MACHINE LEARNING USING SQL

In this report, we have used Portuguese bank data set in order to explore in-database machine learning features in Oracle SQL Developer. We have designed various models that can predict the outcome of a variable based on a selected set of predictors and performed a comparative analysis of their accuracy in making predictions to be employed in real world. We have performed four major steps to achieve the same. In the first step, we have partitioned data set into train and test data sets to be used for modelling. In the second step, we have built three different models namely Decision tree, Naïve Bayes classifier and Support Vector Machine using train data set. In the third step, we have verified the performance of each of the models by applying them to test data set. In the fourth step, we have generated the Confusion Matrix for each of the models and computed their accuracy to perform comparative performance analysis.

Step 1: Data Partitioning

Data Preparation: In order to create various models for the bank_data table, a primary key is required in the table. There is no primary key in this table and a combination of various columns did not help in generating the primary key. Hence, we have created a sequence number starting from 1 and added it as a primary key column to the table.

SQL Query:

```
CREATE SEQUENCE bank_id

 START WITH     1

 INCREMENT BY   1

 MAXVALUE 42000;

alter table bank_data add bank_id number;

update bank_data set bank_id=bank_id.nextval;
```

Data Partitioning: The bank_data table has been partitioned into two subsets train and test data namely bank_train and bank_test by partitioning the original table in 70:30 ratio. This ratio has been selected to increase the stability of the model along with maintaining its applicability in the real world. This has been performed using ORA_HASH function which computes hash value for a variable and generates random sample for different buckets of variable data. ORA_HASH function has been applied for variable age and 100 buckets have been generated containing randomly selected data. Out of these, data of 70 buckets has been selected for training set and of 30 buckets has been selected for test set.

SQL Query:

```
create table bank_train

as

(

select * from bank_data
```

```
where y='yes'

and

ORA_HASH(age, 99, 5)<70

);


create table bank_test

as

(

select * from bank_data

where y='yes'

and

ORA_HASH(age, 99, 5)>=70

);


insert into bank_train

(

select * from bank_data

where y='no'

and

ORA_HASH(age, 99, 5)<70

);


insert into bank_test

(

select * from bank_data

where y='no'
```

```
and

ORA_HASH(age, 99, 5)>=70

);
```

Step 2: Design Models on Train Data

Three different models namely Decision Tree, Support Vector Machine and Naïve Bayes have been created using train dataset bank_train. These models have been created to predict whether customers of Portuguese bank will subscribe to term deposit or not.

<u>SQL Query:</u>

<u>Model 1: Decision Tree</u>

```
--create settings table

CREATE TABLE decision_tree_model_settings (

setting_name VARCHAR2(30),

setting_value VARCHAR2(30)

);


BEGIN

  INSERT INTO decision_tree_model_settings (setting_name, setting_value)

  VALUES (dbms_data_mining.algo_name,dbms_data_mining.algo_decision_tree);


  INSERT INTO decision_tree_model_settings (setting_name, setting_value)

  VALUES (dbms_data_mining.prep_auto,dbms_data_mining.prep_auto_on);

  COMMIT;

END;


--Creating a Decision Tree

BEGIN

DBMS_DATA_MINING.CREATE_MODEL(

  model_name => 'Decision_Tree_Model1',
```

```
  mining_function => dbms_data_mining.classification,

  data_table_name => 'bank_train',

  case_id_column_name => 'bank_id',

  target_column_name => 'y',

  settings_table_name => 'decision_tree_model_settings');

END;


-- describe the model settings tables

--describe user_mining_model_settings

-- List all the ODM models created in your Oracle schema => what machine learning models you have created

SELECT model_name,

  mining_function,

  algorithm,

  build_duration,

  model_size

FROM user_MINING_MODELS;


-- List the algorithm settings used for your machine learning model

SELECT setting_name,

  setting_value,

  setting_type

FROM user_mining_model_settings

WHERE model_name in 'DECISION_TREE_MODEL1';


-- List the attribute the machine learning model uses. It may use a subset of the attributes.

-- This allows you to see what attributes were selected
```

```
SELECT attribute_name,
   attribute_type,
   usage_type,
   target
from all_mining_model_attributes
where model_name = 'DECISION_TREE_MODEL1';
```

