151
DBMS_PREDICTIVE_ANALYTICS

Machine learning can discover useful information buried in vast amounts of data. However, both the programming interfaces and the machine learning expertise required to obtain these results are too complex for use by the wide audiences that can obtain benefits from using Oracle Machine Learning for SQL.

The `DBMS_PREDICTIVE_ANALYTICS` package addresses both of these complexities by automating the entire machine learning process from data preprocessing through model building to scoring new data. This package provides an important tool that makes machine learning possible for a broad audience of users, in particular, business analysts.

This chapter contains the following topics:

DBMS_PREDICTIVE_ANALYTICS Overview

DBMS_PREDICTIVE_ANALYTICS automates parts of the machine learning process.

Machine learning, according to a commonly used process model, requires the following steps:

1. Understand the business problem.
2. Understand the data.
3. Prepare the data for mining.
4. Create models using the prepared data.
5. Evaluate the models.
6. Deploy and use the model to score new data.

`DBMS_PREDICTIVE_ANALYTICS` automates parts of step 3 — 5 of this process.

Predictive analytics procedures analyze and prepare the input data, create and test machine learning models using the input data, and then use the input data for scoring. The results of scoring are returned to the user. The models and supporting objects are not preserved after the operation completes.

DBMS_PREDICTIVE_ANALYTICS Security Model

The `DBMS_PREDICTIVE_ANALYTICS` package is owned by user `SYS` and is installed as part of database installation. Execution privilege on the package is granted to public. The routines in the package are run with invokers' rights (run with the privileges of the current user).

The `DBMS_PREDICTIVE_ANALYTICS` package exposes APIs which are leveraged by the Oracle Machine Learning for SQL option. Users who wish to invoke procedures in this package require the `CREATE MINING MODEL` system privilege (as well as the `CREATE TABLE` and `CREATE VIEW` system privilege).

Summary of DBMS_PREDICTIVE_ANALYTICS Subprograms

This table lists and briefly describes the `DBMS_PREDICTIVE_ANALYTICS` package subprograms.

Table 151-1 DBMS_PREDICTIVE_ANALYTICS Package Subprograms

Subprogram	Purpose
EXPLAIN Procedure	Ranks attributes in order of influence in explaining a target column.
PREDICT Procedure	Predicts the value of a target column based on values in the input data.
PROFILE Procedure	Generates rules that identify the records that have the same target value.

EXPLAIN Procedure

The `EXPLAIN` procedure identifies the attributes that are important in explaining the variation in values of a target column.

The input data must contain some records where the target value is known (not `NULL`). These records are used by the procedure to train a model that calculates the attribute importance.

> **Note:**
>
> `EXPLAIN` supports `DATE` and `TIMESTAMP` datatypes in addition to the numeric, character, and nested datatypes supported by Oracle Machine Learning for SQL models.
>
> Data requirements for Oracle Machine Learning for SQL are described in *Oracle Machine Learning for SQL User's Guide*

The `EXPLAIN` procedure creates a result table that lists the attributes in order of their explanatory power. The result table is described in the Usage Notes.

Syntax

```
DBMS_PREDICTIVE_ANALYTICS.EXPLAIN (
     data_table_name     IN VARCHAR2,
     explain_column_name IN VARCHAR2,
     result_table_name   IN VARCHAR2,
     data_schema_name    IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 151-2 EXPLAIN Procedure Parameters

Parameter	Description
`data_table_name`	Name of input table or view
`explain_column_name`	Name of the column to be explained
`result_table_name`	Name of the table where results are saved

Table 151-2 (Cont.) EXPLAIN Procedure Parameters

Parameter	Description
data_schema_name	Name of the schema where the input table or view resides and where the result table is created. Default: the current schema.

Usage Notes

The EXPLAIN procedure creates a result table with the columns described in Table 151-3.

Table 151-3 EXPLAIN Procedure Result Table

Column Name	Datatype	Description
ATTRIBUTE_NAME	VARCHAR2(30)	Name of a column in the input data; all columns except the explained column are listed in the result table.
EXPLANATORY_VALUE	NUMBER	Value indicating how useful the column is for determining the value of the explained column. Higher values indicate greater explanatory power. Value can range from 0 to 1. An individual column's explanatory value is independent of other columns in the input table. The values are based on how strong each individual column correlates with the explained column. The value is affected by the number of records in the input table, and the relations of the values of the column to the values of the explain column. An explanatory power value of 0 implies there is no useful correlation between the column's values and the explain column's values. An explanatory power of 1 implies perfect correlation; such columns should be eliminated from consideration for PREDICT. In practice, an explanatory power equal to 1 is rarely returned.
RANK	NUMBER	Ranking of explanatory power. Rows with equal values for explanatory_value have the same rank. Rank values are not skipped in the event of ties.

Example

The following example performs an EXPLAIN operation on the SUPPLEMENTARY_DEMOGRAPHICS table of Sales History.

