SANTANDER CUSTOMER TRANSACTION PREDICTION USING R AND PYTHON

Prepared by: **MOULEESHWARAN.S**

Prepared on: 30.08.2019

INTRODUCTION

Background:

At Santander, mission is to help people and businesses prosper. We are always looking for ways to help our customers understand their financial health and identify which products and services might help them achieve their monetary goals.

Our data science team is continually challenging our machine learning algorithms, working with the global data science community to make sure we can more accurately identify new ways to solve our most common challenge, binary classification problems such as:

- is a customer satisfied?
- Will a customer buy this product?
- Can a customer pay this loan?

Problem Statement:

In this challenge, we need to identify which customers will make a specific transaction in the future, irrespective of the amount of money transacted.

Data set:

We have two types of dataset:

- ✓ One is train data set which contains target variable.
- ✓ Another is test data set which does not contain target variable.

TRAIN DATASET:

```
train.head()
```

	ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	...	var_190	var_191	var_192	var_193	var_194	var_195	var_196
0	train_0	0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187	18.6266	...	4.4354	3.9642	3.1364	1.6910	18.5227	-2.3978	7.8784
1	train_1	0	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208	16.5338	...	7.6421	7.7214	2.5837	10.9516	15.4305	2.0339	8.1267
2	train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155	...	2.9057	9.7905	1.6704	1.6858	21.6042	3.1417	-6.5213
3	train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.9250	...	4.4666	4.7433	0.7178	1.4214	23.0347	-1.2706	-2.9275
4	train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405	19.2514	...	-1.4905	9.5214	-0.1508	9.1942	13.2876	-1.5121	3.9267

5 rows × 202 columns

TEST DATASET:

```
test.head()
```

	ID_code	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	...	var_190	var_191	var_192	var_193	var_194	var_195	var_196
0	test_0	11.0656	7.7798	12.9536	9.4292	11.4327	-2.3805	5.8493	18.2675	2.1337	...	-2.1556	11.8495	-1.4300	2.4508	13.7112	2.4669	4.3654
1	test_1	8.5304	1.2543	11.3047	5.1858	9.1974	-4.0117	6.0196	18.6316	-4.4131	...	10.6165	8.8349	0.9403	10.1282	15.5765	0.4773	-1.4852
2	test_2	5.4827	-10.3581	10.1407	7.0479	10.2628	9.8052	4.8950	20.2537	1.5233	...	-0.7484	10.9935	1.9803	2.1800	12.9813	2.1281	-7.1086
3	test_3	8.5374	-1.3222	12.0220	6.5749	8.8458	3.1744	4.9397	20.5660	3.3755	...	9.5702	9.0766	1.6580	3.5813	15.1874	3.1656	3.9567
4	test_4	11.7058	-0.1327	14.1295	7.7506	9.1035	-8.5848	6.8595	10.6048	2.9890	...	4.2259	9.1723	1.2835	3.3778	19.5542	-0.2860	-5.1612

5 rows × 201 columns

Our goal:

We need to predict the target variable in the test dataset with the help of building ML algorithms by using train dataset.

Train data set contains target variable which has two observations

- 0 indicates no which means customer does not make transaction
- 1 indicates yes which means customer makes transactions irrespective to amount

METHODS AND ANALYSIS

LOADING REQUIRED LIBRARIES FOR PYTHON AS WELL AS R

a) For python

- import os
- import pandas as pd
- import numpy as np
- import matplotlib.pyplot as plt
- import seaborn as sns
- import statsmodels.api as sm
- from random import randrange, uniform

b) For Rscript

- library(gridExtra)
- library(grid)
- library(ggplot2)
- library(caret)
- library(rpart)
- library (DataCombine)
- library (ROSE)
- library(e1071)
- library(xgboost)

Here I have mentioned some of the basic libraries required for some operations. If need we need to call the required libraries for performing the operations

EXPLORATORY DATA ANALYSIS:

We need to get more information about our data.