Model 2: Support Vector Machine

```
--create settings table
CREATE TABLE svm_model_settings (
setting_name VARCHAR2(100),
setting_value VARCHAR2(100)
);

--specify the algorithm
BEGIN
 INSERT INTO svm_model_settings (setting_name, setting_value)
 values (dbms_data_mining.algo_name, dbms_data_mining.algo_support_vector_machines);

 INSERT INTO svm_model_settings (setting_name, setting_value)
 VALUES (dbms_data_mining.prep_auto,dbms_data_mining.prep_auto_on);
END;

--Build model
begin
DBMS_DATA_MINING.CREATE_MODEL(
   model_name => 'SVM_Model1',
```

```
  mining_function => dbms_data_mining.classification,
  data_table_name => 'bank_train',
  case_id_column_name => 'bank_id',
  target_column_name => 'y',
  settings_table_name => 'svm_model_settings');
 end;


-- describe the model settings tables
--describe user_mining_model_settings
-- List all the ODM models created in your Oracle schema => what machine learning models you have created
SELECT model_name,
  mining_function,
  algorithm,
  build_duration,
  model_size
FROM user_MINING_MODELS;


-- List the algorithm settings used for your machine learning model
SELECT setting_name,
  setting_value,
  setting_type
FROM user_mining_model_settings
WHERE model_name in 'SVM_MODEL1';


-- List the attribute the machine learning model uses. It may use a subset of the attributes.
-- This allows you to see what attributes were selected
```

```
SELECT attribute_name,
  attribute_type,
  usage_type,
  target
from all_mining_model_attributes
where model_name = 'SVM_MODEL1';
```

Model 3: Naïve Bayes Classifier

```
--create settings table
CREATE TABLE naive_bayes_model_settings (
setting_name VARCHAR2(30),
setting_value VARCHAR2(30)
);

BEGIN
  INSERT INTO naive_bayes_model_settings (setting_name, setting_value)
  VALUES (dbms_data_mining.algo_name,dbms_data_mining.algo_naive_bayes);

  INSERT INTO naive_bayes_model_settings (setting_name, setting_value)
  VALUES (dbms_data_mining.prep_auto,dbms_data_mining.prep_auto_on);
  COMMIT;
END;

-- Create Naive Bayes Classfier
BEGIN
DBMS_DATA_MINING.CREATE_MODEL(
  model_name => 'Naive_Bayes_Model',
```

```
  mining_function => dbms_data_mining.classification,

  data_table_name => 'bank_train',

  case_id_column_name => 'bank_id',

  target_column_name => 'y',

  settings_table_name => 'naive_bayes_model_settings');

END;
```

Step 3: Apply Models to Test Data

Once the models have been created, it is important to apply the model to test data set to determine whether the predicted outcomes match with the actual values. This is important to check whether the models are not specific only to the trained data and whether they can be applied in real world to any data set. Views have been created to compare the predicted and actual value of variable (whether customer subscribes to term deposit or not).

SQL Query:

Model 1: Decision Tree

```
--First we need to apply the model to the test data set

-- create a view that will contain the predicted outcomes => labeled data set

CREATE OR REPLACE VIEW bank_test_results

AS

SELECT bank_id,

  prediction(DECISION_TREE_MODEL1 USING *) predicted_value,

  prediction_probability(DECISION_TREE_MODEL1 USING *) probability

FROM bank_test;



-- Select the data containing the applied/labeled/scored data set

-- This will be used as input to the calculation of the confusion matrix

SELECT *

FROM BANK_TEST_RESULTS;
```

Model 2: Support Vector Machine

```
-- create a view that will contain the predicted outcomes => labeled data set
```

```
CREATE OR REPLACE VIEW svm_results

AS

SELECT bank_id,

  prediction(SVM_MODEL1 USING *) predicted_value,

  prediction_probability(SVM_MODEL1 USING *) probability

FROM bank_test;

-- Select the data containing the applied/labeled/scored data set

-- This will be used as input to the calculation of the confusion matrix

SELECT *

FROM svm_results;
```

<u>Model 3: Naïve Bayes Classifier</u>

```
--First we need to apply the model to the test data set

-- create a view that will contain the predicted outcomes => labeled data set

CREATE OR REPLACE VIEW Naive_Bayes_results

AS

SELECT BANK_ID,

  prediction(Naive_Bayes_Model Using *) predicted_value,

  prediction_probability(Naive_Bayes_Model USING *) probability

FROM BANK_TEST;
```

Step 4: Performance Evaluation using Confusion Matrix and Accuracy

In order to better understand the performance of the created models, confusion matrix has been created which gives the count of True Positives, True Negatives, False Positives and False Negatives. These parameters are used to determine accuracy of models for comparison.