```
--Perform EXPLAIN operation
BEGIN
    DBMS_PREDICTIVE_ANALYTICS.EXPLAIN(
        data_table_name      => 'supplementary_demographics',
        explain_column_name  => 'home_theater_package',
        result_table_name    => 'demographics_explain_result');
END;
/
--Display results
SELECT * FROM demographics_explain_result;

ATTRIBUTE_NAME                         EXPLANATORY_VALUE       RANK
-------------------------------------- ----------------- ----------
Y_BOX_GAMES                                .524311073          1
YRS_RESIDENCE                              .495987246          2
HOUSEHOLD_SIZE                             .146208506          3
AFFINITY_CARD                                .0598227          4
EDUCATION                                  .018462703          5
OCCUPATION                                 .009721543          6
```

```
FLAT_PANEL_MONITOR                        .00013733          7
PRINTER_SUPPLIES                                  0          8
OS_DOC_SET_KANJI                                  0          8
BULK_PACK_DISKETTES                               0          8
BOOKKEEPING_APPLICATION                           0          8
COMMENTS                                          0          8
CUST_ID                                           0          8
```

The results show that `Y_BOX_GAMES`, `YRS_RESIDENCE`, and `HOUSEHOLD_SIZE` are the best predictors of `HOME_THEATER_PACKAGE`.

PREDICT Procedure

The `PREDICT` procedure predicts the values of a target column.

The input data must contain some records where the target value is known (not `NULL`). These records are used by the procedure to train and test a model that makes the predictions.

> **Note:**
>
> `PREDICT` supports `DATE` and `TIMESTAMP` datatypes in addition to the numeric, character, and nested datatypes supported by Oracle Machine Learning for SQL models.
>
> Data requirements for Oracle Machine Learning for SQL are described in *Oracle Machine Learning for SQL User's Guide*

The `PREDICT` procedure creates a result table that contains a predicted target value for every record. The result table is described in the Usage Notes.

Syntax

```
DBMS_PREDICTIVE_ANALYTICS.PREDICT (
    accuracy                OUT NUMBER,
    data_table_name         IN VARCHAR2,
    case_id_column_name     IN VARCHAR2,
    target_column_name      IN VARCHAR2,
    result_table_name       IN VARCHAR2,
    data_schema_name        IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 151-4 PREDICT Procedure Parameters

Parameter	Description
`accuracy`	Output parameter that returns the predictive confidence, a measure of the accuracy of the predicted values. The predictive confidence for a categorical target is the most common target value; the predictive confidence for a numerical target is the mean.
`data_table_name`	Name of the input table or view.
`case_id_column_name`	Name of the column that uniquely identifies each case (record) in the input data.
`target_column_name`	Name of the column to predict.

Table 151-4 (Cont.) PREDICT Procedure Parameters

Parameter	Description
result_table_name	Name of the table where results will be saved.
data_schema_name	Name of the schema where the input table or view resides and where the result table is created. Default: the current schema.

Usage Notes

The PREDICT procedure creates a result table with the columns described in Table 151-5.

Table 151-5 PREDICT Procedure Result Table

Column Name	Datatype	Description
Case ID column name	VARCHAR2 or NUMBER	The name of the case ID column in the input data.
PREDICTION	VARCHAR2 or NUMBER	The predicted value of the target column for the given case.
PROBABILITY	NUMBER	For classification (categorical target), the probability of the prediction. For regression problems (numerical target), this column contains NULL.

> **Note:**
>
> Make sure that the name of the case ID column is not 'PREDICTION' or 'PROBABILITY'.

Predictions are returned for all cases whether or not they contained target values in the input.

Predicted values for known cases may be interesting in some situations. For example, you could perform deviation analysis to compare predicted values and actual values.

Example

The following example performs a PREDICT operation and displays the first 10 predictions. The results show an accuracy of 79% in predicting whether each customer has an affinity card.

```
--Perform PREDICT operation
DECLARE
    v_accuracy NUMBER(10,9);
BEGIN
    DBMS_PREDICTIVE_ANALYTICS.PREDICT(
        accuracy            => v_accuracy,
        data_table_name     => 'supplementary_demographics',
        case_id_column_name => 'cust_id',
        target_column_name  => 'affinity_card',
        result_table_name   => 'pa_demographics_predict_result');
    DBMS_OUTPUT.PUT_LINE('Accuracy = ' || v_accuracy);
END;
/

Accuracy = .788696903

--Display results
```

```
SELECT * FROM pa_demographics_predict_result WHERE rownum < 10;

   CUST_ID PREDICTION PROBABILITY
---------- ---------- -----------
    101501          1  .834069848
    101502          0  .991269965
    101503          0   .99978311
    101504          1  .971643388
    101505          1  .541754127
    101506          0  .803719133
    101507          0  .999999303
    101508          0  .999999987
    101509          0  .999953074
```

PROFILE Procedure

The `PROFILE` procedure generates rules that describe the cases (records) from the input data.