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Columns: 202 entries, ID_code to var_199
dtypes: float64(200), int64(1), object(1)
memory usage: 308.2+ MB
```

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Columns: 201 entries, ID_code to var_199
dtypes: float64(200), object(1)
memory usage: 306.7+ MB
```

Above diagram shows the dimensions, data types and memory consumed by the data

```
In [8]: train.describe()
```

Out[8]:

	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_
count	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.00000
mean	0.100490	10.679914	-1.627622	10.715192	6.796529	11.078333	-5.065317	5.408949	16.545850	0.28416
std	0.300653	3.040051	4.050044	2.640894	2.043319	1.623150	7.863267	0.866607	3.418076	3.33263
min	0.000000	0.408400	-15.043400	2.117100	-0.040200	5.074800	-32.562600	2.347300	5.349700	-10.50550
25%	0.000000	8.453850	-4.740025	8.722475	5.254075	9.883175	-11.200350	4.767700	13.943800	-2.31780
50%	0.000000	10.524750	-1.608050	10.580000	6.825000	11.108250	-4.833150	5.385100	16.456800	0.39370
75%	0.000000	12.758200	1.358625	12.516700	8.324100	12.261125	0.924800	6.003000	19.102900	2.93790
max	1.000000	20.315000	10.376800	19.353000	13.188300	16.671400	17.251600	8.447700	27.691800	10.15130

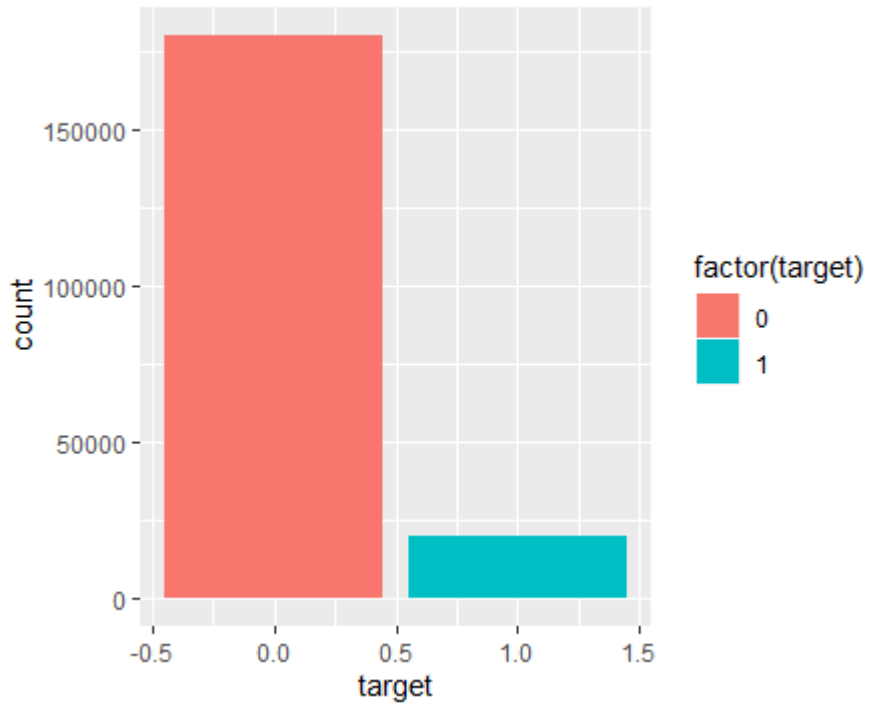
```
test.describe()
```

	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	var_
count	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.00000
mean	10.658737	-1.624244	10.707452	6.788214	11.076399	-5.050558	5.415164	16.529143	0.277135	7.56940
std	3.036716	4.040509	2.633888	2.052724	1.616456	7.869293	0.864686	3.424482	3.333375	1.23186
min	0.188700	-15.043400	2.355200	-0.022400	5.484400	-27.767000	2.216400	5.713700	-9.956000	4.24330
25%	8.442975	-4.700125	8.735600	5.230500	9.891075	-11.201400	4.772600	13.933900	-2.303900	6.62380
50%	10.513800	-1.590500	10.560700	6.822350	11.099750	-4.834100	5.391600	16.422700	0.372000	7.63200
75%	12.739600	1.343400	12.495025	8.327600	12.253400	0.942575	6.005800	19.094550	2.930025	8.58482
max	22.323400	9.385100	18.714100	13.142000	16.037100	17.253700	8.302500	28.292800	9.665500	11.00360

