Machine Learning using SQL

First step before performing any machine learning model we have to split our dataset into training and test datasets. This is done so that over-sampling does not occur.

Here, the data set is split into 40 for training data and 60 for test/validation dataset. To ensure that same values in training data are not present in validation dataset, a not in selection based on customer id is performed.

```
create table trainingdata as (select * from bankdata sample(40) seed(40));

create table validdata as (select * from bankdata where customer_id not in (select customer_id from trainingdata));
```

First model – Decision tree:

Settings:

```
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 the model for the decision tree, with y as the target attribute/feature:

```
BEGIN
DBMS_DATA_MINING.CREATE_MODEL(
  model_name => 'Decision_Tree_Model3',
  mining_function => dbms_data_mining.classification,
  data_table_name => 'trainingdata',
  case_id_column_name => 'customer_id',
  target_column_name => 'y',
  settings_table_name => 'decision_tree_model_settings');
END;
```

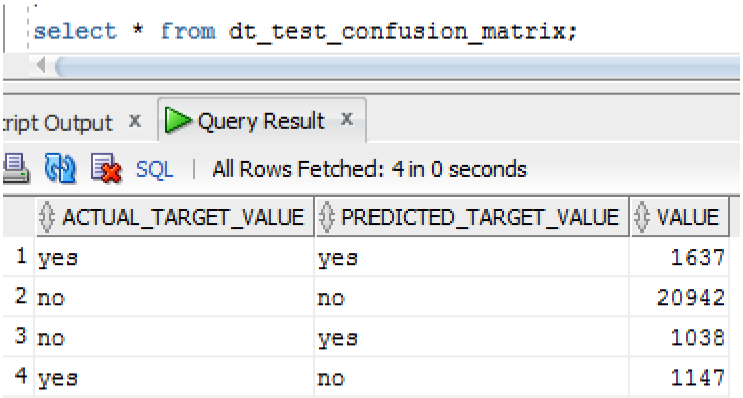
Creating a new view to store the predicted values using Valid dataset.

```
CREATE OR REPLACE VIEW dt_test_results
AS
SELECT customer_id,
  prediction(Decision_Tree_Model3 USING *) predicted_value,
  prediction_probability(Decision_Tree_Model3 USING *) probability
FROM validdata;
```

Generating the confusion matrix to get the accuracy of prediction performed by the model

```
DECLARE
  v_accuracy NUMBER;
BEGIN
DBMS_DATA_MINING.COMPUTE_CONFUSION_MATRIX (
  accuracy => v_accuracy,
  apply_result_table_name => 'dt_test_results',
  target_table_name => 'validdata',
  case_id_column_name => 'customer_id',
  target_column_name => 'y',
  confusion_matrix_table_name => 'dt_test_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 dt_test_confusion_matrix;
```



```
select * from dt_test_confusion_matrix;
```

All Rows Fetched: 4 in 0 seconds

	ACTUAL_TARGET_VALUE	PREDICTED_TARGET_VALUE	VALUE
1	yes	yes	1637
2	no	no	20942
3	no	yes	1038
4	yes	no	1147

**** DT MODEL ACCURACY ****: 91.18%

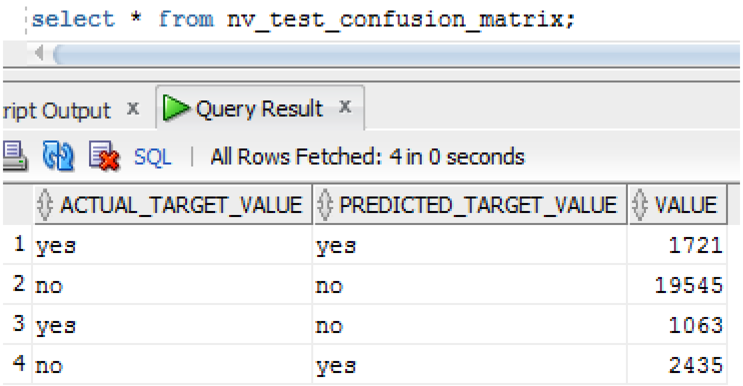
Decision tree model generates an accuracy of 91.18%

NAÏVE BAYES Model:

```
CREATE TABLE nv_model_setting ( setting_name VARCHAR2(30), setting_value
VARCHAR2(30));
```

```
BEGIN
  INSERT INTO nv_model_setting (setting_name, setting_value)
  VALUES (dbms_data_mining.algo_name,dbms_data_mining.ALGO_NAIVE_BAYES);

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

/
```

```
BEGIN
DBMS_DATA_MINING.CREATE_MODEL(
  model_name => 'NV_MODEL_TEST',
  mining_function => dbms_data_mining.classification,
  data_table_name => 'trainingdata',
  case_id_column_name => 'customer_id',
  target_column_name => 'y',
  settings_table_name => 'nv_model_setting');
END;
/
CREATE OR REPLACE VIEW nv_test_results_v
AS
SELECT customer_id,
  prediction(NV_MODEL_TEST USING *) predicted_value,
  prediction_probability(NV_MODEL_TEST USING *) probability
FROM validdata;
/
DECLARE
  v_accuracy NUMBER;
BEGIN
DBMS_DATA_MINING.COMPUTE_CONFUSION_MATRIX (
  accuracy => v_accuracy,
  apply_result_table_name => 'nv_test_results_v',
  target_table_name => 'validdata',
  case_id_column_name => 'customer_id',
  target_column_name => 'y',
  confusion_matrix_table_name => 'nv_test_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('**** NB MODEL ACCURACY ****: ' ||
ROUND(v_accuracy,4)*100||'%');

END;

/
```



```
select * from nv_test_confusion_matrix;
```

All Rows Fetched: 4 in 0 seconds

	ACTUAL_TARGET_VALUE	PREDICTED_TARGET_VALUE	VALUE
1	yes	yes	1721
2	no	no	19545
3	yes	no	1063
4	no	yes	2435

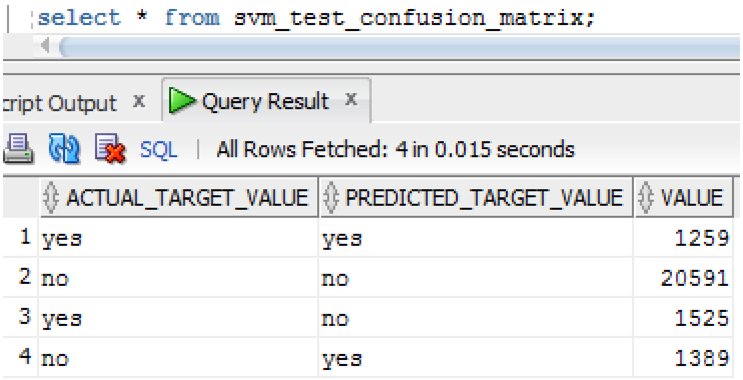
```
**** NB MODEL ACCURACY ****: 85.87%
```

The Naïve Bayes model predicts with an accuracy of 85.87%

Support Vector Machine model:

```
CREATE TABLE svm_settings (

setting_name VARCHAR2(30),

setting_value VARCHAR2(30));

/

BEGIN

  INSERT INTO svm_settings (setting_name, setting_value)

  VALUES
(dbms_data_mining.algo_name,dbms_data_mining.ALGO_SUPPORT_VECTOR_MACHINES);


  INSERT INTO svm_settings (setting_name, setting_value)

  VALUES (dbms_data_mining.prep_auto,dbms_data_mining.prep_auto_on);

  COMMIT;

END;

/
```

```
BEGIN
DBMS_DATA_MINING.CREATE_MODEL(
  model_name => 'SVM_MODEL',
  mining_function => dbms_data_mining.classification,
  data_table_name => 'trainingdata',
  case_id_column_name => 'customer_id',
  target_column_name => 'y',
  settings_table_name => 'svm_settings');
END;
/
CREATE OR REPLACE VIEW svm_test_results_v
AS
SELECT customer_id,
  prediction(SVM_MODEL USING *) predicted_value,
  prediction_probability(SVM_MODEL USING *) probability
FROM validdata;
/
DECLARE
  v_accuracy NUMBER;
BEGIN
DBMS_DATA_MINING.COMPUTE_CONFUSION_MATRIX (
  accuracy => v_accuracy,
  apply_result_table_name => 'svm_test_results_v',
  target_table_name => 'validdata',
  case_id_column_name => 'customer_id',
  target_column_name => 'y',
  confusion_matrix_table_name => 'svm_test_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('**** SVM MODEL ACCURACY ****: ' ||
ROUND(v_accuracy,4)*100||'%');

