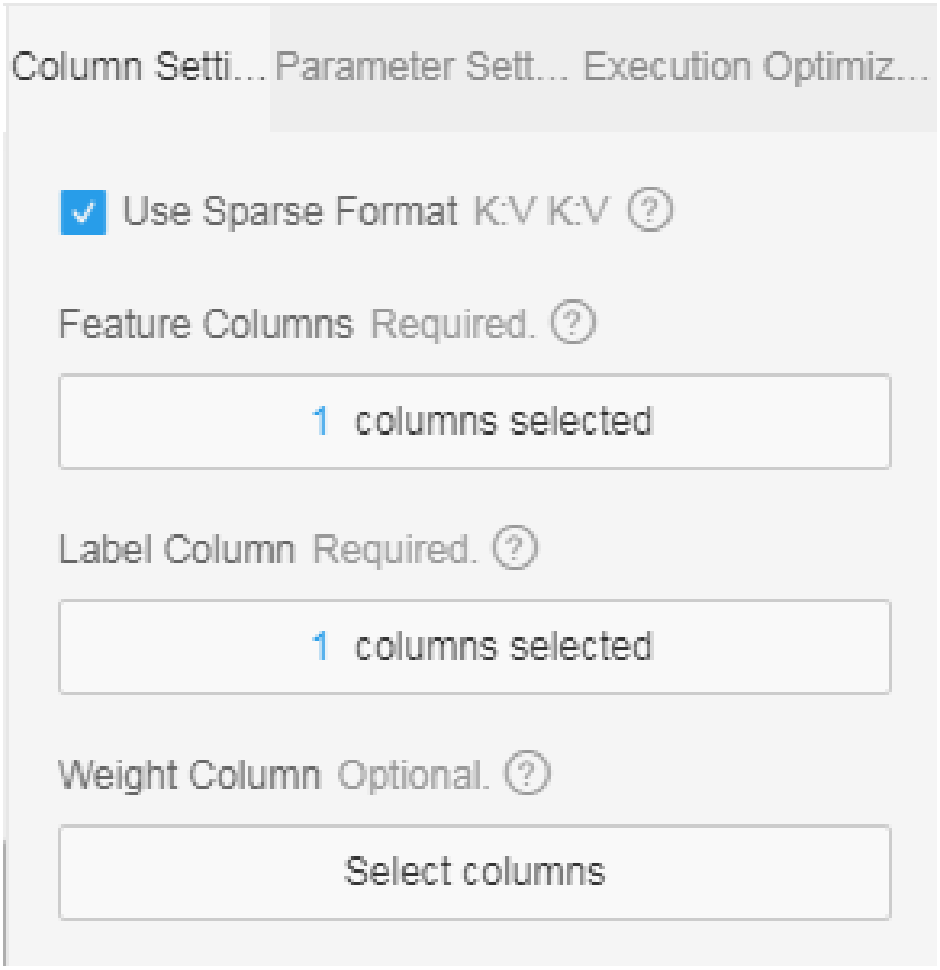
The following figure shows the generated data.

Index	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

The table contains five dimensions of features.

Training

Configure training data and the training component according to Quick start. Select the “label” column as the target column and “features” column as the feature column. The following figures show the data parameter settings page and algorithm parameter settings page.


Column Setti...
Parameter Sett...
Execution Optimiz...
Use Sparse Format
K:V
K:V
Feature Columns
Required.
1 columns selected
Label Column
Required.
1 columns selected
Weight Column
Optional.
Select columns

Column Setti... Parameter Sett... Execution Optimiz...

Number of Classes Required. Non-negative i...

3

Evaluation Index Type Optional.

Multiclass Negative Log Likelihood

Number of Trees Optional. Integer.

5

Maximum Tree Depth Optional. Positive inte...

5

Data Sampling Ratio Optional. Range: (0, 1]

1.0

Feature Sampling Ratio Optional. Range: (0,...

1.0

L1 Penalty Coefficient Optional. Non-negativ...

0

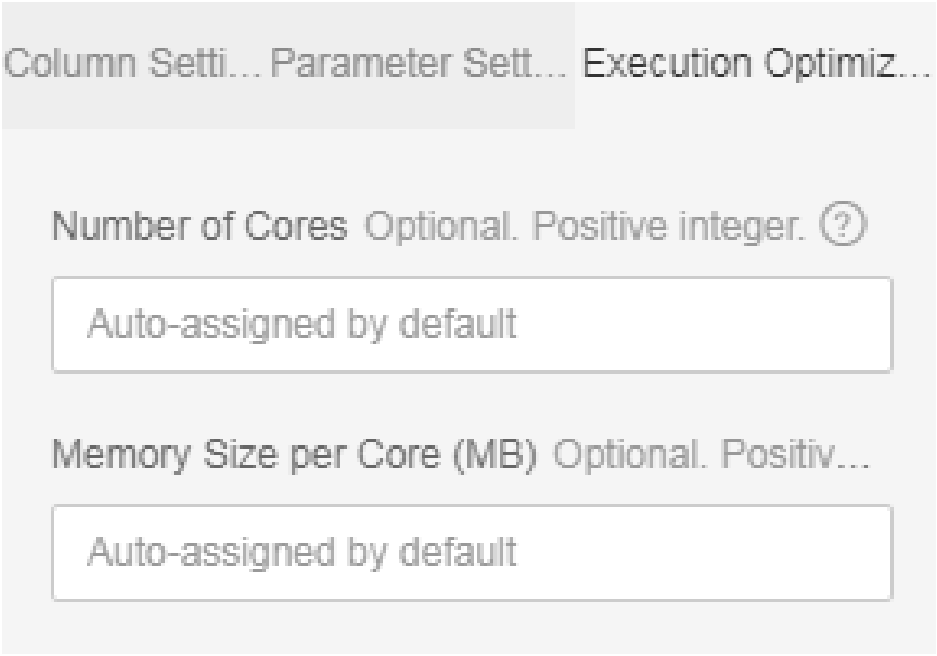
L2 Penalty Coefficient Optional. Non-negativ...

1.0

Learning Rate Optional. Range: (0, 1]

You do not need to set the number of features because this number is calculated automatically by the algorithm.If you have a large number of features and want the algorithm to accurately estimate the amount of resources required, enter the actual number of features here.

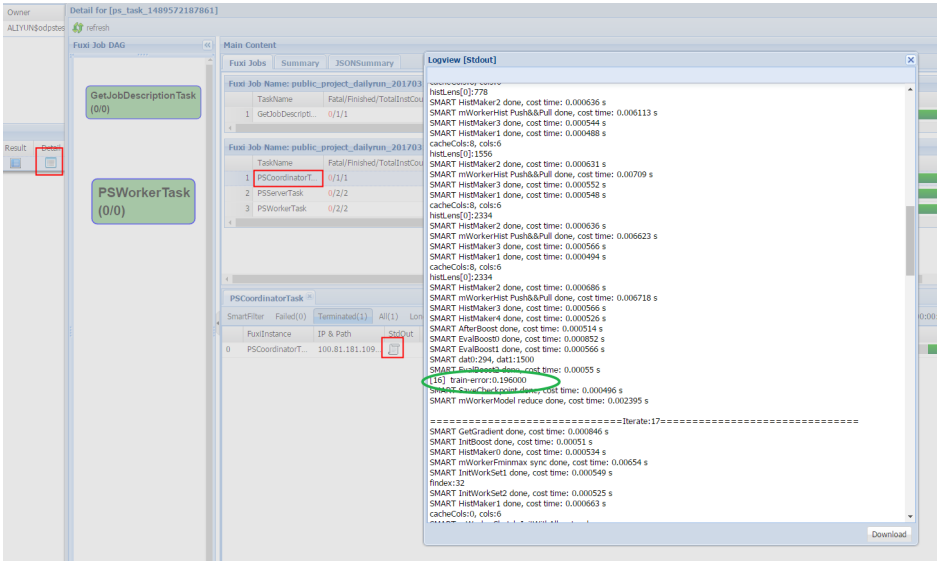
To accelerate the training, you can set the number of cores on the execution optimization page. The larger the number is, the faster the algorithm runs.Generally, you do not need to enter the memory size per core because the algorithm can accurately estimate the memory size.In addition, the PS algorithm starts to run only when all hosts obtain resources. Therefore, you may need to wait for a longer time when the cluster is busy and requires many resources.



You can view metrics output values in stdout of the coordinator in a logview (http link starting with " http://logview.odps.aliyun-inc.com:8080/logview "). One PS-SMART training job has multiple tasks; therefore, multiple logviews are available. Select the logview with logs starting with PS, as shown in the following figure.

The one in the red box is the logview of the PS job. You can identify different tasks by information in the green circle.

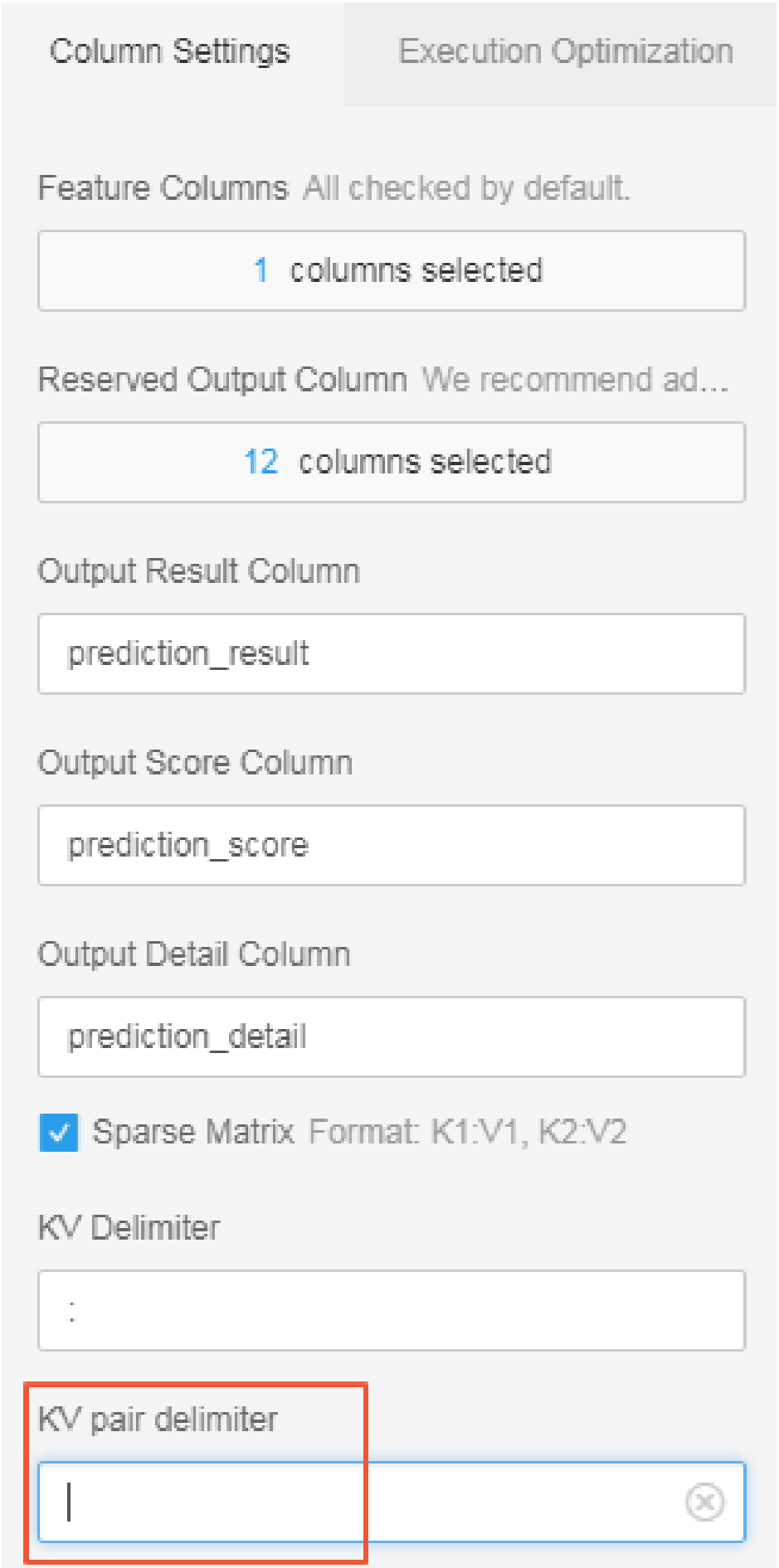
Then, perform operations in the logview according to the following figure.



Prediction

Use the unified prediction component

The output model obtained after training is saved in binary format and can be used for prediction.Configure the input model and test data for the prediction component according to Quick start and set parameters according to the following figure.


Column Settings
Execution Optimization
Feature Columns All checked by default.
1 columns selected
Reserved Output Column We recommend ad...
12 columns selected
Output Result Column
prediction_result
Output Score Column
prediction_score
Output Detail Column
prediction_detail
Sparse Matrix Format: K1:V1, K2:V2
KV Delimiter
:
KV pair delimiter

If the dense format is used, you only need to select feature columns. (All columns are selected by default, and extra columns do not affect the prediction.)If the KV format is used, set the data format to sparse format and select a correct delimiter.In the SMART model, key-value pairs are separated by space characters. Therefore, the delimiter must be set to space or “\u0020” (escape expression of space).

The following figure shows the prediction result.

Index	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": 0.1672592610120773}
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}

In the “prediction_detail” column, values 0, 1, and 2 indicate classes, and the values following them indicate probabilities of the corresponding classes. The “predict_result” column lists the selected classes with the highest probability, and the “predict_score” column lists the probability of each selected class.

Use the PS-SMART prediction component

The output model table obtained after training is saved in binary format and can be used by the PS-SMART prediction component for prediction.Configure the input model and test data for the prediction component according to Quick start and set parameters, including the data format, feature columns, target column, and number of classes. The ID column can only be a string type column other than feature columns and the target column. The loss function must be explicitly set to “multi:softprob”. The following figure shows the prediction result.

Index	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

The “score_class_k” columns list probabilities of class k. The class with the highest probability is the predicted class. The “leaf_index” column lists the predicted leaf node numbers. Each sample has N*M numbers, where N is the number of decision trees, and M is the number of classes. In this example, each sample has 15 numbers (5*3 = 15).Each tree maps to a number, which indicates the number of the leaf node on this tree.

NOTE:

The output model table is a binary table that is not readable. To ensure compatibility with the PS-SMART prediction component, the table provides outputs such as leaf node numbers and evaluation metrics.However, the output table has strict requirements on data formats, resulting in poor user experiences. It will be improved gradually or be replaced by another component.

A string type column must be selected as the label column. You can enter character strings in the column but cannot leave it blank or enter Null in it. A feature column can be converted into the string type by using the data type conversion component and used as the label column.

The loss function must be explicitly set to “multi:softprob”. By default, the loss function does not work.

View feature importance

To view feature importance, you can export the third output stud to an output table, or directly right-click in the PS-SMART component and choose View Data > Output Feature Importance Table. The following figure shows the output feature importance table.

Index ▲	id ▲	value ▲
1	1	0.276059627532959
2	3	0.2085445970296859 7
3	4	0.31002077460289
4	5	0.20537501573562622

If the values in the id column are input feature numbers and the KV format is used, the IDs represent keys in key-value pairs. If the dense format is used and input features are “f0, f1, f2, f3, f4, f5”, ID 0 represents f0, and ID 4 represents f4.Each value indicates a feature importance type. The default value is “gain”, indicating the sum of information gains brought by a feature in the model. The preceding figure shows only four features because only these four features are used in the split of trees. The importance of unused features is considered 0.

Important notes

The target column in a PS-SMART multiclass classification model supports only positive

integer IDs (class numbers are 0, 1, 2, ..., n-1, where n is the number of classes). Even if the values in the ODPS table are strings, they are saved as numerical values. If the classification target is a type string similar to “Good”, “Medium”, or “Bad”, convert it into a numerical value (0, 1, 2, ..., n-1).

In the key-value format, feature IDs must be positive integers, and feature values must be real numbers.If feature IDs are character strings, use the serialization component to serialize them.If feature values are type character strings, perform engineering on the features, such as discretization.

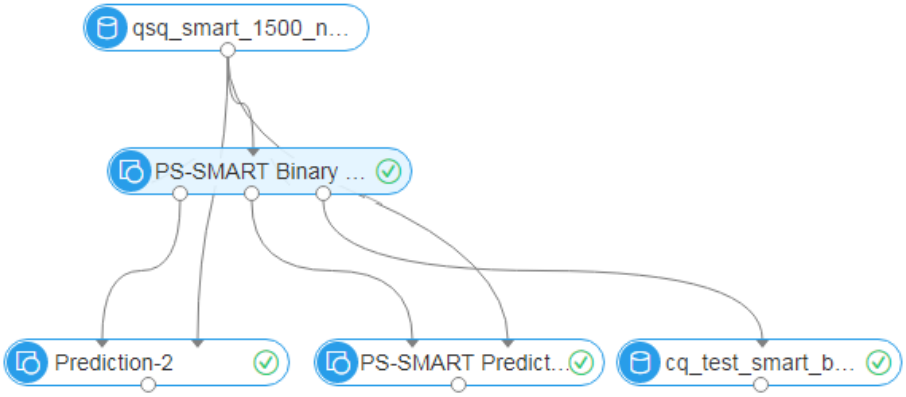
Although PS-SMART supports tasks with hundreds of thousands of features, such tasks consume many resources and run slowly. Therefore, we do not recommend such a large number of features.The GBDT algorithm is suitable for training with continuous features. Continuous type features require one-hot coding (to filter out infrequent features) before they can be used for training. Continuous numerical features can be used for training with the GBDT algorithm directly. Discretization is not recommended for numerical features.

The PS-SMART algorithm applies randomness in many scenarios. For example, the data_sample_ratio and fea_sample_ratio items require data or feature sampling.In addition, the PS-SMART algorithm uses histograms to show similarity. When multiple workers run in a cluster in distributed mode, local sketches are merged to global sketches in a random order. Although different merging orders result in different tree structures, this does not bring great variation in the output model theoretically.Therefore, it is a normal situation if different results are obtained after the algorithm runs multiple times with the same data and same parameter settings.

PS-SMART regression

PS stands for parameter server, which is used for online and offline training tasks of large-scale models. Scalable Multiple Additive Regression Tree (SMART) is an implementation of Gradient boosting decision tree (GBDT) on PS. PS-SMART can run training tasks containing up to tens of billions of samples and hundreds of thousands of features on thousands of nodes. It also supports failover to maintain a high stability. Additionally, PS-SMART supports various data formats, training targets, evaluation targets, output feature importance, and training acceleration (such as histogram similarity).

Quick Start



As shown in the figure, a PS-SMART regression model is learned based on training data.The model has three output studs:

Output model: offline model, which is connected to the unified prediction component. This model does not support output of leaf node numbers.

Output model table: a binary table that is not readable. To ensure compatibility with the PS-SMART prediction component, the table provides outputs such as leaf node numbers and evaluation metrics.However, the output table has strict requirements on data formats, resulting in poor user experiences. It will be improved gradually or be replaced by another component.

Output feature importance table: lists importance of each feature. Three importance types are supported (see parameter description).

PAI command

Training

```
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"
```

Prediction

```
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"
```

Parameter description

Data parameters

Command option	Parameter	Description	Value range	Required/Optional, default value
featureColNames	Feature Column	Name of the feature column selected from the input table for training	The column name must be bigint or double type in dense format or string type in sparse KV format. If the sparse KV format is used, the keys and values must be numerical type data.	Required
labelColName	Label Column	Name of the label column in the input table	The column name can be either a string or a numeric value, but it can only be a numerical value when saved in internal storage. For example, the value can be 0 or 1 for	Required

			regression.	
weightCol	Weight Column	You can specify a weight for each row of samples.	Column name, numerical type	(Optional) It is left blank by default.
enableSparse	Sparse Format	Whether data is in sparse format, in which key-value pairs are separated by spaces whereas keys and values are separated by colons, for example, "1:0.3 3:0.9"	true, false	(Optional) The default value is false.
inputTableName	Name of the input table	NA	NA	Required
modelName	Name of the output model	NA	NA	Required
outputImportanceTableName	Output Feature Importance Table Name	NA	NA	(Optional) It is left blank by default.
inputTablePartitions	Input Table Partitions	NA	NA	Optional, in the format of ds=1/pt=1
outputTableName	Output Model Table Name	The output table is an ODPS table that uses the binary format and is not readable. The self-contained prediction component of SMART can be used to provide output of leaf node numbers.	true, false	(Optional) The default value is false.
lifecycle	Output Table Lifecycle	NA	Positive integer	(Optional) The default value is 3.

Algorithm parameters

Command option	Parameter	Description	Value range	Required/Optional, default value
objective	Objective	The objective	NA	(Required) The

	Function Type	function type affects learning directly and must be selected correctly. Multiple loss functions can be used for regression. See the notes below.		default type is linear regression.
metric	Evaluation Metric Type	Evaluation metrics in the training set, which must be corresponding to the objective function type and are output to stdout of the coordinator in a logview. See the notes and samples below.	NA	(Optional) It is left blank by default.
treeCount	Number of Decision Trees	Number of decision trees. The training time is in direct proportion to this number.	Positive integer	(Optional) The default value is 1.
maxDepth	Maximum Decision Tree Depth	Maximum depth of a decision tree, recommended value: 5 (a maximum of 32 leaf nodes)	Positive integer, [1, 20]	(Optional) The default value is 5.
sampleRatio	Data Sampling Ratio	Data sampling ratio for creating a weak learner to accelerate training when building each decision tree	(0, 1]	(Optional) The default value is 1.0, indicating that data sampling is disabled.
featureRatio	Feature Sampling Ratio	Feature sampling ratio for creating a weak learner to accelerate training when building each decision tree	(0, 1]	(Optional) The default value is 1.0, indicating that feature sampling is disabled.

l1	L1 Penalty Coefficient	This parameter controls the number of leaf nodes. The larger the value is, the fewer the leaf nodes are.Set a larger value for overfitting.	Non-negative real number	(Optional) The default value is 0.
l2	L2 Penalty Coefficient	This parameter controls distribution of leaf nodes. The larger the value is, the more evenly the leaf nodes are distributed.Set a larger value for overfitting.	Non-negative real number	(Optional) The default value is 1.0.
shrinkage	Learning Rate		(0, 1]	(Optional) The default value is 0.3.
sketchEps	Sketch Precision	Threshold of the splitting point when creating a sketch. The number of bins is O (1.0/sketchEps). The smaller the value is, the more bins are divided. This value does not need to be adjusted under normal conditions.	(0, 1)	(Optional) The default value is 0.03.
minSplitLoss	Minimum Split Loss	Minimum split loss required for splitting a node. The larger the value is, the more conservatively the node splits.	Non-negative real number	(Optional) The default value is 0.
featureNum	Feature Quantity	Number of features or the largest feature ID.Specify this parameter for resource usage	Positive integer	Optional

		estimation.		
baseScore	Global Offset	Original predicted values of all samples	Real number	(Optional) The default value is 0.5.
featureImportanceType	Feature Importance Type	Type of feature importance. "weight" indicates the number of times features are split. "gain" indicates information gain provided by the feature. "cover" indicates the number of samples that feature covers on the splitting node.	"weight", "gain", "cover"	(Optional) The default value is "gain".
tweedieVarPower	Tweedie Distribution Index	Tweedie distribution index indicating the relationship between the variance and mean. $Var(y) ~ E(y)^tweedie_variance_power$	(1, 2)	(Optional) The default value is 1.5.

NOTE:

- Specify different values for the objective parameter in different learning models. The regression web GUI provides multiple objective functions, which are described as follows:

reg:linear (Linear regression) Note: The range of label numbers is (-∞, +∞).
reg:logistic (Logistic regression) Note: The range of label numbers is [0, 1].
count:poisson (Poisson regression for count data, output mean of poisson distribution) Note: Label numbers must be greater than 0.
reg:gamma (Gamma regression for modeling insurance claims severity, or for any outcome that might be [gamma-distributed](https://en.wikipedia.org/wiki/Gamma_distribution#Applications)) Note: Label numbers must be greater than 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).) Note: Label numbers must be greater than 0.

- Metrics for these objective functions are:

```
rmse (rooted mean square error, corresponding to objective reg:linear)
mae (mean absolute error, corresponding to objective reg:linear)
poisson-nloglik (negative loglikelihood for poisson regression, corresponding to objective count:poisson)
gamma-deviance (Residual deviance for gamma regression, corresponding to objective reg:gamma)
gamma-nloglik (Negative log-likelihood for gamma regression, corresponding to 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)
```

Execution optimization

Command option	Parameter	Description	Value range	Required/Optional, default value
coreNum	Number of Cores	Number of cores used for computing. The larger the value is, the faster the computing algorithm runs.	Positive integer	(Optional) Cores are automatically assigned by default.
memSizePerCore	Memory Size per Core (MB)	Memory sized used by each core, where 1024 represents 1 GB memory	Positive integer	(Optional) Memory is automatically assigned by default.

Example

Data generation

The following example uses the sparse KV data format.

```
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;
```

The following figure shows the generated data.

Index	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

In the table, feature IDs are numbered starting from 1, and the maximum feature ID is 5.

Training

Select the “label” column as the target column and “features” column as the feature column. The following figure shows the algorithm parameter settings page for linear regression.

Column Setti... Parameter Sett... Execution Optimiz...

Loss Function Type

Regression Loss

Exponent Base of Gbrank and Regression L...

1

Metric Type

NDCG

Tree Quantity [1, 10000]

500

Learning Rate (0, 1)

0.05

Max Leaf Quantity [1, 1000]

32

Max Tree Depth [1, 100]

10

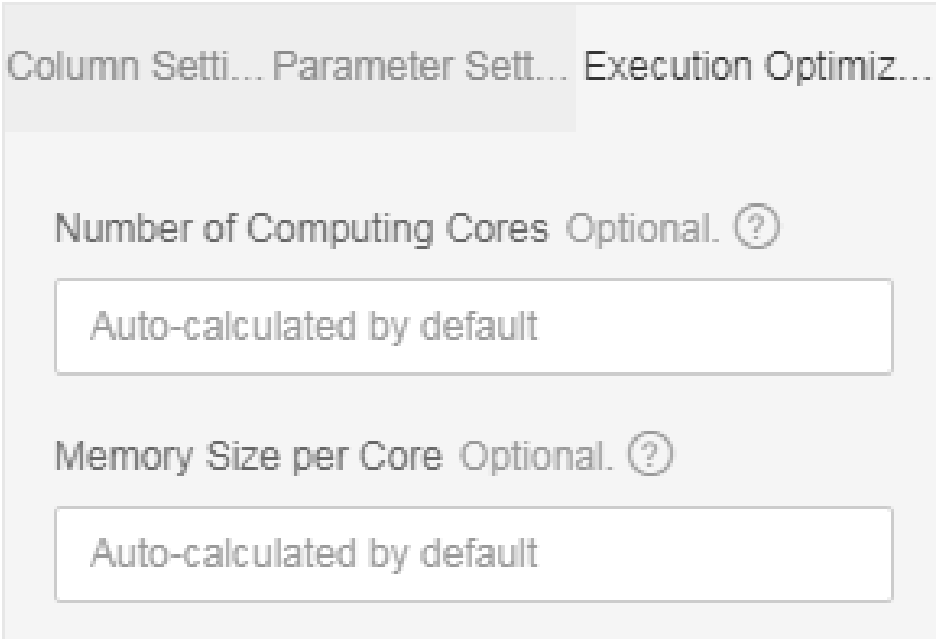
Min Sample Quantity on a Leaf Node [1, 1000]

500

Sample Ratio (0, 1)

You do not need to set the number of features because this number is calculated automatically by the algorithm.If you have a large number of features and want the algorithm to accurately estimate the amount of resources required, enter the actual number of features here.

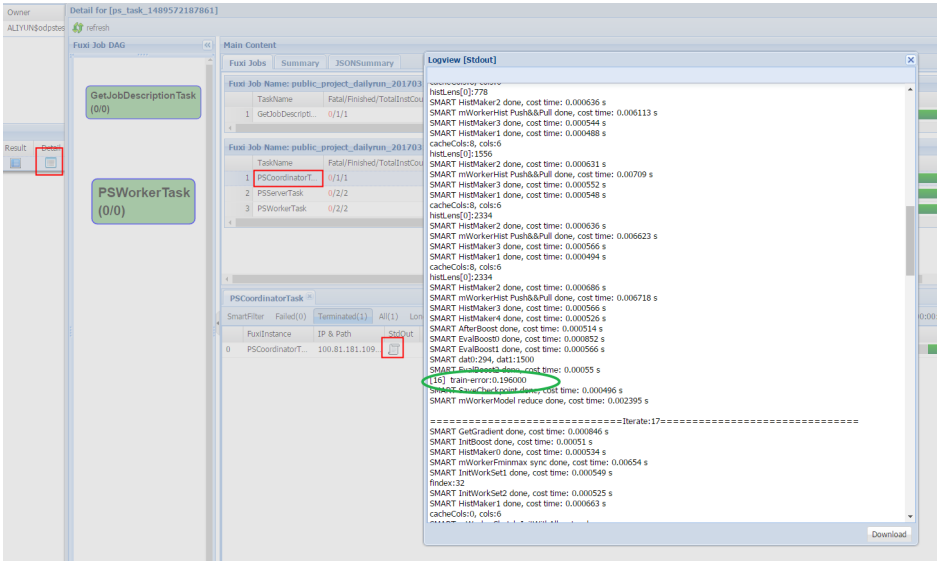
To accelerate the training, you can set the number of cores on the execution optimization page. The larger the number is, the faster the algorithm runs.Generally, you do not need to enter the memory size per core because the algorithm can accurately estimate the memory size.In addition, the PS algorithm starts to run only when all hosts obtain resources. Therefore, you may need to wait for a longer time when the cluster is busy and requires many resources.



You can view metrics output values in stdout of the coordinator in a logview (http link starting with " http://logview.odps.aliyun-inc.com:8080/logview "). One PS-SMART training job has multiple tasks; therefore, multiple logviews are available. Select the logview with logs starting with PS, as shown in the following figure.

The one in the red box is the logview of the PS job. You can identify different tasks by information in the green circle.

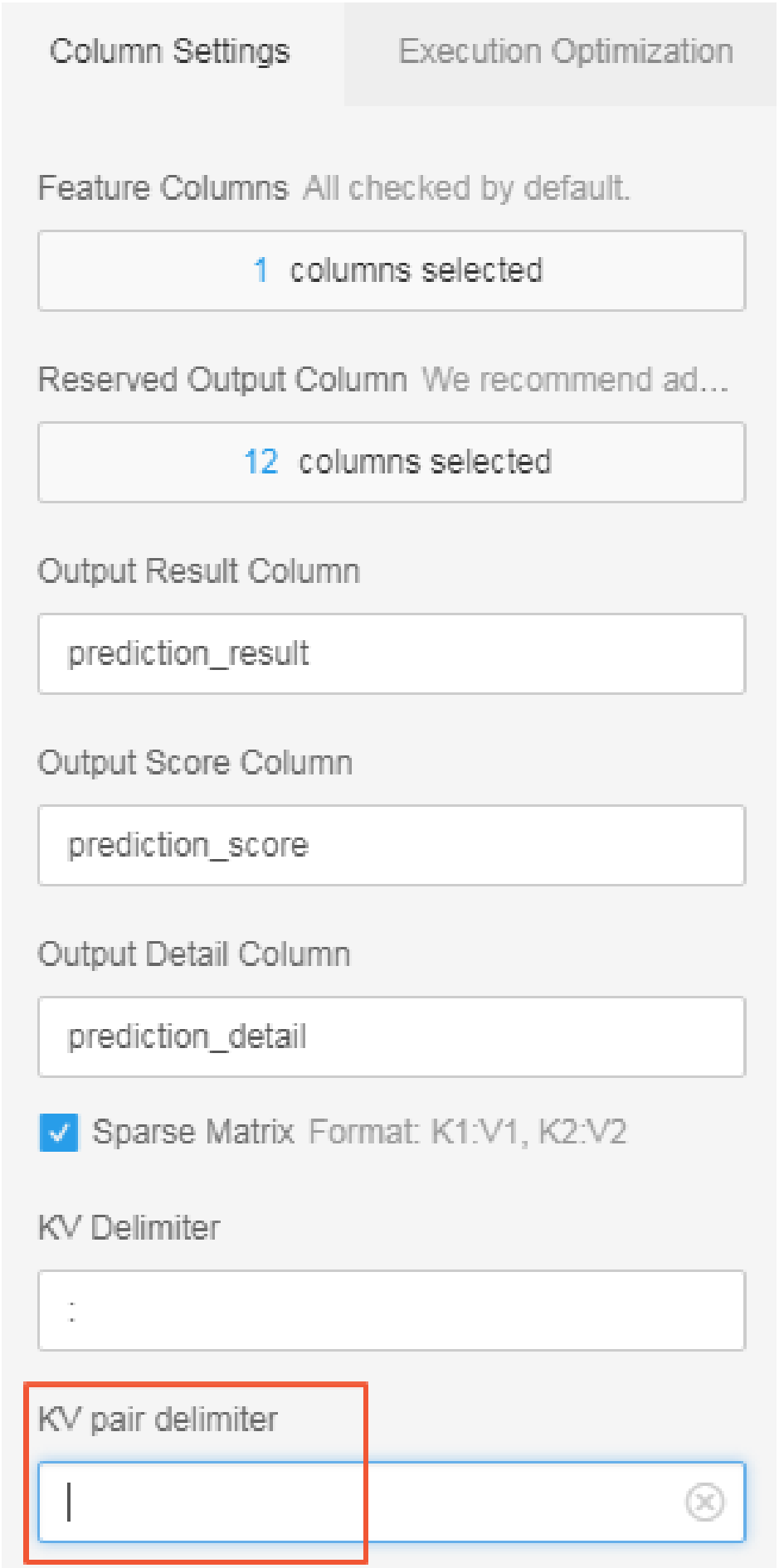
Then, perform operations in the logview according to the following figure.



Prediction

Use the unified prediction component

The output model obtained after training is saved in binary format and can be used for prediction.Configure the input model and test data for the prediction component according to Quick start and set parameters according to the following figure.


Column Settings
Execution Optimization
Feature Columns All checked by default.
1 columns selected
Reserved Output Column We recommend ad...
12 columns selected
Output Result Column
prediction_result
Output Score Column
prediction_score
Output Detail Column
prediction_detail
Sparse Matrix Format: K1:V1, K2:V2
KV Delimiter
:
KV pair delimiter

If the dense format is used, you only need to select feature columns. (All columns are selected by default, and extra columns do not affect the prediction.)If the KV format is used, set the data format to sparse format and select a correct delimiter.In the SMART model, key-value pairs are separated by space characters. Therefore, the delimiter must be set to space or “\u0020” (escape expression of space).

The following figure shows the prediction result.

Index	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}

The “predict_result” column lists the predicted values.

Use the PS-SMART prediction component

The output model table obtained after training is saved in binary format and can be used by the PS-SMART prediction component for prediction.Configure the input model and test data for the prediction component according to Quick start and set parameters, including the data format, feature columns, target column, and number of classes. The ID column can only be a string type column other than feature columns and the target column. The loss function must be explicitly set to the objective function used for training.The following figure shows the prediction result.

Index	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

The “prediction_score” column lists the predicted values. The “leaf_index” column lists the predicted leaf node numbers. Each sample has N numbers, where N is the number of decision trees.Each tree maps to a number, which indicates the number of the leaf node on this tree.

NOTE:

The output model table is a binary table that is not readable. To ensure compatibility with the PS-SMART prediction component, the table provides outputs such as leaf node numbers and evaluation metrics.However, the output table has strict requirements on data formats, resulting in poor user experiences. It will be improved gradually or be replaced by another component.

A string type column must be selected as the label column. You can enter character strings in the column but cannot leave it blank or enter Null in it. A feature column can be converted into the string type by using the data type conversion component and used as the label column.

The loss function must be explicitly set to the objective function used for training. By default, the loss function does not work.

View feature importance

To view feature importance, you can export the third output stud to an output table, or directly right-click in the PS-SMART component and choose View Data > Output Feature Importance Table. The following figure shows the output feature importance table.

Index ▲	id ▲	value ▲
1	1	0.14059734344482422
2	4	0.8594027161598206

If the values in the id column are input feature numbers and the KV format is used, the IDs represent keys in key-value pairs.If the dense format is used and input features are “f0, f1, f2, f3, f4, f5”, ID 0 represents f0, and ID 4 represents f4.Each value indicates a feature importance type. The default value is “gain”, indicating the sum of information gains brought by a feature in the model. The preceding figure shows only two features because only these two features are used in split of trees. The importance of unused features is considered 0.

Important notes

The target column in a PS-SMART regression model supports only numerical values. Even if values in the ODPS table are strings, they are saved as numerical values.

In the key-value format, feature IDs must be positive integers, and feature values must be

real numbers.If feature IDs are character strings, use the serialization component to serialize them.If feature values are type character strings, perform engineering on the features, such as discretization.

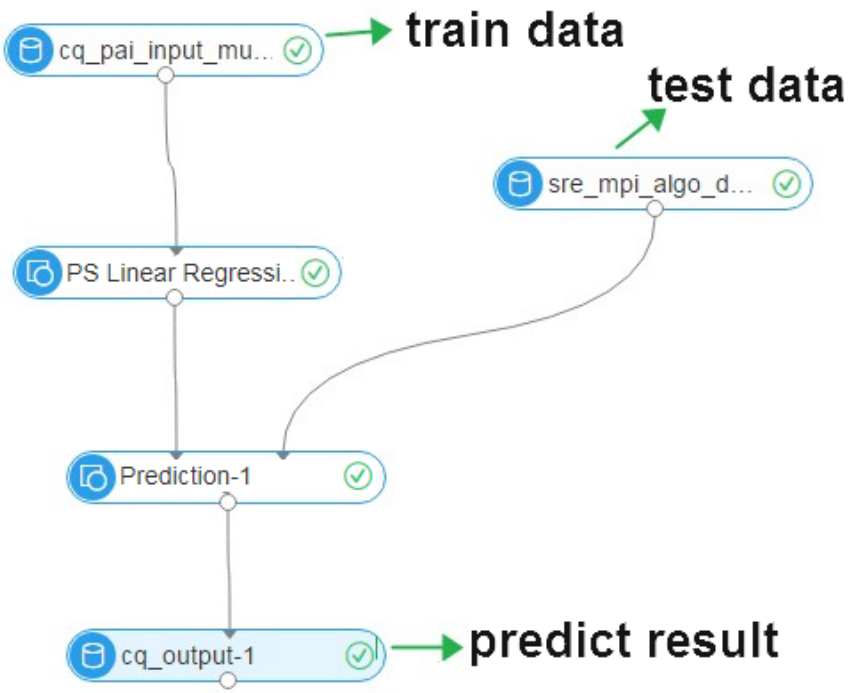
Although PS-SMART supports tasks with hundreds of thousands of features, such tasks consume many resources and run slowly. Therefore, we do not recommend such a large number of features.The GBDT algorithm is suitable for training with continuous features. Continuous type features require one-hot coding (to filter out infrequent features) before they can be used for training. Continuous numerical features can be used for training with the GBDT algorithm directly. Discretization is not recommended for numerical features.

The PS-SMART algorithm applies randomness in many scenarios. For example, the data_sample_ratio and fea_sample_ratio items require data or feature sampling.In addition, the PS-SMART algorithm uses histograms to show similarity. When multiple workers run in a cluster in distributed mode, local sketches are merged to global sketches in a random order. Although different merging orders result in different tree structures, this does not bring great variation in the output model theoretically.Therefore, it is a normal situation if different results are obtained after the algorithm runs multiple times with the same data and same parameter settings.

PS linear regression

Linear regression is a classic regression algorithm used to analyze the linear relationship between a dependent variable and multiple independent variables. A parameter server (PS) is used to run online and offline training tasks of large-scale models. Parameter servers can use over a hundred billion rows of samples to train tens of billions of feature models at high efficiency. The PS linear regression model can run training tasks with hundreds of billions of samples and billions of features, and supports L1 and L2 regular expressions. To run tasks of larger scales, contact the author of the algorithm.

Quick Start



PAI command

Training

```
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
```

Prediction

```
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
```

Parameter description

Data parameters

Command option	Parameter	Description	Value range	Required/Optional, default value
featureColNames	Feature Columns	Feature columns selected from the input table for training	The column names must be bigint or double type in dense format or string type in sparse KV format.	Required
labelColName	Label Column	Name of the label column in the input table	The column name must be a numerical value (bigint or double type).	Required
enableSparse	Sparse Format	If the sparse KV format is used, do not use feature ID 0. We recommend that you number feature IDs starting from 1.	true, false	(Optional) The default value is false.
itemDelimiter	KV Pair Delimiter	Delimiter used between key-value pairs when data in the input table is in sparse format	NA	(Optional) The default delimiter is space.
kvDelimiter	Key and Value Delimiter	Delimiter used between keys and values when data in the input table is in sparse format	NA	(Optional) The default delimiter is colon.
inputTableName	Name of the input table	NA	NA	Required
modelName	Name of the output model	NA	NA	Required
inputTablePartitions	Input Table Partitions	NA	NA	Optional, in the format of ds=1/pt=1

enableModelIo	Output to Offline Model	When this parameter is set to false, output data is exported to the ODPS table, where you can view weights of models	true, false	(Optional) The default value is true.

Algorithm parameters

Command option	Parameter	Description	Value range	Required/Optional, default value
l1Weight	L1 weight	L1 regularization coefficient. The larger the value is, the less zeros a model has.Set a larger value for overfitting.	Non-negative real number	(Optional) The default value is 1.0.
l2Weight	L2 weight	L2 regularization coefficient. The larger the value is, the smaller the model parameter absolute values are.Set a larger value for overfitting.	Non-negative real number	(Optional) The default value is 0.
maxIter	Maximum Iterations	Maximum number of L-BFGS/OWL-QN iterations. The value 0 indicates that the number of iterations is unlimited.	Non-negative integer	(Optional) The default value is 100.
epsilon	Minimum Convergence Deviation	Mean value of the relative loss change rate in ten iterations, which is used as the condition for determining whether to terminate the	Real number between 0 and 1	(Optional) The default value is 1.0e-06.

		optimization algorithm. The smaller the value is, the more strict the condition is and the longer the algorithm runs.		
modelSize	Largest Feature ID	Largest feature ID among all features (feature dimension). It can be larger than the actual largest feature ID. The larger the value is, the higher the memory usage is. If you leave this parameter blank, the system starts an SQL task to calculate the largest feature ID automatically.	Non-negative integer	(Optional) The default value is 0.

Both the maximum iterations and minimum convergence deviation determine when the algorithm stops. If both the parameters are set, the algorithm stops when either condition is triggered.

Execution optimization

Command option	Parameter	Description	Value range	Required/Optional, default value
coreNum	Number of Cores	Number of cores used for computing. The larger the value is, the faster the computing algorithm runs.	Positive integer	(Optional) Cores are automatically assigned by default.
memSizePerCore	Memory Size per Core (MB)	Memory sized used by each core, where 1024 represents 1 GB memory	Positive integer	(Optional) Memory is automatically assigned by default. Generally, you do not need to set this parameter

				because the algorithm can accurately estimate the memory size required.