8 rows × 200 columns

Above diagram shows the summary of the dataset

Now let us see the distribution of target value with diagrammatic illustrations



Above diagram clearly shows that about 10% of the dataset have value of 1(yes) and remaining are 0. our dependent variable is biased.

MISSING VALUE ANALYSIS:

We need to check whether there are any missing values present in the dataset

Missing values can be occurred due to human error or if the field marked as an optional. There is no use of that data hence we need to do either drop the variable or observation or impute with any one of the methods like mean, mode and KNN imputation for numerical variables and median method for categorical variables.

```
In [11]: missing_val
Out[11]: ID_code      0
         target       0
         var_0        0
         var_1        0
         var_2        0
         var_3        0
         var_4        0
         var_5        0
         var_6        0
         var_7        0
         var_8        0
         var_9        0
         var_10       0
         var_11       0
         var_12       0
         var_13       0
         var_14       0
         var_15       0
         var_16       0
```

```
In [11]: missing_val
         var_182    0
         var_183    0
         var_184    0
         var_185    0
         var_186    0
         var_187    0
         var_188    0
         var_189    0
         var_190    0
         var_191    0
         var_192    0
         var_193    0
         var_194    0
         var_195    0
         var_196    0
         var_197    0
         var_198    0
         var_199    0
         Length: 202, dtype: int64
```

In our dataset there is no missing values. So, we are free from the use of missing value analysis.

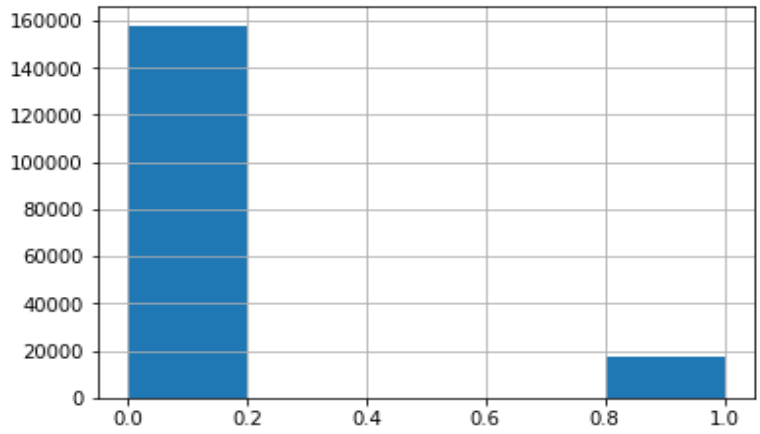
We can simply jump to outlier analysis.

OUTLIER ANALYSIS

Outliers are inconsistent data which is not lying under the range of other variables. outliers may lead the model towards inconsistent variable and biased towards it in our data set outliers are detected and removed.it makes our dataset small and doesn't affect our whole dataset.

A box plot is a method for graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending vertically from the boxes (whiskers) indicating variability outside the upper and lower quartiles, hence the terms box-and-whisker plot and box-and-whisker diagrams. Outliers may be plotted as individual points.

```
In [24]: train.shape

Out[24]: (175073, 202)

In [25]: train['target'].value_counts()

Out[25]: 0    157970
         1     17103
         Name: target, dtype: int64

In [27]: train['target'].value_counts()*100/len(train)

In [29]: train['target'].hist(bins=5)

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x247e5ad0ef0>
```