<u>Model 1: Decision Tree</u>

```
--generate confusion matrix

DECLARE

  v_accuracy NUMBER;
```

```
BEGIN

DBMS_DATA_MINING.COMPUTE_CONFUSION_MATRIX (

  accuracy => v_accuracy,

  apply_result_table_name => 'BANK_TEST_RESULTS',

  target_table_name => 'bank_test',

  case_id_column_name => 'bank_id',

  target_column_name => 'y',

  confusion_matrix_table_name => 'bank_confusion_matrix',

  score_column_name => 'PREDICTED_VALUE',

  score_criterion_column_name => 'PROBABILITY',

  cost_matrix_table_name => null,

  apply_result_schema_name => null,

  target_schema_name => null,

  cost_matrix_schema_name => null,

  score_criterion_type => 'PROBABILITY');

  DBMS_OUTPUT.PUT_LINE('**** MODEL ACCURACY ****: ' || ROUND(v_accuracy,4));

END;

SELECT * FROM BANK_CONFUSION_MATRIX;
```

Model 2: Support Vector Machine

```
DECLARE

  v_accuracy NUMBER;

BEGIN

DBMS_DATA_MINING.COMPUTE_CONFUSION_MATRIX (

  accuracy => v_accuracy,

  apply_result_table_name => 'svm_results',

  target_table_name => 'bank_test',
```

```
  case_id_column_name => 'bank_id',

  target_column_name => 'y',

  confusion_matrix_table_name => 'svm_confusion_matrix',

  score_column_name => 'PREDICTED_VALUE',

  score_criterion_column_name => 'PROBABILITY',

  cost_matrix_table_name => null,

  apply_result_schema_name => null,

  target_schema_name => null,

  cost_matrix_schema_name => null,

  score_criterion_type => 'PROBABILITY');

  DBMS_OUTPUT.PUT_LINE('**** MODEL ACCURACY ****: ' || ROUND(v_accuracy,4));

END;

SELECT * FROM svm_confusion_matrix;
```

Model 3: Naïve Bayes Classifier

```
DECLARE v_accuracy NUMBER;

BEGIN

DBMS_DATA_MINING.COMPUTE_CONFUSION_MATRIX (

  accuracy => v_accuracy,

  apply_result_table_name => 'Naive_Bayes_results',

  target_table_name => 'bank_test',

  case_id_column_name => 'bank_id',

  target_column_name => 'y',

  confusion_matrix_table_name => 'Naive_Bayes_confusion_matrix',

  score_column_name => 'PREDICTED_VALUE',

  score_criterion_column_name => 'PROBABILITY',

  cost_matrix_table_name => null,
```

```
  apply_result_schema_name => null,

  target_schema_name => null,

  cost_matrix_schema_name => null,

  score_criterion_type => 'PROBABILITY');

  DBMS_OUTPUT.Put_LINE('**** MODEL ACCURACY ****: ' || ROUND(v_accuracy,4));

END;

SELECT * FROM Naive_Bayes_confusion_matrix;
```

Outputs:

<u>Confusion matrix for Decision Tree Model:</u>

Script Output × Query Result ×

SQL | All Rows Fetched: 4 in 0.019 seconds

	ACTUAL_TARGET_VALUE	PREDICTED_TARGET_VALUE	VALUE
1	yes	yes	725
2	no	no	9606
3	no	yes	398
4	yes	no	529

<u>Confusion matrix for Support Vector Machine Model:</u>

Script Output × Query... ×

SQL | All Rows Fetched: 4 in 0 seconds

	ACTUAL_TARGET_VALUE	PREDICTED_TARGET_VALUE	VALUE
1	yes	yes	712
2	no	no	9029
3	yes	no	542
4	no	yes	975

<u>Confusion matrix for Naïve Bayes Classifier Model:</u>

Script Output × Query Result ×

SQL | All Rows Fetched: 4 in 0.02 seconds

	ACTUAL_TARGET_VALUE	PREDICTED_TARGET_VALUE	VALUE
1	yes	yes	780
2	no	no	8928
3	yes	no	474
4	no	yes	1076

Based on the results obtained from confusion matrix, model accuracy has been computed using ((True Positives+True Negatives)/(Total number of predicted outcomes))*100. The accuracy of Decision Tree, Support Vector Machine and Naïve Bayes Classifier are 91.77%, 86.52% and 86.23% respectively. Hence, this implies Decision Tree model has maximum accuracy and is better than other models considered for predicting the outcome.