For example, if a target column `CHURN` has values 'Yes' and 'No', `PROFILE` generates a set of rules describing the expected outcomes. Each profile includes a rule, record count, and a score distribution.

The input data must contain some cases where the target value is known (not `NULL`). These cases are used by the procedure to build a model that calculates the rules.

> **Note:**
>
> `PROFILE` does not support nested types or dates.
>
> Data requirements for Oracle Machine Learning for SQL are described in *Oracle Machine Learning for SQL User's Guide*

The `PROFILE` procedure creates a result table that specifies rules (profiles) and their corresponding target values. The result table is described in the Usage Notes.

Syntax

```
DBMS_PREDICTIVE_ANALYTICS.PROFILE (
     data_table_name           IN VARCHAR2,
     target_column_name        IN VARCHAR2,
     result_table_name         IN VARCHAR2,
     data_schema_name          IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 151-6 PROFILE Procedure Parameters

Parameter	Description
`data_table_name`	Name of the table containing the data to be analyzed.
`target_column_name`	Name of the target column.
`result_table_name`	Name of the table where the results will be saved.
`data_schema_name`	Name of the schema where the input table or view resides and where the result table is created. Default: the current schema.

Usage Notes

The PROFILE procedure creates a result table with the columns described in Table 151-7.

Table 151-7 PROFILE Procedure Result Table

Column Name	Datatype	Description
PROFILE_ID	NUMBER	A unique identifier for this profile (rule).
RECORD_COUNT	NUMBER	The number of records described by the profile.
DESCRIPTION	SYS.XMLTYPE	The profile rule. See "XML Schema for Profile Rules".

XML Schema for Profile Rules

The DESCRIPTION column of the result table contains XML that conforms to the following XSD:

```
<xs:element name="SimpleRule">
  <xs:complexType>
    <xs:sequence>
      <xs:group ref="PREDICATE"/>
      <xs:element ref="ScoreDistribution" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="optional"/>
    <xs:attribute name="score" type="xs:string" use="required"/>
    <xs:attribute name="recordCount" type="NUMBER" use="optional"/>
  </xs:complexType>
</xs:element>
```

Example

This example generates a rule describing customers who are likely to use an affinity card (target value is 1) and a set of rules describing customers who are not likely to use an affinity card (target value is 0). The rules are based on only two predictors: education and occupation.

```
SET serveroutput ON
SET trimspool ON
SET pages 10000
SET long 10000
SET pagesize 10000
SET linesize 150
CREATE VIEW cust_edu_occ_view AS
             SELECT cust_id, education, occupation, affinity_card
             FROM sh.supplementary_demographics;
BEGIN
    DBMS_PREDICTIVE_ANALYTICS.PROFILE(
         DATA_TABLE_NAME    => 'cust_edu_occ_view',
         TARGET_COLUMN_NAME => 'affinity_card',
         RESULT_TABLE_NAME  => 'profile_result');
END;
/
```

This example generates eight rules in the result table profile_result. Seven of the rules suggest a target value of 0; one rule suggests a target value of 1. The score attribute on a rule identifies the target value.

This SELECT statement returns all the rules in the result table.

```
SELECT a.profile_id, a.record_count, a.description.getstringval()
 FROM profile_result a;
```

This `SELECT` statement returns the rules for a target value of 0.

```
SELECT *
  FROM profile_result t
  WHERE extractvalue(t.description, '/SimpleRule/@score') = 0;
```

The eight rules generated by this example are displayed as follows.

```
<SimpleRule id="1" score="0" recordCount="443">
  <CompoundPredicate booleanOperator="and">
    <SimpleSetPredicate field="OCCUPATION" booleanOperator="isIn">
      <Array type="string">"Armed-F" "Exec." "Prof." "Protec."
      </Array>
    </SimpleSetPredicate>
    <SimpleSetPredicate field="EDUCATION" booleanOperator="isIn">
      <Array type="string">"< Bach." "Assoc-V" "HS-grad"
      </Array>
    </SimpleSetPredicate>
  </CompoundPredicate>
  <ScoreDistribution value="0" recordCount="297" />
  <ScoreDistribution value="1" recordCount="146" />
</SimpleRule>

<SimpleRule id="2" score="0" recordCount="18">
  <CompoundPredicate booleanOperator="and">
    <SimpleSetPredicate field="OCCUPATION" booleanOperator="isIn">
      <Array type="string">"Armed-F" "Exec." "Prof." "Protec."
      </Array>
    </SimpleSetPredicate>
    <SimpleSetPredicate field="EDUCATION" booleanOperator="isIn">
      <Array type="string">"10th" "11th" "12th" "1st-4th" "5th-6th" "7th-8th" "9th" "Presch."
      </Array>
    </SimpleSetPredicate>
  </CompoundPredicate>
  <ScoreDistribution value="0" recordCount="18" />
</SimpleRule>