Example

Data generation

The following example uses the sparse KV data format.

```
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;
```

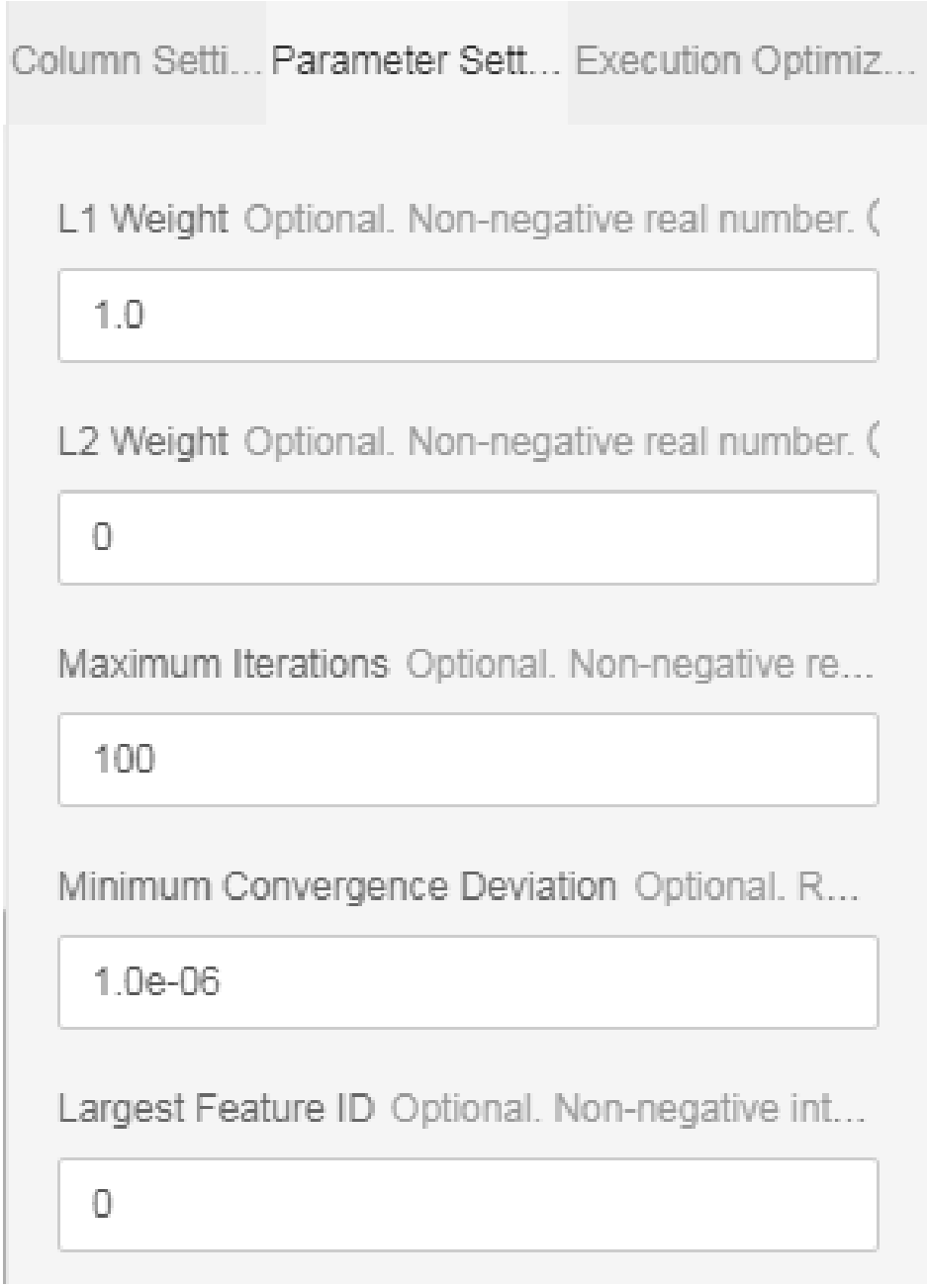
The following figure shows the generated data.

Index ▲	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

In the table, feature IDs are numbered starting from 1, and the maximum feature ID is 5.

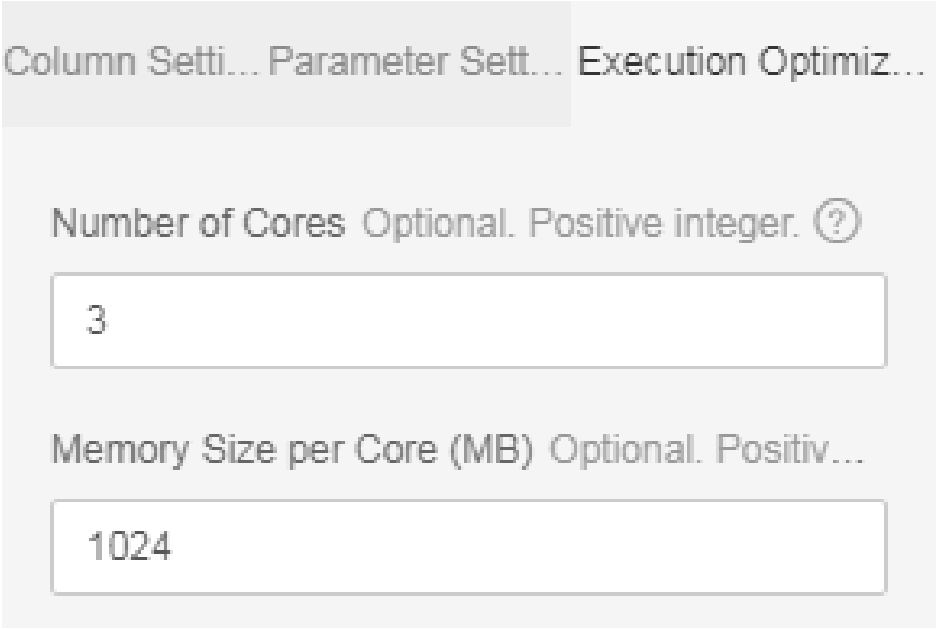
Training

Configure training data and the training component according to Quick start. Select the “label” column as the target column and “features” column as the feature column. Select the sparse data format.The following figure shows the algorithm parameter settings page.



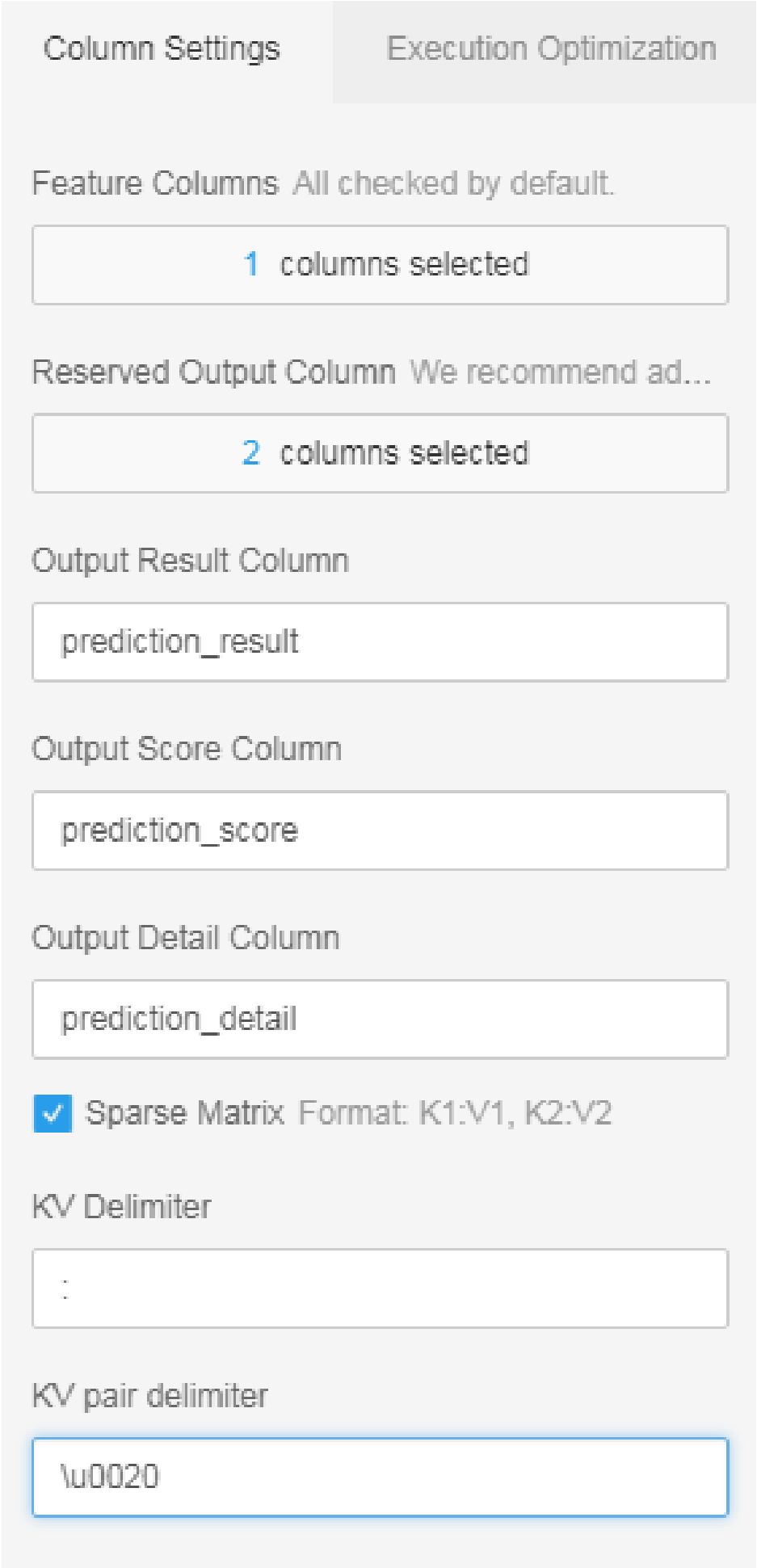
You can retain the default value 0 of the largest feature ID. The algorithm can start an SQL task to calculate the largest feature ID automatically. If you do not want to start the SQL task, enter a value greater than 5. This value indicates the number of feature columns in dense format and indicates the largest feature ID in KV format.

To accelerate the training, you can set the number of cores on the execution optimization page. The larger the number is, the faster the algorithm runs.Generally, you do not need to enter the memory size per core because the algorithm can accurately estimate the memory size.In addition, the PS algorithm starts to run only when all hosts obtain resources. Therefore, you may need to wait for a longer time when the cluster is busy and requires many resources.



Prediction

The model obtained after training is saved in binary format and can be used for prediction.Configure the input model and test data for the prediction component according to Quick start and set parameters according to the following figure.


Column Settings
Execution Optimization
Feature Columns All checked by default.
1 columns selected
Reserved Output Column We recommend ad...
2 columns selected
Output Result Column
prediction_result
Output Score Column
prediction_score
Output Detail Column
prediction_detail
Sparse Matrix Format: K1:V1, K2:V2
KV Delimiter
:
KV pair delimiter
\u0020

Select the KV format used for training and set a correct delimiter. When the KV format is used, key-value pairs are separated by space characters. Therefore, the delimiter must be set to space or "\u0020" (escape expression of space).

The following figure shows the prediction result.

Index	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}

You only need to view the predict_result column.

Important notes

In the key-value format, feature IDs must be positive integers, and feature values must be real numbers.If feature IDs are character strings, use the serialization component to serialize them.If feature values are type character strings, perform engineering on the features, such as discretization.

Clustering model evaluation

This component evaluates clustering models based on metrics and icons.

PAI command

```
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;
```

Parameter description

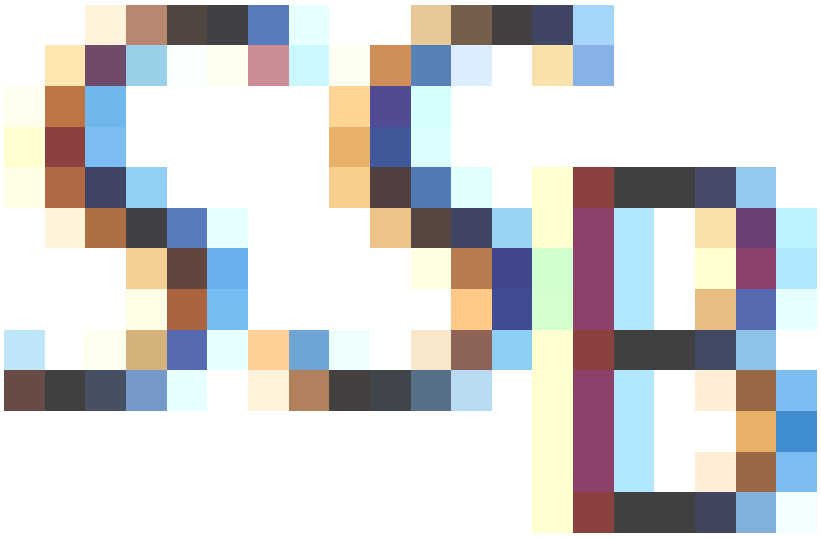
Parameter	Description	Value range	Required/Optional, default value/act
inputTableName	Input table	Table name	Required
selectedColNames	Names of the columns used for evaluation in the input table, which are separated by commas. The column names must	Column names	(Optional) All columns in the input table are selected by default.

	be the same as feature names saved in the model.		
inputTablePartitions	Partitions used for evaluation in the input table, in the format of name1=value1/name2=value2. Multiple partition names are separated by commas.	NA	(Optional) All partitions are selected by default.
enableSparse	Whether data in the input table is in sparse format	true, false	Optional, default: false
itemDelimiter	Delimiter used between key-value pairs when data in the input table is in sparse format	NA	Optional, default: space
kvDelimiter	Delimiter used between keys and values when data in the input table is in sparse format	NA	Optional, default: colon
modelName	Name of the input clustering model	Model name	Required
outputTableName	Name of the output table	Table name	Required
lifecycle	(Optional) Lifecycle of the output table	Positive integer	No lifecycle

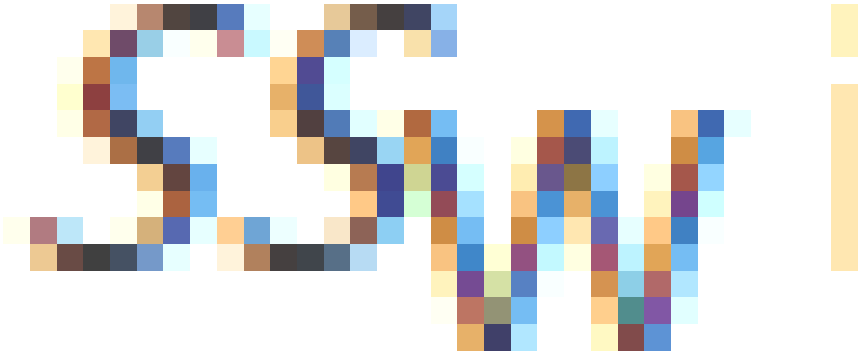
Evaluation formula

The Calinski-Harabasz metric is also known as the variance ratio criterion (VRC). It is defined as follows:

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

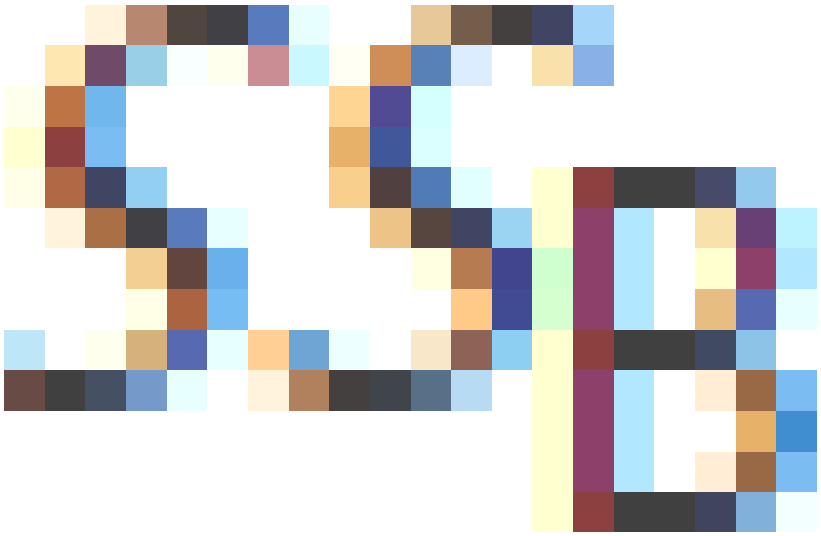


is the inter-cluster variance.



is the intra-cluster variance.

N is the total number of records, and k is the number of cluster centers.



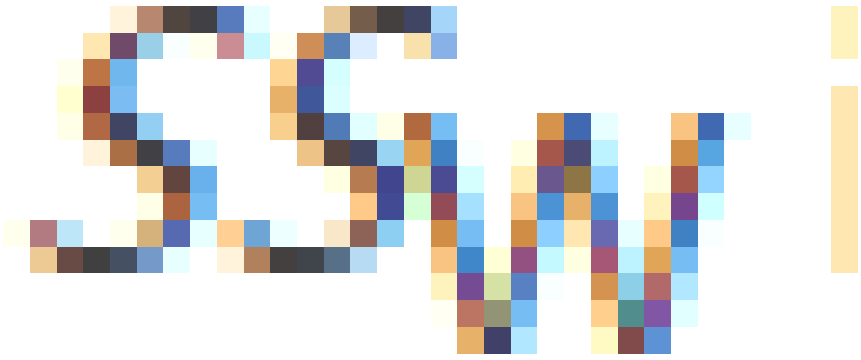
is defined as

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

follows:

- k is the number of the cluster centers.

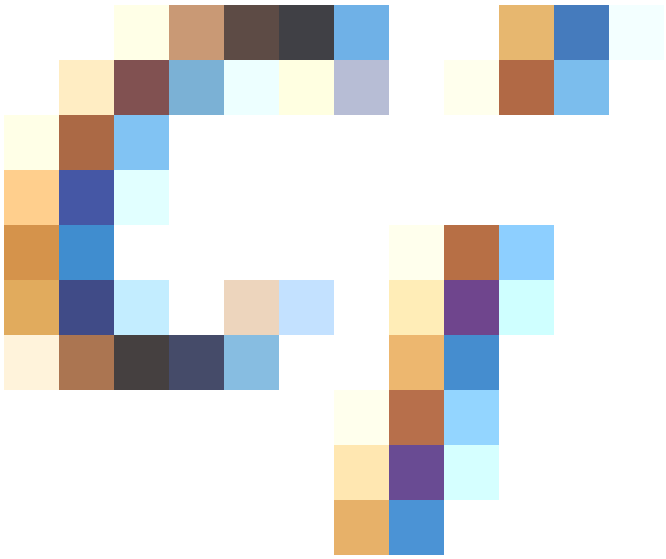
- is the center of cluster i, and m is the mean of input data.



is defined as follows:

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

- k is the number of cluster centers, and x is a data point.



- indicates cluster i.

- indicates the center of cluster i.

Example

Test data

```
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;
```

Build a clustering model

```
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 command

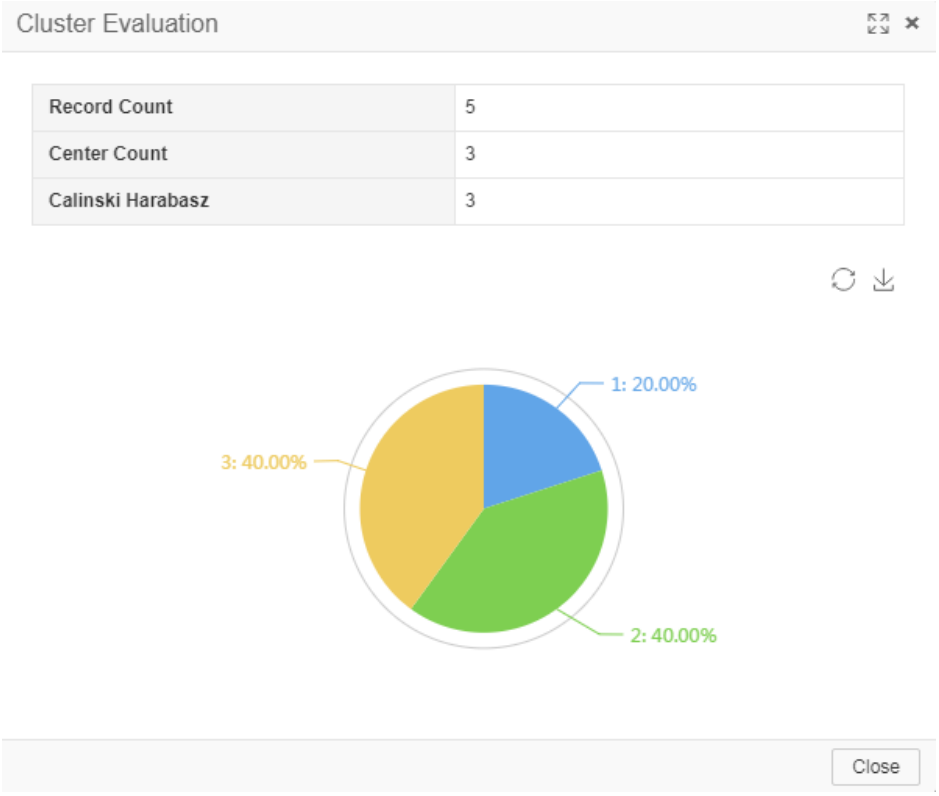
```
PAI -name cluster_evaluation
-project algo_public
-DinputTableName=pai_cluster_evaluation_test_input
-DselectedColNames=f0,f3
-DmodelName=pai_kmeans_test_model
-DoutputTableName=pai_ft_cluster_evaluation_out;
```

Output description

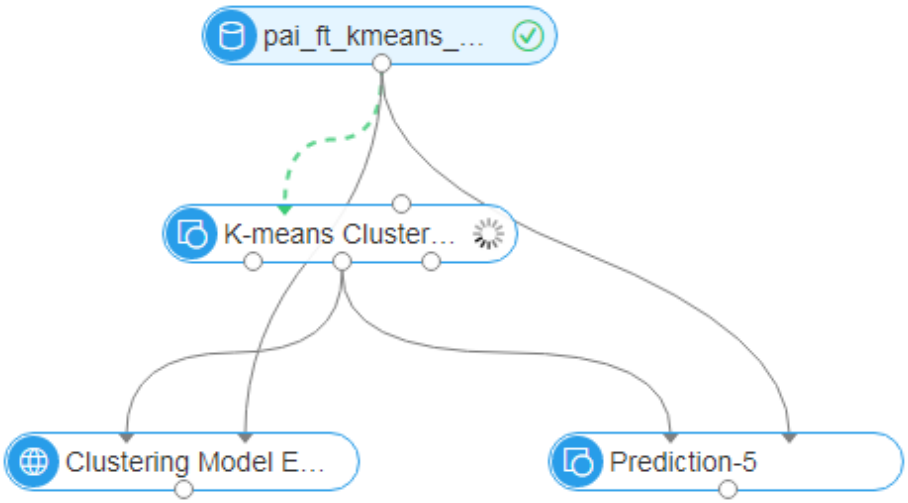
Output table outputTableName, with the following fields:

column name	comment
count	Total number of records
centerCount	Number of cluster centers
calinhara	Calinski Harabasz metric Evaluation formula
clusterCounts	Total number of points in each cluster

Display on the Machine Learning Platform For AI web GUI:



PaiWeb-Pipeline:



Deep learning

Contents

Introduction to deep learning

Alibaba Cloud Machine Learning Platform for AI supports multiple deep learning frameworks and provides powerful GPU clusters that contain both M40 and P100 GPU nodes. You can use these frameworks and hardware resources to train your deep learning algorithms.

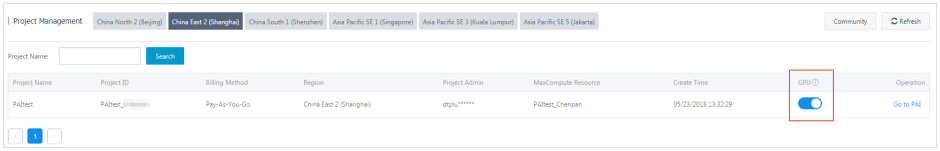
Supported frameworks currently include TensorFlow (versions 1.0, 1,1, and 1,2), MXNet 0.9.5, and Caffe RC3. TensorFlow and MXNet support Python. Caffe supports custom net files.

Before using deep learning frameworks, you must upload your data to Alibaba Cloud Object Storage Service (OSS). The algorithms can read data from specified OSS directories when running. Note that machine learning GPU clusters are currently only available in the China (Shanghai) region. If your algorithms only read OSS data from the China (Shanghai) region, no traffic fees are incurred.

Enable deep learning

Currently, the deep learning feature is in beta testing. Three deep learning frameworks: TensorFlow,

Caffe, and MXNet are supported. To enable deep learning, log on to your machine learning platform console and enable GPU resources, as shown in the following figure:



After GPU resources have been enabled, the corresponding projects are allowed to access the public resource pool and dynamically use the underlying GPU resources.

Upload data to OSS

Before using deep learning to process your data, you must upload your data to OSS. First, create OSS buckets. Since deep learning GPU clusters are only available in the China (Shanghai) region, we recommend that you choose this region when creating OSS buckets. Your data is then transmitted over the Alibaba Cloud classic network. No traffic fees are incurred when you run your algorithms. After the OSS buckets have been created, you can log on to the [OSS console] to create folders and upload your data.

OSS supports multiple data upload methods. For more information, see: https://www.alibabacloud.com/help/doc-detail/31848.html.

OSS also provides many tools to help you make full use of this service. For more information, see: https://www.alibabacloud.com/help/doc-detail/44075.html.

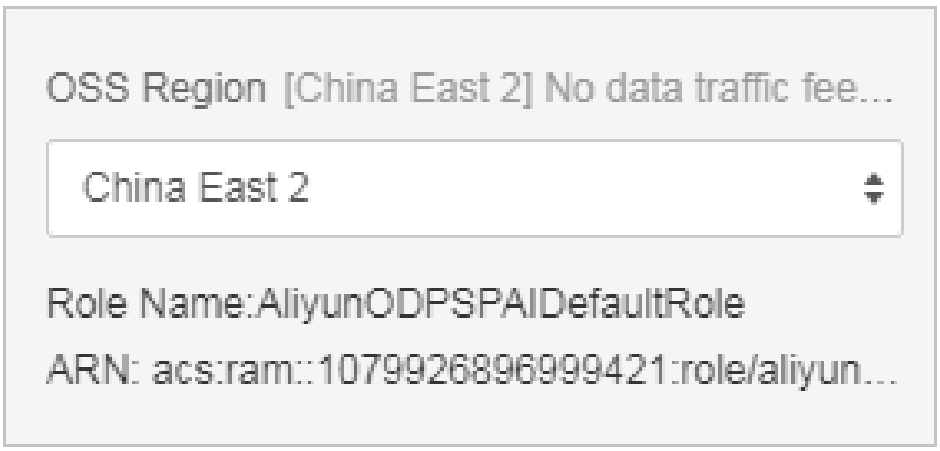
We recommend that you use the ossutil or osscmd tool to upload or download files using command lines. These tools can resume a transmission from breakpoints.

Note: You must specify your Access Key ID to use these tools. You can log on to the Access Key console to create or view Access Keys.

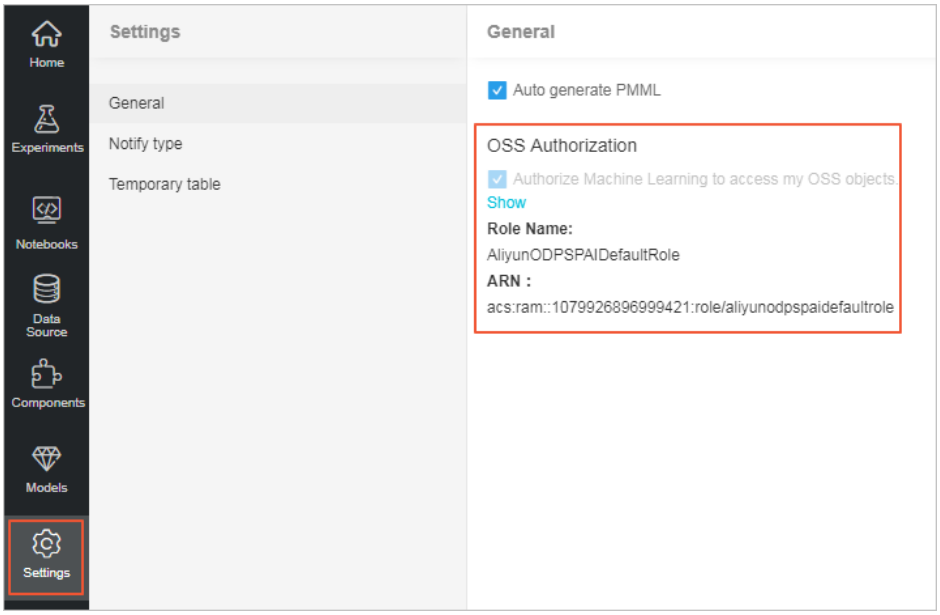
Read OSS buckets

Before reading data from OSS buckets, you must grant “AliyunODPSPAIDefaultRole” access to your DTplus account.

> **Note**: The machine learning platform is based on MaxCompute. Accounts are shared between both platforms. The default role is granted to the MaxCompute account.



You must grant OSS read and write permissions in **Settings**. For more information, see RAM settings.

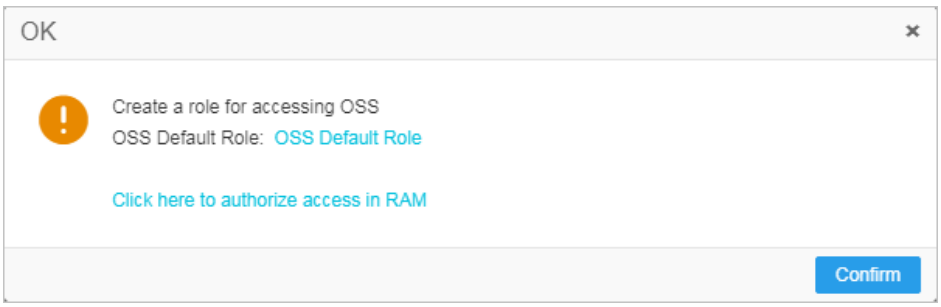


RAM settings

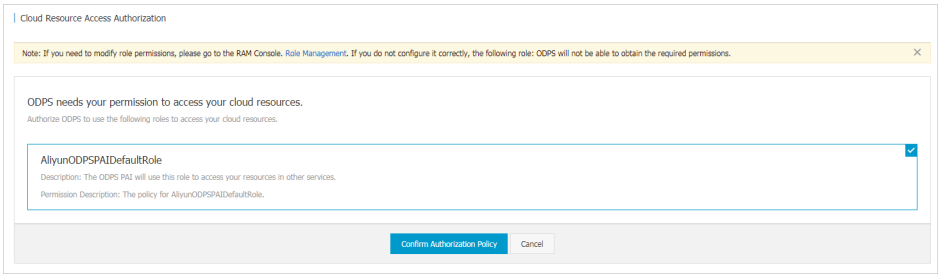
Log on to Machine Learning Platform for AI console, click **Settings** in the left menu bar, and select **General**.

Choose **Authorize Machine Learning to access my OSS objects** on the **OSS Authorization** tab.

Click **Click here to authorize access in RAM**.



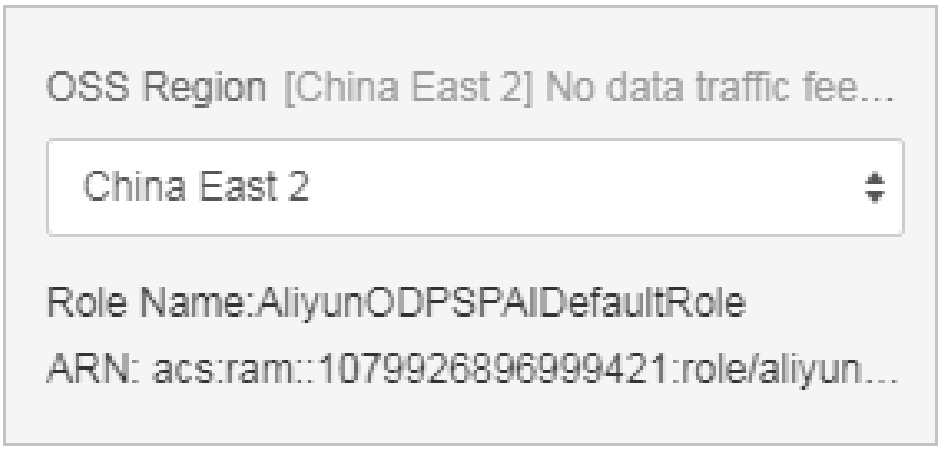
Click **Confirm Authorization Policy**.



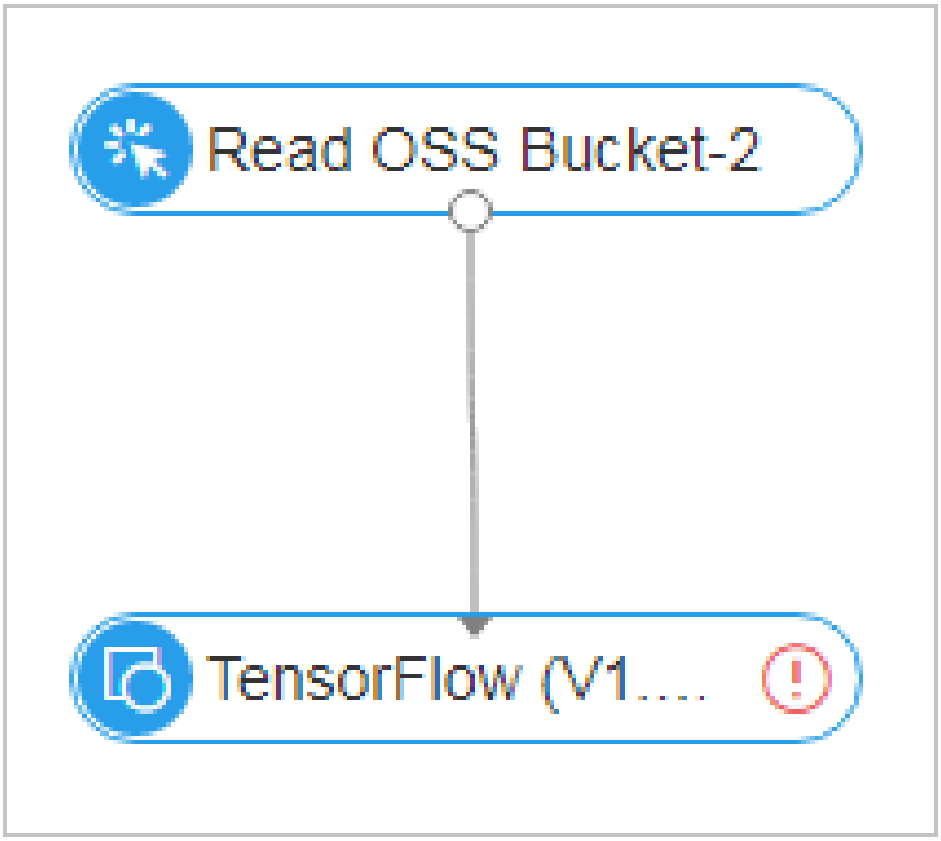
Note: For more information about AliyunODPSPAIDefaultRole, visit the **RAM console**. The default role AliyunODPSPAIDefaultRole can perform the following actions:

Action	Description
oss:PutObject	Upload file or folder objects
oss:GetObject	Obtain file or folder objects
oss:ListObjects	View listed files
oss:DeleteObjects	Delete objects

Navigate to the machine learning platform, and click **Refresh** to load RAM information, as shown in the following figure.



Link the **Read OSS Bucket** component with the corresponding deep learning component to grant OSS permissions.

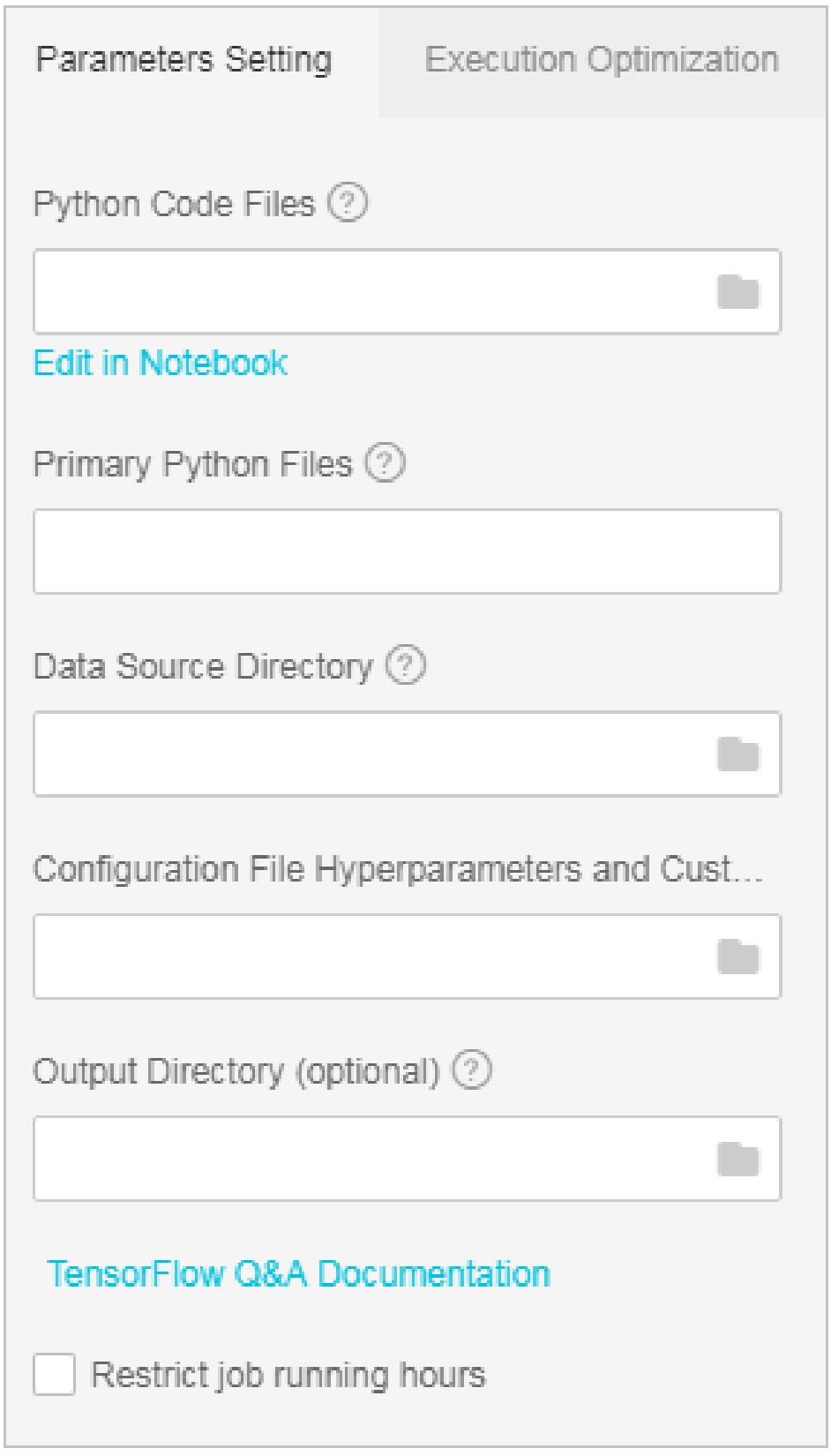


TensorFlow

TensorFlow (TF) is an open source machine learning framework developed by Google. TF is simple and easy to use. Alibaba Cloud Machine Learning Platform for AI supports TensorFlow. You can write code in TF and dynamically adjust the GPU resources.

Parameters

Parameter settings



Python Code File: You can compress multiple code files into a tar.gz file.

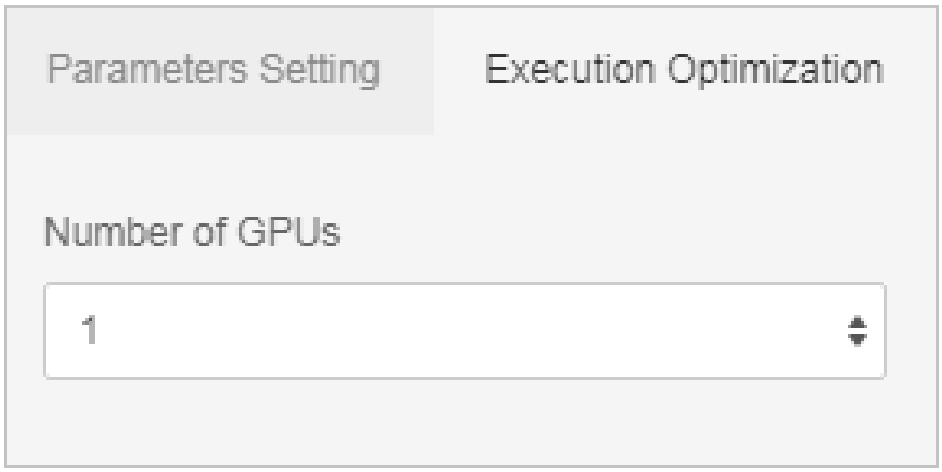
Main Python File: Specify the main file in the compressed file. This step is optional.

Data Source Directory: Select the OSS data source.

Hyperparameters and Custom Parameters: You can write commands to specify hyperparameters and custom parameters. Using these parameters, you can try different learning rates and batch sizes in your models.

Output Directory: Specify the output directory of your model.

Tuning



You can specify the number of GPUs based on the complexity of your tasks.

PAI commands

Not all parameters are needed for actual use. Specify your own parameters based on your needs. For more information about these parameters, see the following table.