Confusion Matrix without Function

In this portion, we have designed a confusion matrix by writing the SQL query without using the Confusion_Matrix function. We have considered Decision Tree Model for manually creating confusion matrix and compared the results obtained here with the one obtained by using in-built function in order to check the accuracy of manually created confusion matrix.

First, a view is created which contains predicted value and actual values for the test dataset. Then true positives, true negatives, false positives and false negatives have been computed. Using these values, total negative rate, % correct values and accuracy have been manually calculated. Later, additional formatting and labelling has been performed on the output values to generate result in the specified format.

SQL Query:

Model Considered: Decision Tree

```
--Create View for comparison results

Create or Replace View DT_View

AS

SELECT a.BANK_ID, a.Y, b.predicted_value FROM

BANK_TEST a INNER JOIN BANK_TEST_RESULTS b ON a.bank_id=b.bank_id;



select * from DT_View;



--Manually create confusion matrix



create table model_output

(

Negative number,

Positive number,

Num number,
```

```
perc_correct number

);


insert into model_output values

(

(select count(*) as from DT_View where Y = 'no' AND predicted_value= 'no'),

(select count(*)from DT_View where Y = 'no' AND predicted_value= 'yes'),

((select count(*) as from DT_View where Y = 'no' AND predicted_value= 'no')+

(select count(*)from DT_View where Y = 'no' AND predicted_value= 'yes')

),

round(

(

((select count(*)from DT_View where Y = 'no' AND predicted_value= 'no')/

((select count(*)from DT_View where Y = 'no' AND predicted_value= 'no')+

(select count(*)from DT_View where Y = 'no' AND predicted_value= 'yes')

))*100

),2

)

);


insert into model_output values

(

(select count(*)from DT_View where Y = 'yes' AND predicted_value= 'no'),

(select count(*)from DT_View where Y = 'yes' AND predicted_value= 'yes'),

((select count(*)from DT_View where Y = 'yes' AND predicted_value= 'no')+

(select count(*)from DT_View where Y = 'yes' AND predicted_value= 'yes')
```

```
),

round(

(

((select count(*)from DT_View where Y = 'yes' AND predicted_value= 'yes')/

((select count(*)from DT_View where Y = 'yes' AND predicted_value= 'no')+

(select count(*)from DT_View where Y = 'yes' AND predicted_value= 'yes')

))*100

),2

)

);


create table temp

(

total number,

negative_rate number,

accuracy number

);


insert into temp values

(

(SELECT count(*) FROM dt_view),

100-round(

 (

 (

 (select count(*)from DT_View where Y = 'yes' AND predicted_value= 'yes')+(select count(*)from DT_View
where Y = 'no' AND predicted_value= 'no')

 )/
```

```
(select count(*) from dt_view)

)*100,2),

round(

(

(

(select count(*)from DT_View where Y = 'yes' AND predicted_value= 'yes')+(select count(*)from DT_View
where Y = 'no' AND predicted_value= 'no')

)/

(select count(*) from dt_view)

)*100,2)

);


select * from model_output;


create table temp1(

tot_neg number,

tot_pos number,

pos_neg number,

tot_neg_prc number,

tot_pos_prc number

);


insert into TEMP1 values

(

((select count(*)from DT_View where Y = 'no' AND predicted_value= 'no')+(select count(*)from DT_View
where Y = 'yes' AND predicted_value= 'no')),

((select count(*)from DT_View where Y = 'no' AND predicted_value= 'yes')+(select count(*)from DT_View
```

```
where Y = 'yes' AND predicted_value= 'yes')),

((select count(*)from DT_View where Y = 'no' AND predicted_value= 'no')+(select count(*)from DT_View
where Y = 'yes' AND predicted_value= 'no')) +

((select count(*)from DT_View where Y = 'no' AND predicted_value= 'yes')+(select count(*)from DT_View
where Y = 'yes' AND predicted_value= 'yes')),

round(((select count(*)from DT_View where Y = 'no' AND predicted_value= 'no')/((select count(*)from
DT_View where Y = 'no' AND predicted_value= 'no') +  (select count(*)from DT_View where Y = 'yes' AND
predicted_value= 'no')))*100,2),

round(((select count(*)from DT_View where Y = 'yes' AND predicted_value= 'yes')/((select count(*)from
DT_View where Y = 'yes' AND predicted_value= 'yes') +  (select count(*)from DT_View where Y = 'no' AND
predicted_value= 'yes')))*100,2)

 );


  SET SERVEROUTPUT ON

declare

total number;

negative_rate number;

accuracy number;

true_positive number;

true_negative number;

false_negative number;

false_positive number;

actual_negative_sum number;

actual_positive_sum number;

perc_actual_neg number;

perc_actual_pos number;

 tot_neg number;