<SimpleRule id="3" score="0" recordCount="458">
  <CompoundPredicate booleanOperator="and">
    <SimpleSetPredicate field="OCCUPATION" booleanOperator="isIn">
      <Array type="string">"Armed-F" "Exec." "Prof." "Protec."
      </Array>
    </SimpleSetPredicate>
    <SimpleSetPredicate field="EDUCATION" booleanOperator="isIn">
      <Array type="string">"Assoc-A" "Bach."
      </Array>
    </SimpleSetPredicate>
  </CompoundPredicate>
  <ScoreDistribution value="0" recordCount="248" />
  <ScoreDistribution value="1" recordCount="210" />
</SimpleRule>

<SimpleRule id="4" score="1" recordCount="276">
  <CompoundPredicate booleanOperator="and">
    <SimpleSetPredicate field="OCCUPATION" booleanOperator="isIn">
      <Array type="string">"Armed-F" "Exec." "Prof." "Protec."
      </Array>
    </SimpleSetPredicate>
    <SimpleSetPredicate field="EDUCATION" booleanOperator="isIn">
      <Array type="string">"Masters" "PhD" "Profsc"
      </Array>
    </SimpleSetPredicate>
  </CompoundPredicate>
```

```
  <ScoreDistribution value="1" recordCount="183" />
  <ScoreDistribution value="0" recordCount="93" />
</SimpleRule>

<SimpleRule id="5" score="0" recordCount="307">
  <CompoundPredicate booleanOperator="and">
    <SimpleSetPredicate field="EDUCATION" booleanOperator="isIn">
      <Array type="string">"Assoc-A" "Bach." "Masters" "PhD" "Profsc"
      </Array>
    </SimpleSetPredicate>
    <SimpleSetPredicate field="OCCUPATION" booleanOperator="isIn">
      <Array type="string">"Crafts" "Sales" "TechSup" "Transp."
      </Array>
    </SimpleSetPredicate>
  </CompoundPredicate>
  <ScoreDistribution value="0" recordCount="184" />
  <ScoreDistribution value="1" recordCount="123" />
</SimpleRule>

<SimpleRule id="6" score="0" recordCount="243">
  <CompoundPredicate booleanOperator="and">
    <SimpleSetPredicate field="EDUCATION" booleanOperator="isIn">
      <Array type="string">"Assoc-A" "Bach." "Masters" "PhD" "Profsc"
      </Array>
    </SimpleSetPredicate>
    <SimpleSetPredicate field="OCCUPATION" booleanOperator="isIn">
      <Array type="string">"?" "Cleric." "Farming" "Handler" "House-s" "Machine" "Other"
      </Array>
    </SimpleSetPredicate>
  </CompoundPredicate>
  <ScoreDistribution value="0" recordCount="197" />
  <ScoreDistribution value="1" recordCount="46" />
</SimpleRule>

<SimpleRule id="7" score="0" recordCount="2158">
  <CompoundPredicate booleanOperator="and">
    <SimpleSetPredicate field="EDUCATION" booleanOperator="isIn">
      <Array type="string">
        "10th" "11th" "12th" "1st-4th" "5th-6th" "7th-8th" "9th" "< Bach." "Assoc-V" "HS-grad"
        "Presch."
      </Array>
    </SimpleSetPredicate>
    <SimpleSetPredicate field="OCCUPATION" booleanOperator="isIn">
      <Array type="string">"?" "Cleric." "Crafts" "Farming" "Machine" "Sales" "TechSup" " Transp."
      </Array>
    </SimpleSetPredicate>
  </CompoundPredicate>
  <ScoreDistribution value="0" recordCount="1819"/>
  <ScoreDistribution value="1" recordCount="339"/>
</SimpleRule>

<SimpleRule id="8" score="0" recordCount="597">
  <CompoundPredicate booleanOperator="and">
    <SimpleSetPredicate field="EDUCATION" booleanOperator="isIn">
      <Array type="string">
        "10th" "11th" "12th" "1st-4th"  "5th-6th" "7th-8th" "9th" "< Bach." "Assoc-V" "HS-grad"
        "Presch."
      </Array>
    </SimpleSetPredicate>
    <SimpleSetPredicate field="OCCUPATION" booleanOperator="isIn">
      <Array type="string">"Handler" "House-s" "Other"
      </Array>
```

```
    </SimpleSetPredicate>
  </CompoundPredicate>
<ScoreDistribution value="0" recordCount="572"/>
<ScoreDistribution value="1" recordCount="25"/>
</SimpleRule>
```