```
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";
```

The description of the parameters are as follows:

Parameter	Description	Format	Default
script	Required. The TF algorithm file. This file can be a regular Python file or a TAR.GZ file.	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_tensorflow/mnist_ext.py	NA

entryFile	Optional. The entry file for the algorithm. This parameter is required if you specified a TAR.GZ file for the script parameter.	train.py	Null
buckets	Required. You can specify multiple OSS buckets by separating them with commas (,). Each bucket must end with a backslash ().	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_tensorflow/mnist/	Null
arn	Required. OSS role_arn	NA	Null
gpuRequired	Required. The number of GPUs that are used.	200	100
checkpointDir	Optional. The TF checkpoint directory.	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_tensorflow/mnist/	Null
hyperParameters	Optional. The path of the hyperparameter file.	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_tensorflow/mnist/hyper_parameters.txt	Null

script and **entryFile** are used to specify the algorithm scripts. If your algorithm is complex and consists of multiple files, you can compress these files into a TAR.GZ file and use **entryFile** to specify the entry file for the algorithm.

checkpointDir is used to specify the OSS directory that the algorithm writes to. This parameter is required if you must save models in TensorFlow.

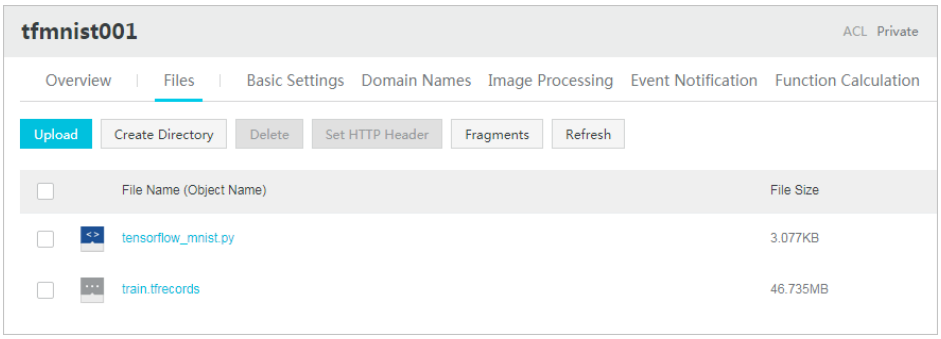
buckets is used to specify the OSS directory from which the algorithm read data. **arn** is required to use OSS.

Example

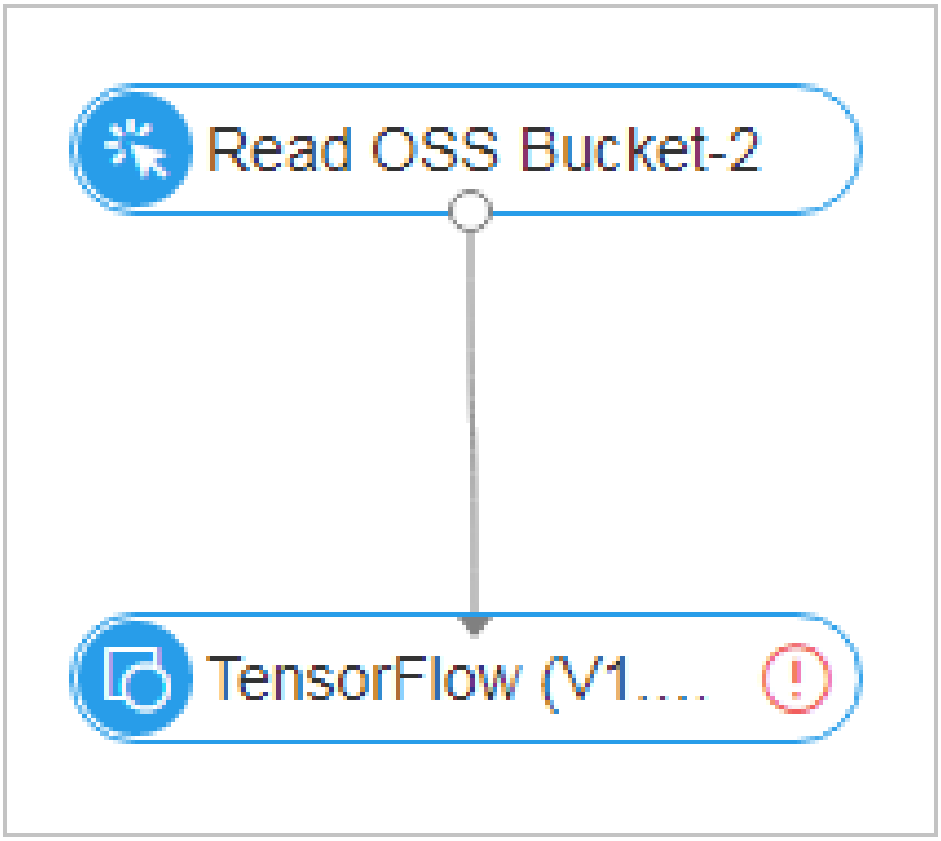
MNIST is the official tutorial provided by TF. This tutorial trains a model that scans images of

handwritten digits and predicts what digits they are.

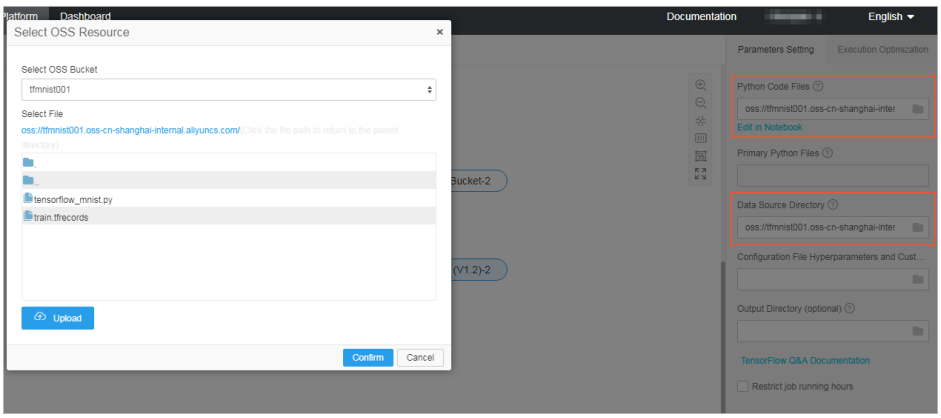
Upload Python code files and training datasets to OSS. In this case, a bucket named tfmnist is created in the China (Shanghai) region to store Python files and training datasets.



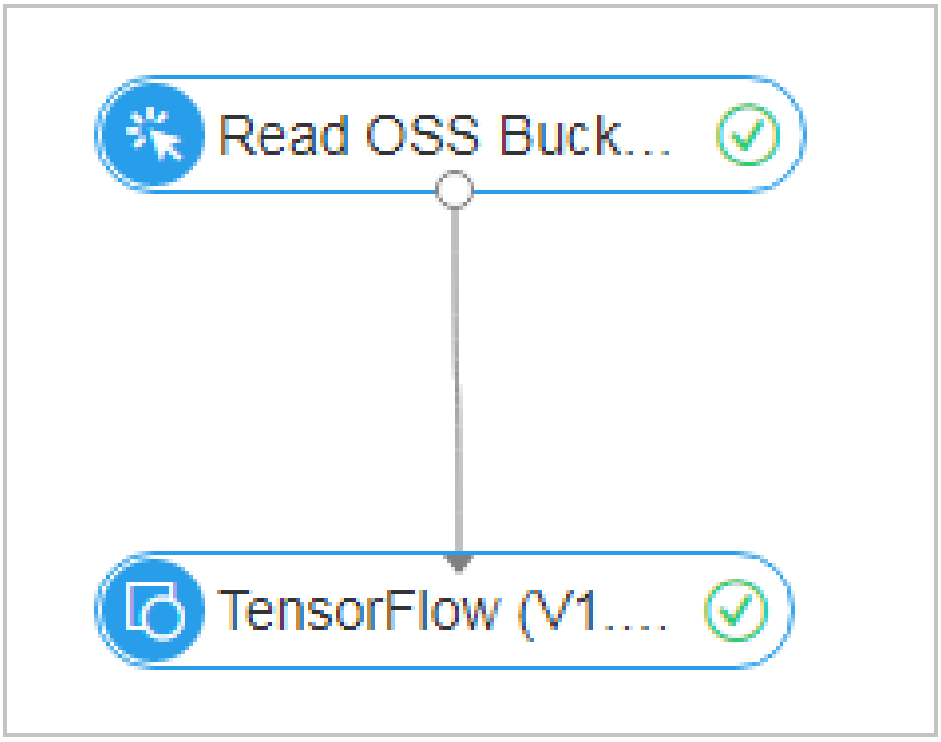
Drag a Read OSS Bucket and a TensorFlow component to the canvas and link them as follows.



Specify the parameters for the TensorFlow component as shown in the following figure.



Click **Run** and wait for the process to finish.



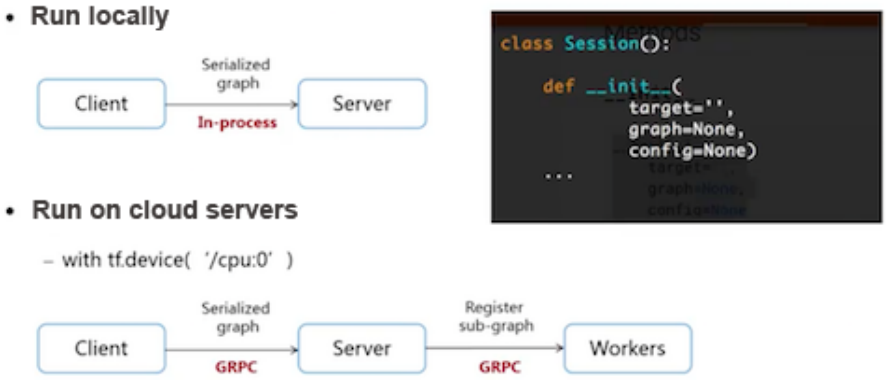
Right-click the TensorFlow component to view the runtime logs.

```
[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%]
```

Read data in TensorFlow

Inefficient I/O approaches

The major difference exists between the TensoFlow code that is run locally and on cloud servers:



Reading local data: The server directly obtains graphs from the client.

Cloud servers: After obtaining graphs, they need to distribute these graphs to workers for computing. For more information, see TensorFlow advanced article).

The following example reads data from a CVS file and demonstrates how to read data efficiently using TensorFlow. The CSV file is as follows:

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

Note the following issues:

Do not use the built-in I/O methods in Python.

The machine learning platform supports the built-in I/O methods in Python. To use these methods, you must compress the data source and code into a package and upload this package to OSS. This approach writes data to the memory for computing and is low in efficiency. The example code is as follows.

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

Do not use third-party methods for file I/O.

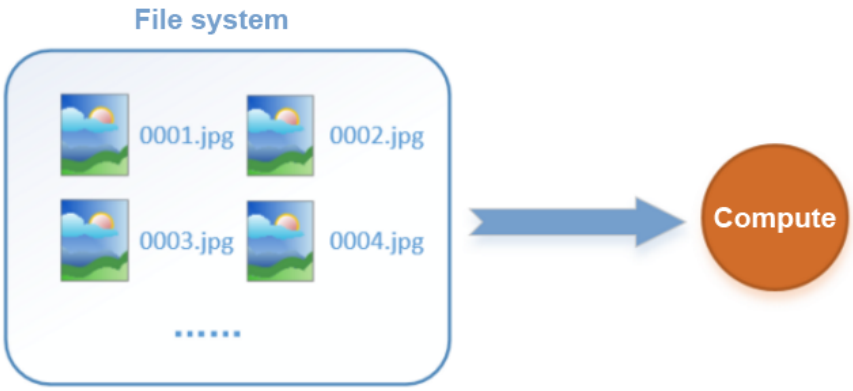
Many developers use third-party methods for file I/O. For example, both TFLearn and Panda provide file I/O methods. However, many of these methods are based on Python I/O methods and still cause inefficiencies.

Do not use preload when reading files.

You may find that the GPUs are not significantly faster than the CPUs when using the machine learning platform. The main cause is in data I/O.

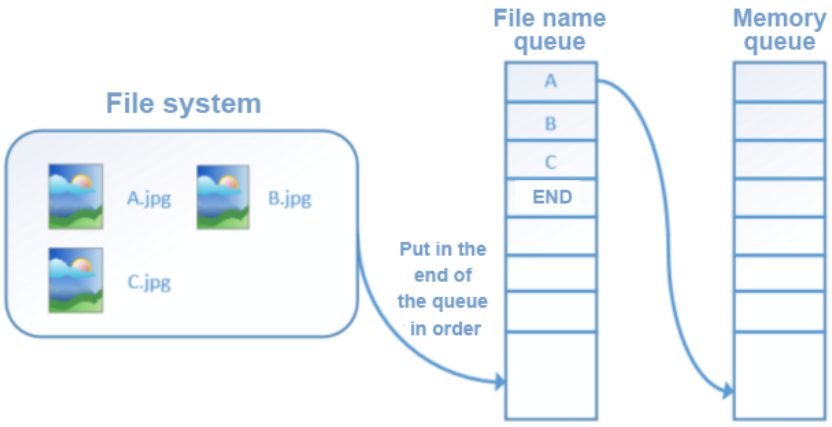
To preload data into the memory, you may use feed methods to read the data first and then use session methods for computing. This approach wastes computing resources and does not support large volumes of data due to the memory limit.

For example, if we have a hard disk that contains an image dataset, we need to load the data set into the memory and use either the GPU or the CPU for computing processes. This procedure sounds simple enough but is not that easy to implement. We need to first load the data before the computing process starts. Assume that loading one image takes 0.1s and computing takes 0.9s. For every second the GPU remains idle for 0.1s. This greatly reduces efficiency.



Efficient I/O approach

The efficient I/O approach uses TensorFlow operators to process the data and uses the **session.run** method to pull the data. One read thread continuously loads images from the file system into a memory queue, and one compute thread directly retrieves the data for computing from the memory queue. No GPU is left idle due to the data I/O.



The following code demonstrates how to read data using TensorFlow operators.

```
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: The OSS file path. This parameter can be an array type.

reader: TF provides multiple reader APIs. You can select these APIs based on your needs.

tf.train.string_input_producer: Generates a queue from the file.

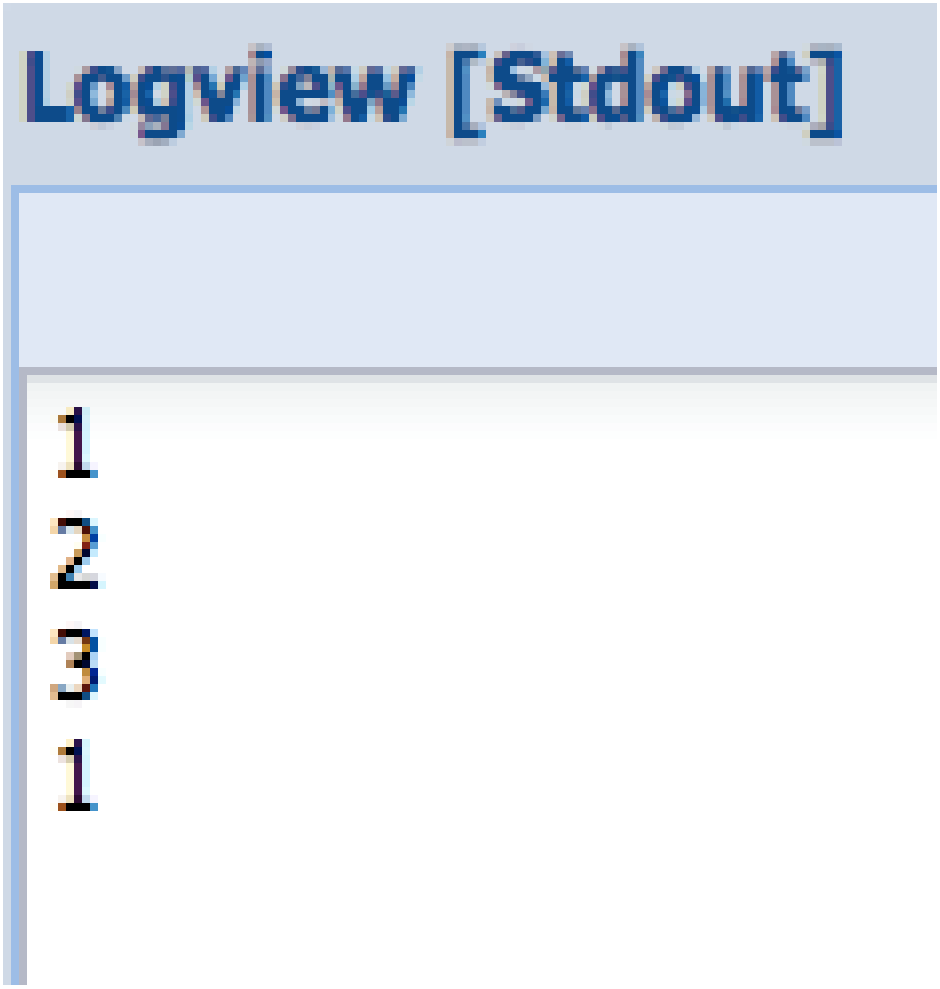
tf.decode_csv: Provides the splite feature and returns a specified parameter in each line.

To retrieve data using OP you must call tf.train.Coordinator() and tf.train.start_queue_runners(sess=sess,coord=coord) in session.

To test the code, enter the following three rows as input:

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

Run the code four times to print the second field in each row. The result is shown in the following figure.



The result indicates that the data structure is a queue.

Others

The machine learning platform has released the notebook feature, which allows you to modify your code online and provides built-in support for multiple deep learning frameworks. To start using this feature, click https://www.alibabacloud.com/product/machine-learning.

Configure multiple workers and tasks in TensorFlow

The machine learning platform currently supports multiple workers and multiple tasks in TensorFlow. This feature is only available in the China (Beijing) region. Using this feature, you can perform large scale data training. Contact us for detailed billing information.

Concepts

Parameter Server (PS) node: Stores the parameters generated during the compute process. When multiple PS nodes have been created, these parameters are automatically sliced and stored in different PS ndoes to facilitate the communication between worker nodes and PS

nodes.

Worker node: A worker node is where the GPU resides.

Task node: In TensorFlow, data is sliced and distributed on different task nodes to perform parameter training.

How to use multiple workers and tasks

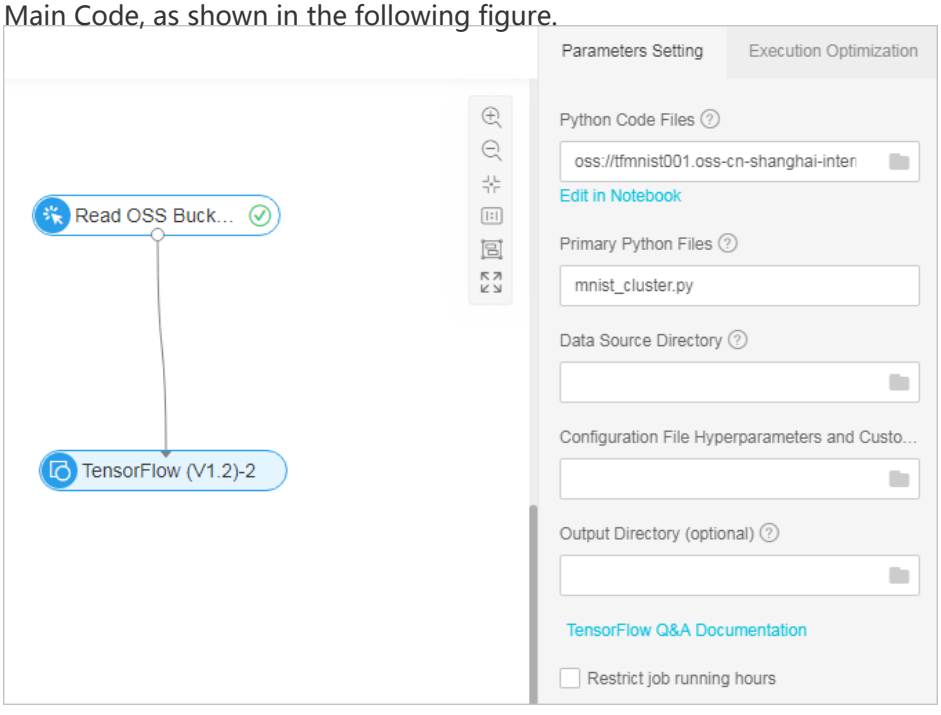
To use multiple workers and tasks in TensorFlow, you do not need to worry about resource scheduling in the backend. You can create a distributed computing network using simple configurations. For more information about this service, see the following steps.

Basic configurations.

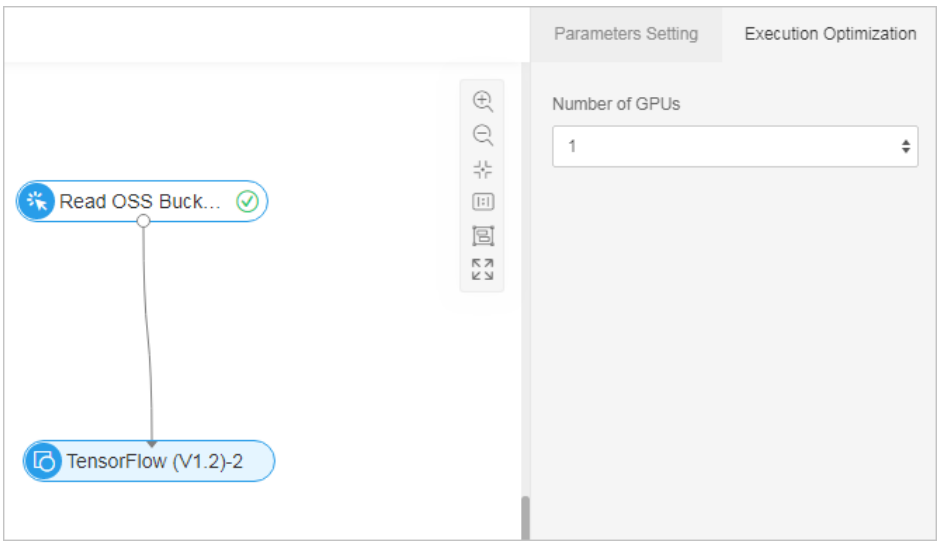
Download the **mnist_cluster.tar.gz** file and upload this file to OSS.

Configure OSS access permissions.

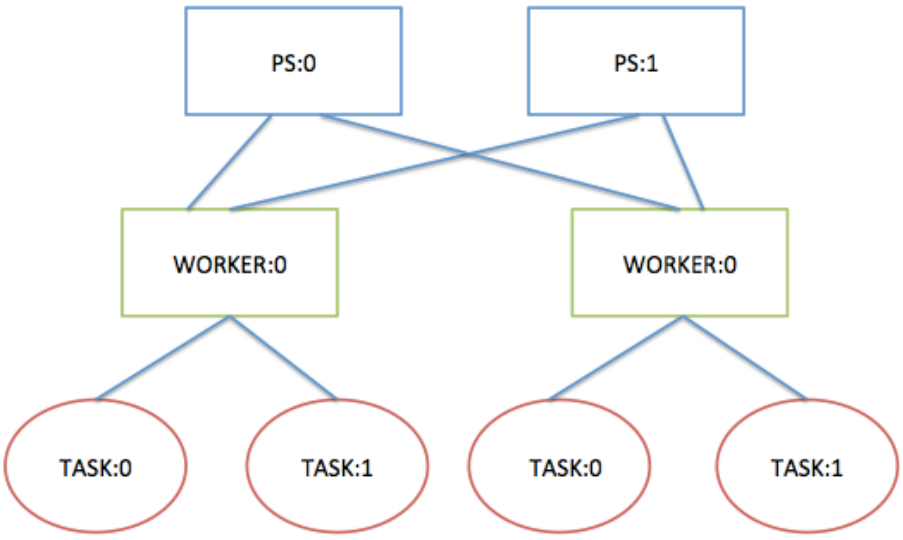
Drag a TensorFlow component and link it with a Read OSS Bucket. Select the path of **mnist_cluster.tar.gz** for Python Code File and enter **mnist_cluster.py** for Python Main Code, as shown in the following figure.



Select **Tuning** to specify the other parameters.



After completing the configuration, you have created a computing network with multiple workers and tasks, as shown in the following figure. **PS** represents the Parameter Server node, **WORKER** represents the compute node, and **TASK** represents the GPU.



Code configurations.

In traditional TF experiments, you must specify the port number of each computing node in the code, as shown in the following figure.

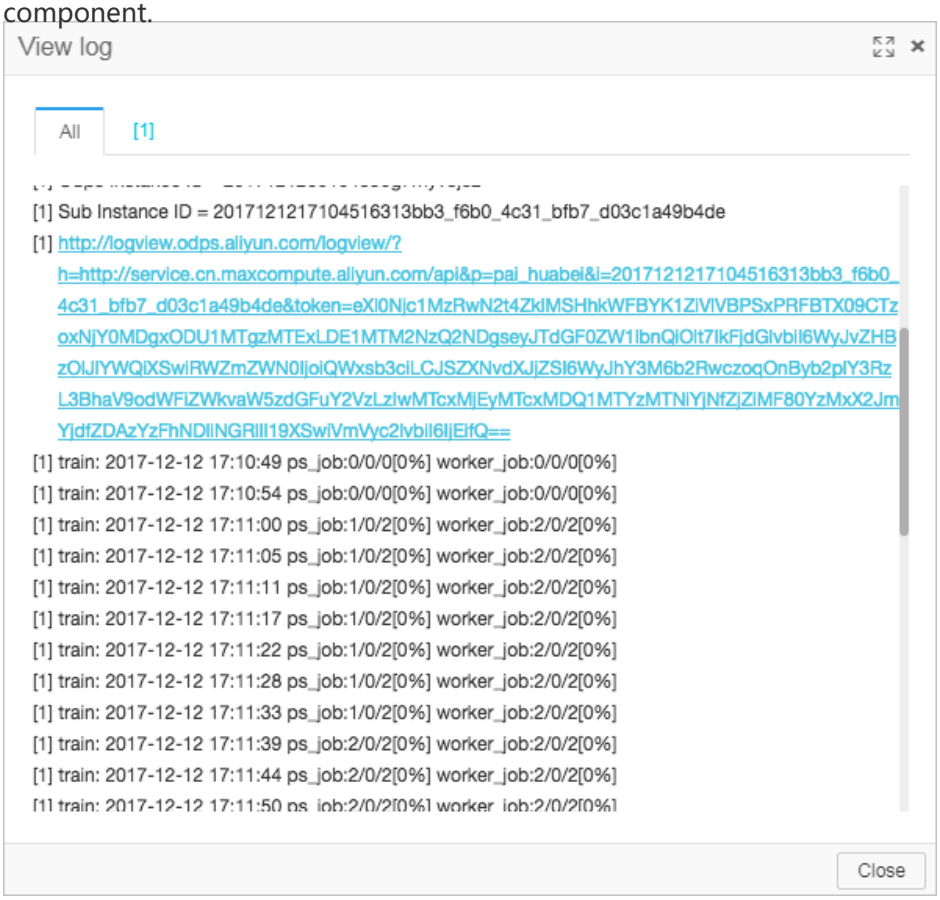
```
tf.train.ClusterSpec({                                          /job:worker/task:0
    "worker": [                                                 /job:worker/task:1
        "worker0.example.com:2222",                             /job:worker/task:2
        "worker1.example.com:2222",                             /job:ps/task:0
        "worker2.example.com:2222"                              /job:ps/task:1
    ],
    "ps": [
        "ps0.example.com:2222",
        "ps1.example.com:2222"
    ]})
```

When the number of compute nodes increases, the port configuration becomes more complex. In the machine learning platform, port configuration is much easier. You can use the following code to retrieve the port number of each compute node.

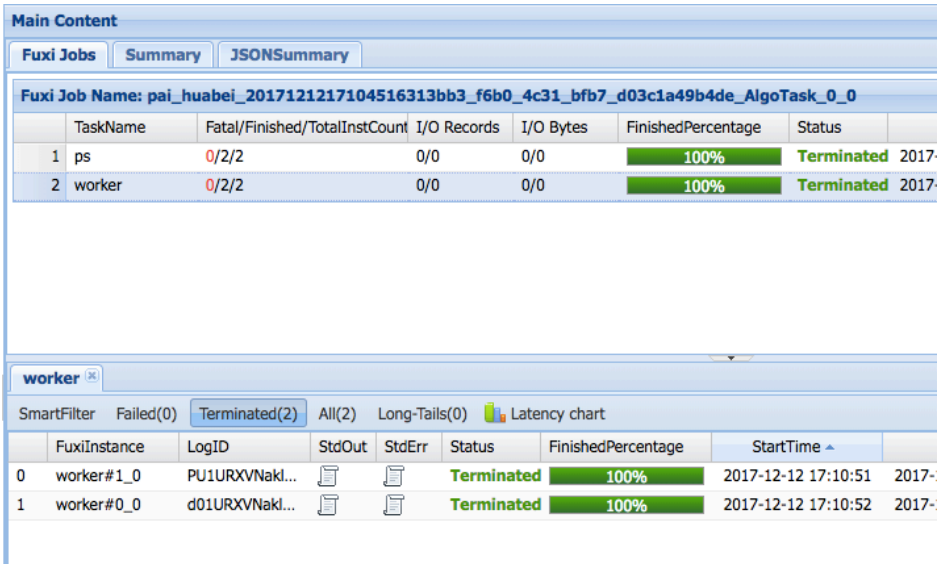
```
ps_hosts = FLAGS.ps_hosts.split(",") #Get ports of ps_hosts from the framework layer.
worker_hosts = FLAGS.worker_hosts.split(",") #Get ports of worker_hosts from the framework layer.
```

View runtime logs.

Right-click a TensorFlow component to view the logs. In the following example, two **PS** nodes and two **WORKER** nodes are allocated to this TensorFlow component.



Click the blue link to view the running conditions of a worker node in logview.



Download sample code

https://www.alibabacloud.com/help/doc-detail/107974.html

Use hyperparameters in TensorFlow

You can specify hyperparameter files in the Hyperparameters and Custom Parameters field of the parameters setting page. For example:

```
batch_size=10
learning_rate=0.01
```

You can reference a hyperparameter as follows:

```
import tensorflow as tf
tf.app.flags.DEFINE_string("learning_rate", "", "learning_rate")
tf.app.flags.DEFINE_string("batch_size", "", "batch size")
FAGS = tf.app.flags.FLAGS
print("learning rate:" + FAGS.learning_rate)
print("batch size:" + FAGS.batch_size)
```

Third-party libraries supported by TensorFlow

Third-party libraries supported by TensorFlow 1.0.0

```
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)
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)
opencv-python (3.3.0.10)

Third-party libraries supported by TensorFlow 1.1.0

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)
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)
opencv-python (3.3.0.10)

Third-party libraries supported by TensorFlow 1.2.1

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)

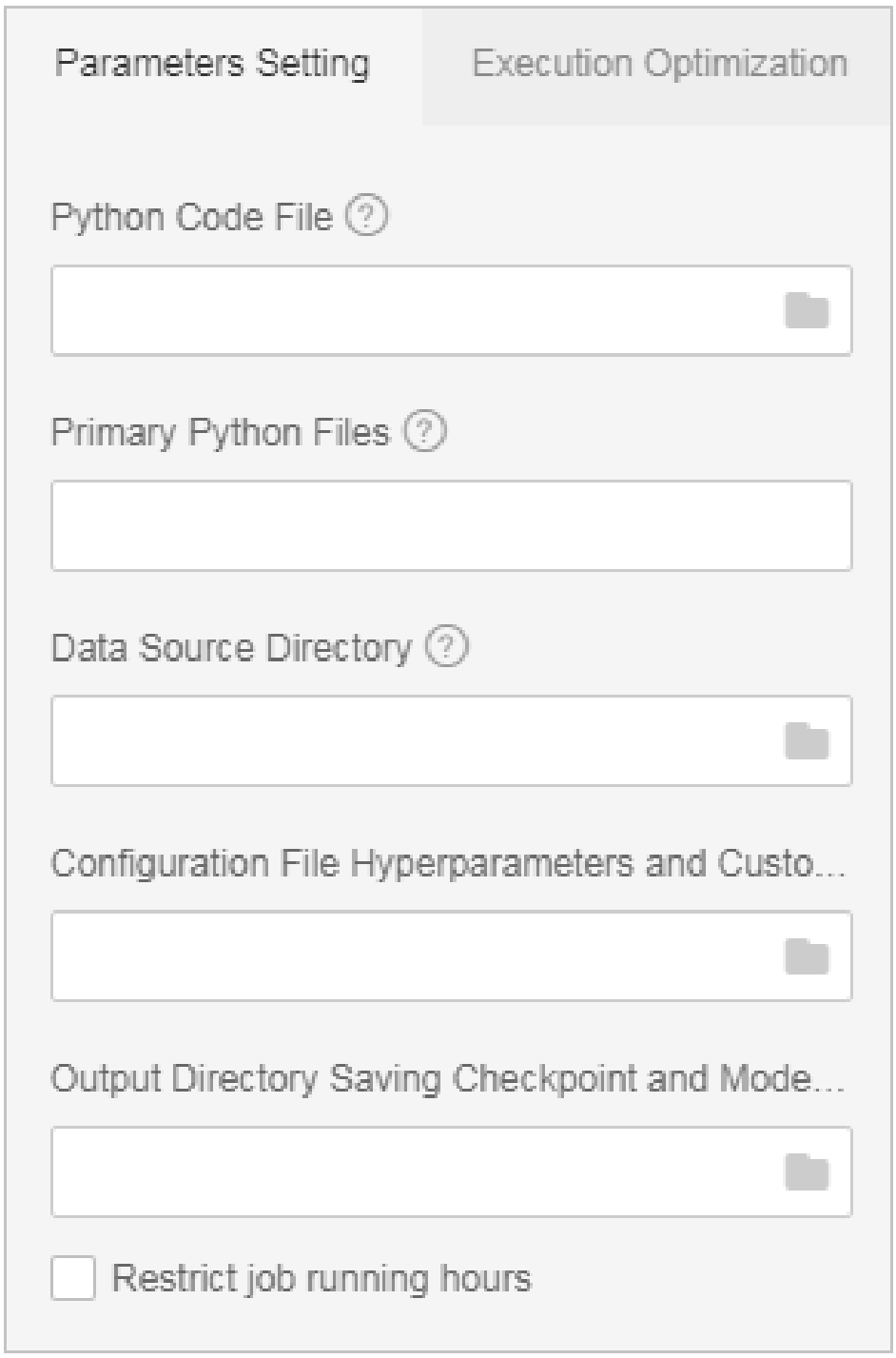
pip (9.0.1)
prompt-toolkit (1.0.15)
protobuf (3.1.0)
ptyprocess (0.5.2)
pycrypto (2.6.1)
pycurl (7.19.0)
Pygments (2.2.0)
pygobject (3.14.0)
pygpgme (0.3)
pyliblzma (0.5.3)
pyparsing (2.2.0)
python-dateutil (2.6.1)
pytz (2017.2)
PyWavelets (0.5.2)
pyxattr (0.5.1)
PyYAML (3.12)
pyzmq (16.0.2)
qtconsole (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)
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)
opencv-python (3.3.0.10)

MXNet

MXNet is a deep learning framework that supports both imperative and symbolic programming. It can be distributed on either CPU or GPU clusters. MXNet allows you to write more efficient applications than CXXNet, a distributed deep learning framework that was inspired by Minerva.

Parameters

Parameter settings



Python Code File: You can compress multiple code files into a tar.gz file.

Main Python File: Specify the main file in the compressed file. This step is optional.

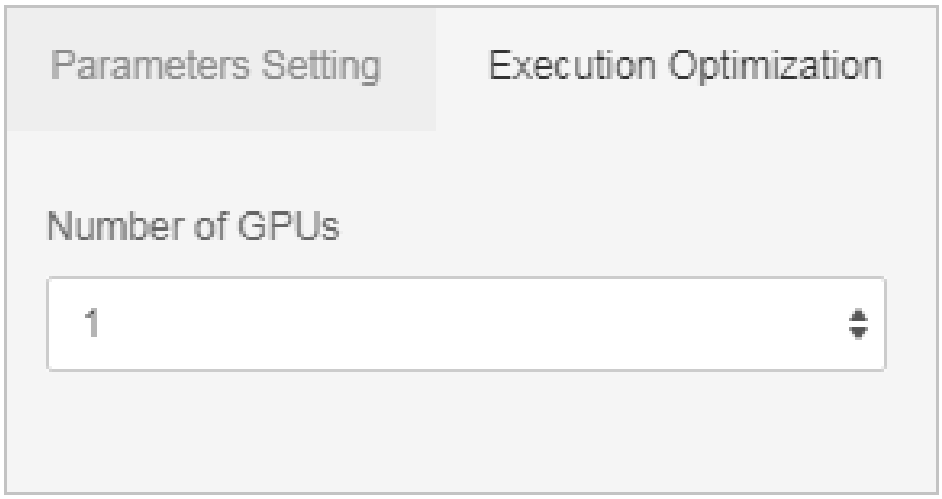
Data Source Directory: Select the OSS data source.

Hyperparameters and Custom Parameters: You can write commands to specify

hyperparameters and custom parameters. Using these parameters, you can test different learning rates and batch sizes in your models.

Output Directory: Specify the output directory of your model.

Tuning



You can specify the number of GPUs based on the complexity of your tasks.

PAI commands

Not all parameters are needed for actual use. Specify your own parameters based on your needs. For more information about these parameters, see the following table.

```
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";
```

The description of the parameters are as follows:

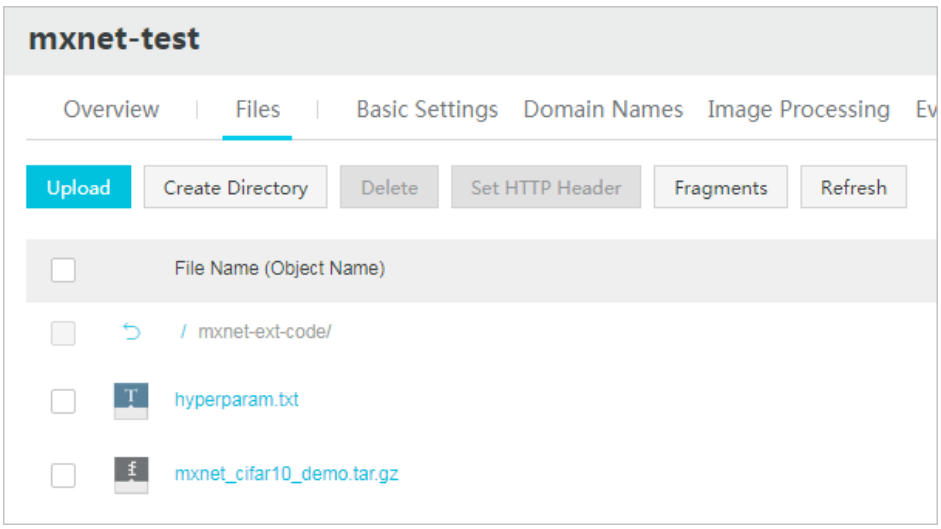
Parameter	Description	Format	Default
script	Required. The TF algorithm file. This file can be a regular Python file or a TAR.GZ file.	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_mxnet/mnist_ext.py	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_mxnet/mnist_ext.py
entryFile	Optional. The entry file for the algorithm. This	train.py	Null

	parameter is required if you specified a TAR.GZ file for the script parameter.		
buckets	Required. You can specify multiple OSS buckets by separating them with commas (,). Each bucket must end with a backslash ().	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com	Null
hyperParameters	Optional. The path of the hyperparameter file.	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/mxnet-ext-code/	Null
gpuRequired	Required. The number of GPUs that are used.	200	100
checkpointDir	Optional. The checkpoint directory.	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/mxnet-ext-code/	Null

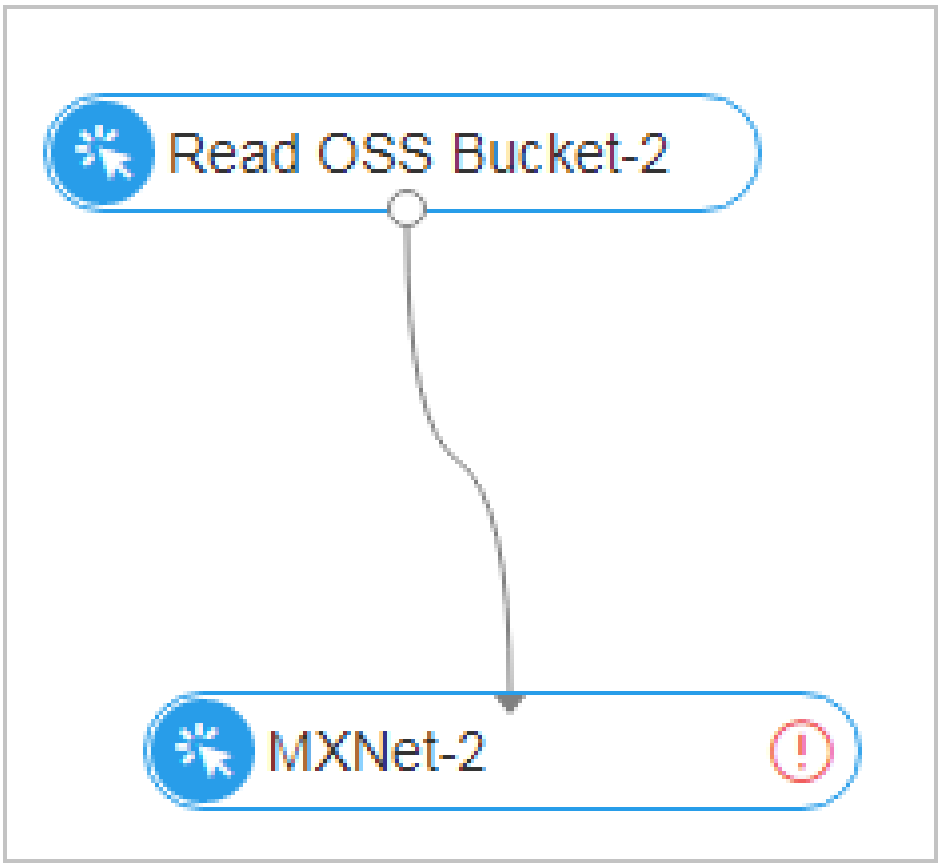
Example

The CIFAR-10 dataset contains 60,000 32x32 color images in 10 different categories. This dataset is commonly used to train machine learning algorithms to recognize objects and sort them into airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, or trucks. For more information, see: https://www.cs.toronto.edu/~kriz/cifar.html

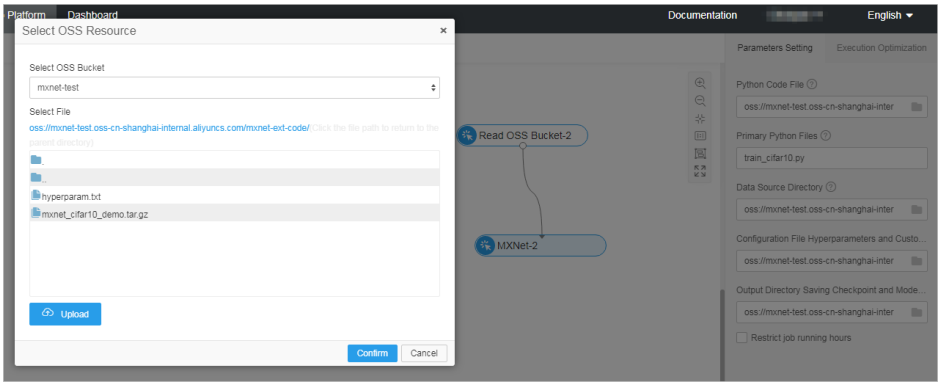
Upload Python code files and training datasets to OSS. In this case, a bucket named tfmnist is created in the China (Shanghai) region to store Python files and training datasets.



Drag a Read OSS Bucket and a MXNet component to the canvas and link them as follows.

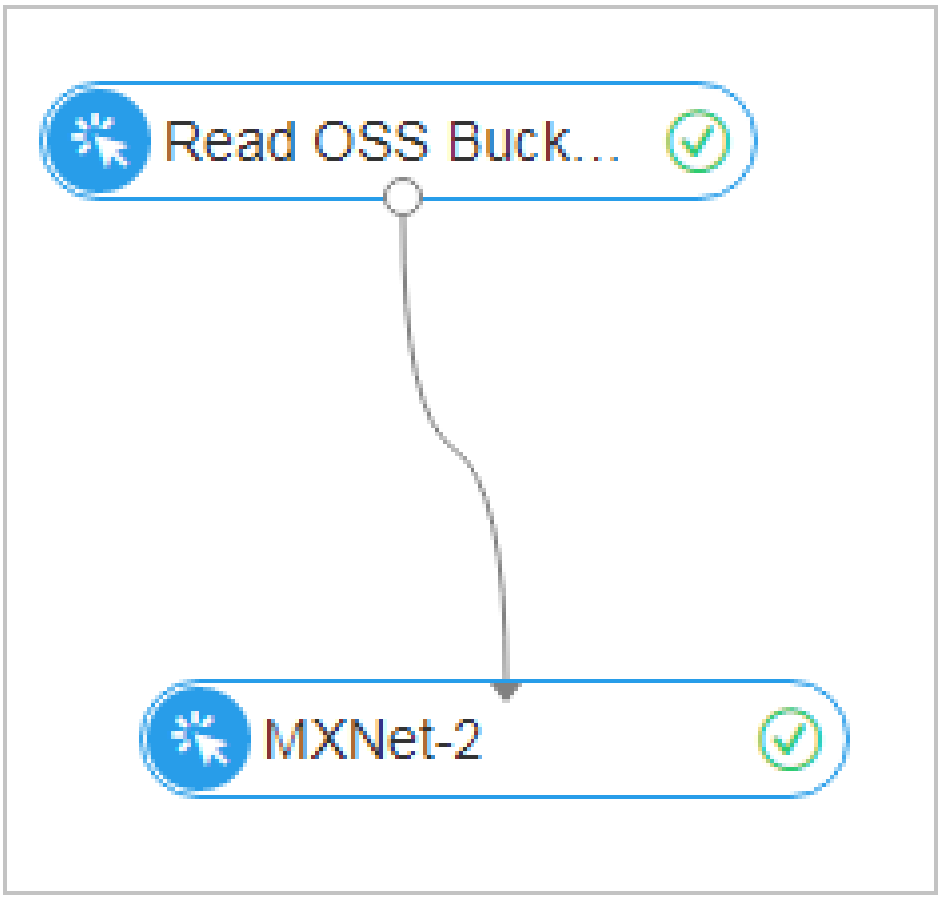


Specify the parameters for the MXNet component as shown in the following figure.



- Select a TAR.GZ file for the Python Code File.
- Specify the entry file for the algorithm as the Main Python File.
- Select a TXT file to specify hyperparameters and custom parameters.
- The checkpoint directory is the output directory of the model.

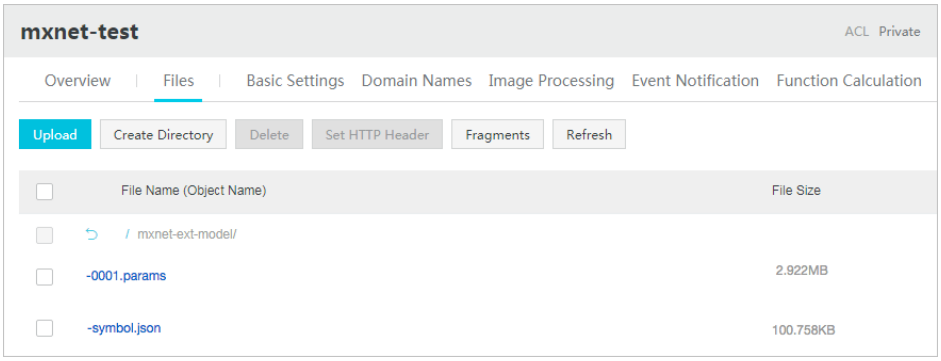
Click **Run** and wait for the process to finish.



Right-click the MXNet component to view the runtime logs.