END;
```



```
select * from svm_test_confusion_matrix;
```

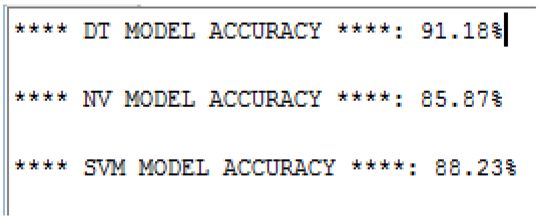
All Rows Fetched: 4 in 0.015 seconds

	ACTUAL_TARGET_VALUE	PREDICTED_TARGET_VALUE	VALUE
1	yes	yes	1259
2	no	no	20591
3	yes	no	1525
4	no	yes	1389

**** SVM MODEL ACCURACY ****: 88.23%

The SVM Model predicts with an accuracy of 88.23%

MODEL COMPARISON:



```
**** DT MODEL ACCURACY ****: 91.18%

**** NV MODEL ACCURACY ****: 85.87%

**** SVM MODEL ACCURACY ****: 88.23%
```

Based on the results of the three models, we can confidently conclude that the Decision tree model performs the best with an accuracy of 91.18%.

PL/SQL CODE:

In this section we have to create a confusion matrix using a PL/SQL procedure, without using the in-built confusion matrix function.

Definition:

A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known.

This confusion matrix is built upon the decision tree model using the prediction function.

A view is created to store the customer id, actual value or target variable y, the predicted variable using prediction function and the probability of the variable using probability function:

```
CREATE OR REPLACE VIEW dt_test_labeled_conf

AS

SELECT customer_id, y as actual_value,

  prediction(Decision_Tree_Model3 USING *) predicted_value,

  prediction_probability(Decision_Tree_Model3 USING *) probability

FROM validdata;
```

A PL/SQL procedure to select values from the view that was created above:

The actual value and predicted values are grouped and selected into four variables as following:

True negative TN - If the actual value and predicted value have no.

True positive TP - If the actual value and predicted value have yes.

False negative FN - If the actual value is yes and predicted value is no.

False positive FP - If the actual value is no and predicted value is yes.

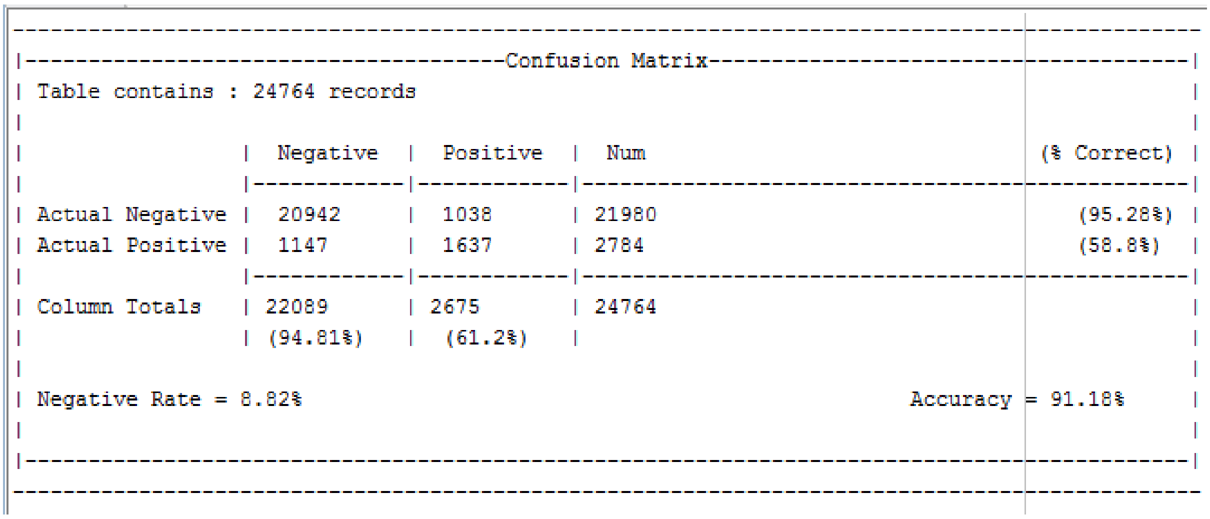
And then these values are tabulated into the output as shown in the assignment specification:

```
DECLARE

TN NUMBER;

FP NUMBER;
```

```
FN NUMBER;
TP NUMBER;
test_data NUMBER;

BEGIN
select count INTO TN from
(SELECT actual_value,predicted_value, count(*) as count
FROM dt_test_labeled_conf
group by actual_value,predicted_value)
where actual_value='no' and predicted_value='no';

select count INTO FP from
(SELECT actual_value,predicted_value, count(*) as count
FROM dt_test_labeled_conf
group by actual_value,predicted_value)
where actual_value='no' and predicted_value='yes';

select count INTO FN from
(SELECT actual_value,predicted_value, count(*) as count
FROM dt_test_labeled_conf
group by actual_value,predicted_value)
where actual_value='yes' and predicted_value='no';

select count INTO TP from
(SELECT actual_value,predicted_value, count(*) as count
FROM dt_test_labeled_conf
group by actual_value,predicted_value)
where actual_value='yes' and predicted_value='yes';

select count(*) into test_data from dt_test_labeled_conf;
```

```
dbms_output.put_line ('------------------------------------------------------------------------------------------
----');

dbms_output.put_line ('|--------------------------------------Confusion Matrix----------------------------
----------|');

dbms_output.put_line ('| Table contains : '  || test_data || ' records
|');

dbms_output.put_line ('|                                                                      |');

dbms_output.put_line ('|              |  Negative  |  Positive  |  Num                         (%
Correct) |');

dbms_output.put_line ('|              |------------|------------|---------------------------------------------
-|');

dbms_output.put_line ('| Actual Negative |  '||TN||'     |  '||FP||'     | '||(TN+FP) ||'
('|| ROUND( (TN/(TN+FP)),4)*100 ||'%) |');

dbms_output.put_line ('| Actual Positive |  '||FN||'     |  '||TP||'     | '||(FN+TP) ||'
('|| ROUND( (TP/(FN+TP)),4)*100 ||'%) |');

dbms_output.put_line ('|              |------------|------------|---------------------------------------------
-|');

dbms_output.put_line ('| Column Totals   | '||(TN+FN)||'     | '||(FP+TP)||'      | '
||test_data||'                              |');

dbms_output.put_line ('|              | ('||ROUND( (TN/(TN+FN)),4)*100||'%)   |  (' ||ROUND(
(TP/(TP+FP)),4)*100 ||'%)  |                                  |');

dbms_output.put_line ('|                                                                      |');

dbms_output.put_line ('| Negative Rate = '||ROUND( ((FN+FP)/test_data),4)*100||'%
Accuracy = '||ROUND( ((TN+TP)/test_data),4)*100||'%     |');

dbms_output.put_line ('|                                                                      |');

dbms_output.put_line ('|------------------------------------------------------------------------------------------
---|');

dbms_output.put_line ('------------------------------------------------------------------------------------------
----');

END;
```

Screenshot generated using above function:



```
----------------------------------------------------------------------------------------------
|-------------------------------------Confusion Matrix---------------------------------------|
| Table contains : 24764 records                                                              |
|                                                                                            |
|                 |  Negative  |  Positive  |  Num                               (% Correct) |
|                 |------------|------------|------------------------------------------------|
| Actual Negative |  20942     |  1038      | 21980                                (95.28%)  |
| Actual Positive |  1147      |  1637      | 2784                                 (58.8%)   |
|                 |------------|------------|------------------------------------------------|
| Column Totals   | 22089      | 2675       | 24764                                          |
|                 | (94.81%)   |  (61.2%)   |                                                |
|                                                                                            |
| Negative Rate = 8.82%                                                Accuracy = 91.18%     |
|                                                                                            |
|--------------------------------------------------------------------------------------------|
----------------------------------------------------------------------------------------------
```