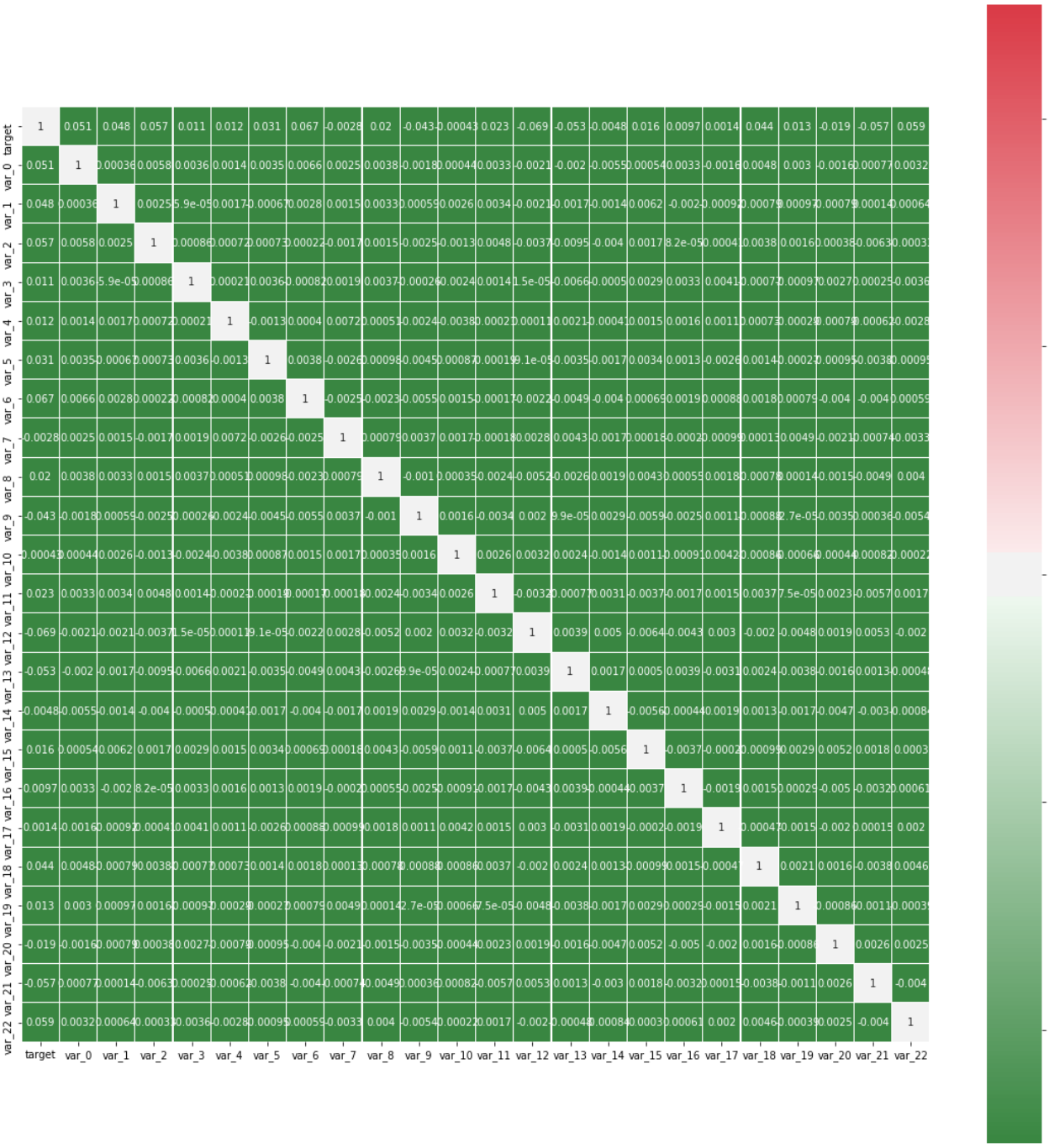


Above represents the shape and distribution of target value after performing outlier analysis.