```
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
wiRWZmZWN0IjoiQWxsb3ciLCJSZXNvdXJjZSI6WyJhY3M6b2RwczoqOnByb2plY3RzL3NoZXF1L2I
uc3RhbmNlcy8yMDE3MDIxNTEwNDk0MTJkZTNlMTM2XzljZTJfNGJjNl85MWU5XzA4OGZkZDgxND
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%]
```

The model is generated under the checkpoint directory as follows.



Format conversion

Currently, custom data file formats are not supported in PAI Caffe. You must convert your training data files to the required format by using the format conversion component.

Link the Input to a Read OSS Bucket component.

Parameters

- OSS input directory, the OSS training data in a file_list file, for example, bucket.hz.aliyun.com/train_img/train_file_list.txt. The format of file_list is as follows:

```
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
```

OSS output directory, for example, bucket_name.oss-cn-hangzhou-zmf.aliyuncs.com/ilsvrc12_val_convert. The converted data_file_list.txt file and corresponding data files are stored in this directory. The format of data_file_list is as follows:

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

- Encoding type: Optional. You can select jpg, png, or raw.
- Shuffle: The default setting is true.
- File prefix: The default value is data.
- resize_height: The default value is 256.
- resize_width: The default value is 256.
- isGray: The default setting is false.
- Generate the image mean file: The default setting is false.

PAI commands

```
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
```

Parameters

Parameter	Description	Format	Default
ossHost	The OSS host address	For example, “oss-test.aliyun-inc.com”	Optional, the default value is “oss-cn-hangzhou-zmf.aliyuncs.com”.
arn	The ARN of the default Role of the OSS bucket	For example, “acs:ram::XXXXXXXXXXXXXXXX:role/ossaccessroleforodps”; The 16-digit number in the middle	Required

		represents the RoleArn.	
ossImageList	The image file list	For example, “bucket_name/image_list.txt”	Required
ossOutputDir	The OSS output directory	For example, “bucket_name/your/dir”	Required
encodeType	The encoding type	For example: jpg, png, or raw	Optional. The default setting is jpg.
shuffle	Whether or not to shuffle the data	Boolean type	Optional. The default setting is true.
dataFilePrefix	The prefix of the data file	String type, for example, train or val	Required
resizeHeight	The image resize height	Int type	Optional. The default value is 256.
resizeWidth	The image resize width	Int type	Optional. The default value is 256.
isGray	Whether or not the image is grayscale	Boolean type	Optional. The default setting is false.
imageMeanFile	Whether or not to generate the image mean file	Boolean type	Optional. The default setting is false.

Caffe

Caffe is a lightweight, scalable, and fast deep learning framework developed by Berkeley AI Research (BAIR) and by community contributors. Yangqing Jia created the project during his PhD at UC Berkeley. Caffe is released under the BSD 2-Clause. Caffe’ s official website is http://caffe.berkeleyvision.org/.

Parameters

First, configure OSS access permissions.

You only need to specify the OSS path of the solver.prototxt file. Note the following parameters in this file:

- net: “bucket.hz.aliyun.com/alexnet/train_val.prototxt”. The OSS path of the net file.
- type: “ParallelSGD”. The type is denoted as a string value.
- model_average_iter_interval: 1. The synchronization frequency. 1 means synchronize after each running.
- snapshot_prefix: “bucket/snapshot/alexnet_train”. The OSS output directory of the model.

```
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
```

Select BinaryDataLayer for the datalayer in train_val, as shown in the following example:

```
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
}
}
```

The new data Layer is named **BinaryData**. You can also use transform param to perform image data conversion operations. The parameters are consistent with the native parameters in Caffe.

binary_data_param specifies the parameter settings for the data layer, including the following parameters.

source: the data source. The path of the data source is the same as the path that is specified in filelist. It starts with a bucket name and does not contain oss://.

num_threads: the number of concurrent threads for reading OSS data. The default value is 10. You can modify this parameter based on your needs.

PAI commands

```
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
```

Parameters

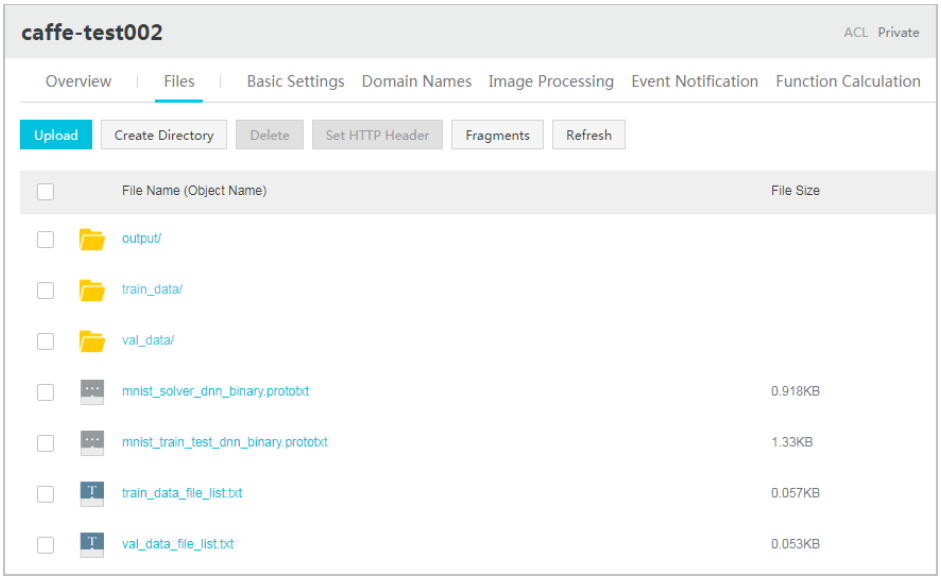
Parameter	Description	Format	Default
ossHost	The OSS host address	For example, “oss-test.aliyun-inc.com”	Optional. The default value is “oss-cn-hangzhou-zmf.aliyuncs.com”.
arn	The ARN of the default Role of the OSS bucket	For example, “acs:ram::XXXXXXXXXXXXXXXX:role/ossaccessroleforodps”. The 16-digit number in the middle represents the RoleArn.	Required
solverPrototxtFile	The solver file	The OSS path of the solver file, which starts with a bucket name	Required
gpuRequired	The number of GPUs	Int type	Optional. The default value is 1.

Example

The following example trains a model with MNIST data using Caffe.

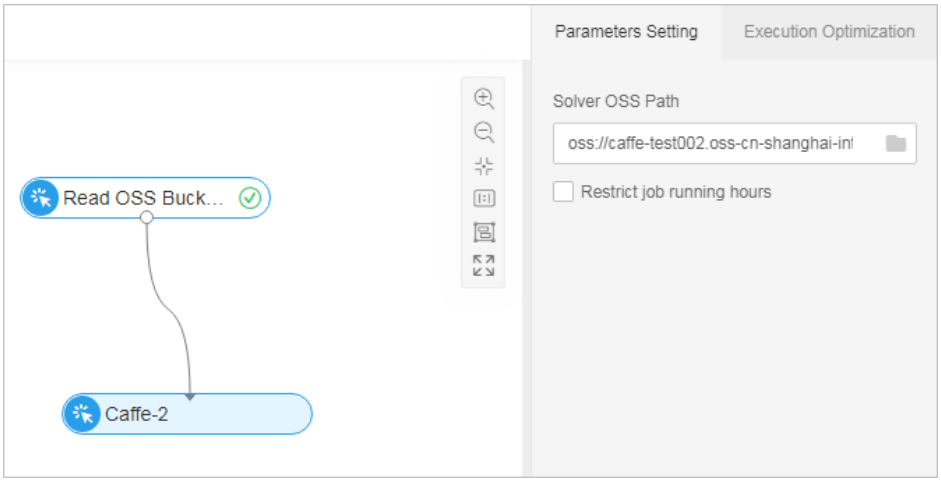
Prepare the data source.

Download Caffe data in the Deep learning example resources section and extract the data. Upload the data to OSS as follows.



Implement the experiment.

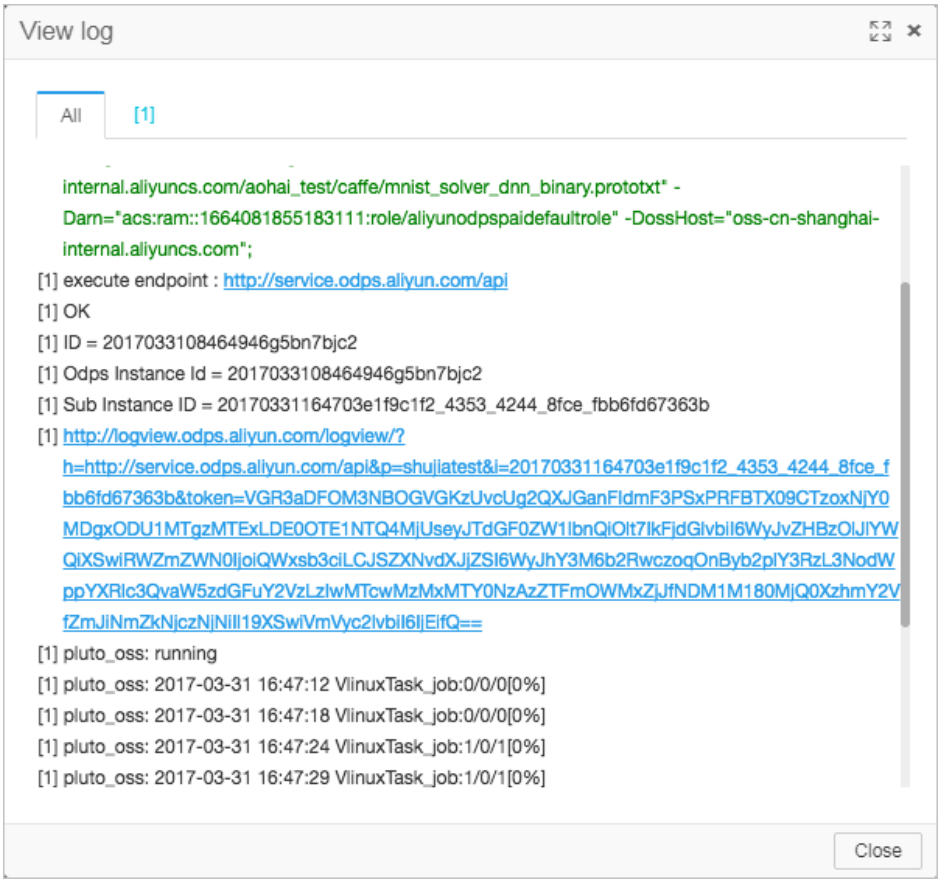
Drag a Caffe component and link it with a Read OSS Bucket component as follows.



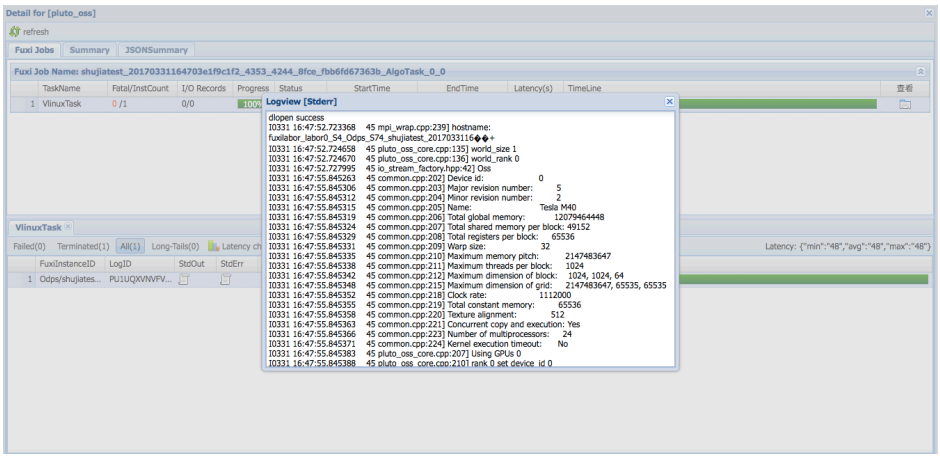
Set Solver OSS Path to the path of **mnist_solver_dnn_binary.prototxt**. Click **Run**.

View logs.

Right-click a Caffe component to view logs.



Click a **logview link** > **ODPS Tasks** > **VlinuxTask** > **StdErr** to view the logs generated during the training.



Time series

Contents

x13_arima

Autoregressive Integrated Moving Average Model (ARIMA) is a famous time series prediction method defined by Box and Jenkins in the early 1970s. Therefore, this model is also called the Box-Jenkins model or the Box-Jenkins approach.

x13-arima is an ARIMA algorithm based on the open-source X-13ARIMA-SEATS seasonal adjustment.

For more information about X-13ARIMA-SEATS Seasonal Adjustment Program, visit wiki.

For more information about ARIMA, visit wiki.

PAI command

```
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
```

Parameter description

Parameter	Description	Value range	Required/Optional, default value/act
inputTableName	Input table	Table name	Required
inputTablePartitions	Partitions used for	Partition name	(Optional) All

	training in the input table, in the format of partition_name=value. The multilevel format is name1=value1/name2=value2. Multiple partitions are separated by commas (,).		partitions are selected by default.
seqColName	Time series column	Column name	(Required) It is used only to sort valueColNames, and the value is irrelevant to the algorithm.
valueColName	Value column	Column name	Required
groupColNames	Grouping column. Multiple columns are separated by commas (,), such as col0,col1. A time series is created for each group.	Column name	Optional
order	p, d, and q respectively represent the autoregressive coefficient, difference, and moving regression coefficient.	p, d, and q are non-negative integers in the range of [0, 36].	Required
start	Time series start date	String, in the format of year.seasonal, such as 1986.1 Time series format description	Optional; default value: 1.1
frequency	Frequency of time series	Positive integer in the range of (0, 12] Time series format description	Optional; default value: 12, indicating 12 months/year
seasonal	sp, sd, and sq respectively represent the seasonal autoregressive coefficient, seasonal difference, and seasonal moving regression coefficient.	sp, sd, and sq are all non-negative integers in the range of [0, 36].	Optional; default value: not seasonal

period	Seasonal period	Number in the range of (0, 100]	Optional; default value: frequency
maxiter	Maximum number of iterations	Positive integer	Optional; default value: 1500
tol	Tolerance	Double type	Optional; default value: 1e-5
predictStep	Number of prediction items	Number in the range of (0, 365]	Optional; default value: 12
confidenceLevel	Prediction confidence level	Number in the range of (0, 1)	Optional; default: 0.95
outputPredictTableName	Prediction output table name	Table name	Required
outputDetailTableName	Detail table	Table name	Required
outputTablePartition	Partitions in the output table	Partition name	Optional, not output to the partition by default
coreNum	Number of nodes	Paired with the memSizePerCore parameter, positive integer	(Optional) Automatically calculated by default
memSizePerCore	Memory size of each node, in MB	Positive integer in the range of [1024, 64*1024]	(Optional) Automatically calculated by default
lifecycle	(Optional) Lifecycle of the output table	Positive integer	No lifecycle

Time series format

The start and frequency parameters specify the two time dimensions of data (valueColName), TS1 and TS2. The frequency parameter represents the data frequency within a cycle, namely, the frequency of TS2 in each TS1. The start parameter is in the format of n1.n2, which indicates that the start date is the N2 TS2 in the N1 TS1.

Unit time	ts1	ts2	frequency	start
12 months/Year	Year	Month	12	1949.2 represents the 2nd month of the 1949th year.
Four seasons/year	Year	Season	4	1949.2 represents the second quarter of the 1949th year.
Seven	Week	Day	7	1949.2

days/week				represents the second day of the 1949th week.
1	Any time unit	1	1	1949.1 represents the 1949th (year, day, hour, and so on).

For example, value=[1,2,3,5,6,7,8,9,10,11,12,13,14,15]

- start=1949.3, frequency=12 indicates that the data frequency is monthly per year, and the prediction start date is 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							

- start=1949.3, frequency=4 indicates that the data frequency is quarterly per year, and the prediction start date is 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			

- start=1949.3, frequency=7 indicates that the data frequency is daily per week, and the prediction start date is 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				

- start=1949.1, frequency=1 can represent any time unit, and the prediction start date is 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

Example

Test data

Used data: AirPassengers.

This data set is the number of passengers for international airlines each month from 1949 to 1960, as shown in the following table.

id	number
1	112
2	118
3	132
4	129
5	121
...	...

Upload data by tuunel. The command is as follows.

```
create table pai_ft_x13_arima_input(id bigint,number bigint);
tunnel upload data/airpassengers.csv pai_ft_x13_arima_input -h true;
```

PAI command

```
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
```

Output description

- Output table: outputPredictTableName.
  The fields are:

column name	comment
pdate	Prediction date
forecast	Prediction conclusion
lower	Lower threshold of the prediction conclusion when the confidence level is confidenceLevel (default: 0.95)
upper	Upper threshold of the prediction conclusion when the confidence level is confidenceLevel (default: 0.95)

Data display:

index	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

Output table: outputDetailTableName.
The fields are:

column name	comment
key	model: indicates the model. evaluation: indicates the evaluation result. parameters: indicates training parameters. log: indicates the training log.
summary	Storage details

Data display:

index	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...

Display on the PAI web - Model factor (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

Display on the PAI web - Evaluation index (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

Algorithm scale

Supported scale

Rows: a maximum of 1200 rows in a single group

Columns: 1 data column

Resource calculation method

Default calculation method used when groupColNames is not set
coreNum = 1
memSizePerCore = 4096

Default calculation method used when groupColNames is set
coreNum = floor (total number of data rows / 120,000)
memSizePerCore = 4096

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.

The x13-auto-arima selection process is as follows:

Default model estimation.

- When frequency is 1, the default model is (0,1,1).
- When frequency is greater than 1, the default model is (0,1,1)(0,1,1).

Identication of dierencing orders.

- Skip this step if you set diff and seasonalDiff.
- Determine the difference d and the seasonal difference D by using the Unit root test (wiki).

Identication of ARMA model orders.

Select the optimal model based on BIC(wiki), and the maxOrder and maxSeasonalOrder parameters are used in this step.

Comparison of identified model with default model.

Compare models by using Ljung-Box Q statistic(wiki). If both models are unacceptable, use the (3,d,1)(0,D,1) model.

Final model checks.

PAI command line

```
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
```

Parameter description

Parameter	Description	Value range	Required/Optional, default value/act
inputTableName	Input table	Table name	Required
inputTablePartitions	Partitions used for training in the input table, in the format of partition_name=value. The multilevel format is name1=value1/name2=value2. Multiple partitions are separated by commas (,).	Partition name	(Optional) All partitions are selected by default.
seqColName	Time series column	Column name	(Required) It is used only to sort valueColNames, and the value is irrelevant to the algorithm.

valueColName	Value column	Column name	Required
groupColNames	Grouping column. Multiple columns are separated by commas (,), such as col0,col1. A time series is created for each group.	Column name	Optional
start	Time series start date	String, in the format of year.seasonal, such as 1986.1 Time series format description	Optional; default value: 1.1
frequency	Frequency of time series	Positive integer in the range of (0, 12] Time series format description	Optional; default value: 12, indicating 12 months/year
maxOrder	Maximum values of p and q	Non-negative integer in the range of [0, 4]	Optional; default value: 2
maxSeasonalOrder	Maximum values of seasonal p and q	Non-negative integer in the range of [0, 2]	Optional; default value: 1
maxDiff	Maximum value of differential d	Non-negative integer in the range of [0, 2]	Optional; default value: 2
maxSeasonalDiff	Maximum value of seasonal differential d	Non-negative integer in the range of [0,1]	Optional; default value: 1
diff	Differential d	Non-negative integer in the range of [0, 2] When both diff and maxDiff are set, maxDiff is ignored. diff and seasonalDiff must be both set.	Optional; default value: -1; no diff specified
seasonalDiff	Seasonal differential d	Non-negative integer in the range of [0, 1] When both seasonalDiff and maxSeasonalDiff are set, maxSeasonalDiff is ignored.	Optional; default value: -1; no seasonalDiff specified
maxiter	Maximum number of iterations	Positive integer	Optional; default value: 1500
tol	Tolerance	Double type	Optional; default value: 1e-5

predictStep	Number of prediction items	Number in the range of (0, 365]	Optional; default value: 12
confidenceLevel	Prediction confidence level	Number in the range of (0, 1)	Optional; default: 0.95
outputPredictTableName	Prediction output table name	Table name	Required
outputDetailTableName	Detail table	Table name	Required
outputTablePartition	Partitions in the output table	Partition name	Optional, not output to the partition by default
coreNum	Number of nodes	Paired with the memSizePerCore parameter, positive integer	(Optional) Automatically calculated by default
memSizePerCore	Memory size of each node, in MB	Positive integer in the range of [1024, 64*1024]	(Optional) Automatically calculated by default
lifecycle	(Optional) Lifecycle of the output table	Positive integer	No lifecycle

Time series format

The start and frequency parameters specify the two time dimensions of data (valueColName), TS1 and TS2.The frequency parameter represents the data frequency within a cycle, namely, the frequency of TS2 in each TS1. The start parameter is in the format of n1.n2, which indicates that the start date is the N2 TS2 in the N1 TS1.

Unit time	ts1	ts2	frequency	start
12 months/Year	Year	Month	12	1949.2 represents the 2nd month of the 1949th year.
Four seasons/year	Year	Season	4	1949.2 represents the second quarter of the 1949th year.
Seven days/week	Day	Week	7	1949.2 represents the second day of the 1949th week.
1	Any time unit	1	1	1949.1 represents the 1949th (year,

				day, hour, and so on).

For example, value=[1,2,3,5,6,7,8,9,10,11,12,13,14,15]

- start=1949.3, frequency=12 indicates that the data frequency is monthly per year, and the prediction start date is 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							

- start=1949.3, frequency=4 indicates that the data frequency is quarterly per year, and the prediction start date is 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			

- start=1949.3, frequency=7 indicates that the data frequency is daily per week, and the prediction start date is 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				

- start=1949.1, frequency=1 can represent any time unit, and the prediction start date is 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

Example

Test data

Used data: AirPassengers

This data set is the number of passengers for international airlines each month from 1949 to 1960, as shown in the following table.

id	number
1	112
2	118
3	132
4	129
5	121
...	...

Upload data by tuunel. The command is as follows.

```
create table pai_ft_x13_arima_input(id bigint,number bigint);
tunnel upload data/airpassengers.csv pai_ft_x13_arima_input -h true;
```

PAI command

```
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
```

Output description

- Output table: outputPredictTableName.
  The fields are:

column name	comment
pdate	Prediction date
forecast	Prediction conclusion
lower	Lower threshold of the prediction conclusion when the confidence level is confidenceLevel (default: 0.95)
upper	Upper threshold of the prediction conclusion when the confidence level is confidenceLevel (default: 0.95)

Data display:

index	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

- Output table: outputDetailTableName.

The fields are:

column name	comment
key	model: indicates the model. evaluation: indicates the evaluation result. parameters: indicates training parameters. log: indicates the training log.
summary	Storage details

Data display:

index	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...

Display on the PAI web - Model factor (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

Display on the PAI web - Evaluation index (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

Algorithm scale

Supported scale

Rows: a maximum of 1200 rows in a single group

Columns: 1 data column

Resource calculation method

Default calculation method used when groupColNames is not set
coreNum = 1
memSizePerCore = 4096

Default calculation method used when groupColNames is set
coreNum = floor (total number of data rows / 120,000)
memSizePerCore = 4096

Important notes

Why are the prediction results the same?

When an exception occurs during model training, the mean model is called, and all prediction results are the mean of the training data.

Common exceptions include “Not stationary after timing differential diff”, “training does not converge”, “variance is 0”. You can view the stderr file of individual nodes in the logview to obtain exception details.

Text analysis

Contents

Deprecated word filtering

Text summarization

Document similarity

Sentence splitting

Conditional random field

Keyword extraction

ngram-count

Semantic vector distance

Conditional random field

PMI

Word frequency statistics

Function overview

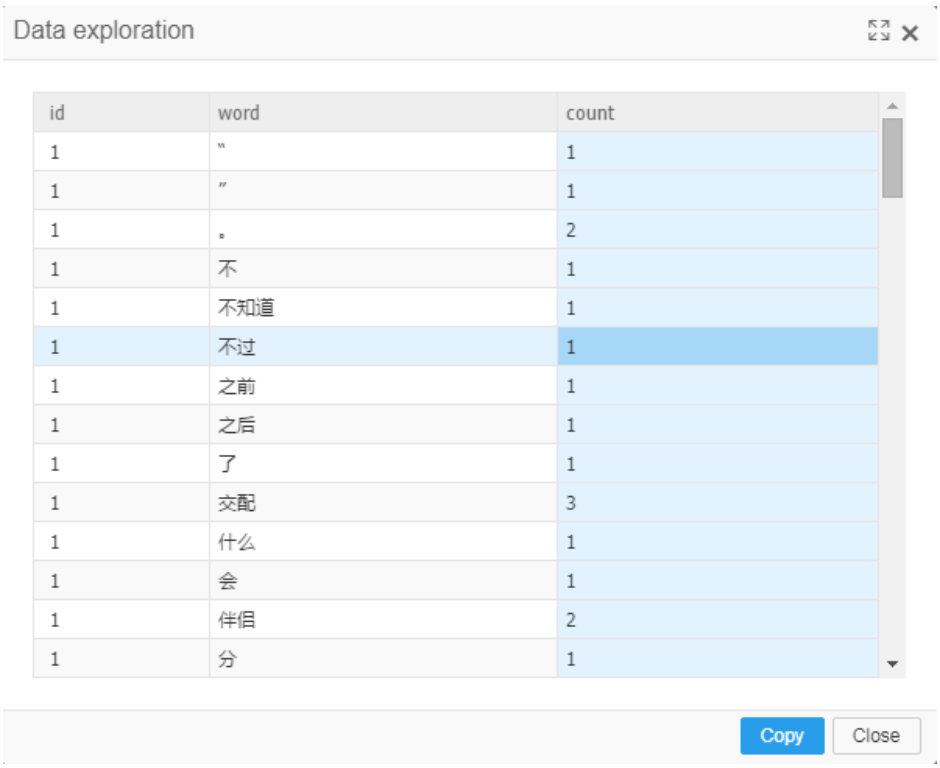
Based on the word splitting result of the document, this component outputs the words in the document in order and counts word frequency in the document content (docContent) specified by the document ID column (docId).

Parameter settings

Input parameters: the docId column and docContent column generated by the word splitting component.

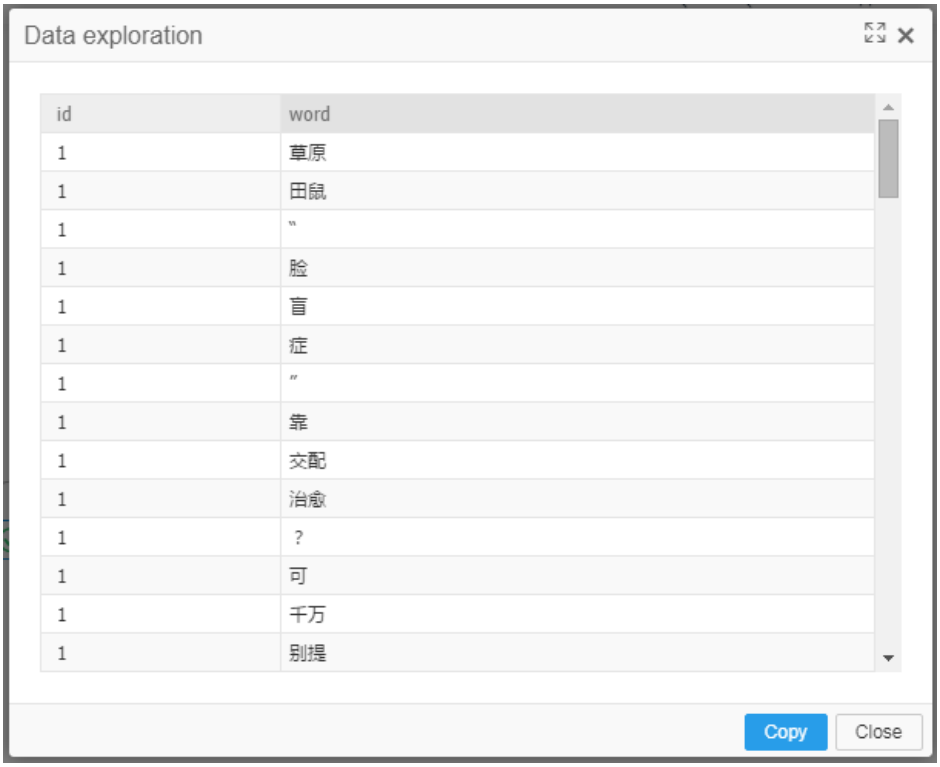
Two output parameters:

The first output end: The output table contains the “id”, “word”, and “count” fields, as shown in the following figure.



count: indicates the number of times a word appears in every document.

The second output end: The output table contains the "id" and "word" fields, as shown in the following figure:



The output table at this output end lists words in order of appearance in the document, but does not count the word frequency. Therefore, there may be multiple records for a word in the same

document. The package output table format is intended to realize compatibility with the Word2Vec component.

Example

In the example that uses Alibaba Cloud Word Splitting, the two columns in the output table are used as the input parameters for word frequency statistics.Select the docId column - id; select the docContent column - textThe statistical results on word frequency are displayed in the first figure showing the output parameters in this component.

PAI command

```
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"
```

Algorithm parameters

Parameter key	Description	Option	Default value
inputTableName	Name of the input table	-	-
docId	Name of the docId column	Only one column can be specified.	-
docContent	Name of the docContent column	Only one column can be specified.	-
outputTableNameMulti	Name of the output table listing words in order	-	-
outputTableNameTriple	Name of the output table listing word frequency statistics	-	-
inputTablePartitions	Partitions used for word splitting in the input table, in the format of partition_name=value. The multilevel format is name1=value1/name2=value2. Multiple partitions are separated by commas (,).	-	All partitions in the input table

Note: The table specified by outputTableNameMulti lists the docIds and the words in order of appearance in the document, which are obtained by splitting the document content (displayed in the docContent column) corresponding to the docId column. The table specified by outputTableNameTriple lists the docIds, words, and the number of times each word appears in the document, which are obtained by splitting the document content (displayed in the docContent column) corresponding to the docId column.

TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF) is a commonly used weighting technique for information retrieval and text mining. TF-IDF is a statistical method for assessing the importance of a word for a document in a collection or corpus.

The importance of a word increases proportionally to the number of times it appears in the document and is offset by the frequency of the word in the corpus. Variations of the TF-IDF weighting scheme are often used by search engines as a tool in scoring and ranking a document’ s relevance to a user query.

For details, see [tf-idf in Wikipedia].

The TF-IDF component is used to calculate the tf-idf value of each word that appears in a collection of documents based on word frequency statistics.

Parameter settings

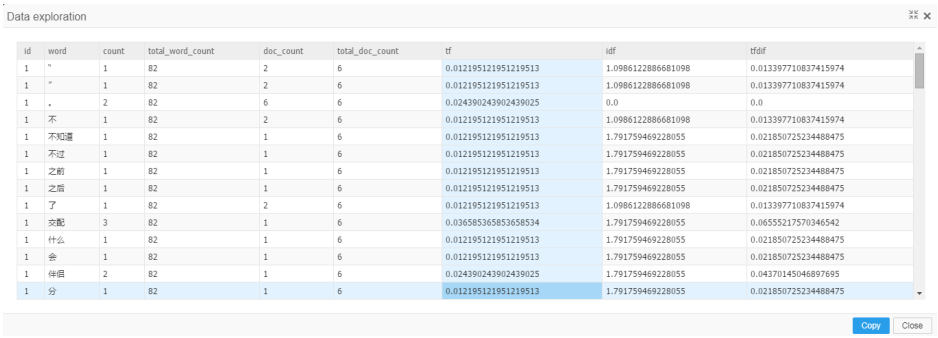
omitted

Example

The output table in the example of the word frequency statistics component is used as the input table for the TF-IDF component. The corresponding parameter settings are as follows:

- Select the docId column: id
- Select the word column: word
- Select the word count column: count

The output table has nine columns: docid, word, word_count (the number of times the current word appears in the current document), total_word_count (total number of words in the current document), doc_count (total number of documents that contain the current word), total_doc_count (total number of documents), tf, idf, and tfidf.
The result is as follows:



PAI command

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

Algorithm parameters

Parameter key	Description	Required/Optional	Default value
inputTableName	Name of the input table	Required	-
inputTablePartitions	Partition in the input table	Optional	All partitions in the input table
docIdCol	Name of the column listing document IDs. Only one column can be specified.	Required	-
wordCol	Name of the word column. Only one column can be specified.	Required	-
countCol	Name of the count column. Only one column can be specified.	Required	-
outputTableName	Name of the output table	Required	-
lifecycle	Life cycle of the output table, in the unit of days	Optional	No life cycle restriction
coreNum	Number of cores, which must be set together with	Optional	Automatically calculated

	memSizePerCore		
memSizePerCore	Size of memory, which must be set together with coreNum	Optional	Automatically calculated

PLDA

A topic model returns the topic corresponding to a document.

Latent Dirichlet Allocation (LDA) is a topic model that provides the topic of each document in a document set by probability distribution. In addition, LDA is an unsupervised learning algorithm that does not require a manually tagged training set during training. Instead, it only requires the document set and the number of topics k.LDA was first proposed by David M. Blei, Andrew Y. Ng, and Michael I. Jordan in 2003. It is now applicable to the text mining field including text topic recognition, text classification, and text similarity calculation.

Parameter settings

- Number of topics: Number of topics output by LDA.
- Alpha: Prior Dirichlet distribution parameter of P(z/d).
- beta: Prior Dirichlet distribution parameter of P(w/z).
- burn In: Number of burn in iterations, which must be less than the total number of iterations. Default value is 100.
- Total number of iterations: Positive integer, Optional, default value is 150.
- Note: z indicates the topic, w indicates the word, and d indicates the document.

Input/Output settings

Input: Data must be in format of sparse matrix. For more information about the format, see the data format description section. Currently, you have to write a MR for data conversion.
The input format is as follows:

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

The first column is docid, the second column is word and word frequency data kv.

The following are output in order:

Output column 1: Word frequency, indicating the number of times the word appears in the topic after internal sampling by using the algorithm.

Output column 2: P (word | topic), indicating the probability of the word in a topic.

Output column 3: P (topic | word), indicating the probability of the word corresponding to a topic.

Output column 4: P (document | topic), indicating the probability of the topic corresponding to a document.

Output column 5: P (topic | document), indicating the probability of the document corresponding to a topic.

Output column 6: P (topic), indicating the probability of each topic and its weight in the whole document.

The output format of the topic-word frequency contribution table is as follows:

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 command

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

Algorithm parameters

Parameter key	Description	Value range	Required/Optional, default value/act
inputTableName	Name of the input table	Table name	Required

inputTablePartitions	Partitions used for word splitting in the input table	Format: partition_name=value. The multilevel format is name1=value1/name2=value2. Multiple partitions are separated by commas.	Optional; default: all partitions in the input table
selectedColNames	Names of the columns used for LDA in the input table	Column name, separated by commas (,)	Optional, default: all columns in the input table
topicNum	Number of topics	[2, 500]	Required
kvDelimiter	Delimiter between the key and the value	Space, comma, and colon	Optional, default: colon
itemDelimiter	Delimiter between keys	Space, comma, and colon	Optional, default: space
alpha	Prior Dirichlet distribution parameter of P(z/d)	(0, ∞)	Optional, default: 0.1
beta	Prior Dirichlet distribution parameter of P(w/z)	(0, ∞)	Optional, default: 0.01
topicWordTableName	topic-word frequency contribution table	Table name	Required
pwzTableName	P(w/z) output table	Table name	Optional, default act: no P(w/z) output table
pzwTableName	P(z/w) output table	Table name	Optional, default act: no P(z/w) output table
pdzTableName	P(d/z) output table	Table name	Optional, default act: no P(d/z) output table
pzdTableName	P(z/d) output table	Table name	Optional, default act: no P(z/d) output table
pzTableName	P(z) output table	Table name	Optional, default act: no P(z) output table
burnInIterations	Number of burn in iterations	Positive integer	Optional; its value must be less than the value of totalIterations; default: 100
totalIterations	Total number of	Positive integer	Optional, default:

	iterations		150

Note: z indicates the topic, w indicates the word, and d indicates the document.

Word2Vec

Function overview

Word2Vec is an algorithm open sourced by Google in 2013, which is used to convert words into vectors. Using neural networks, Word2Vec maps words to K-dimensional space vectors through training, or even maps word vectors to semantics. Word2Vec has been favored by many users for its simplicity and efficiency.

For Google Word2Vec toolkit, visit the website https://code.google.com/p/word2vec/.

Parameter settings

Algorithm parameters:

- Word feature dimension: A value ranging from 0 to 1000 is recommended.
- Downward sampling threshold: The value 1e-3~1e-5 is recommended.
- Input: Word column and vocabulary.
- Output: Output word vector table and vocabulary.

PAI command

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

Algorithm parameters

Parameter key	Description	Value range	Required/Optional, default value/act
inputTableName	Name of the input table	Table name	Required
inputTablePartitions	Partitions used for word splitting in the input table	Format: partition_name=value.The multilevel format is name1=value1/name2=value2. Multiple partitions are	Optional; default: all partitions in the input table

		separated by commas.	
wordColName	Name of the word column. Each row in the word column is a word, and the linefeed in the corpus is represented by .	Column name	Required
inVocabularyTableName	Input vocabulary, which is the wordcount output of inputTableName	Table name	Optional, default act: The program performs wordcount on the output table.
inVocabularyPartitions	Partitions of the input vocabulary	Partition name	Optional, default: all partitions in the corresponding table of inVocabularyTableName
layerSize	Word feature dimension	0 to 1000	Optional, default: 100
cbow	Language model	1: continuous bag-of-words (CBOW) model; 0: skip-gram model	Optional, default: 0
window	Size of word window	Positive integer	Optional, default: 5
minCount	Minimum word truncation frequency	Positive integer	Optional, default: 5
hs	Use of hierarchical softmax	1: to use hierarchical softmax; 0: not to use hierarchical softmax	Optional, default: 1
negative	Negative sampling	0: negative sampling unavailable; recommended value range: 5 to 10	Optional, default: 0
sample	downward sampling threshold	0 or smaller values: downward sampling unavailable; recommended value: 1e-3-1e-5	Optional, default: 0
alpha	Initial learning rate	Greater than 0	Optional, default: 0.025
iterTrain	Number of training iterations	greater than or equal to 1	Optional, default: 1
randomWindow	Random size of window	1: the window has a random size ranging from 1 to 5; 0: the	Optional, default: 1

		window size is determined by the window parameter	
outVocabularyTableName	Output vocabulary	Table name	Optional, default act: no 'output vocabulary' output
outVocabularyPartition	Partitions of the output vocabulary	Partition name	Optional, default act: The output vocabulary is not partitioned.
outputTableName	Name of the output table	Table name	Required
outputPartition	Output table partition information	Partition name	Optional, default act: The output table is not partitioned.

Doc2Vec

Function overview

Doc2Vec can map a document to a vector.

Parameter settings

Algorithm parameters:

- Word feature dimension: A value ranging from 0 to 1000 is recommended.
- Downward sampling threshold: The value 1e-3-1e-5 is recommended.
- Input: Word column and vocabulary.
- Output: Output document vector table, word vector table, and word table.

PAI command