 tot_pos number;

 pos_neg number;
```

```
 tot_neg_prc number;

 tot_pos_prc number;

BEGIN

SELECT total

      INTO total

    FROM temp;

SELECT NEGATIVE_RATE

      INTO negative_rate

    FROM temp;

SELECT accuracy

      INTO accuracy

    FROM temp;

SELECT negative

     INTO true_negative

    FROM MODEL_OUTPUT

    where negative=9606;

SELECT positive

     INTO false_positive

    FROM MODEL_OUTPUT

    where positive=398;

SELECT negative

     INTO false_negative

    FROM MODEL_OUTPUT

    where negative=529;

SELECT positive

     INTO true_positive
```

```
  FROM MODEL_OUTPUT
  where positive=725;
SELECT num
   INTO actual_negative_sum
  FROM MODEL_OUTPUT
  where num=10004;
SELECT num
   INTO actual_positive_sum
  FROM MODEL_OUTPUT
  where num=1254;
SELECT perc_correct
   INTO perc_actual_neg
  FROM MODEL_OUTPUT
  where perc_correct=96.02;
SELECT perc_correct
   INTO perc_actual_pos
  FROM MODEL_OUTPUT
  where perc_correct=57.81;
SELECT tot_neg
   INTO tot_neg
  FROM TEMP1;
SELECT tot_pos
   INTO tot_pos
  FROM TEMP1;
SELECT pos_neg
   INTO pos_neg
```

```
   FROM TEMP1;

SELECT tot_neg_prc

    INTO tot_neg_prc

   FROM TEMP1;

SELECT tot_pos_prc

    INTO tot_pos_prc

   FROM TEMP1;

   Dbms_Output.Put_Line('------------------------------------------------------------------------');

   Dbms_Output.Put_Line('|--------------------------Confusion Matrix-----------------------------|');

   Dbms_Output.Put_Line('|                                                           |');

   Dbms_Output.Put_Line('| Table contains : ' ||total ||' records                                   |');

   Dbms_Output.Put_Line('|                                                           |');

   Dbms_Output.Put_Line('|                |   Negative   |   Positive  |  Num        (% Correct)|');

   Dbms_Output.Put_Line('| Actual Negative  |   '||true_negative||'       |  '||false_positive||'        |
'||actual_negative_sum||'        ('||perc_actual_neg||'%)  |');

   Dbms_Output.Put_Line('| Actual Positive  |   '||false_negative||'       |  '||true_positive|| '        |
'||actual_positive_sum||'        ('||perc_actual_pos||'%)  |');

   Dbms_Output.Put_Line('|------------------|--------------|-------------|-------------------------|');

   Dbms_Output.Put_Line('|   Column totals  |   '||tot_neg||'      |  '||tot_pos||'      | '||pos_neg||'
|');

   Dbms_Output.Put_Line('|                |  ('||tot_neg_prc||'%)    |  ('||tot_pos_prc||'%)   |                      |');

   Dbms_Output.Put_Line('|                                                           |');

   Dbms_Output.Put_Line('| Negative Rate = ' || negative_rate|| '%                    Accuracy = ' || accuracy ||'%
|');

   Dbms_Output.Put_Line('|                                                           |');

   Dbms_Output.Put_Line('|-----------------------------------------------------------------------|');

   Dbms_Output.Put_Line('------------------------------------------------------------------------');

END;
```

Outcome:

The output obtained after running above SQL query is:

```
----------------------------------------------------------------------------
|--------------------------Confusion Matrix------------------------------|
|                                                                          |
| Table contains : 11258 records                                           |
|                                                                          |
|                  |   Negative    |   Positive   |  Num        (% Correct)|
| Actual Negative  |   9606        |  398         | 10004        (96.02%)  |
| Actual Positive  |   529         |  725         | 1254         (57.81%)  |
|------------------|---------------|--------------|------------------------|
|   Column totals  |   10135       |  1123        | 11258                  |
|                  |  (94.78%)     |  (64.56%)    |                        |
|                                                                          |
| Negative Rate = 8.23%                        Accuracy = 91.77%           |
|                                                                          |
|--------------------------------------------------------------------------|
----------------------------------------------------------------------------
```

Result Comparison:

For Decision Tree, the accuracy value obtained using Confusion_Matrix in-built function and manually written query is 91.77% which is same in both cases. Hence, the queries written to generate such value are correct.