FEATURE SELECTION:

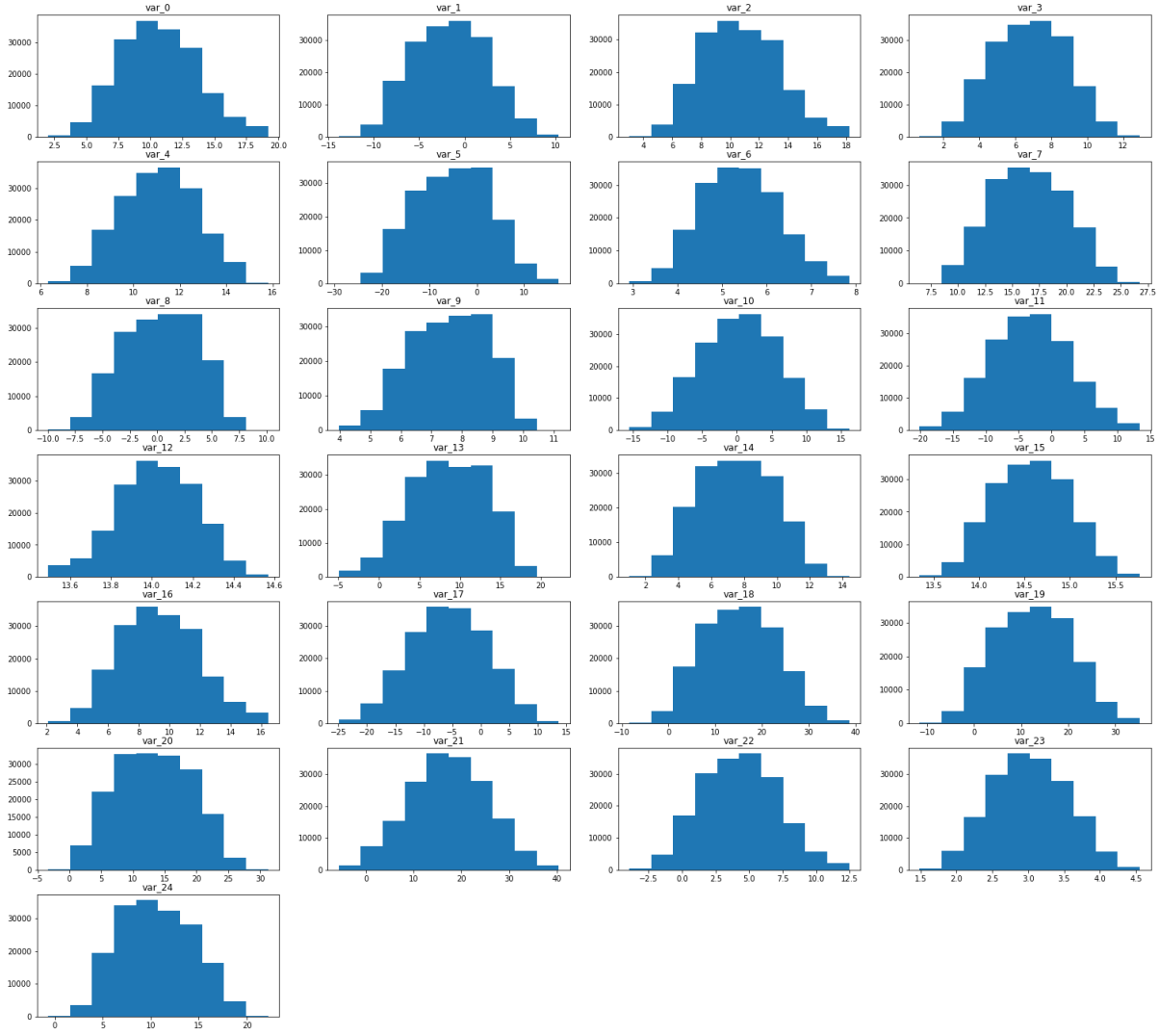
Some have sensitive data which might enhance the prediction of the target variable and some variable can contain noise which have mere any impact on the prediction. We can bring to use various plots and heat maps which will show us the importance of the variable and whether the act as a duplicate to other variable in which case only one of them will suffice for our model development. Co-relation analysis determines whether the correlation between two variables is positive, negative or have no correlation at all. Say if correlation between two variables is 1, we can drop one of them as both of the variable will fetch the same output.