```
PAI -name pai_doc2vec
-project algo_public
-DinputTableName=d2v_input
-DdocIdColName=docid
-DdocColName=text_seg
-DoutputWordTableName=d2v_word_output
-DoutputDocTableName=d2v_doc_output;;
```

Algorithm parameters

Parameter key	Description	Value range	Required/Optional, default value/act
inputTableName	Name of the input table	Table name	Required

inputTablePartitions	Partitions used for word splitting in the input table	Format: partition_name=value.The multilevel format is name1=value1/name2=value2. Multiple partitions are separated by commas.	Optional; default: all partitions in the input table
docIdColName	Name of the docId column	Column name	Required
docColName	docContent, which can be the space-separated word splitting results	Column name	Required
layerSize	Word feature dimension	0 to 1000	Optional, default: 100
cbow	Language model	1: continuous bag-of-words (CBOW) model; 0: skip-gram model	Optional, default: 0
window	Size of word window	Positive integer	Optional, default: 5
minCount	Minimum word truncation frequency	Positive integer	Optional, default: 5
hs	Use of hierarchical softmax	1: to use hierarchical softmax; 0: not to use hierarchical softmax	Optional, default: 1
negative	Negative sampling	0: negative sampling unavailable; recommended value range: 5 to 10	Optional, default: 0
sample	downward sampling threshold	0 or smaller values: downward sampling unavailable; recommended value: 1e-3-1e-5	Optional, default: 0
alpha	Initial learning rate	Greater than 0	Optional, default: 0.025
iterTrain	Number of training iterations	greater than or equal to 1	Optional, default: 1
randomWindow	Random size of window	1: the window has a random size ranging from 1 to 5; 0: the window size is determined by the window parameter	Optional, default: 1
outVocabularyTable	Output vocabulary	Table name	Optional, default act:

Name			no 'output vocabulary' output
outputWordTableName	Output word vector table	Table name	Required
outputDocTableName	Output document vector table	Table name	Required
lifecycle	Life cycle of the output table	-	Optional, default: no life cycle
coreNum	Number of cores, which must be set together with memSizePerCore	-	Automatically calculated
memSizePerCore	Size of memory, which must be set together with coreNum	-	Automatically calculated

SplitWord

Based on Alibaba Word Segmenter (AliWS), this component performs word splitting on documents specified by columns. Segmented words are separated by spaces. If users have set part-of-speech tagging or semantic tagging parameters, the component outputs word splitting results, part-of-speech tagging results, and semantic tagging results. Forward slashes (/) are used as delimiters for part-of-speech tagging. Vertical bars (|) are used as delimiters for semantic tagging. Currently, only Chinese Taobao library and Internet library are supported.

Function overview

Field settings

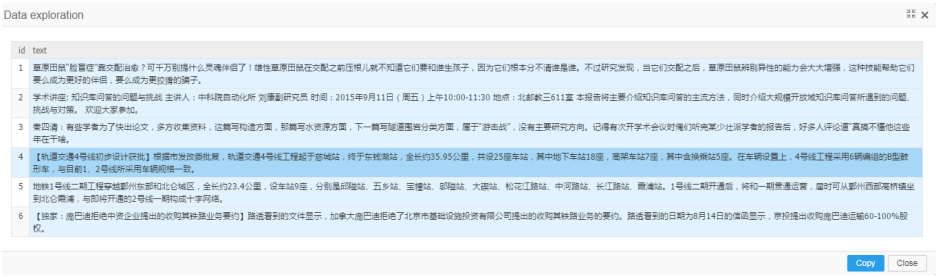
Omitted

Parameter settings:

- Word splitting algorithm: CRF and UNIGRAM.
- Recognition option: whether to recognize nouns with special meanings during word splitting.
- Merge option: to consider the nouns used in special fields as a whole without splitting.
- String split length: =0, indicating that a numerical string is split into retrieval units based on the specified length. The default value is 0, indicating not to split a numerical string by length.
- Use word frequency correction: whether to use the word correction dictionary.
- Part-of-speech tagging: to mark up a word in output results as corresponding to a particular part of speech.

Example

The following input table consists of two columns: docId column and text column.



The output result is as follows:

PAI command example

```
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
```

Algorithm parameters

Parameter key	Description	Option	Default value
inputTableName	Name of the input table	-	-

selectedColNames	Names of the columns used for word splitting in the input table	The names of multiple columns are separated by commas (,).	-
outputTableName	Name of the output table	-	-
inputTablePartitions	Partitions used for word splitting in the input table, in the format of partition_name=value.The multilevel format is name1=value1/name2=value2. Multiple partitions are separated by commas (,).	-	All partitions in the input table
outputTablePartition	Partition in the output table	-	The output table is not partitioned.
tokenizer	Type of the classifier	TAOBAO_CHN and INTERNET_CHN	Default: TAOBAO_CHN, indicating Chinese Taobao library. INTERNET_CHN indicates Internet library.
enableDfa	Simple entity recognition	True/False	True
enablePersonNameTagger	Personal name recognition	True/False	False
enableOrgnizationTagger	Organization name recognition	True/False	False
enablePosTagger	Whether to enable part-of-speech tagging	True/False	False
enableTelephoneRetrievalUnit	Retrieval unit configuration - telephone number recognition	True/False	True
enableTimeRetrievalUnit	Retrieval unit configuration - time ID recognition	True/False	True
enableDateRetrievalUnit	Retrieval unit configuration - date ID recognition	True/False	True
enableNumberLetterRetrievalUnit	Retrieval unit configuration - number and letter	True/False	True

	recognition		
enableChnNumMerge	Merge Chinese numbers into a retrieval unit	True/False	False
enableNumMerge	Merge regular numbers into a retrieval unit	True/False	True
enableChnTimeMerge	Merge the Chinese time into a semantic unit	True/False	False
enableChnDateMerge	Merge Chinese dates into a semantic unit	True/False	False
enableSemanticTagger	Whether to enable semantic tagging	True/False	False

Triples to KV

Function overview

- Given a triple (row, col, value) of XXD or XXL type, X represents arbitrary type, D represents double, and L represents bigint. To convert it to kv format (row, [col_id:value]), the row and value types are consistent with the original input data, the col_id is bigint and the col is mapped to col_id according to the index table.
- The input table format is as follows:

id	word	count
01	a	10
01	b	20
01	c	30

- The output KV table is as follows, which can contain custom KV delimiters:

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

- The output word index table is as follows:

key	key_id
a	1
b	2
c	3

PAI command

```
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
```

Algorithm parameters

Parameter	Description	Option	Default value	Remarks
inputTableName	(Required)Name of the input table	-	-	The table cannot be empty.
idColName	(Required) Columns reserved during sparse conversion	-	-	-
keyColName	(Required) Key in the KV pair	-	-	-
valueColName	(Required) Value in the KV pair	-	-	-
outputTableName	(Required) Name of the output KV table	-	-	-
indexOutputTableName	(Required) Table that outputs the key index	-	-	-
indexInputTableName	(Optional) Existing index table in the input	-	""	The table cannot be empty, and the index of part of the key is allowed.
indexInputKeyColName	(Optional) Name of the	-	""	This parameter is required if

	column that inputs the index table key			indexInputTableName is set.
indexInputKeyIdColName	(Optional) Name of the column that inputs the index ID of the index table key	-	""	This parameter is required if indexInputTableName is set.
inputTablePartitions	(Optional) Partitions in the input table	-	""	Only a single partition can be input.
kvDelimiter	(Optional) Delimiter between the key and the value	-	:	-
pairDelimiter	(Optional) Delimiter between KV pairs	-	;	-
lifecycle	(Optional) Life cycle of the output table	-	Life cycle is not set.	-
coreNum	(Optional) Total number of instances	-	-1	Calculated by the size of input data by default
memSizePerCore	(Optional) Memory size. Value range 100 - 64*1024	-	-1	Calculated based by the size of input data by default

Example

Test data

SQL statement for data generation

```
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;
```

Running command

```
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;
```

Running result*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 |
+------------+------------+
```

String similarity

Function overview

String similarity calculation is a basic operation in machine learning, mainly used for information retrieval, natural language processing, bioinformatics, and so on. This algorithm supports five similarity calculation methods: Levenshtein Distance, Longest Common SubString, String

Subsequence Kernel, Cosine, and simhash_hamming. It also supports two input methods: String-to-string calculation and top n calculation.

Levenshtein Distance (Levenshtein) supports two parameters: distance and similarity. Similarity = 1-Distance. Distance is represented as Levenshtein in the parameters, and similarity is represented as levenshtein_sim.

Longest Common SubString (LCS) supports two parameters: distance and similarity. Similarity =1 - distance. Distance is represented as lcs in the parameters, and similarity is represented as lcs_sim.

String Subsequence Kernel (SSK) supports similarity calculation, represented as SSK in the parameters.

See 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 supports similarity calculation, represented as cosine in the parameters.

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

For Simhash_hamming, the SimHash algorithm is used for mapping the original text into a 64-bit binary fingerprint, and the HammingDistance algorithm is used for calculating the number of characters for binary fingerprint in the same position. Simhash_hamming supports two parameters: distance and similarity. Similarity = 1 - distance/64.0. Distance is represented as simhash_hamming in the parameters, and similarity is represented as simhash_hamming_sim.

For more information about SimHash, see pdf;

For more information about HammingDistance, see wiki.

String-to-string calculation

PAI command

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

Algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	-	-
outputTableName	(Required) Name of the output table	-	-
inputSelectedColNa	(Optional) Name of	-	Name of the first

me1	the first column for similarity calculation		column in string type in the table
inputSelectedColName2	(Optional) Name of the second column for similarity calculation	-	Name of the second column in string type in the table
inputAppendColNames	(Optional) Names of the columns appended to the output table	-	Not appended
inputTablePartitions	(Optional) Partitions selected in the input table	-	Entire table selected
outputColName	(Optional) Name of the similarity column in the output table. A column name must not contain any special character. It can contain only lower- and upper-case letters, numbers, and underscores (_) and must start with a letter. The column name length is no greater than 128 bytes.	-	output
method	(Optional) Similarity calculation method	levenshtein, levenshtein_sim, lcs, lcs_sim, ssk, cosine, simhash_hamming, simhash_hamming_sim	levenshtein_sim
lambda	(Optional) Weight of the matching string, which is available in SSK	(0, 1)	0.5
k	(Optional) Length of the sub-string, which is available in SSK and Cosine	(0, 100)	2
lifecycle	(Optional) Life cycle of the output table	Positive integer	No life cycle
coreNum	(Optional) Number of cores for calculation	Positive integer	Automatically assigned
memSizePerCore	(Optional) Memory size for each core, in MB	Positive integer in the range of (0, 65536)	Automatically assigned

Example

Test data

```
create table pai_ft_string_similarity_input as select * from
(select 0 as id, "Beijing" as col0, "Beijing" as col1 from dual
union all
select 1 as id, "Beijing" as col0, "Beijing Shanghai" as col1 from dual
union all
select 2 as id, "Beijing" as col0, "Beijing Shanghai Hong Kong" as col1 from dual
)tmp;
```

PAI command

```
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;
```

Output description

Output results obtained by using the simhash_hamming method:

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

Output results obtained by using the simhash_hamming_sim method:

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

String similarity - topN

PAI command

```
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";
```

Algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	-	-
mapTableName	(Required) Name of the mapping table	-	-
outputTableName	(Required) Name of the output table	-	-
inputSelectedColName	(Optional) Name of the column from the left table for similarity calculation	-	Name of the first column in string type in the table
mapSelectedColName	(Optional) Name of the column in the mapping table for similarity calculation. The similarities between each row in the left table and all strings in the mapping table are calculated, and the top N results are given in the end.	-	Name of the first column in string type in the table
inputAppendColNames	(Optional) Names of the columns from the input table appended to the output table	-	Not appended
inputAppendRenameColNames	(Optional) Aliases of the columns from the input table appended to the output table. The parameter is valid when inputAppendColNames is not null.	-	Aliases not used
mapAppendColNames	(Optional) Names of the columns from the mapping table appended to the	-	Not appended

	output table		
mapAppendRenameColNames	(Optional) Aliases of the columns from the mapping table appended to the output table	-	Aliases not used
inputTablePartitions	(Optional) Partitions selected in the input table	-	Entire table selected
mapTablePartitions	(Optional) Partitions in the mapping table	-	All partitions in the mapping table
outputColName	(Optional) Name of the similarity column in the output table. A column name must not contain any special character. It can contain only lower- and upper-case letters, numbers, and underscores (_) and must start with a letter. The column name length is no greater than 128 bytes.	-	output
method	(Optional) Similarity calculation method	levenshtein_sim, lcs_sim, ssk, cosine, simhash_hamming_sim	levenshtein_sim
lambda	(Optional) Weight of the matching string, which is available in SSK	(0, 1)	0.5
k	(Optional) Length of the sub-string, which is available in SSK and Cosine	(0, 100)	2
topN	(Optional) Number of similarity maximums in the End	(0, +∞)	10
lifecycle	(Optional) Life cycle of the output table	Positive integer	No life cycle
coreNum	(Optional) Number of cores for calculation	Positive integer	Automatically assigned
memSizePerCore	(Optional) Memory size for each core, in	Positive integer in the range of (0,	Automatically assigned

	MB	65536)	

Example

Test data

```
create table pai_ft_string_similarity_topn_input as select * from
(select 0 as id, "Beijing" as col0 from dual
union all
select 1 as id, "Beijing Shanghai" as col0 from dual
union all
select 2 as id, "Beijing Shanghai Hong Kong" as col0 from dual
)tmp;
```

PAI command

```
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;
```

Output description

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

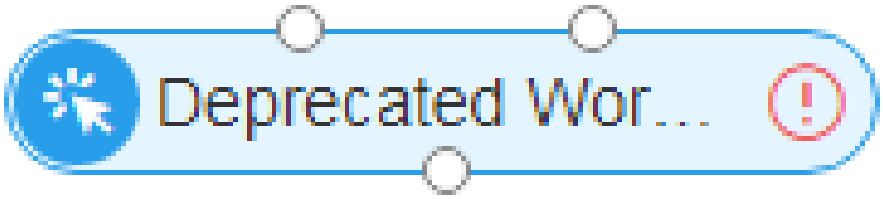
Deprecated word filtering

Function overview

Deprecated word filtering is a preprocessing method in text analysis. Its function is to filter out the noise in word splitting results (for example, of, yes, and ah).

Parameter settings

Component description

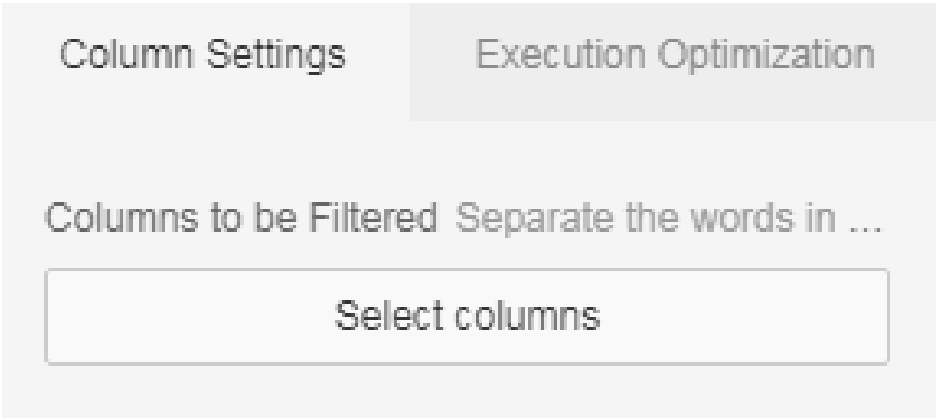


Two input columns from left to right:

Input table, which is a word splitting result table for filtering; parameter: inputTableName

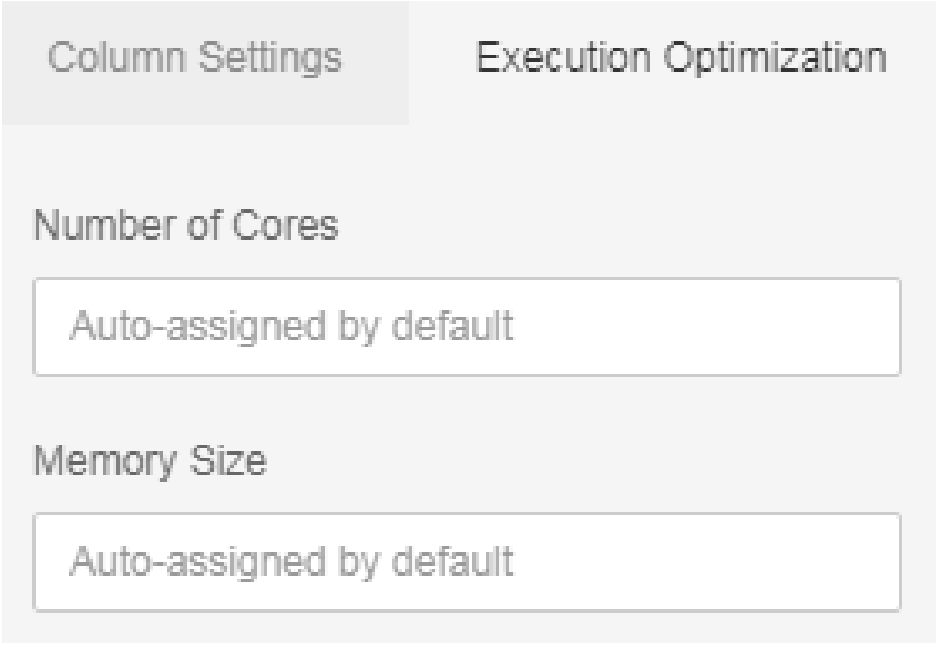
Deprecated word table, which is a one-column table with each row containing a deprecated word; parameter: noiseTableName

Parameter UI description



Columns available for filtering.

Fine-tune description



You can configure the number of concurrent computing cores and memory size. By default, they are automatically assigned.

PAI command

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

Algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	-	-
inputTablePartitions	(Optional) Partitions used for calculation in the input table	-	All partitions in the input table
noiseTableName	(Required) Deprecated word table	One-column table with each row containing a deprecated word	-
noiseTablePartitions	(Optional) Partitions in the deprecated word table		All partitions in the table
outputTableName	(Required) Name of the output table	-	-
selectedColNames	(Required) Names of	-	-

	the columns to be filtered, separated by commas in case of more than one column		
lifecycle	(Optional) Life cycle of the output table	Positive integer	No life cycle
coreNum	(Optional) Number of cores for calculation	Positive integer	Automatically assigned
memSizePerCore	(Optional) Memory size for each core, in MB	Positive integer in the range of (0, 65536)	Automatically assigned

Example

Source data

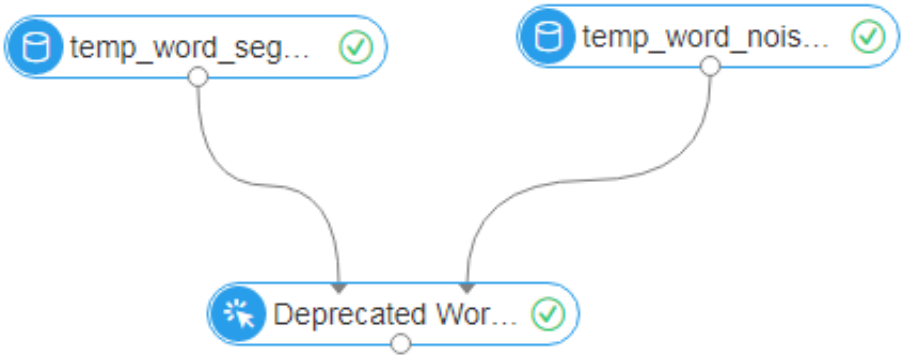
Word splitting result table temp_word_seg_input

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

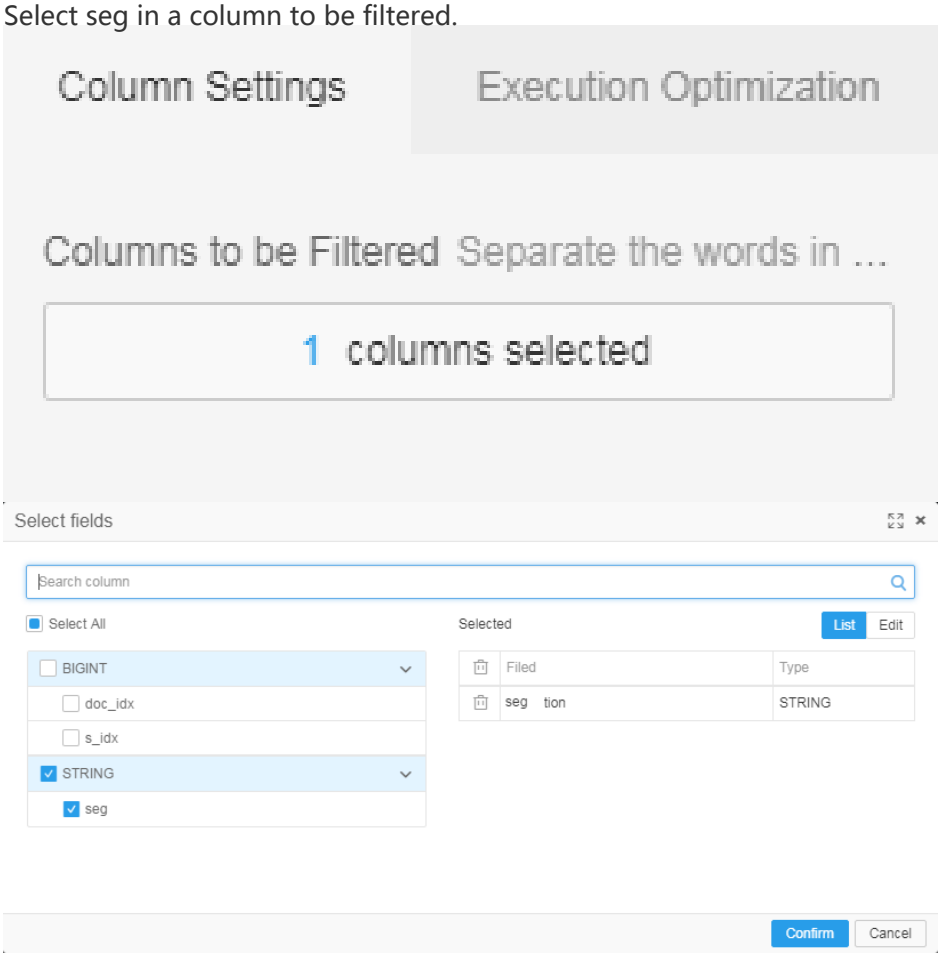
Deprecated word table temp_word_noise_input

noise_word
在
地
时
他
是
比如
而是
的

Create an experiment.



Select seg in a column to be filtered.



Running result.

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

Text summarization

Automatic summarization uses computers to automatically extract summaries from the source document. Summaries are simple, consistent, and short documents that fully and accurately reflect the content of a certain document. Based on TextRank, this algorithm forms summaries by extracting existing sentences in the document.

For detailed principles of this algorithm, see the document TextRank: Bringing Order into Texts.

PAI command

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

Parameter description

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	-	-
inputTablePartitions	(Optional) Partitions used for calculation in the input table	-	All partitions in the input table
outputTableName	(Required) Name of the output table	-	-
docIdCol	(Required) Name of the column listing document IDs	-	-
sentenceCol	(Required) Sentence column. Only one column can be specified.	-	-

topN	(Optional) Top N key sentences that are output	-	3
similarityType	(Optional) Sentence similarity calculation method	lcs_sim, levenshtein_sim, cosine, ssk	lcs_sim
lambda	(Optional) Weight of the matching string, which is available in SSK	(0, 1)	0.5
k	(Optional) Length of the sub-string, which is available in SSK and Cosine	(0, 100)	2
dampingFactor	(Optional) Damping factor	(0, 1)	0.85
maxIter	(Optional) Maximum number of iterations	[1, +]	100
epsilon	(Optional) Convergence factor	(0, ∞)	0.000001
lifecycle	(Optional) Life cycle of the input/output table	Positive integer	No life cycle
coreNum	(Optional) Number of cores for calculation	Positive integer	Automatically calculated
memSizePerCore	(Optional) Memory size for each core	Positive integer	Automatically calculated

The sentence similarity has the following value options: (Symbol description: A and B indicate two strings, and len(A) indicates the length of string A.)

lcs_sim: Formula ‘1.0 - (Length of the longest common subsequence)/max(len(A), len(B))’.

levenshtein_sim: Formula ‘1.0 - (Levenshtein distance)/max(len(A), len(B))’.

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

Output format description

The output table contains the doc_id and abstract columns. For example:

doc_id	abstract
1000894	Shanghai Stock Exchange published the guidelines for disclosure of listed companies' corporate social responsibility (CSR) in 2008, urging three types of companies to disclose their CSR reports and encouraging other qualified listed companies to voluntarily disclose their CSR reports. Statistics show that in 2012, a total of 379 listed companies on Shanghai Stock Exchange disclosed their CSR reports. Among these companies 74 did so voluntarily, and the remaining 305 companies were required by financial regulations, which account for 40% of all listed companies on Shanghai Stock Exchange. According to Hu Ruyin, Shanghai Stock Exchange will explore how to expand the scope of CSR report disclosure, revise, and refine the guidelines on disclosure of the CSR reports, and encourage more organizations to promote CSR product innovation.

Document similarity

Based on the similarity of strings, the word-based document similarity is calculated by comparing the similarity between documents or sentences that are separated by spaces. The document similarity is calculated in a similar way to string similarity. Types of document similarity are the same as those of string similarity.

Levenshtein Distance (Levenshtein) supports two parameters: distance and similarity. Similarity = 1-Distance. Distance is represented as Levenshtein in the parameters, and similarity is represented as levenshtein_sim.

Longest Common SubString (LCS) supports two parameters: distance and similarity. Similarity =1 - distance. Distance is represented as lcs in the parameters, and similarity is represented as lcs_sim.

String Subsequence Kernel (SSK) supports similarity calculation, represented as SSK in the parameters.

See 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 supports similarity calculation, represented as cosine in the parameters.

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

For Simhash_hamming, the SimHash algorithm is used for mapping the original text into a 64-bit binary fingerprint, and the HammingDistance algorithm is used for calculating the number of characters for binary fingerprint in the same position. Simhash_hamming supports two parameters: distance and similarity. Similarity = 1 - distance/64.0. Distance is represented as simhash_hamming in the parameters, and similarity is represented as simhash_hamming_sim.

For more information about SimHash, see pdf;

For more information about HammingDistance, see wiki.

PAI command

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

Parameter description

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	-	-
outputTableName	(Required) Name of the output table	-	-
inputSelectedColName1	(Optional) Name of the first column for similarity calculation	-	Name of the first column in string type in the table
inputSelectedColName2	(Optional) Name of the second column for similarity calculation	-	Name of the second column in string type in the table
inputAppendColNames	(Optional) Names of the columns appended to the output table	-	Not appended
inputTablePartitions	(Optional) Partitions selected in the input table	-	Entire table selected
outputColName	(Optional) Name of the similarity column in the output table.A column name must not contain any special character. It can contain only lower- and upper-case letters,	-	output

	numbers, and underscores (_) and must start with a letter. The column name length is no greater than 128 bytes.		
method	(Optional) Similarity calculation method	levenshtein, levenshtein_sim, lcs, lcs_sim, ssk, cosine, simhash_hamming, simhash_hamming_sim	levenshtein_sim
lambda	(Optional) Weight of the matching word combination, which is available in SSK	(0, 1)	0.5
k	(Optional) Length of the sub-string, which is available in SSK and Cosine	(0, 100)	2
lifecycle	(Optional) Life cycle of the output table	Positive integer	No life cycle
coreNum	(Optional) Number of cores for calculation	Positive integer	Automatically assigned
memSizePerCore	(Optional) Memory size for each core, in MB	Positive integer in the range of (0, 65536)	Automatically assigned

Example

Data generation

```
drop table if exists pai_doc_similarity_input;
create table pai_doc_similarity_input as
select * from
(
select 0 as id, "Beijing Shanghai" as col0, "Beijing Shanghai" as col1 from dual
union all
select 1 as id, "Beijing Shanghai" as col0, "Beijing Shanghai Hong Kong" as col1 from dual
)tmp
```

PAI command line

```
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;
```

Input description

pai_doc_similarity_input

id	col0	col1
1	Beijing Shanghai	Beijing Shanghai Hong Kong
0	Beijing Shanghai	Beijing Shanghai

Output description

pai_doc_similarity_output

id	col0	col1	output
1	Beijing Shanghai	Beijing Shanghai Hong Kong	0.6666666666666667
0	Beijing Shanghai	Beijing Shanghai	1.0

Important notes

The similarity calculation is based on the word splitting result, that is, each word separated by space is used as a unit for similarity calculation. You can use the string similarity calculation method if the words are input in a string as a whole.

In the parameter Method, levenshtein, lcs, and simhash_hamming represents the distance for calculation. levenshtein_sim, lcs_sim, ssk, cosine, and simhash_hamming_sim represent the distance for calculation. Distance = 1.0 - similarity.

The presence of parameter k in the Cosine and SSK methods indicates that k words are used as a combination for similarity calculation. Therefore, when k is greater than the number of words, the similarity output may be 0 even if the two strings are the same. In this case, the value of k can be changed to the minimum number of words.

Sentence splitting

Component description

A piece of text is split by punctuations.

This component is used for preprocessing of text summarization. It splits a piece of text so that each row contains only one sentence.

PAI command

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

Parameter description

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	-	-
inputTablePartitions	(Optional) Partitions used for calculation in the input table	-	All partitions in the input table
outputTableName	(Required) Name of the output table	-	-
docIdCol	(Required) Name of the column listing document IDs	-	-
docContent	(Required) Name of the column displaying the document content. Only one column can be specified.	-	-
delimiter	(Optional) Delimiter set of a sentence	-	! ?
lifecycle	(Optional) Life cycle of the input/output table	Positive integer	No life cycle
coreNum	(Optional) Number of cores for calculation	Positive integer	Automatically calculated
memSizePerCore	(Optional) Memory size for each core	Positive integer	Automatically calculated

Output format description

The output table contains the doc_id and sentence columns.For example:

doc_id	sentence
1000894	Shanghai Stock Exchange published the guidelines for disclosure of listed

	companies' corporate social responsibility (CSR) in 2008, urging three types of companies to disclose their CSR reports and encouraging other qualified listed companies to voluntarily disclose their CSR reports.
1000894	Statistics show that in 2012, a total of 379 listed companies on Shanghai Stock Exchange disclosed their CSR reports. Among these companies 74 did so voluntarily, and the remaining 305 companies were required by financial regulations, which account for 40% of all listed companies on Shanghai Stock Exchange.

Conditional random field

A conditional random field (CRF) is a conditional probability distribution model of a group of output random variables based on a group of input random variables.This model presumes that the output random variables constitute a Markov random field (MRF). CRFs can be used in different prediction scenarios. The linear chain CRF is mostly used, especially in annotation scenarios.

For more information about CRFs, see wiki.

PAI command

```
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
```

Algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input feature data table	-	-
inputTablePartitions	(Optional) Partitions selected in the input feature data table	-	All partitions in the table by default
featureColNames	(Optional) Feature columns selected in the input table	-	All columns but the label column are selected by default.
labelColName	(Required) Target	-	-

	column		
idColName	(Required) Sample ID column	-	-
outputTableName	(Required) Name of the output model table	-	-
outputTablePartitions	(Optional) Partitions selected in the output model table	-	All partitions in the table by default
template	(Optional) Template generated by algorithm features	Definition format: as follows	Default value: as follows
freq	(Optional) Parameter for filtering features. Only the feature appears equal to or more than the value of freq is retained in the algorithm.	Positive integer	Default value: 1
iterations	(Optional) Maximum number of optimization iterations	-	Default value: 100
l1Weight	(Optional) L1 norm parameter weight	-	Default value: 1.0
l2Weight	(Optional) L2 norm parameter weight	-	Default value: 1.0
epsilon	(Optional) Convergence deviation, which is the termination condition for L-BFGS, that is, the log-likelihood deviation between two iterations	-	Default value: 0.0001
lbfgsStep	(Optional) Historical size during lbfgs optimization, valid only for lbfgs	-	default value: 10
threadNum	(Optional) Number of the threads enabled in parallel during model training	-	Default value: 3
lifecycle	(Optional) Life cycle of the output table	-	Unspecified by default

coreNum	(Optional) Number of cores	-	Automatically calculated by default
memSizePerCore	(Optional) Size of memory	-	Automatically calculated by default

Feature template definition

```
<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
```

Default algorithm 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]
```

Data example

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

PAI command example for prediction algorithm

```
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
```

Prediction algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input feature data table	-	-
inputTablePartitions	(Optional) Partitions selected in the input feature data table	-	All partitions in the table by default
featureColNames	(Optional) Feature columns selected in the input table	-	All columns but the label column are selected by default.
labelColName	(Optional) Target column	-	-
IdColName	(Required) Sample ID column	-	-
resultColName	(Optional) Name of the result column in the output table	-	Default: prediction_result
scoreColName	(Optional) Name of the score column in the output table	-	Default: prediction_score
detailColName	(Optional) Name of the detail column in the output table	-	Default: null
outputTableName	(Required) Name of the output prediction result table	-	-
outputTablePartitions	(Optional) Partitions selected in the output prediction result table	-	All partitions in the table by default
modelTableName	(Required) Name of the algorithm model table	-	-
modelTablePartitions	(Optional) Partitions selected in the algorithm model table	-	All partitions in the table by default
lifecycle	(Optional) Life cycle of the output table	-	Unspecified by default

coreNum	(Optional) Number of cores	-	Automatically calculated by default
memSizePerCore	(Optional) Size of memory	-	Automatically calculated by default

Test data

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

Note: The label column can be absent.

Keyword extraction

Keyword extraction is an important natural language processing technique. Specifically, it refers to the process of extracting words that are most relevant to the meaning of a document. This algorithm is based on TextRank and inspired by PageRank (an algorithm that describes the relationship between webpages). It uses the relationship between partial words (co-occurrence window) to construct a network, calculate the importance of words, and select greater weight words as keywords.

PAI command

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

Parameter description

Parameter	Description	Value range	Required/Optional, default value

inputTableName	Input table	Table name	Required
inputTablePartitions	Partitions used for training in the input table, in the format of partition_name=value. The multilevel partition name format is name1=value1/name2=value2. If you specify multiple partitions, separate them with a comma (,).		(Optional) Default: all partitions
outputTableName	Name of the output table	Required	
docIdCol	Name of the column listing document IDs. Only one column can be specified.	Required	
docContent	Word column. Only one column can be specified.	Required	
topN	Number of first keywords to be output. If this number is smaller than the number of all words, all words are output.	(Optional) Default value: 5	
windowSize	Size of the window of the TextRank algorithm	(Optional) Default value: 2	
dumpingFactor	Damping factor of the TextRank algorithm	(Optional) Default value: 0.85	
maxIter	Maximum number of iterations of the TextRank algorithm	(Optional) Default value: 100	
epsilon	Convergence residual threshold of the TextRank algorithm	(Optional) Default value: 0.000001	
lifecycle	(Optional) Life cycle of the output table	Positive integer	No life cycle
coreNum	Number of nodes	Used together with the memSizePerCore parameter. Positive	(Optional) Automatically calculated by default

		integer in the range of [1, 9999]. Detailed description	
memSizePerCore	Memory size of each node, in MB	Positive integer in the range of [1024, 64*1024] Detailed description	(Optional) Automatically calculated by default

Example

Data generation

The words in the input table are separated by space, and the deprecated words and all punctuation marks are filtered out.

docid:string	word:string
doc0	The blended-wing-body aircraft is a new direction for the future development in the aviation field Many research institutions in and outside China have carried out researches on the blended-wing-body aircraft while its fully automated shape optimization algorithm has become a new hot topic Based on the existing research achievements in and outside China common modeling and flow solver tools have been analyzed and compared The geometric modeling meshing flow field solver and shape optimization modules have been designed The pros and cons between different algorithms have been compared to get the optimized shape of the blended-wing-body aircraft in the conceptual design stage Geometric modeling and mesh generation module include the transfinite interpolation algorithm and spline based mesh generation method The flow solver module includes the finite difference solver the finite element solver and the panel method solver wherein the finite difference solver includes mathematical modeling of the potential flow the derivation of the Cartesian grid based variable step length difference scheme Cartesian grid generation and indexing algorithm the Cartesian grid based Neumann boundary conditions The aerodynamic parameters of a two-dimensional airfoil are calculated based on the finite difference solver Finite element solver includes potential flow modeling based on the variational principle of the finite element theory the derivation of the two-dimensional finite element Kutta conditional least squares based speed solving algorithm Gmsh based two-dimensional field mesh generator of airfoil with wakes design The aerodynamic

	parameters of a two-dimensional airfoil are calculated based on the finite element solver The panel method solver includes modeling and automatic wake generation the design of the three-dimensional flow solver of the blended-wing-body drag estimation based on the Blasius solution solver implemented in the Fortran language a mixed compilation of Python and Fortran OpenMP and CUDA based acceleration algorithm The aerodynamic parameters of a three-dimensional wing body are calculated based on the panel method solver The shape optimization module includes a free form deformation algorithm genetic algorithms differential evolution algorithm aircraft surface area calculation algorithm and the moments integration algorithm which can calculate the volume of an aircraft.

PAI command line

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

Output description

Output table

docid	keywords	weight
doc0	Based on	0.041306752223538405
doc0	Algorithm	0.03089845626854151
doc0	Modeling	0.021782865850562882
doc0	Grid	0.020669749212693957
doc0	Solver	0.020245609506360847
doc0	Aircraft	0.019850761705313365
doc0	Research	0.014193732541852615
doc0	Finite element	0.013831122054200538
doc0	Solving	0.012924593244133104
doc0	Module	0.01280216562287212
doc0	Derivation	0.011907588923852495
doc0	Shape	0.011505456605632607
doc0	Difference	0.011477831662367547

doc0	Flow	0.010969269350293957
doc0	Design	0.010830986516637251
doc0	Implementation	0.010747536556701583
doc0	Two-dimensional	0.010695570768457084
doc0	Development	0.010527342662670088
doc0	New	0.010096978306668461

Algorithm scale

Sina data set, 1,080,162 documents, 30 initial nodes, 10 minutes.

ngram-count

Ngram-count is one of the steps in the language model, which generates n-grams based on the words and counts the number of corresponding n-grams on all corpus. It counts the number of n-grams in the global documents rather than in a single document. See ngram-count, which is a subset of the tool.

PAI command

```
PAI -name ngram_count
-project algo_public
-DinputTableName=pai_ngram_input
-DoutputTableName=pai_ngram_output
-DinputSelectedColNames=col0
-DweightColName=weight
-DcoreNum=2
-DmemSizePerCore=1000;
```

Parameter description

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	-	-
outputTableName	(Required) Name of the output table	-	-
inputSelectedColNames	(Optional) Names of the columns selected in the input table	-	The first column of character type
weightColName	(Optional) Name of the weight column	-	Default weight: 1
inputTablePartitions	(Optional) Partitions	-	Entire table selected

	in the input table		by default
countTableName	(Optional) Previous ngram-count output table, which is merged into the final result	-	-
countWordColName	(Optional) Name of the word column in the count table	-	The second column is selected by default.
countCountColName	(Optional) Name of the count column in the count table	-	The third column is selected by default.
countTablePartitions	(Optional) Partitions in the count table	-	-
vocabTableName	(Optional) Bag-of-words table. The words not contained in the bag-of-words are identified as \<unk\ in the result.	-	-
vocabSelectedColName	(Optional) Name of the bag-of-words column	-	The first column of character type is selected by default.
vocabTablePartitions	(Optional) Partitions in the bag-of-words table	-	-
order	(Optional) Maximum length of N-grams	-	Default value: 3
lifecycle	(Optional) Life cycle of the output table	-	-
coreNum	(Optional) Number of cores	-	-
memSizePerCore	(Optional) Memory size for each core	-	-

Semantic vector distance

Calculate the extension words (sentences) of the specified words (sentences) based on the calculated semantic vectors (such as word vectors calculated by the Word2vec component). The extension words (sentences) are a set of vectors closest to a certain vector. The following example shows how to generate a list of words that are most similar to the word that you entered based on the word vector results calculated by the Word2vec component.

PAI command

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

Parameter description

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	-	-
inputTablePartitions	(Optional) Partitions used for calculation in the input table	-	All partitions in the input table
outputTableName	(Required) Name of the output table	-	-
idTableName	(Optional) Name of the table containing the column listing IDs of the close vectors. One-column table with each row assigned with an ID. Null by default, that is, all vectors in the input table are calculated.	-	-
idTablePartitions	(Optional) List of partitions used for calculation in the ID table. All partitions are selected by default.	-	-
idColName	(Required) Name of the ID column	-	3
vectorColNames	(Optional) List of vector column names, such as f1, f2,...	-	-
topN	(Optional) Number of the output closest vectors	[1, ∞]	5
distanceType	(Optional) Distance calculation method	euclidean, cosine, and manhattan	euclidean
distanceThreshold	(Optional) Distance threshold. The distance between	(0, ∞)	∞

	two vectors is output if it is smaller than the distance threshold.		
lifecycle	(Optional) Life cycle of the input/output table	Positive integer	No life cycle
coreNum	(Optional) Number of cores for calculation	Positive integer	Automatically calculated
memSizePerCore	(Optional) Memory size for each core	Positive integer	Automatically calculated

Example

The output table contains the original_id, near_id, distance, and rank columns. For example:

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

Conditional random field

A conditional random field (CRF) is a conditional probability distribution model of a group of output random variables based on a group of input random variables.This model presumes that the output random variables constitute a Markov random field (MRF).CRFs can be used in different prediction scenarios. The linear chain CRF is mostly used, especially in annotation scenarios.

For more information about CRFs, see wiki.

PAI command

```
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
```

Algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input feature data table	-	-
inputTablePartitions	(Optional) Partitions selected in the input feature data table	-	All partitions in the table by default
featureColNames	(Optional) Feature columns selected in the input table	-	All columns but the label column are selected by default.
labelColName	(Required) Target column	-	-
idColName	(Required) Sample ID column	-	-
outputTableName	(Required) Name of the output model table	-	-
outputTablePartitions	(Optional) Partitions selected in the output model table	-	All partitions in the table by default
template	(Optional) Template generated by algorithm features	Definition format: as follows	Default value: as follows
freq	(Optional) Parameter for filtering features. Only the feature appears equal to or more than the value of freq is retained in the algorithm.	Positive integer	Default value: 1
iterations	(Optional) Maximum number of optimization iterations	-	Default value: 100
l1Weight	(Optional) L1 norm parameter weight	-	Default value: 1.0
l2Weight	(Optional) L2 norm parameter weight	-	Default value: 1.0
epsilon	(Optional) Convergence deviation, which is the termination condition for L-BFGS, that is, the log-likelihood	-	Default value: 0.0001

	deviation between two iterations		
lbfgsStep	(Optional) Historical size during lbfgs optimization, valid only for lbfgs	-	default value: 10
threadNum	(Optional) Number of the threads enabled in parallel during model training	-	Default value: 3
lifecycle	(Optional) Life cycle of the output table	-	Unspecified by default
coreNum	(Optional) Number of cores	-	Automatically calculated by default
memSizePerCore	(Optional) Size of memory	-	Automatically calculated by default

Feature template definition

```
<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
```

Default algorithm 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]
```

Data example

Training data

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

PAI command example for prediction algorithm

```
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
```

Prediction algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input feature data table	-	-
inputTablePartitions	(Optional) Partitions selected in the input feature data table	-	All partitions in the table by default
featureColNames	(Optional) Feature columns selected in the input table	-	All columns but the label column are selected by default.
labelColName	(Optional) Target column	-	-
IdColName	(Required) Sample ID column	-	-
resultColName	(Optional) Name of the result column in the output table	-	Default: prediction_result
scoreColName	(Optional) Name of the score column in the output table	-	Default: prediction_score
detailColName	(Optional) Name of the detail column in the output table	-	Default: null
outputTableName	(Required) Name of the output prediction result table	-	-
outputTablePartition	(Optional) Partitions	-	All partitions in the

s	selected in the output prediction result table		table by default
modelTableName	(Required) Name of the algorithm model table	-	-
modelTablePartitions	(Optional) Partitions selected in the algorithm model table	-	All partitions in the table by default
lifecycle	(Optional) Life cycle of the output table	-	Unspecified by default
coreNum	(Optional) Number of cores	-	Automatically calculated by default
memSizePerCore	(Optional) Size of memory	-	Automatically calculated by default

Test data

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

Note: The label column can be absent.

PMI

Mutual information (MI) is a useful measure of information in the information theory. It can be regarded as the amount of information contained in a random variable about the other, or the reduction in uncertainty of a random variable due to the known other.

This algorithm counts the co-occurrence of all words in several documents and calculates PMI (point mutual information). MI definition:
PMI(x,y)=ln(p(x,y)/(p(x)p(y)))=ln(#(x,y)D/(#x#y))

Where, #(x,y) is the counted number of pair(x,y) and D is the total number of pairs. If x and y appear in the same window, then #x+=1; #y+=1;#(x,y)+=1.

PAI command line

```
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;
```

Parameter description

Parameter	Description	Value range	Required/Optional, default value
inputTableName	Input table	Table name	Required
outputTableName	Name of the output table	Table name	Required
docColName	Column names of documents after word splitting, where words are separated by space	Column name	Required
windowSize	Size of window. For example, the value 5 refers to the five adjacent words to the right of the current word (excluding the current word). Words appearing in the window are considered related to the current word.	[1, sentence length]	(Optional) Entire row by default
minCount	Minimum word truncation frequency. Words that appear for a number of times less than this value are	[0, 2e63]	(Optional) Default value: 5

	filtered out.		
inputTablePartitions	Partitions used for training in the input table, in the format of partition_name=value. The multilevel partition name format is name1=value1/name2=value2. If you specify multiple partitions, separate them with a comma (,).		(Optional) Default: all partitions
lifecycle	(Optional) Life cycle of the output table	Positive integer	No life cycle
coreNum	Number of nodes	Used together with the memSizePerCore parameter, positive integer in the range of [1, 9999]	(Optional) Automatically calculated by default
memSizePerCore	Memory size of each node, in MB	Positive integer in the range of [1024, 64*1024]	(Optional) Automatically calculated by default

Example

Data generation

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 command line

```
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;
```

Output description

Output table

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

Graph analysis

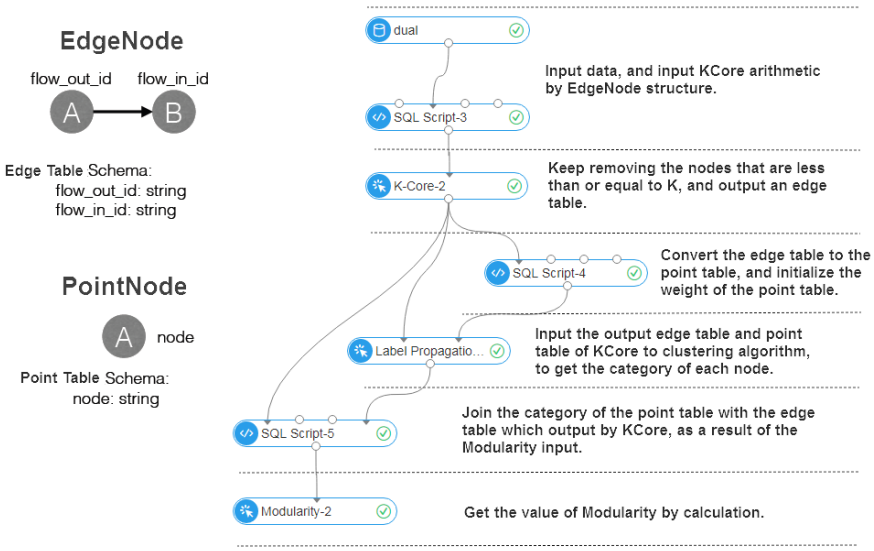
Contents

Edge clustering coefficient

Counting triangle

Decision tree depth

The network analysis column provides analytic algorithms which are based on the Graph data structure. The following figure shows an example of the analysis process developed with the network analysis component of the platform.



The running parameters need to be set for the algorithm components in the network analysis column.
The parameters are described as follows:

- Process count: The workerNum parameter specifies the number of nodes for concurrent job execution. The concurrency level and framework communication costs increase with the value of this parameter.
- Work memory: The workerMem parameter specifies the maximum memory size that a single worker can use. The default value is 4096 MB. The OutOfMemory exception is thrown if memory usage of a single process exceeds the maximum.

k-Core

Function overview

The KCore of a graph is the subgraph that is left after all nodes whose degrees are less than or equal to K are removed. If a node is included in the KCore but is removed from the (K+1)Core, the coreness of this node is K. Therefore, the coreness of a node whose degree is 1 must be 0. The maximum node

coreness is the graph coreness.

Parameter settings

K: Coreness value, required, default value is 3.

PAI command

```
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;
```

Algorithm parameters

Parameter key	Description	Required/Optional	Default value
inputEdgeTableName	Name of the input edge table	Required	NA
inputEdgeTablePartitions	Partitions in the input edge table	Optional	Entire table
fromVertexCol	Start column in the edge table	Required	NA
toVertexCol	End column in the edge table	Required	NA
outputTableName	Name of the output table	Required	NA
outputTablePartitions	Partitions in the output table	Optional	NA
lifecycle	Life cycle of the output table	Optional	NA
workerNum	Worker count	Optional	Not set
workerMem	Worker memory size	Optional	4096
splitSize	Data split size	Optional	64
k	Number of cores	Required	3

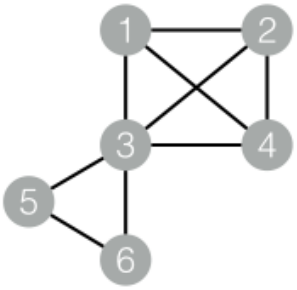
Example

Test data

SQL statement for data generation:

```
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;
```

Structure of the graph corresponding to the data:



Running result

Set K to 2.
The result is as follows:

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

Single-source shortest path

Function overview

The single-source shortest path (SSSP) refers to the Dijkstra algorithm. In this algorithm, if the start node is specified, the shortest paths between this node and all other nodes are output.

Parameter settings

Start node ID: ID of the start node used to calculate the shortest paths, which is required.

PAI command

```
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;
```

Algorithm parameters

Parameter key	Description	Required/Optional	Default value
inputEdgeTableName	Name of the input edge table	Required	NA
inputEdgeTablePartitions	Partitions in the input edge table	Optional	Entire table
fromVertexCol	Start column in the input edge table	Required	NA
toVertexCol	End column in the input edge table	Required	NA
outputTableName	Name of the output table	Required	NA
outputTablePartitions	Partitions in the output table	Optional	NA
lifecycle	Life cycle of the output table	Optional	NA
workerNum	Worker count	Optional	Not set

workerMem	Worker memory size	Optional	4096
splitSize	Data split size	Optional	64
startVertex	Start node ID	Required	NA
hasEdgeWeight	Indicates whether the edge of the input edge table has weight	Optional	false
edgeWeightCol	Edge weight column in the input edge table	Optional	NA

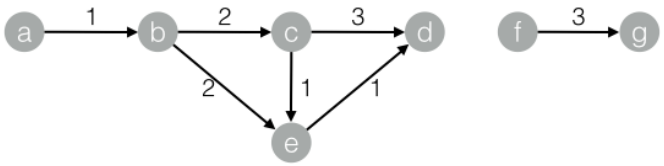
Example

Test data

SQL statement for data generation:

```
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;
```

Structure of the graph corresponding to the data:



Running result

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

PageRank

Function overview

PageRank is an algorithm used by Google Search to rank websites based on the structure of links to a page. The underlying assumption is as follows:

More important or quality websites are likely to receive more links from other websites.

Besides the quantity of links to a web page, the weight of the web page and the number of outgoing links also count in the calculation.

For the social networks of a user, the edge weight is an important factor in addition to the influence of the user.

For example, a Sina Weibo user is more likely to have influence on the family, friends, classmates, and colleagues among the user’ s followers, while less on the followers of weak relationship such as strangers. In social networks, the edge weight is equivalent to the user-user relationship strength index.

The PageRank formula with the link weight is as follows:

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

- W(i): The weight of node i.
- C(Ai): The link weight.
- d: The damping factor.
- W(A): The node weight after the algorithm iteration is stable, which is the influence index of each user.

Parameter settings

Maximum number of iterations: (Optional) The number of iterations before the algorithm automatically converges. The default value is 30.

PAI command

```
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;
```

Algorithm parameters

Parameter key	Description	Required/Optional	Default value
inputEdgeTableName	Name of the input edge table	Required	NA
inputEdgeTablePartitions	Partitions in the input edge table	Optional	Entire table
fromVertexCol	Start column in the input edge table	Required	NA
toVertexCol	End column in the input edge table	Required	NA
outputTableName	Name of the output table	Required	NA
outputTablePartitions	Partitions in the output table	Optional	NA
lifecycle	Life cycle of the output table	Optional	NA
workerNum	Worker count	Optional	Not set
workerMem	Worker memory size	Optional	4096
splitSize	Data split size	Optional	64
hasEdgeWeight	Indicates whether the edge of the input edge table has weight	Optional	false
edgeWeightCol	Edge weight column in the input edge table	Optional	NA
maxIter	Maximum number of iterations	Optional	30

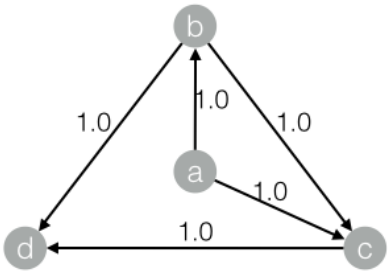
Example

Test data

SQL statement for data generation:

```
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;
```

Structure of the graph corresponding to the data:



Running result

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

Label propagation clustering

Function overview

Graph clustering is used to divide subgraphs based on the graph topology such that the links between the nodes in the subgraphs are more than those between subgraphs. The label propagation

algorithm (LPA) is a graph-based semi-supervised machine learning algorithm. The labels of a node (community) depend on those of the neighboring nodes. The dependence degree is determined by the similarity between nodes, and data becomes stable by iterative propagation update.

Parameter settings

Maximum number of iterations: (Optional) The default value is 30.

PAI command

```
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;
```

Algorithm parameters

Parameter key	Description	Required/Optional	Default value
inputEdgeTableName	Name of the input edge table	Required	NA
inputEdgeTablePartitions	Partitions in the input edge table	Optional	Entire table
fromVertexCol	Start column in the input edge table	Required	NA
toVertexCol	End column in the input edge table	Required	NA
inputVertexTableName	Name of the input vertex table	Required	NA
inputVertexTablePartitions	Partitions in the input vertex table	Optional	Entire table
vertexCol	Vertex column in the input vertex table	Required	NA
outputTableName	Name of the output table	Required	NA
outputTablePartitions	Partitions in the output table	Optional	NA

lifecycle	Life cycle of the output table	Optional	NA
workerNum	Worker count	Optional	Not set
workerMem	Worker memory size	Optional	4096
splitSize	Data split size	Optional	64
hasEdgeWeight	Indicates whether the edge of the input edge table has weight	Optional	false
edgeWeightCol	Edge weight column in the input edge table	Optional	NA
hasVertexWeight	Indicates whether the vertexes of the input vertex table have weights	Optional	false
vertexWeightCol	Vertex weight column in the input vertex table	Optional	NA
randSelect	Indicates whether the maximum label is randomly selected	Optional	false
maxIter	Maximum number of iterations	Optional	30

Example

Test data

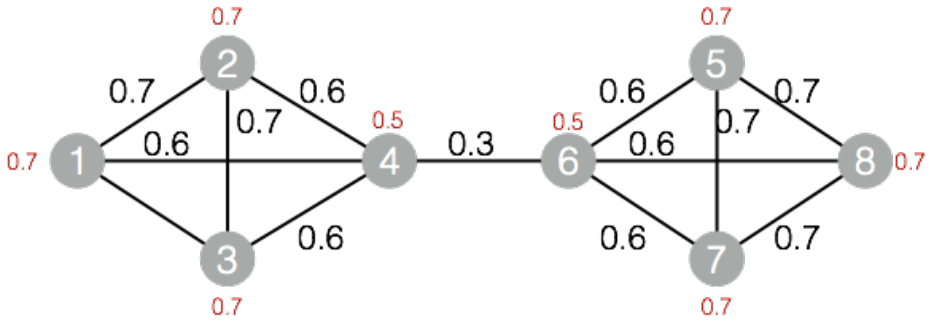
SQL statement for data generation:

```
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;
```

Structure of the group corresponding to the data:



Running result

```
+------+------------+
| node | group_id |
+------+------------+
| 1 | 1 |
| 2 | 1 |
```

```
|3|1|
|4|1|
|5|5|
|6|5|
|7|5|
|8|5|
+------+------------+
```

Label propagation classification

Function overview

Label propagation classification is a semi-supervised classification algorithm that uses the label information of labeled nodes to predict that of unlabeled nodes.

During algorithm execution, the labels of each node are propagated to the neighboring nodes based on the similarity between the nodes. In each step of propagation, a node updates its label based on the labels of the neighboring nodes such that the node is more similar with the neighboring nodes. The higher the similarity, the greater weight the neighboring nodes have on that node, and the easier for labels to be propagated. During label propagation, the labels of the labeled data remain unchanged, which serve as sources for propagation to the unlabeled data.

After the iterations end, the probability distributions of similar nodes are also similar such that the nodes can be divided into the same category to complete label propagation.

Parameter settings

- Damping factor: The default value is 0.8.
- Convergence factor: The default value is 0.000001.

PAI command

```
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;
```

Algorithm parameters

Parameter key	Description	Required/Optional	Default value
inputEdgeTableName	Name of the input edge table	Required	NA
inputEdgeTablePartitions	Partitions in the input edge table	Optional	Entire table
fromVertexCol	Start column in the input edge table	Required	NA
toVertexCol	End column in the input edge table	Required	NA
inputVertexTableName	Name of the input vertex table	Required	NA
inputVertexTablePartitions	Partitions in the input vertex table	Optional	Entire table
vertexCol	Vertex column in the input vertex table	Required	NA
vertexLabelCol	Vertex label column in the input vertex table	Required	NA
outputTableName	Name of the output table	Required	NA
outputTablePartitions	Partitions in the output table	Optional	NA
lifecycle	Life cycle of the output table	Optional	NA
workerNum	Worker count	Optional	Not set
workerMem	Worker memory size	Optional	4096
splitSize	Data split size	Optional	64
hasEdgeWeight	Indicates whether the edge of the input edge table has weight	Optional	false
edgeWeightCol	Edge weight column in the input edge table	Optional	NA
hasVertexWeight	Indicates whether the vertexes of the input vertex table have weights	Optional	false
vertexWeightCol	Vertex weight column in the input vertex table	Optional	NA
alpha	Damping factor	Optional	0.8
epsilon	Convergence factor	Optional	0.000001

maxIter	Maximum number of iterations	Optional	30

Example

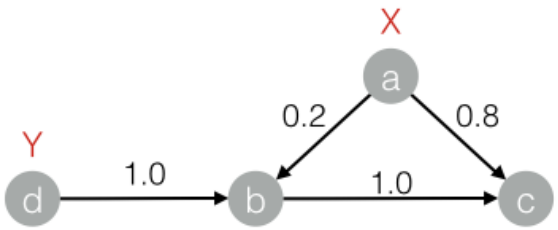
Test data

SQL statement for data generation:

```
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;
```

Structure of the graph corresponding to the data:



Running result

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

Modularity

Function overview

Modularity is a measure of the structure of networks. It measures the closeness of communities divided from a network structure. A value larger than 0.3 represents an obvious community structure.

PAI command

```
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;
```

Algorithm parameters

Parameter key	Description	Required/Optional	Default value
inputEdgeTableName	Name of the input edge table	Required	NA
inputEdgeTablePartitions	Partitions in the input edge table	Optional	Entire table
fromVertexCol	Start column in the input edge table	Required	NA
fromGroupCol	Group of the start node in the input edge table	Required	NA
toVertexCol	End column in the input edge table	Required	NA
toGroupCol	Group of the end vertex in the input edge table	Required	NA
outputTableName	Name of the output table	Required	NA
outputTablePartitions	Partitions in the output table	Optional	NA
lifecycle	Life cycle of the output table	Optional	NA
workerNum	Worker count	Optional	Not set
workerMem	Worker memory size	Optional	4096
splitSize	Data split size	Optional	64

Example

Test data

Same as the data in Label propagation clustering.

Running result

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

Maximum connected subgraphs

Function overview

In an undirected graph G, vertex A is connected to vertex B if a path exists between them. Graph G contains several subgraphs, where each vertex is connected to every other vertex in the same subgraph but is separated from those in other subgraphs. These subgraphs of graph G are maximum connected subgraphs.

Parameter settings

None

PAI command

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

Algorithm parameters

Parameter key	Description	Required/Optional	Default value
inputEdgeTableNam e	Name of the input edge table	Required	NA
inputEdgeTableParti tions	Partitions in the input edge table	Optional	Entire table
fromVertexCol	Start column in the input edge table	Required	NA
toVertexCol	End column in the	Required	NA

	input edge table		
outputTableName	Name of the output table	Required	NA
outputTablePartitions	Partitions in the output table	Optional	NA
lifecycle	Life cycle of the output table	Optional	NA
workerNum	Worker count	Optional	Not set
workerMem	Worker memory size	Optional	4096
splitSize	Data split size	Optional	64

Example

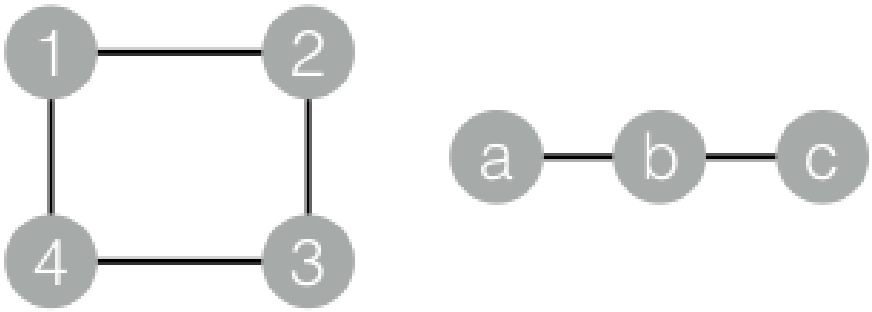
Test data

SQL statement for data generation:

```
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
);
```

Structure of the graph corresponding to the data:



Running result

```
+-------+-------+
| node | grp_id|
+-------+-------+
| 1 | 4 |
| 2 | 4 |
| 3 | 4 |
| 4 | 4 |
| a | c |
| b | c |
| c | c |
+-------+-------+
```

Node clustering coefficient

Function overview

This coefficient is used to calculate the peripheral density of a node in an undirected graph G. The density of a star network is 0, and that of a fully meshed network is 1.

Parameter settings

maxEdgeCnt: If the node degree is greater than the value of this parameter, sampling is performed. This parameter is optional, and the default value is 500.

PAI command

```
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;
```

Algorithm parameters

Parameter key	Description	Required/Optional	Default value
inputEdgeTableName	Name of the input edge table	Required	NA
inputEdgeTablePartitions	Partitions in the input edge table	Optional	Entire table
fromVertexCol	Start column in the input edge table	Required	NA
toVertexCol	End column in the input edge table	Required	NA
outputTableName	Name of the output table	Required	NA
outputTablePartitions	Partitions in the output table	Optional	NA
lifecycle	Life cycle of the output table	Optional	NA
maxEdgeCnt	If the node degree is greater than the value of this parameter, sampling is performed.	Optional	500
workerNum	Worker count	Optional	Not set
workerMem	Worker memory size	Optional	4096
splitSize	Data split size	Optional	64

Example

Test data

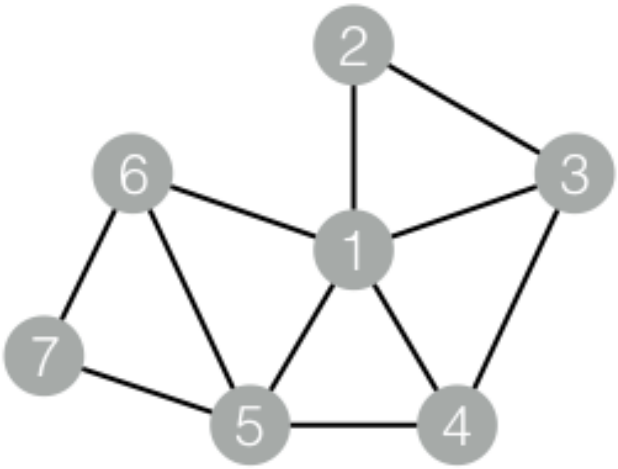
SQL statement for data generation:

```
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
);
```

Structure of the graph corresponding to the data:



Running result

```
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
```

Edge clustering coefficient

Function overview

This coefficient is used to calculate the peripheral density of each edge in an undirected graph G.

Parameter settings

None

PAI command

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

Algorithm parameters

Parameter key	Description	Required/Optional	Default value
inputEdgeTableName	Name of the input edge table	Required	NA
inputEdgeTablePartitions	Partitions in the input edge table	Optional	Entire table
fromVertexCol	Start column in the input edge table	Required	NA
toVertexCol	End column in the input edge table	Required	NA
outputTableName	Name of the output table	Required	NA
outputTablePartitions	Partitions in the output table	Optional	NA
lifecycle	Life cycle of the output table	Optional	NA
workerNum	Worker count	Optional	Not set
workerMem	Worker memory size	Optional	4096
splitSize	Data split size	Optional	64

Example

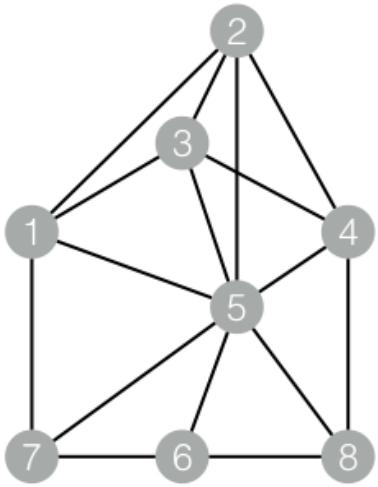
Test data

SQL statement for data generation:

```
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
);
```

Structure of the graph corresponding to the data:



Running result

```
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
```

Triangle count

Function overview

Output all triangles in an undirected graph G.

Parameter settings

maxEdgeCnt: If the node degree is greater than the value of this parameter, sampling is performed. This parameter is optional, and the default value is 500.

PAI command

```
PAI -name TriangleCount
```

```
-project algo_public
-DinputEdgeTableName=TriangleCount_func_test_edge
-DfromVertexCol=flow_out_id
-DtoVertexCol=flow_in_id
-DoutputTableName=TriangleCount_func_test_result;
```

Algorithm parameters

Parameter key	Description	Required/Optional	Default value
inputEdgeTableName	Name of the input edge table	Required	NA
inputEdgeTablePartitions	Partitions in the input edge table	Optional	Entire table
fromVertexCol	Start column in the input edge table	Required	NA
toVertexCol	End column in the input edge table	Required	NA
outputTableName	Name of the output table	Required	NA
outputTablePartitions	Partitions in the output table	Optional	NA
lifecycle	Life cycle of the output table	Optional	NA
maxEdgeCnt	If the node degree is greater than the value of this parameter, sampling is performed.	Optional	500
workerNum	Worker count	Optional	Not set
workerMem	Worker memory size	Optional	4096
splitSize	Data split size	Optional	64

Example

Test data

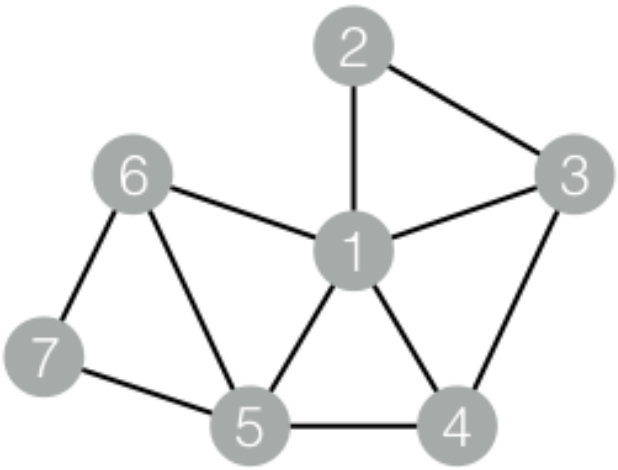
SQL statement for data generation:

```
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
);
```

Structure of the graph corresponding to the data:



Running result

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

Decision tree depth

Function overview

Output depth and tree ID of each node in a network composed of many trees.

Parameter settings

None

PAI command

```
PAI -name TreeDepth
-project algo_public
-DinputEdgeTableName=TreeDepth_func_test_edge
-DfromVertexCol=flow_out_id
-DtoVertexCol=flow_in_id
-DoutputTableName=TreeDepth_func_test_result;
```

Algorithm parameters

Parameter key	Description	Required/Optional	Default value
inputEdgeTableName	Name of the input edge table	Required	NA
inputEdgeTablePartitions	Partitions in the input edge table	Optional	Entire table
fromVertexCol	Start column in the input edge table	Required	NA
toVertexCol	End column in the input edge table	Required	NA
outputTableName	Name of the output table	Required	NA
outputTablePartitions	Partitions in the output table	Optional	NA
lifecycle	Life cycle of the output table	Optional	NA
workerNum	Worker count	Optional	Not set
workerMem	Worker memory size	Optional	4096
splitSize	Data split size	Optional	64

Example

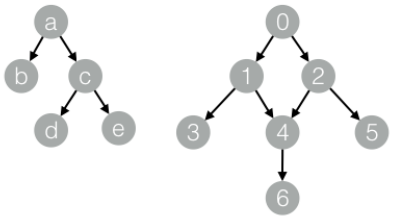
Test data

SQL statement for data generation:

```
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
);
```

Structure of the graph corresponding to the data:



Running result

```
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
```

Tools

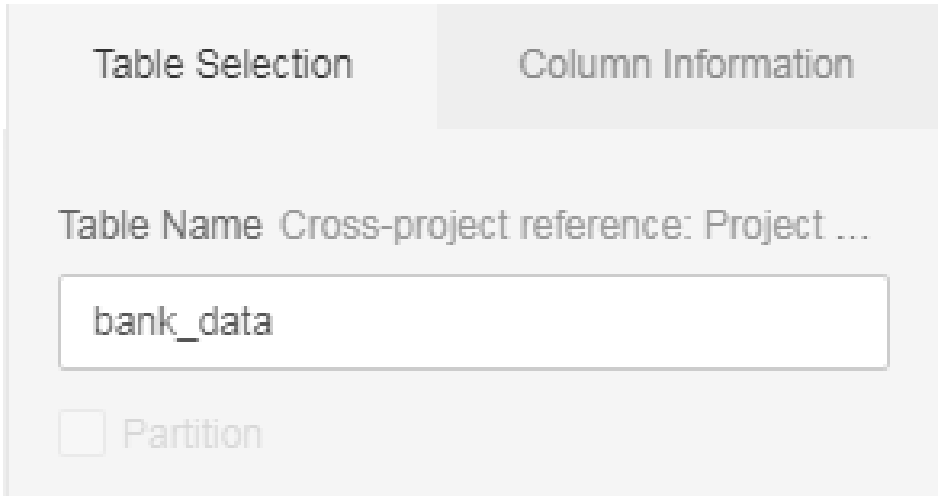
Contents

SQL scripts

You can use the SQL script editor to write SQL syntaxes. For more information, see MaxCompute SQL Introduction.

Demo

Drag a MaxCompute source component to the canvas and enter the table name in the right pane, as shown in the following figure.

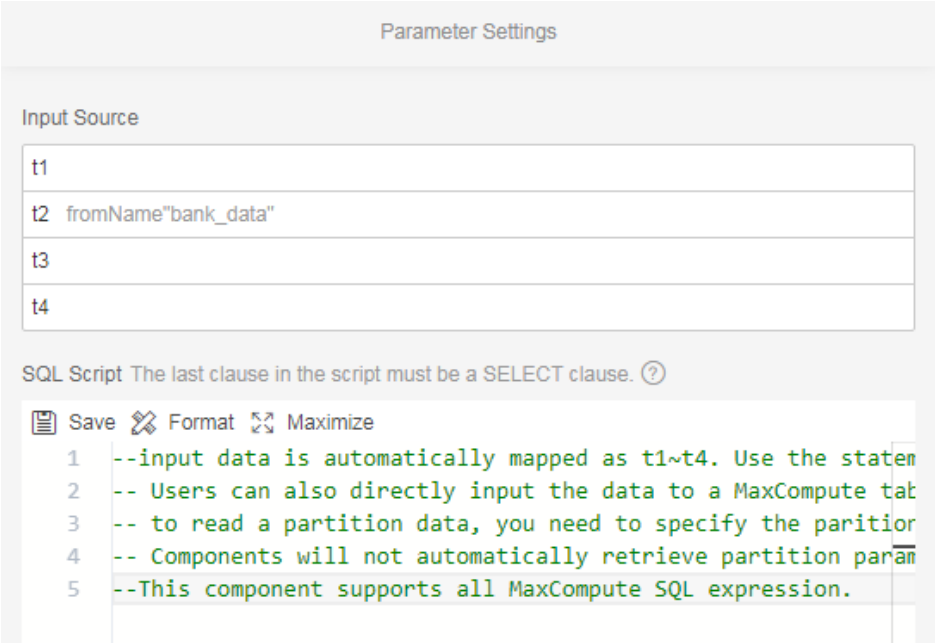


A SQL Script component supports a maximum of two input ports.

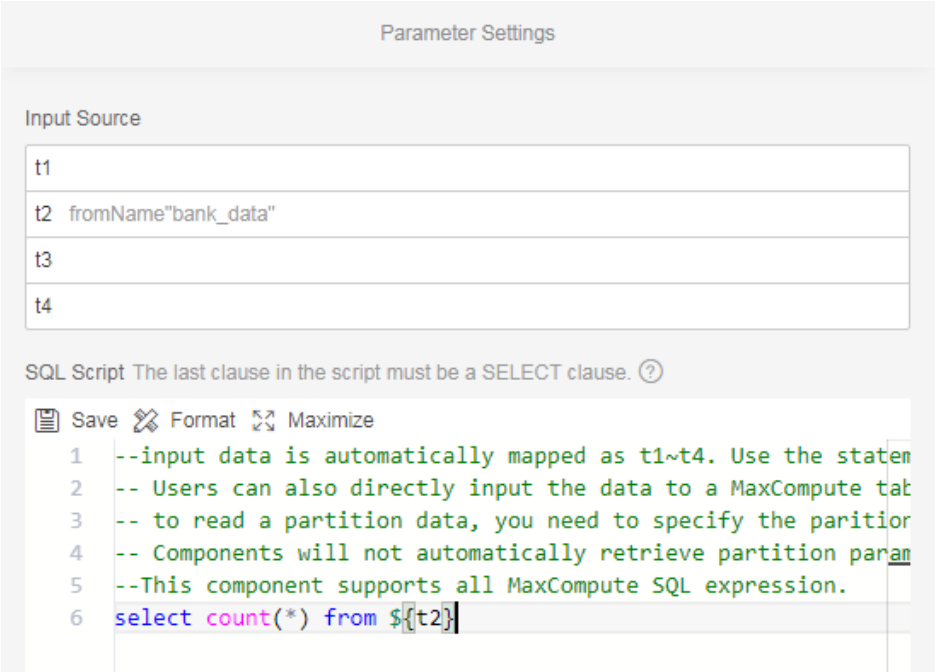
If a partitioned table is input, the system automatically selects the Partition option for the table. You can then select a partition or enter a partition name. Currently, only one partition can be specified. If the Partition option is not selected or no partition name is specified, the system determines that a full table is input.

The Partition option is unavailable when an unpartitioned table is input.

Drag a SQL script component to the canvas, connect the component to the MaxCompute source component, and click the SQL Script component. The parameters appear, as shown in the following figure.



Enter SQL syntaxes, as shown in the following figure:



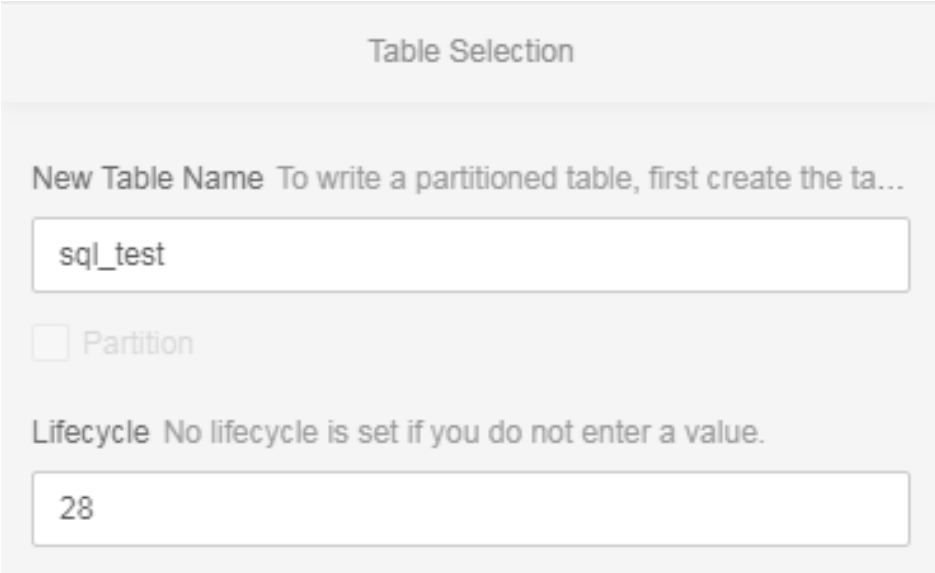
A SQL Script Component supports a maximum of two input ports and one output port.

You can enter only one SQL syntax, for details see SQL Expressions.

Input tables are mapped to t1 and t2, respectively. You can use ${t1} or ${t2} to make a call to the relevant table without a need to specify the table name.

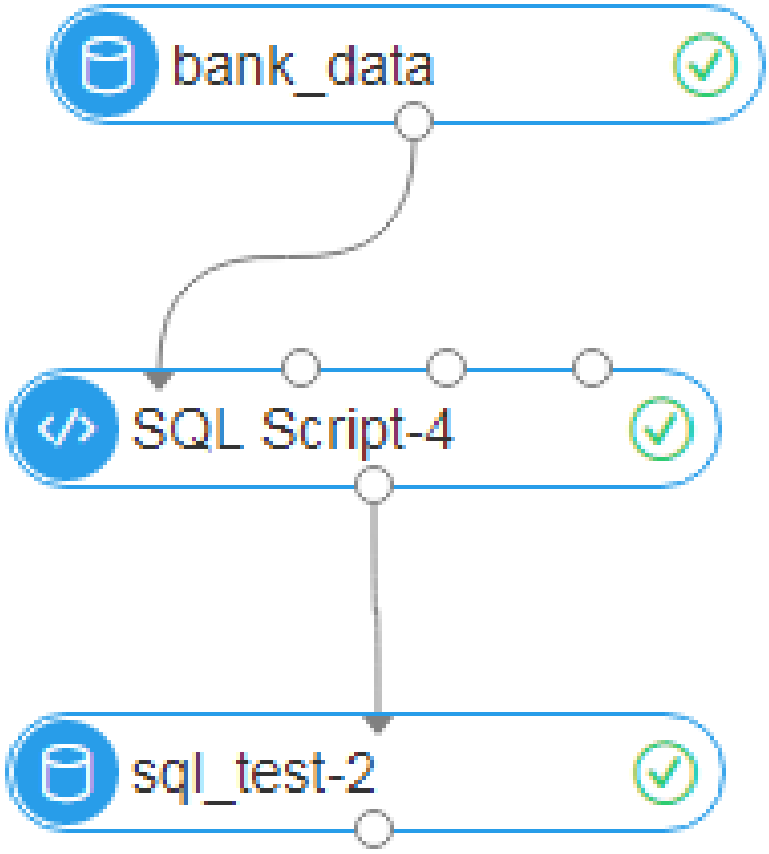
The SQL script in the preceding figure is used to calculate the number of rows in the table.

Drag a MaxCompute target component to the Canvas and enter a new table name, as shown in the following figure. The system then automatically creates the table. If you want to input a partitioned table, you must enter partition information.



Currently, partitioned tables are not supported.

Connect all components and then click Run.



5. Right-click the MaxCompute target component and select View Data.

PAI commands

No PAI commands are available.

Financial algorithms

Contents

Binning

Binning with equal frequency or equal width.

PAI command

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

Parameter description

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input table	table name	NA
outputTableName	(Required) Name of the output table	table name	NA
selectedColNames	(Optional) Binning columns in the input table	column name	All columns except the label column, or all columns when no label exists
labelColumn	(Optional) Label column	column name	No label
validTableName	(Optional) Validation table name entered when binningMethod is	table name	Null

	auto. Required in auto mode.		
validTablePartitions	(Optional) Partitions selected for the validation table	partition name	Entire table selected
inputTablePartitions	(Optional) Names of the selected partitions in the input table	partition name	All partitions are selected by default.
inputBinTableName	(Optional) Input binning table	table name	No binning table
selectedBinColNames	(Optional) Columns selected in the binning table	column name	Null
positiveLabel	(Optional) Type of output positive samples	NA	1
nDivide	(Optional) Number of bins	positive integer	10
colsNDivide	(Optional) Number of bins in custom columns, for example, col0:3 and col2:5 The columns that are selected in colsNDivide but are not included in selectedColNames are also calculated. For example, the value is calculated based on col0:3, col1:10, and col2:5 if selectedColNames is col0 and col1, colsNDivide is col0:3 and col2:5, and nDivide is 10.	positive integer	Null
isLeftOpen	(Optional) Interval open mode: left open and right closed, or left closed and right open	true/false	true
stringThreshold	(Optional) Threshold of discrete values in other bins	NA	No other bins by default
colsStringThreshold	(Optional) Threshold of custom columns, the same as colsNDivide	NA	Null

binningMethod	(Optional) Binning method. quantile: equal frequency binning; bucket: equal width binning; auto: monotonous binning automatically selected in quantile mode	quantile, bucket, and auto	quantile
lifecycle	(Optional) Life cycle of the output table	positive integer	Unspecified by default
coreNum	(Optional) Number of cores	positive integer	Automatically calculated by default
memSizePerCore	(Optional) Size of memory	positive integer	Automatically calculated by default

Data conversion module

Parameter description

Parameter	Description
inputFeatureTableName	(Required) Name of the input feature table
inputBinTableName	(Required) Name of the input binning result table
inputFeatureTablePartitions	(Optional) Partitions selected in the input feature table. Entire table selected by default.
outputTableName	(Required) Name of the output table
featureColNames	(Optional) Names of feature columns selected in the input table. Entire table selected by default.
metaColNames	(Optional) Columns without data conversion. The selected columns are output without any change. (No meta column by default, and you can specify label, sample_id, and other columns.)
transformType	(Optional) Data conversion method, which is normalize (normalization), dummy (discretization), or woe (WOE conversion). The default value is “dummy”.
itemDelimiter	(Optional) Feature delimiter. Default value: comma. Valid only for discretization.
kvDelimiter	(Optional) KV delimiter. Default value: colon. Valid only for discretization.
lifecycle	(Optional) Output table lifecycle. No lifecycle

	by default.
coreNum	(Optional) Number of CPU cores used (calculated automatically by default)
memSizePerCore	(Optional) Memory size used by each CPU core, in MB. Calculated automatically by default.

Example

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

Normalization algorithm

Normalization converts variable values into values between 0 and 1 based on input binning information, and sets missing values to 0. The following algorithm is used.

```
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
```

Output format

Normalization and WOE conversion output tables are normal tables.

When variables are converted into dummy variables through discretization, the output table is in KV format and the output variable format is [${feaname}]\_bin\_${bin_id}. For example:

- If the sns variable falls into bin 2, the output variable is [sns]_bin_2.
- If the variable is null, it falls into the null bin and the output variable is [sns]_bin_null.
- If the variable is not null and does not fall into any defined bin, it falls into the else bin and the output variable is [sns]_bin_else.

Scorecard training

The scorecard is a modeling tool commonly used in credit risk evaluation. Scorecard training discretizes the original variables by binning input and uses a linear model (such as logistic regression

and linear regression) to carry out model training. The scorecard supports various features, including feature selection and score conversion. In addition, you can add constraints to variables during model training.

Note: If no binning input is specified, scorecard training is completely equivalent to common logistic regression or linear regression.

Feature engineering

The scorecard model differs from ordinary linear models in that it performs feature engineering on the data before the linear model is used for training. The scorecard provides two feature engineering modes, both of which need to discretize the features through binning.

- One method is to generate N dummy variables (N is the number of variable bins) for each variable through One-Hot encoding based on the binning result.
- The other is WOE conversion that replaces the original values of variables with the WOE values of the bins where the variables fall.

Note: When using dummy variables for conversion, you can set constraints between dummy variables of each original variable. For more information, see the subsequent chapters.

Score conversion

In scorecard scenarios such as credit scoring, the odds of predicted samples must be transformed into scores by linear transformation, typically as follows:

log(odds) = \sum(w*x) = a * scaled_score + b

Specify the linear transformation relationship by using the following parameters:

- scaledValue: Specifies a benchmark for a score.
- odds: Specifies the odds value at a given score benchmark.
- pdo(Point Double Odds): Doubles the odds for multiple scores.

For example, if scaledValue is 800, odds is 50, and pdo is 25, the two points in a straight line are specified.

log(40) = a * 800 + b

log(80) = a * 825 + b

Calculate a and b, and conduct linear transformation on the scores in the model to obtain the transformed variables.