From the below plot, we can clearly know that the variables are not correlated to each other.


target var_0 var_1 var_2 var_3 var_4 var_5 var_6 var_7 var_8 var_9 var_10 var_11 var_12 var_13 var_14 var_15 var_16 var_17 var_18 var_19 var_20 var_21 var_22
1.08
1.04
1.00
0.96
0.92

FEATURE SCALING:

We can convert each variable on a single comparison number starting from 0 to 1. This can be applied on the dataset which contains multiple variables and huge size of data like our current dataset. Variables containing different measure scale can cause anomalies which while doing a comparison may not be interpreted in the correct manner. This step will reduce unwanted variation either within or between variables. It also brings all of the variables into proportion with one another. Normalization was used to get all the variables in line to a single range.



The above plot shows the distribution of data for the first 25 columns

MODEL DEVELOPMENT

In our dataset, I am going to use different models in both languages. They are mentioned below as follows:

- **IN PYTHON:**
    1. Decision tree
    2. Random forest
    3. KNN Imputation
    4. Logistic Regression
    5. Naive Bayes
    6. SGDClassifier

- **IN RSCRIPT:**
    1. Logistic Regression
    2. Decision tree
    3. Naive Bayes
    4. XGBoost

SAMPLING METHODS:

Before training the model, we need to treat the imbalanced data.so that it may give accurate results. We can achieve it using sampling methods. There are many different methods for drawing samples from data; the ideal one depends on the data set and situation. Sampling can be based on probability an approach that uses random numbers that correspond to points in the data set to ensure that there is no correlation between points chosen for the sample. They are

1. Simple random sampling
2. Stratified sampling
3. Over sampling
4. Under sampling

IN PYTHON - I had used stratified sampling method for treating the imbalanced dependent variable for all models

In R SCRIPT – I had used random over sampling, stratified sampling and non-sampled data based on the different models

DECISION TREE:

It is very powerful tool for both classification and prediction. It has two most used algorithm methods.

1. C5.0
2. CART

C5.0 uses entropy (Information Gain) as major splitting criteria and **CART** uses Gini index as prominent criteria

In my model **C5.0** is used and the evaluation metrics are mentioned below.

IN PYTHON:

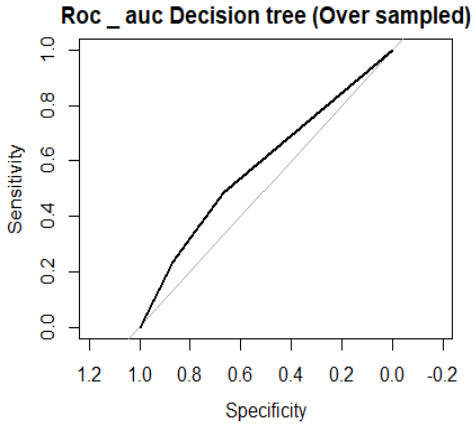
After sampling I had split the whole dataset into train data and test data for predicting the results of my trained data. Here is my result of the trained data using my model.

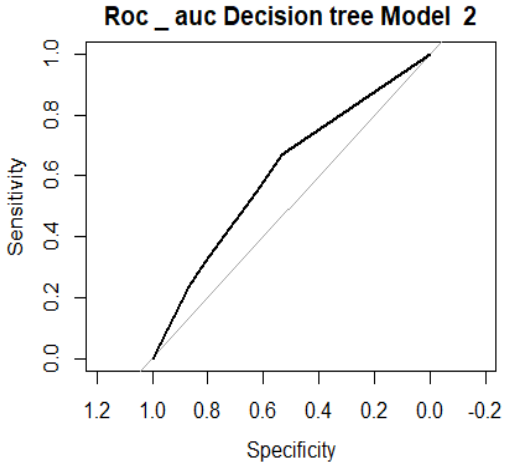
Accuracy =58.71
FNR = 41.62
Recall = 0.58
Precision = 0.58