The scaling information is specified by the **-Dscale** parameter in the JSON format, as follows:

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

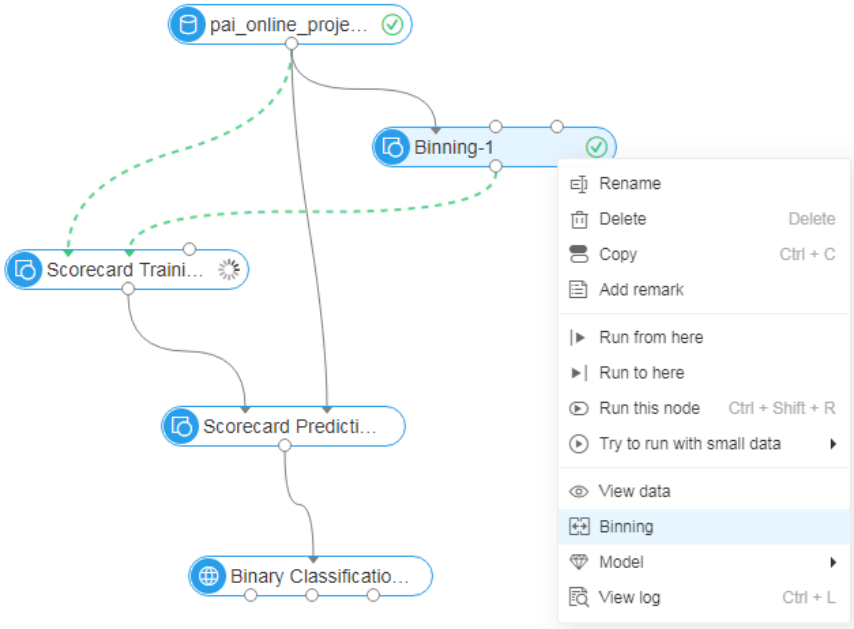
When this parameter is not null, set the values of all the preceding three parameters.

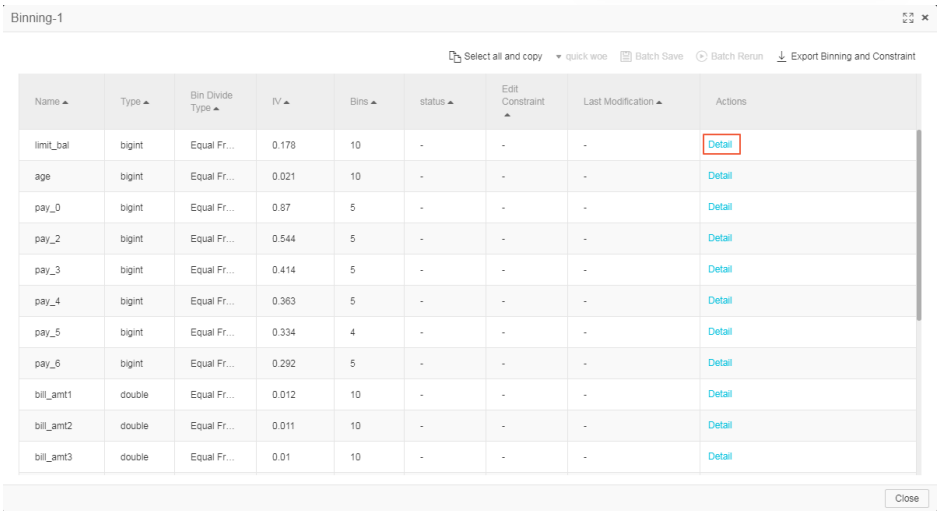
Support constraints in the training process

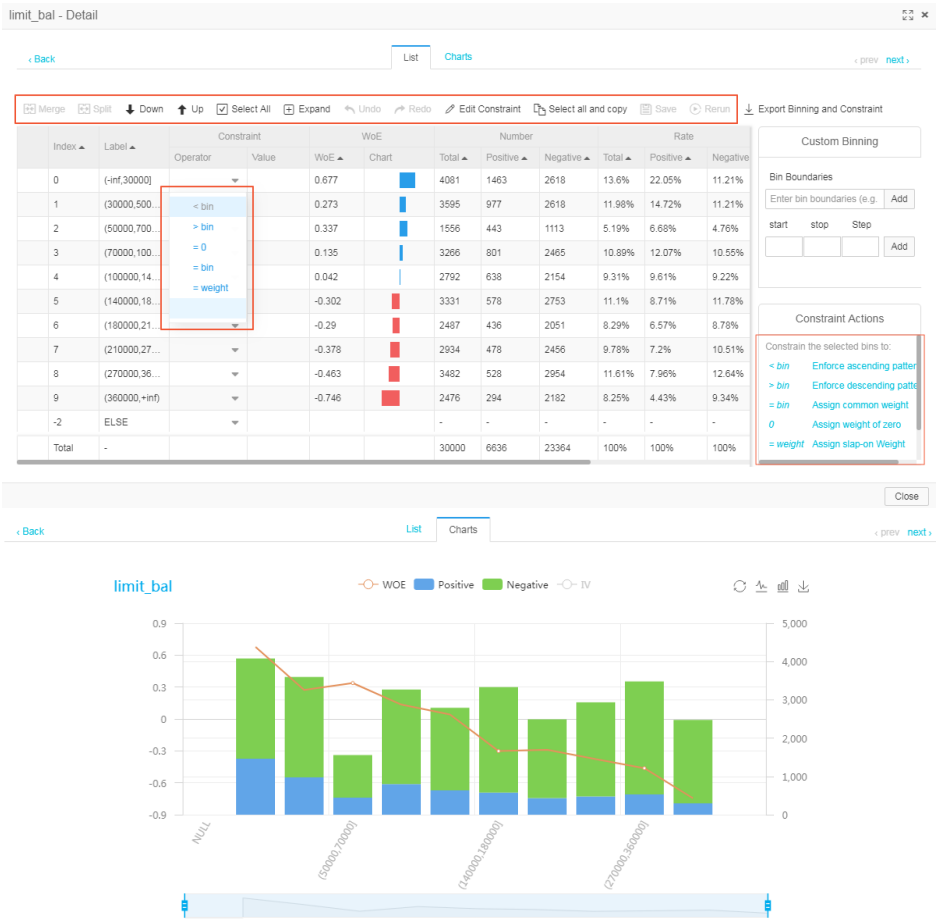
In the scorecard training, you can add constraints to variables. The implementation of constraints depends on the underlying constrained optimization algorithm. Set the constraints in the binning UI. After the setting, the bin generates a constraint in JSON format and automatically transfers it to the training components connected. The scorecard training component supports the following method to add constraints to variables:

- Specify the score of a certain bin to be a fixed value.
- Specify the scores of two bins to a certain proportion.
- Limit the scores between bins, such as sorting bin scores by the WOE values in the bins.

The following is a demonstration of binning component operations:







Currently, the following JSON constraints are supported:

- "<" : The variable weights satisfy constraints in ascending order.
- "<" : The variable weights satisfy constraints in descending order.
- "=" : The variable weights are fixed values.
- "%" : The weights between variables meet a proportional relationship.
- "UP" : upper threshold of variable weight constraints.
- "LO" : lower threshold of variable weight constraints.

The JSON constraints are stored as a character string in a table. The table is single-row and single-column (character string), storing the following JSON character strings:

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

```
"%": [
["6:1.0","7:1.0"]
]
}
```

Built-in constraints

Each original variable has an implicit constraint, which does not need to be specified by the user. That is, the average score of a single variable’ s population is 0. With this constraint, scaled_weight of the model intercept is the average score for the entire population.

Optimization algorithm

In the advanced options, you can select the optimization algorithm used in the training. Currently, the following optimization algorithms are supported:

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

L-BFGS is a first-order optimization algorithm that supports large-scale feature data. Newton’ s method is a classic second-order algorithm that features fast convergence and high accuracy, but it is not suitable for large feature sizes because the second-order Hessian Matrix must be calculated. Both of the algorithms are unconstrained optimization algorithms. When these two optimization algorithms are selected, the constraints are automatically ignored.

Algorithm	Sort	Feature	Others
L-BFGS	first-order algorithm, unconstrained	Supports large-scale feature data.	When the optimization algorithms are selected, the constraints are automatically ignored.
Newton’ s Method	second-order algorithm, unconstrained	Features fast convergence and high accuracy	It is not suitable for large feature sizes because the second-order Hessian Matrix must be calculated. When the optimization algorithms are selected, the constraints are automatically ignored.
Barrier Method	second-order algorithm, constrained	The algorithms is similar with SQP in computational performance and	Equivalent to Newton’ s method when no constraints are involved.

		accuracy. We recommend SQP by default.	
SQP	second-order algorithm, unconstrained	The algorithms is similar with Barrier Method in computational performance and accuracy. We recommend SQP by default.	Equivalent to Newton’ s method when no constraints are involved.

If you are unfamiliar with optimization algorithms, we recommend the default option “Automatic select”. An appropriate optimization algorithm is automatically selected based on the data size and constraints of user tasks.

Feature selection

The training module supports stepwise feature selection. Stepwise is a combination of forward selection and backward selection. After a new variable is selected to enter the model in each forward feature selection, backward selection needs to be performed on the variables in the model to remove the variables whose significance does not meet requirements. Because multiple objective functions and multiple feature transformation methods are supported, stepwise feature selection supports different selection criteria:

- Marginal contribution: Applies to all objective functions and feature engineering modes.
- Score test: Supports only WOE conversion or logistic regression selection without feature engineering.
- F test: Supports only WOE conversion or linear regression selection without feature engineering.

Marginal contribution

Concept

Two models need to be trained. Model A does not include variable X, and model B includes variable X in addition to all variables of model A. The difference of objective functions in the final convergence of the two models is the marginal contribution between all variables of variable X in model B.

In the scenario where feature engineering is a dummy transformation, the marginal contribution of X of the original variable is defined as the difference between objective functions of all the dummy variables when the two models respectively include and do not include the variable.

Feature selection using marginal contribution supports all feature engineering modes.

Feature

The advantage is that this method is flexible and is not limited to a certain model. The variables with an optimal objective function are directly selected to enter the model.

The disadvantage is that marginal contribution is different from statistical significance. The threshold of statistical significance is typically 0.05, while no fixed threshold of marginal contribution is available for new users. The recommended default value is 10E-5.

Score test

Notice: The score test is applicable only to feature selection of logistic regression.

Forward selection:

Training a model with only the intercept firstly.

In each subsequent iteration, the **Score Chi-Square** for the variables that have not entered the model is calculated, the variable with the largest **Score Chi-Square** is selected to enter the model.

calculating the significance P Value based on the **Score Chi-Square**.

If the P Value of the variable with the largest **Score Chi-Square** is greater than the maximum significance threshold (slentry) specified for entering the model, the variable is not included in the model and the selection stops.

Else, go to forward selection.

Forward selection:

The backward selection is performed on variables that have been selected to enter the model.

In the backward selection, the **Wald Chi-Square** and the significance P Value are calculated for variables that have entered the model.

If P Value is greater than the maximum significance threshold (slstay) specified for removal, the variable is removed from the model and the iteration selection starts.

Else, including the variable in the model and start iteration selection.

F test

Notice: The F test is applicable only to feature selection of linear regression.

Forward selection

Training variable with only the intercept firstly.

In each subsequent iteration, F Values of the variables that have not entered the model are calculated.

F Value calculation is similar to edge contribution calculation. Two models are trained to calculate F Value of a variable.

If F Value meets F distribution, the significance P Value can be calculated based on the probability density function of its F distribution.

If P Value is greater than the maximum significance threshold (slentry) specified for entering the model, the variable is not included in the model and the selection stops.

Else, go to forward selection.

Backward selection

Uses F Value to calculate the significance, and the process is similar to the score test.

Variables forcedly selected

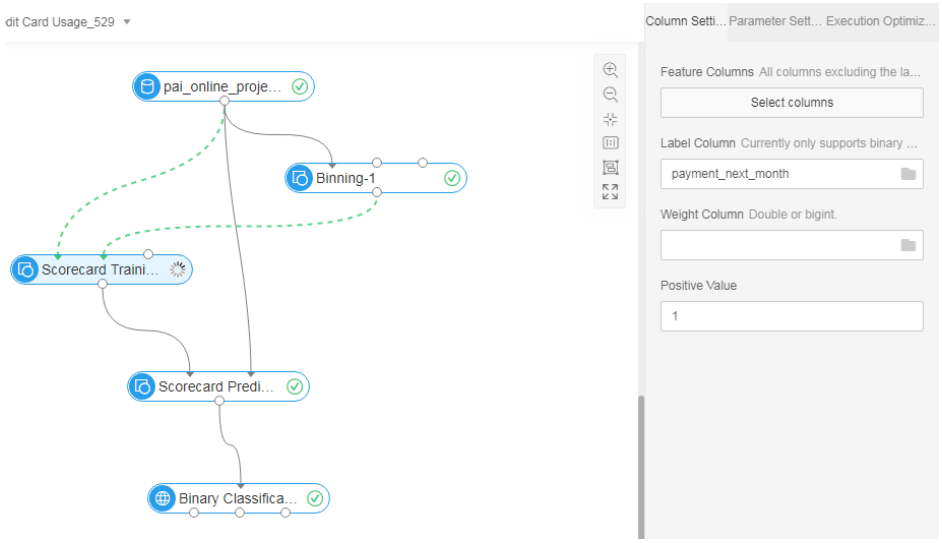
Before feature selection, you can set variables that are forced into the model. Regardless of the significance, the selected variables directly enter the model and do not participate in the forward and backward feature selections.

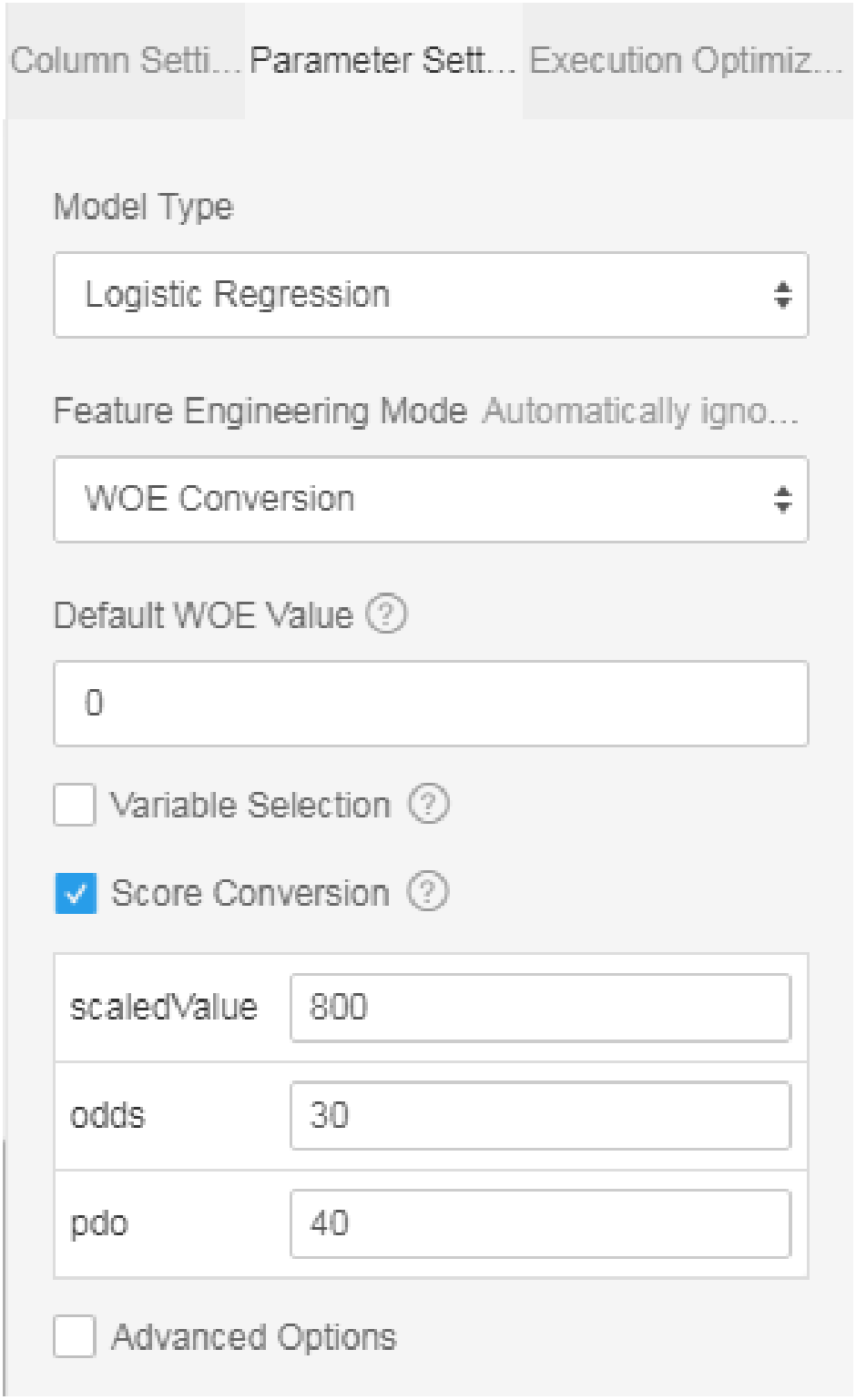
The number of iterations and significance threshold are specified in CLI by using the **-Dselected** parameter in the JSON format:

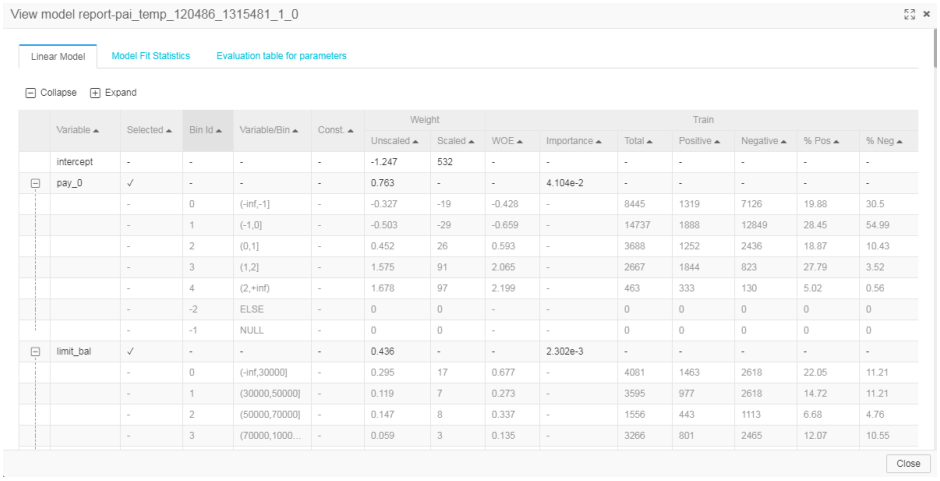
```
{"max_step":2, "slentry": 0.0001, "slstay": 0.0001}
```

If this parameter is null or **max_step** is “0”, normal training is performed without feature selection.

We recommend that you use the scorecard on the PAI web. The following is a simple demonstration of comparison between STEPWISE feature selection and feature WOE transformation + logistic regression STEPWISE feature selection:


dit Card Usage_529
pai_online_proje...
Binning-1
Scorecard Traini...
Scorecard Predi...
Binary Classifica...
Column Setti...
Parameter Sett...
Execution Optimiz...
Feature Columns
All columns excluding the la...
Select columns
Label Column
Currently only supports binary ...
payment_next_month
Weight Column
Double or bigint.
Positive Value
1


Column Setti...
Parameter Sett...
Execution Optimiz...
Model Type
Logistic Regression
Feature Engineering Mode
Automatically igno...
WOE Conversion
Default WOE Value
0
Variable Selection
Score Conversion
scaledValue
800
odds
30
pdo
40
Advanced Options



Variable	Selected	Bin Id	Variable/Bin	Const.	Weight				Train				
					Unscaled	Scaled	WOE	Importance	Total	Positive	Negative	% Pos	% Neg
intercept	-	-	-	-	-1.247	532	-	-	-	-	-	-	-
pay_0	✓	-	-	-	0.763	-	-	4.104e-2	-	-	-	-	-
	-	0	(-inf,-1]	-	-0.327	-19	-0.428	-	8445	1319	7126	19.88	30.5
	-	1	(-1,0]	-	-0.503	-29	-0.659	-	14737	1888	12849	28.45	54.99
	-	2	(0,1]	-	0.452	26	0.593	-	3688	1252	2436	18.87	10.43
	-	3	(1,2]	-	1.575	91	2.065	-	2667	1844	823	27.79	3.52
	-	4	(2,+inf)	-	1.678	97	2.199	-	463	333	130	5.02	0.56
	-	-2	ELSE	-	0	0	-	-	0	0	0	0	0
	-	-1	NULL	-	0	0	-	-	0	0	0	0	0
limit_bal	✓	-	-	-	0.436	-	-	2.302e-3	-	-	-	-	-
	-	0	(-inf,30000]	-	0.295	17	0.677	-	4081	1463	2618	22.05	11.21
	-	1	(30000,50000]	-	0.119	7	0.273	-	3595	977	2618	14.72	11.21
	-	2	(50000,70000]	-	0.147	8	0.337	-	1556	443	1113	6.68	4.76
	-	3	(70000,1000...	-	0.059	3	0.135	-	3266	801	2465	12.07	10.55

Model report

The scorecard model output is a model report, which contains variable binning information, binning constraint information, basic statistical indicators such as WOE and marginal contribution, and scorecard model assessment reports displayed on the PAI web.

The related columns are described as follows:

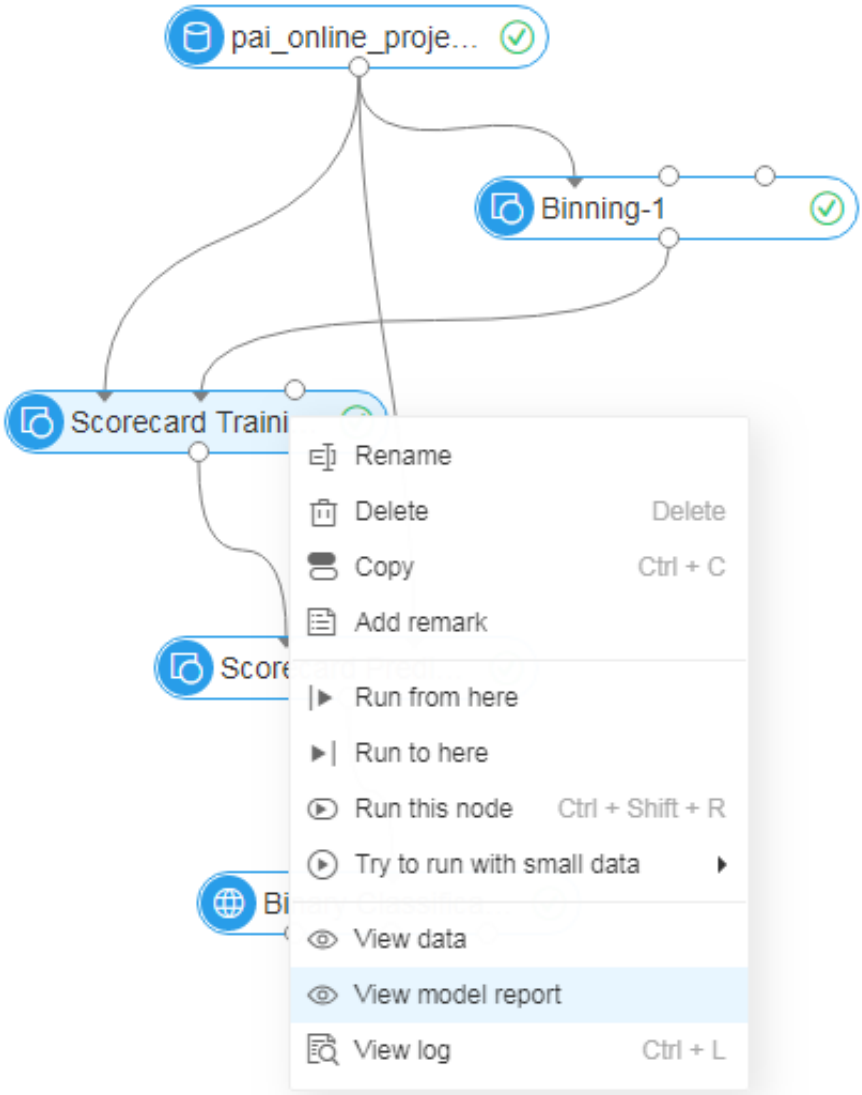
Column Name	Column Type	Column Description
feaname	string	Feature name
binid	bigint	Bin ID
bin	string	Bin description, used to indicate the value range of the bin
constraint	string	Constraints added to the bin during training
weight	double	Bin variable weight after training or model variable weight in a non-scorecard model with no bin input specified
scaled_weight	double	Score value obtained after linear transformation of the bin variable weight when the score conversion information is specified in scorecard model training
Woe	double	Statistical indicator: WOE value of the bin in the training set
Contribution	double	Statistical indicator: marginal contribution value of the bin in the training set
total	bigint	Statistical indicator: total

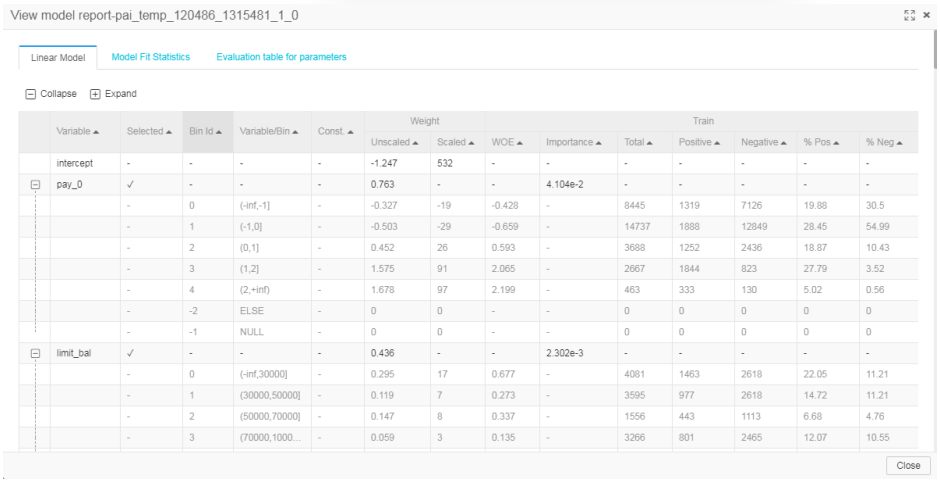
		number of samples in the bin in the training set
positive	bigint	Statistical indicator: number of positive samples in the bin in the training set
negative	bigint	Statistical indicator: number of negative samples in the bin in the training set
percentage_pos	double	Statistical indicator: ratio of the number of positive samples to the total number of samples in the bin in the training set
percentage_neg	double	Statistical indicator: ratio of the number of negative samples to the total number of samples in the bin in the training set
test_woe	double	Statistical indicator: WOE value of the bin in the test set
test_contribution	double	Statistical indicator: marginal contribution value of the bin in the test set
test_total	bigint	Statistical indicator: total number of samples in the bin in the test set
test_positive	bigint	Statistical indicator: number of positive samples in the bin in the test set
test_negative	bigint	Statistical indicator: number of negative samples in the bin in the test set
test_percentage_pos	double	Statistical indicator: ratio of the number of positive samples to the total number of samples in the bin in the test set
test_percentage_neg	double	Statistical indicator: ratio of the number of negative samples to the total number of samples in the bin in the test set

Demonstration

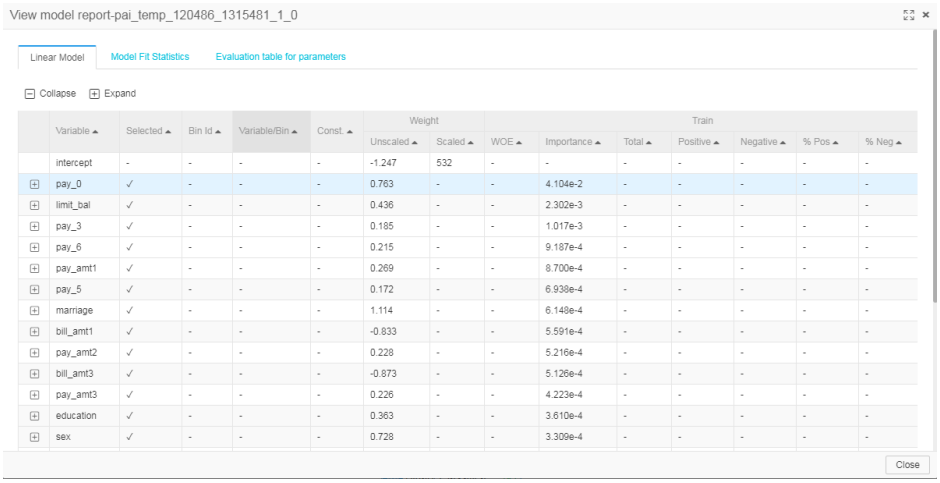
The following is a simple demonstration of comparison between scorecard training and feature WOE

conversion + logistic regression.



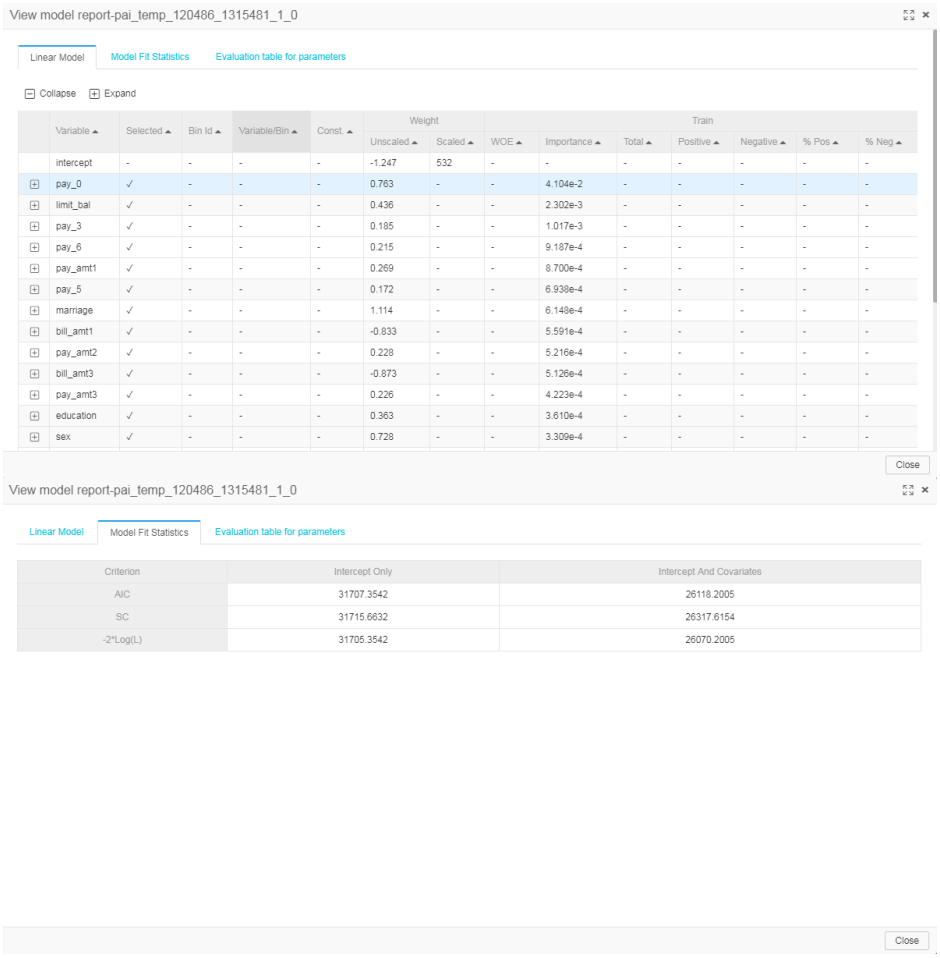


Variable	Selected	Bin Id	Variable/Bin	Const.	Weight		Train						
					Unscaled	Scaled	WOE	Importance	Total	Positive	Negative	% Pos	% Neg
intercept	-	-	-	-	-1.247	532	-	-	-	-	-	-	-
pay_0	✓	-	-	-	0.763	-	-	4.104e-2	-	-	-	-	-
	-	0	(-inf,-1)	-	-0.327	-19	-0.428	-	8445	1319	7126	19.88	30.5
	-	1	(-1,0]	-	-0.503	-29	-0.659	-	14737	1888	12849	28.45	54.99
	-	2	(0,1]	-	0.452	26	0.593	-	3688	1252	2436	18.87	10.43
	-	3	(1,2]	-	1.575	91	2.065	-	2667	1844	823	27.79	3.52
	-	4	(2,+inf)	-	1.678	97	2.199	-	463	333	130	5.02	0.56
	-	-2	ELSE	-	0	0	-	-	0	0	0	0	0
	-	-1	NULL	-	0	0	-	-	0	0	0	0	0
limit_bal	✓	-	-	-	0.436	-	-	2.302e-3	-	-	-	-	-
	-	0	(-inf,30000]	-	0.295	17	0.677	-	4081	1463	2618	22.05	11.21
	-	1	(30000,50000]	-	0.119	7	0.273	-	3595	977	2618	14.72	11.21
	-	2	(50000,70000]	-	0.147	8	0.337	-	1556	443	1113	6.68	4.76
	-	3	(70000,1000...	-	0.059	3	0.135	-	3266	801	2465	12.07	10.55



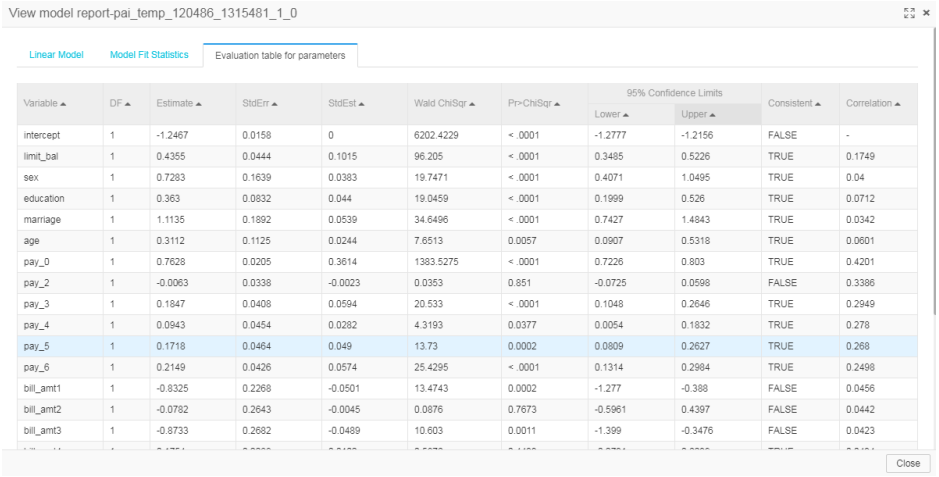
	Variable	Selected	Bin Id	Variable/Bin	Const.	Weight		Train						
						Unscaled	Scaled	WOE	Importance	Total	Positive	Negative	% Pos	% Neg
	intercept	-	-	-	-	-1.247	532	-	-	-	-	-	-	-
⊞	pay_0	✓	-	-	-	0.763	-	-	4.104e-2	-	-	-	-	-
⊞	limit_bal	✓	-	-	-	0.436	-	-	2.302e-3	-	-	-	-	-
⊞	pay_3	✓	-	-	-	0.185	-	-	1.017e-3	-	-	-	-	-
⊞	pay_6	✓	-	-	-	0.215	-	-	9.187e-4	-	-	-	-	-
⊞	pay_amt1	✓	-	-	-	0.269	-	-	8.700e-4	-	-	-	-	-
⊞	pay_5	✓	-	-	-	0.172	-	-	6.938e-4	-	-	-	-	-
⊞	marriage	✓	-	-	-	1.114	-	-	6.148e-4	-	-	-	-	-
⊞	bill_amt1	✓	-	-	-	-0.833	-	-	5.591e-4	-	-	-	-	-
⊞	pay_amt2	✓	-	-	-	0.228	-	-	5.216e-4	-	-	-	-	-
⊞	bill_amt3	✓	-	-	-	-0.873	-	-	5.126e-4	-	-	-	-	-
⊞	pay_amt3	✓	-	-	-	0.226	-	-	4.223e-4	-	-	-	-	-
⊞	education	✓	-	-	-	0.363	-	-	3.610e-4	-	-	-	-	-
⊞	sex	✓	-	-	-	0.728	-	-	3.309e-4	-	-	-	-	-

When the test set is connected to the input training component, the output model report shows statistical indicators of the model in the test set, such as WOE and MC. The following is a simple training demonstration of a test set:



	Variable	Selected	Bin Id	Variable/Bin	Const.	Weight		Train						
						Unscaled	Scaled	WOE	Importance	Total	Positive	Negative	% Pos	% Neg
	intercept	-	-	-	-	-1.247	532	-	-	-	-	-	-	-
⊞	pay_0	✓	-	-	-	0.763	-	-	4.104e-2	-	-	-	-	-
⊞	limit_bal	✓	-	-	-	0.436	-	-	2.302e-3	-	-	-	-	-
⊞	pay_3	✓	-	-	-	0.185	-	-	1.017e-3	-	-	-	-	-
⊞	pay_6	✓	-	-	-	0.215	-	-	9.187e-4	-	-	-	-	-
⊞	pay_amt1	✓	-	-	-	0.269	-	-	8.700e-4	-	-	-	-	-
⊞	pay_5	✓	-	-	-	0.172	-	-	6.938e-4	-	-	-	-	-
⊞	marriage	✓	-	-	-	1.114	-	-	6.148e-4	-	-	-	-	-
⊞	bill_amt1	✓	-	-	-	-0.833	-	-	5.591e-4	-	-	-	-	-
⊞	pay_amt2	✓	-	-	-	0.228	-	-	5.216e-4	-	-	-	-	-
⊞	bill_amt3	✓	-	-	-	-0.873	-	-	5.126e-4	-	-	-	-	-
⊞	pay_amt3	✓	-	-	-	0.226	-	-	4.223e-4	-	-	-	-	-
⊞	education	✓	-	-	-	0.363	-	-	3.610e-4	-	-	-	-	-
⊞	sex	✓	-	-	-	0.728	-	-	3.309e-4	-	-	-	-	-

Criterion	Intercept Only	Intercept And Covariates
AIC	31707.3542	26118.2005
SC	31715.6632	26317.6154
-2*Log(L)	31705.3542	26070.2005



Variable	DF	Estimate	StdErr	StdEst	Wald ChiSqr	Pr>ChiSqr	95% Confidence Limits		Consistent	Correlation
							Lower	Upper		
intercept	1	-1.2467	0.0158	0	6202.4229	< .0001	-1.2777	-1.2156	FALSE	-
limit_bal	1	0.4355	0.0444	0.1015	96.205	< .0001	0.3485	0.5226	TRUE	0.1749
sex	1	0.7283	0.1639	0.0383	19.7471	< .0001	0.4071	1.0495	TRUE	0.04
education	1	0.363	0.0832	0.044	19.0459	< .0001	0.1999	0.526	TRUE	0.0712
marriage	1	1.1135	0.1892	0.0539	34.6496	< .0001	0.7427	1.4843	TRUE	0.0342
age	1	0.3112	0.1125	0.0244	7.6513	0.0057	0.0907	0.5318	TRUE	0.0601
pay_0	1	0.7628	0.0205	0.3614	1383.5275	< .0001	0.7226	0.803	TRUE	0.4201
pay_2	1	-0.0063	0.0338	-0.0023	0.0353	0.851	-0.0725	0.0598	FALSE	0.3386
pay_3	1	0.1847	0.0408	0.0594	20.533	< .0001	0.1048	0.2646	TRUE	0.2949
pay_4	1	0.0943	0.0454	0.0282	4.3193	0.0377	0.0054	0.1832	TRUE	0.278
pay_5	1	0.1718	0.0464	0.049	13.73	0.0002	0.0809	0.2627	TRUE	0.268
pay_6	1	0.2149	0.0426	0.0574	25.4295	< .0001	0.1314	0.2984	TRUE	0.2498
bill_amt1	1	-0.8325	0.2268	-0.0501	13.4743	0.0002	-1.277	-0.388	FALSE	0.0456
bill_amt2	1	-0.0782	0.2643	-0.0045	0.0876	0.7673	-0.5961	0.4397	FALSE	0.0442
bill_amt3	1	-0.8733	0.2682	-0.0489	10.603	0.0011	-1.399	-0.3476	FALSE	0.0423

PAI command

```
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
```

Algorithm parameters

Parameter	Description	Option	Default value
inputTableName	(Required) Name of the input feature data table	table name	NA
inputTablePartitions	(Optional) Partitions selected in the input feature data table	partition name	All partitions in the table by default
inputBinTableName	(Optional) Input binning result table. If the table is specified, the original features are automatically discretized based on the binning rules of the table before training.	table name	NA
featureColNames	(Optional) Feature columns selected in the input table	column name	All columns but the label column are selected by default.
labelColName	(Required) Target	column name	NA

	column		
outputTableName	(Required) Name of the output model table	table name	NA
inputConstraintTableName	(Optional) Input constraint in JSON format, stored in a cell of the table	table name	NA
optimization	(Optional) optimization type	Value options: lbfgs, newton, barrier_method, sqp, and auto. Currently, only sqp and barrier_method support constraints. auto is to automatically select an appropriate optimization algorithm based on user data and related parameters. We recommend that you select auto if you are unfamiliar with the preceding optimization algorithms.	auto
loss	(Optional) Loss type	logistic_regression and least_square	logistic_regression
iterations	(Optional) Maximum number of optimization iterations	positive integer	Default value: 100
l1Weight	(Optional) L1 regular parameter weight. Currently, only lbfgs supports l1weight.	NA	0
l2Weight	(Optional) L2 regular parameter weight	NA	0
m	(Optional) Historical length during lbfgs optimization, valid only for lbfgs	NA	10
scale	(Optional) Weight scale information on the scorecard	NA	Null
selected	(Optional) Scorecard feature selection	NA	Null
convergenceToleran	(Optional)	NA	1e-6

ce	Convergence conditions		
positiveLabel	(Optional) Class of the positive sample	NA	1
lifecycle	(Optional) Life cycle of the output table	positive integer	Unspecified by default
coreNum	(Optional) Number of cores	positive integer	Automatically calculated by default
memSizePerCore	(Optional) Size of memory	positive integer	Automatically calculated by default

Scorecard prediction

- The scorecard prediction component predicts and scores raw data based on the output model produced by the training component.
- The supported training components include scorecard training, binary logistic regression (Financials), and linear regression (Financials).

Input parameters

The prediction component supports the following parameters:

Feature column: Select the original feature columns used for prediction. By default, all columns are selected.

Columns reserved in result table: Select the columns attached directly to the prediction result table, such as the common ID column and target column.

Output variable score: Choose whether to output the score of each feature variable. The final total prediction score is the intercept score plus all the variable scores.

Output score table

The following is an example of an output score table:

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}

The first column, churn, is added to the result table by the user as is and is irrelevant to the prediction result. The other three columns are prediction result columns, described as follows:

Column Name	Column Type	Column Description
prediction_score	double	Prediction score column, which lists the sum of feature values multiplied by model weights in the linear model. If score conversion is conducted in the scorecard model, converted scores are output.
prediction_prob	double	Positive probability value predicted in binary classification, obtained by sigmoid transformation of the raw score (without fraction conversion)
prediction_detail	string	Probability value of each class described in JSON format, where 0 represents a negative class and 1 represents a positive class. For example, { “0” :0.1813110520,” 1” :0. 8186889480}.

PAI command

```
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
```

Algorithm parameters

Parameter	Description	Option	Default value

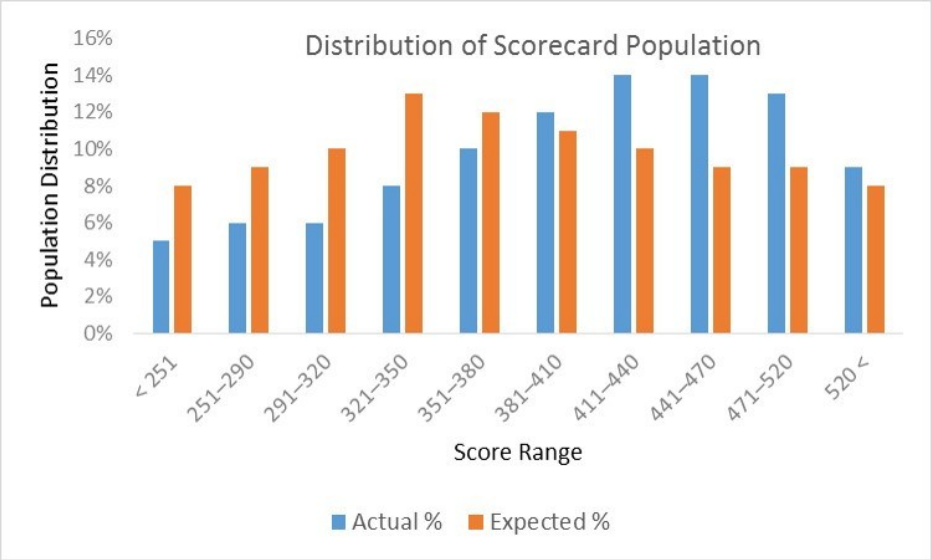
inputFeatureTableName	(Required) Name of the input feature data table	table name	NA
inputFeatureTablePartitions	(Optional) Partitions selected in the input feature table	partition name	Entire table selected by default
inputModelTableName	(Required) Name of the input model table	table name	NA
featureColNames	(Optional) Feature columns selected in the input table	column name	Entire table selected by default
metaColNames	(Optional) Data columns not converted. The selected column is output as is.	column name	No meta column by default, and you can specify label, sample_id, and other columns.
outputFeatureScore	(Optional) Whether to include variable scores in the prediction result	true/false	false
outputTableName	(Required) Name of the output prediction result table	table name	NA
lifecycle	(Optional) Life cycle of the output table	positive integer	Unspecified by default
coreNum	(Optional) Number of cores	positive integer	Automatically calculated by default
memSizePerCore	(Optional) Size of memory	positive integer	Automatically calculated by default

PSI

The population stability index (PSI) is an important indicator that measures the offset caused by sample changes. It is used to measure the sample stability, for example, whether a sample changes stably between two months.

Typically, if the PSI of a variable is less than 0.1, the variable does not have obvious changes. If the PSI is between 0.1 and 0.25, the variable changes greatly. If the PSI is greater than 0.25, the variable changes dramatically and requires special attention.

Drawings can be used to check the sample stability. Discretize the variables to be compared into N bins, calculate the number and proportion of samples in each bin, and draw a histogram based on the data, as shown in the following figure.

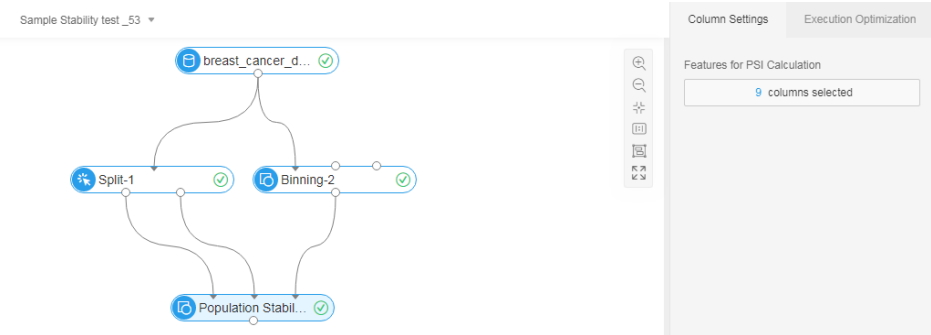


This method helps you intuitively check whether a certain variable has drastic changes in the two samples. However, this method cannot quantify the changes or automatically monitor the sample stability. Therefore, the PSI is particularly important. The feature data must be divided into bins before the PSI can be used. Therefore, a binning component is required. The PSI calculation formula is as follows:

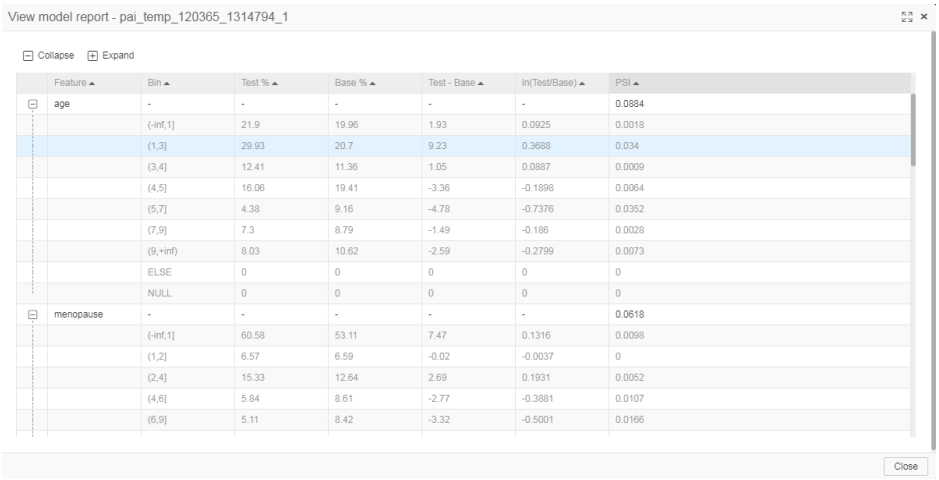
$$PSI = \sum((Actual\ \% - Expected\ \%) \times (ln(\frac{Actual\ \%}{Expected\ \%}))$$

Example

Connect the two sample data sets and binning component with PSI component. And set the parameter of the PSI component, choose **Feature for PSI Calculation**, as shown in the following figure.



Example of results



Feature	Bin	Test %	Base %	Test - Base	ln(Test/Base)	PSI
age	-	-	-	-	-	0.0884
	(-inf,1]	21.9	19.96	1.93	0.0925	0.0018
	(1,3]	29.93	20.7	9.23	0.3688	0.034
	(3,4]	12.41	11.36	1.05	0.0887	0.0009
	(4,5]	16.06	19.41	-3.36	-0.1898	0.0064
	(5,7]	4.38	9.16	-4.78	-0.7376	0.0352
	(7,9]	7.3	8.79	-1.49	-0.186	0.0028
	(9,+inf)	8.03	10.62	-2.59	-0.2799	0.0073
	ELSE	0	0	0	0	0
	NULL	0	0	0	0	0
menopause	-	-	-	-	-	0.0618
	(-inf,1]	60.58	53.11	7.47	0.1316	0.0098
	(1,2]	6.57	6.59	-0.02	-0.0037	0
	(2,4]	15.33	12.64	2.69	0.1931	0.0052
	(4,6]	5.84	8.61	-2.77	-0.3881	0.0107
	(6,9]	5.11	8.42	-3.32	-0.5001	0.0166

PAI command

```
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
```

Algorithm parameters

Parameter	Description	Option	Default value
inputBaseTableName	(Required) Name of the input base table name. Calculate the offset of the test table relative to the base table.	table name	NA
inputBaseTablePartitions	(Optional) Input base table partitions	partition name	Entire table selected by default
inputTestTableName	(Required) Input test table name. Calculate the offset of the test table relative to the base table.	table name	NA
inputTestTablePartitions	(Optional) Input test table partitions	partition name	Entire table selected by default
inputBinTableName	(Required) Name of the input binning result table	table name	NA
featureColNames	(Optional) Features of which the PSIs	column name	All features selected by default

	need to be calculated		
outputTableName	(Required) Output indicator table	table name	NA
lifecycle	(Optional) Life cycle of the output table	positive integer	Unspecified by default
coreNum	(Optional) Number of cores	positive integer	Automatically calculated by default
memSizePerCore	(Optional) Size of memory	positive integer	Automatically calculated by default

Automatic parameter tuning with Auto ML

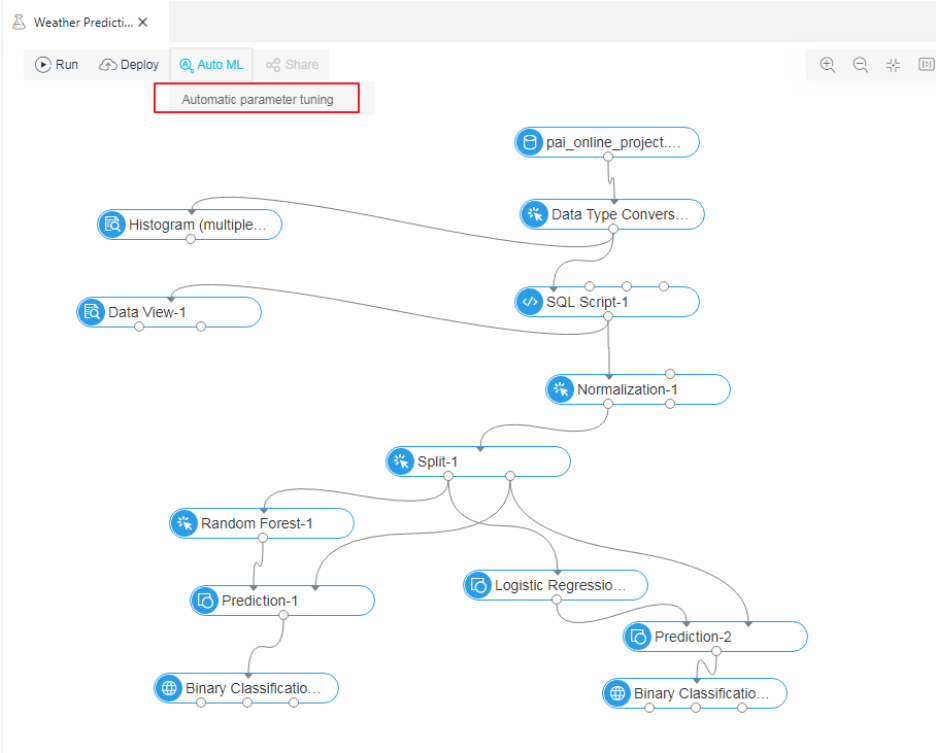
Configure parameters

Log on to the Machine Learning Platform for AI console.

In the left-side navigation pane, click **Experiments**.

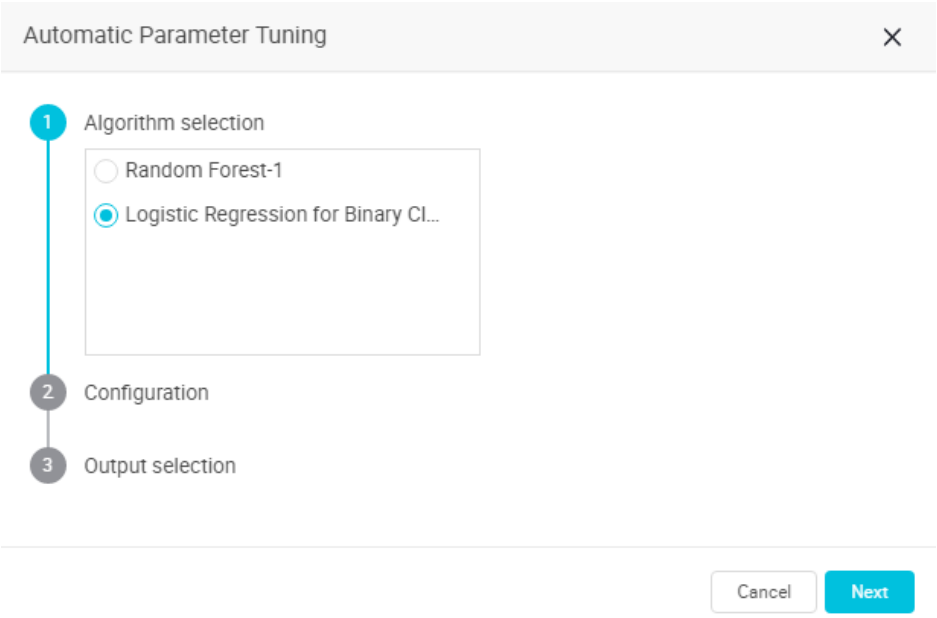
Click your target experiment to enter the canvas of the experiment. This topic takes the Weather Prediction experiment as an example.

On the upper-left corner of the canvas, select **Auto ML** > **Automatic parameter tuning**.

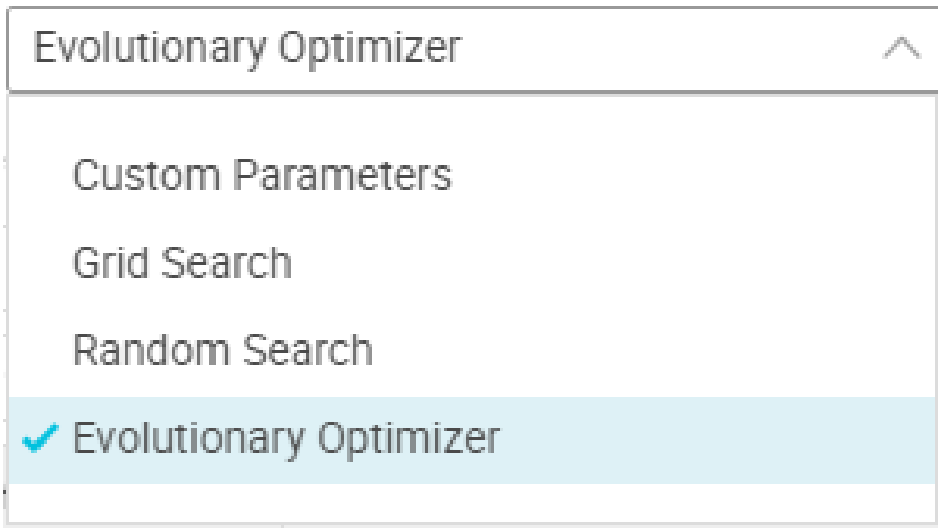


Select one algorithm for parameter tuning, and then click **Next**.

Note: You can only select one algorithm to tune at a time.



In the **Configuration** module, select a parameter tuning method, and then click **Next**.



Alibaba Cloud Machine Learning Platform for AI provides the following parameter tuning methods:

Evolutionary Optimizer

Concept:

Randomly selects A parameter candidate sets (where A indicates the **Number of exploration samples**).

Takes the N parameter candidate sets with higher evaluation indicators as the parameter candidate sets of the next iteration.

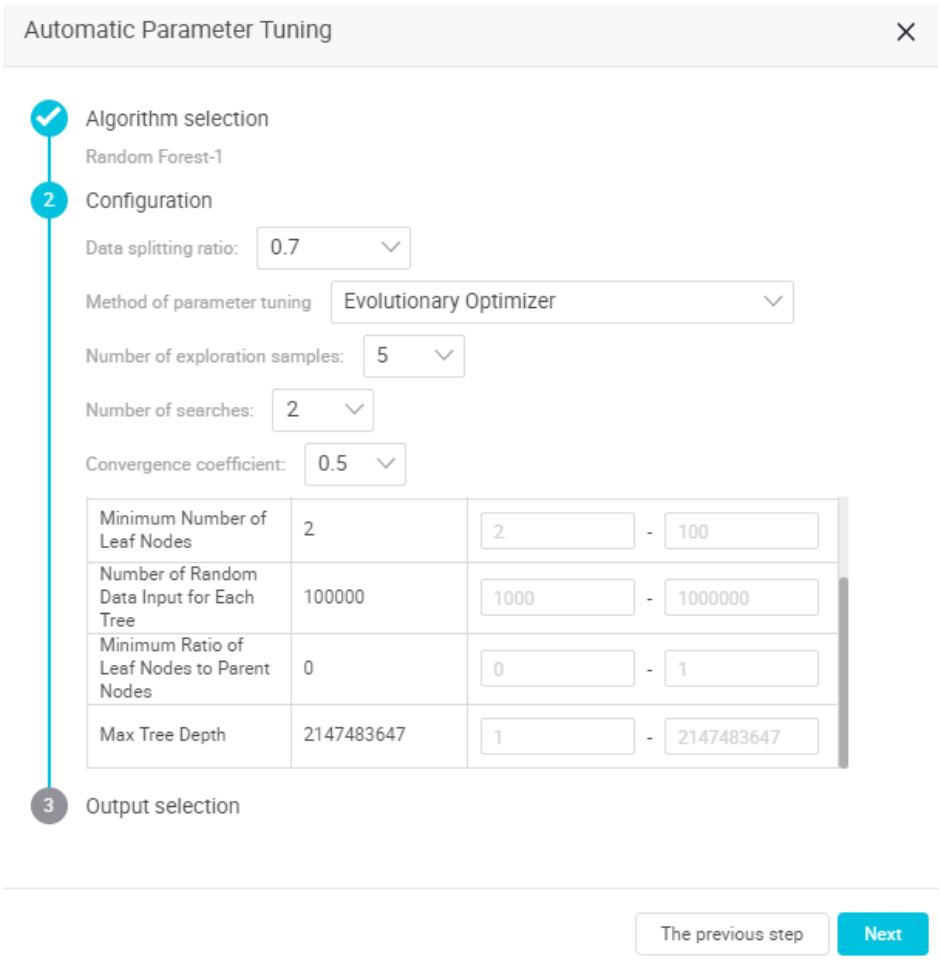
Continues the exploration within R times (where R indicates **Convergence coefficient**) as the standard deviation range around these parameters to explore new parameter sets. The new parameter sets replace the last A-N parameter sets by the evaluation indicator in the previous round.

Iterates the exploration for M rounds (where M indicates **Number of searches**) until the optimal parameter set is found, according to the preceding logic.

According to the preceding principle, the final number of models is A+(A-N)*M.

Note: Continues the exploration within R times (where R indicates **Convergence coefficient**) as the standard deviation range around these parameters to explore new parameter sets to replace the last A-N parameter sets by the evaluation indicator in the previous round. The first value of N is

A/2-1, which defaults to N/2-1 during the iteration (decimal values are rounded up to an integer).



Data splitting ratio: Splits input data sources into training and evaluation sets. 0.7 indicates that 70% of the data is for training a model, and the remaining 30% of data is for evaluation.

Number of exploration samples: The number of parameter sets of each iteration. The higher the number, the greater the accuracy, the larger the calculation. The value range is 5-30.

Number of searches: The number of iterations. The higher the number of iterations, the greater the search accuracy, the larger the calculation. The value range is 1-10.

Convergence coefficient: Tunes the exploration ranges (R times the standard deviation range search). The smaller the range, the faster the convergence (however, optimal parameters may be missed). The value

range is 0.1-1 (one floating point after the decimal point).

- You must enter the tuning ranges for each parameter. **If the current parameter range is not configured, the parameter range is set by default**.

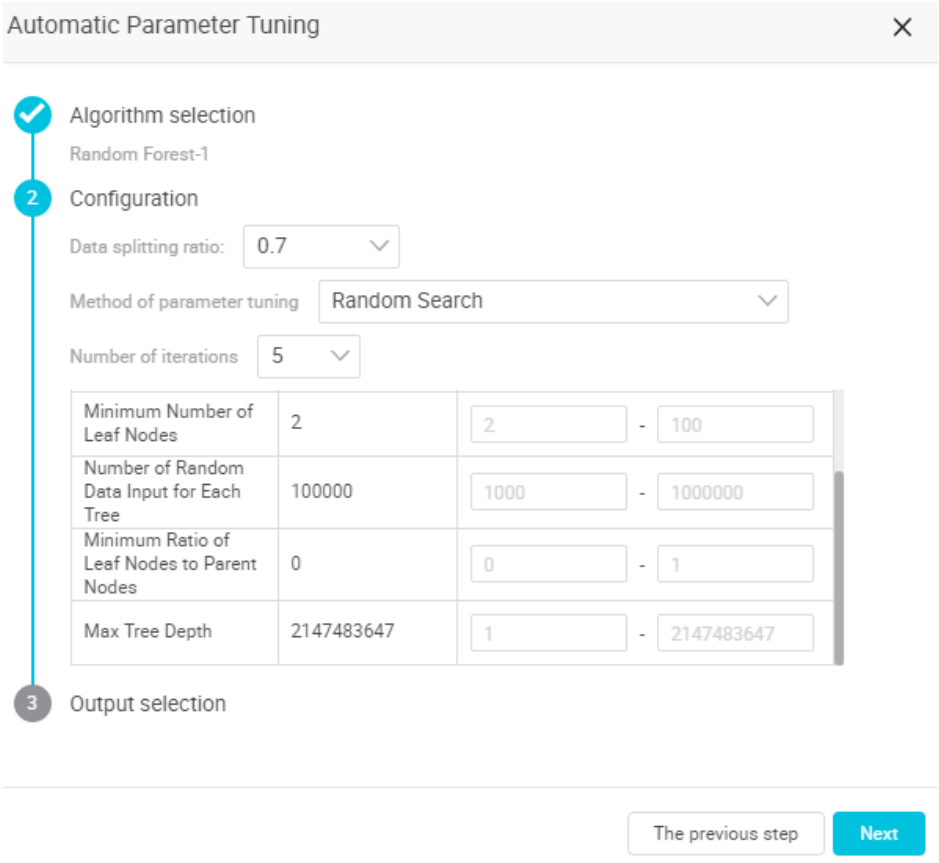
Random Search

Concept:

Each parameter randomly selects a value within its range.

Enters random values into a set of parameters for model training.

Performs M rounds (where M indicates the number of iterations) and then orders the output models.



Number of iterations: Indicates the number of searches in the configured interval. The value range is 2 to 50.

Data splitting ratio: Splits input data sources into training and evaluation sets. 0.7 indicates that 70% of the data is for training a model, and the

remaining 30% of data is for evaluation.

You must enter the tuning ranges for each parameter. **If the current parameter range is not configured, the parameter range is set by default**.

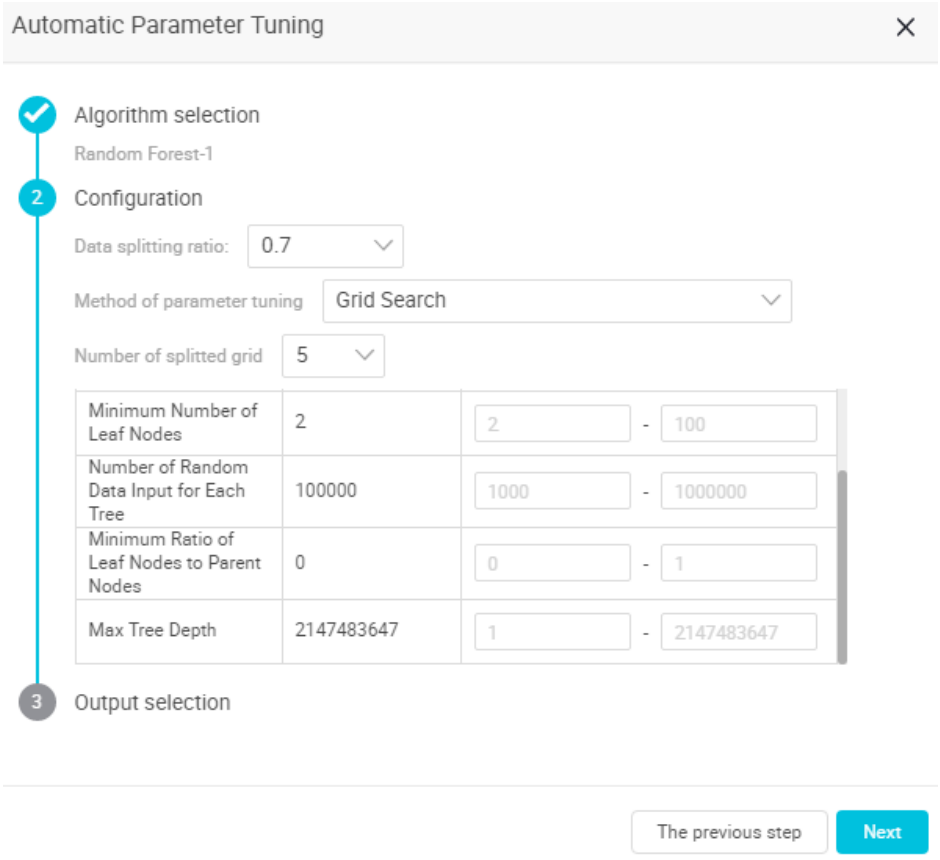
Grid Search

Concept:

Splits the value range of each parameter into N segments (grid split score).

Randomly takes a value from the N segments. Assuming that there are M parameters, N^M parameter groups can be combined.

According to the N^M parameter groups, N^M models are generated by training. The models are then ordered.

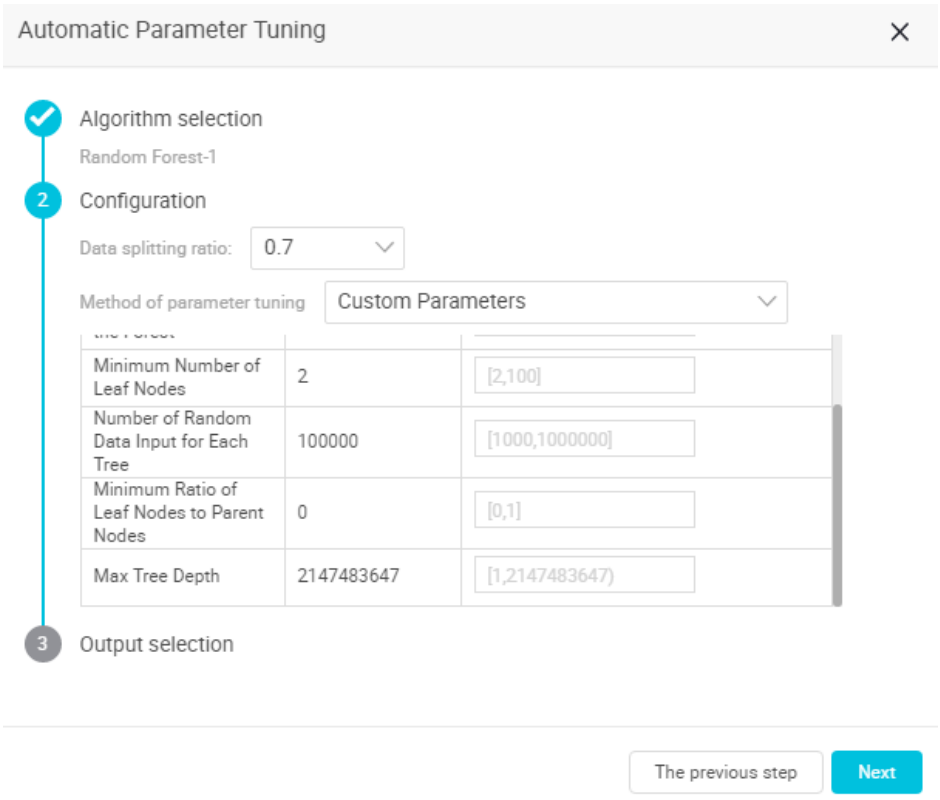


Number of splitted grid: Indicates the number of split grids. The value range is 2-10.

Data splitting ratio: Splits input data sources into training and evaluation sets. 0.7 indicates that 70% of the data is for training a model, and the remaining 30% of data is for evaluation.

You must enter the tuning ranges for each parameter. **If the current parameter range is not configured, the parameter range is set by default**.

Custom Parameters



You can enumerate parameter candidate sets. The system then helps you to score all the combinations of the candidate sets.

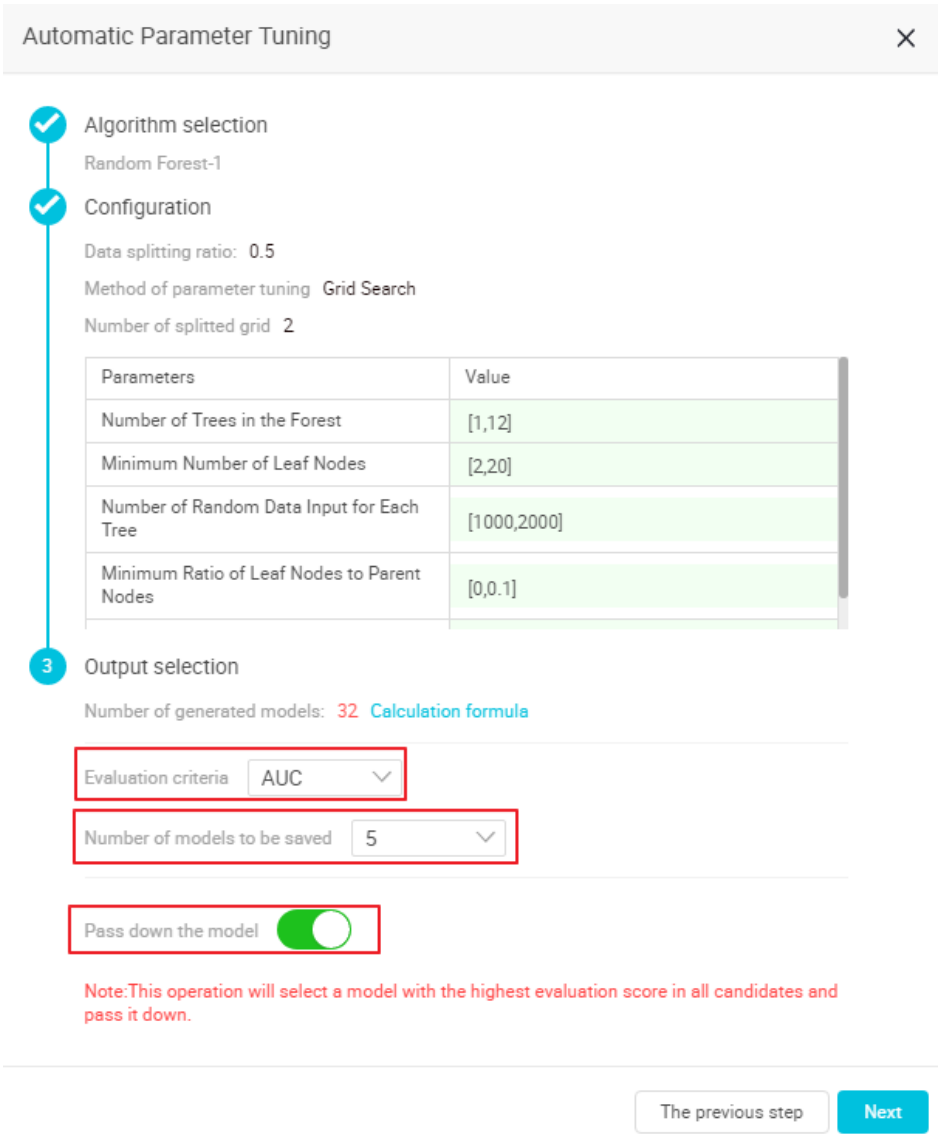
You can define enumeration ranges. Parameters are separated by commas. If the ranges are not configured, the parameters are tuned by default.

In the **Output selection** module, configure model Output parameters, and then click **Next**.

Evaluation criteria: Select one evaluation standard from the following four dimensions: **AUC**, **F1 Score**, **Precision**, and **RECALL**.

Number of models to be saved: You can save up to five models. The system ranks models and save the top ranked models according to the number entered in the **Number of models to be saved** field.

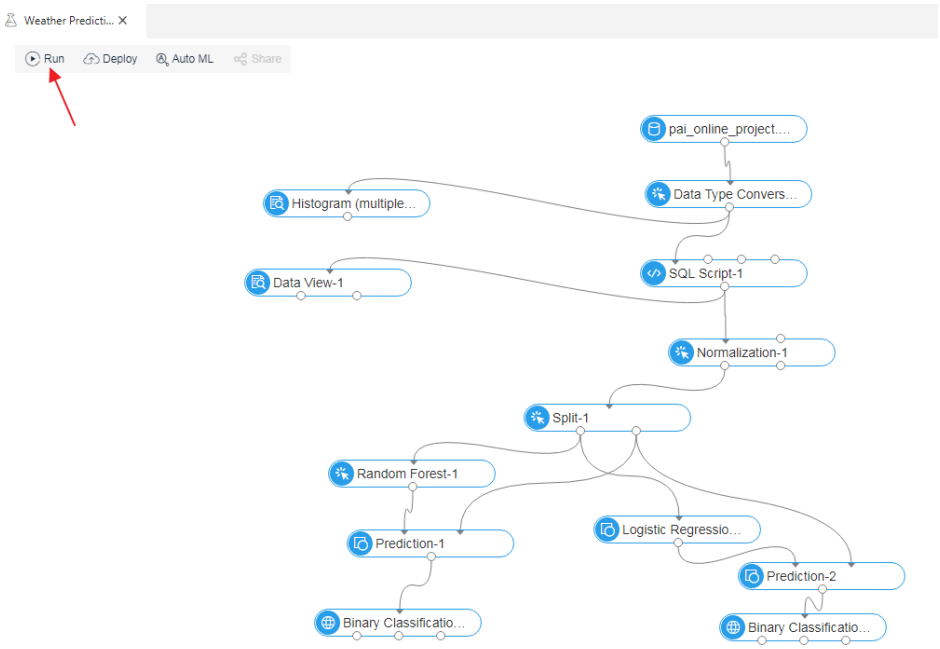
Pass down the model: The switch is turned ON by default. If the switch is OFF, the model generated by the default parameters of the current component are passed down to the node of the subsequent component. If the switch is ON, the optimal model generated by automatic parameter tuning are passed down to the node of the subsequent component.



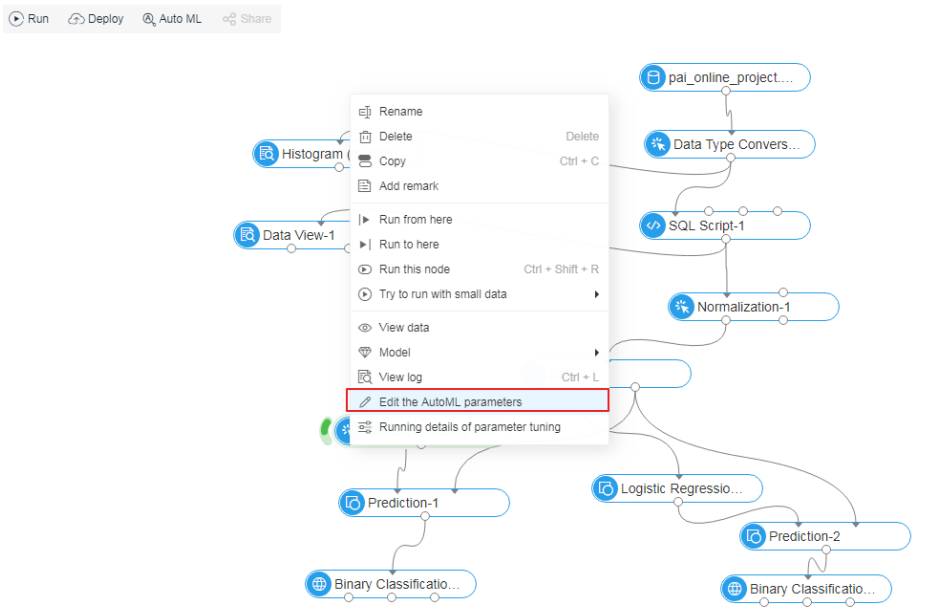
In the upper-left corner of the canvas, click **Run** to run the automatic parameter tuning algorithm.

Note: After the preceding configuration is run, the **Auto ML** switch of the related

algorithm is turned ON. You can turn the switch ON or OFF as required.

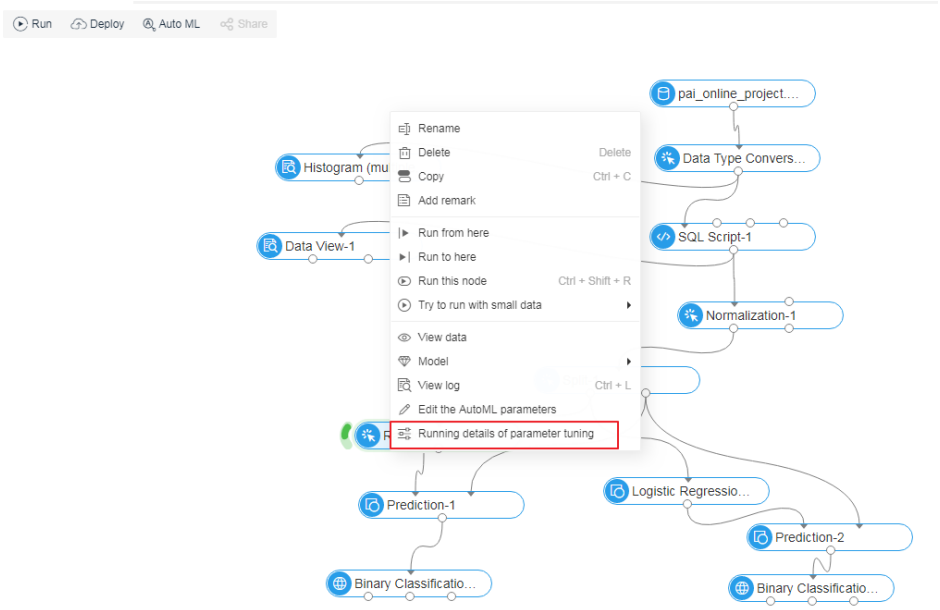


(Optional) Right-click a model component, and then select **Edit the AutoML parameters** to modify its Auto ML configuration parameters.

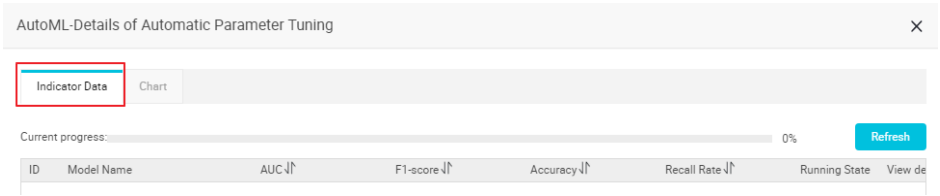


Output model display

During parameter tuning, right click the target model component and then select **Running details of parameter tuning**.

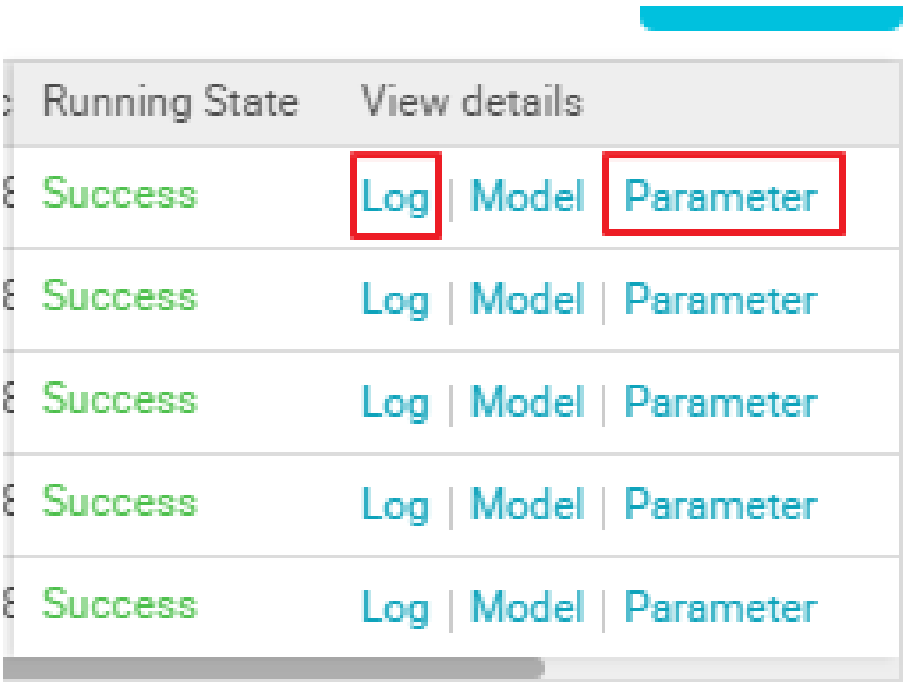


In the **AutoML-Details of Automatic Parameter Tuning** pane, click **Indicator Data** to view the current tuning progress and the running status of each model.



You can order candidate models according to indicators (**AUC**, **F1-score**, **Accuracy**, and **Recall Rate**).

In the **View details** column, you can click **Log** or **Parameter** to view the logs and parameters of each candidate model.



Parameter tuning effect display

Click **Chart** on the **Indicator Data** page to view the **Model evaluation & comparison** and **Effect Comparison of Hyper-parameters Iteration** charts.

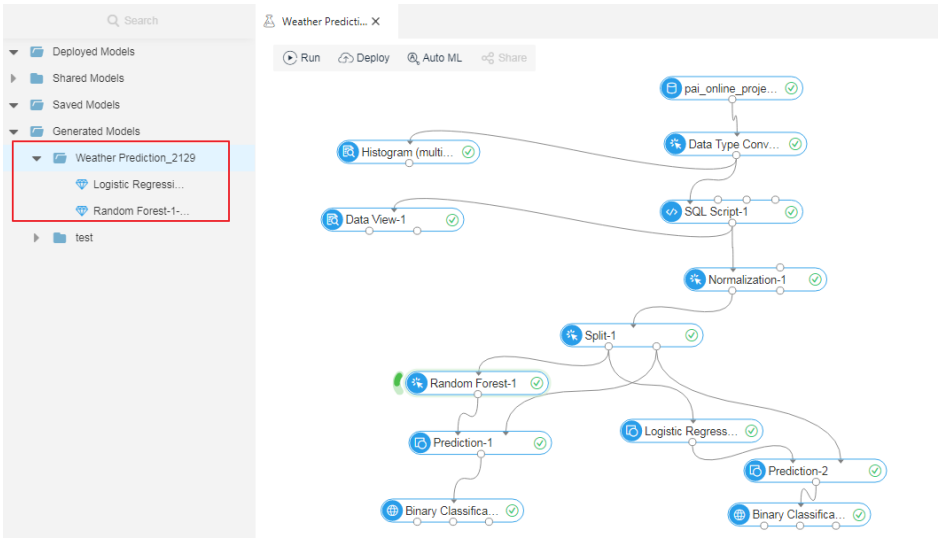
For example, you can view the growth trend of the evaluation indicators of updated parameters in **Effect Comparison of Hyper-parameters Iteration** as shown in the following figure:

Model storage

In the left-side navigation pane, click **Models**.

Click **My Experiments**.

Click the corresponding experiment folder to view the model saved with Auto ML.



(Optional) You can apply a model to other experiments by dragging the model to the canvas of the target experiment.