IN R SCRIPT:

I had used over sampling method (MODEL 1) and manually initialising the depth of the tree (MODEL 2) for evaluating my test data. Below is the result of my trained data.

```
#####################  MODEL 1  ########
# [1] "accuracy 0.65"
# [1] "precision 0.14"
# [1] "recall 0.33"
# [1] "fpr 0.33"
# [1] "fnr 0.51"
# [1] "f1 0.2"
#Area under the curve: 0.5843
```






```
#######   MODEL 2  ######
#[1] "accuracy 0.55"
#[1] "precision 0.14"
#[1] "recall 0.47"
#[1] "fpr 0.47"
#[1] "fnr 0.33"
#[1] "f1 0.21"
#Area under the curve: 0.6124
```

LOGISTIC REGRESSION:

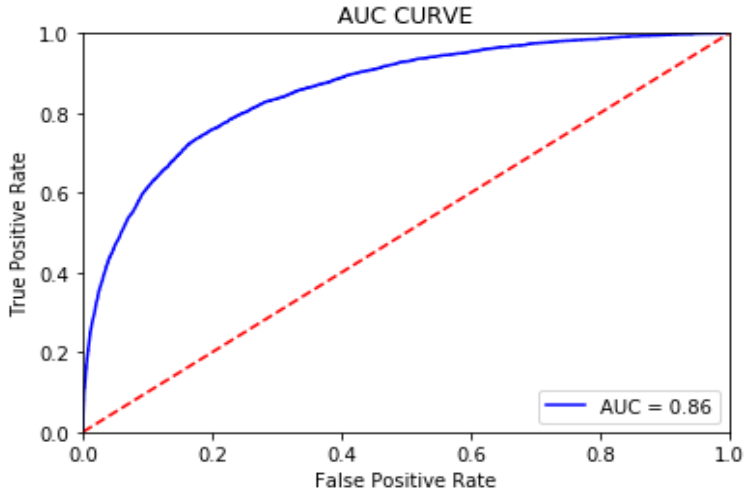
Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. For using this model there should be no outliers.it is also predictive analysis. It mainly identifies relationships between the data.

Logistic Regression is a supervised machine learning algorithm where the predicted output is categorical.

IN PYTHON:

Data is fragmented to train and test data using under sampling methods. In our dataset we got,

Accuracy = 76.98
FNR = 23.03
Recall = 0.77
precision = 0.76
AUC = 0.86



IN R SCRIPT:

Here I have used stratified sample data for logistic base model **(MODEL 1)** and over sampling method for **MODEL 2**. Output of the trained model is mentioned below

```
################### MODEL 1 ############
#[1] "accuracy 0.91"
#[2] "precision 0.69"
#[3] "recall 0.01"
#[4] "fpr 0.01"
#[5] "fnr 0.73"
#[6] "f1 0.03"
#Area under the curve: 0.86
```

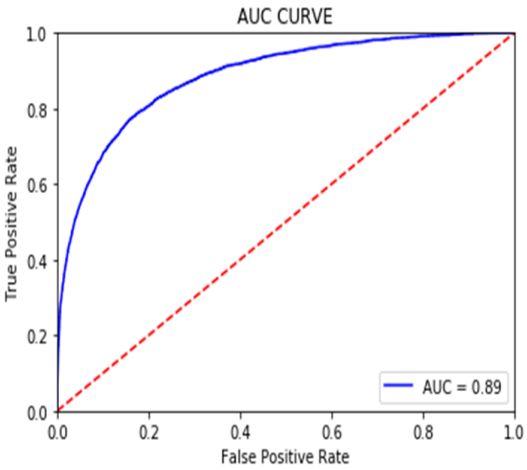
```
######## MODEL 2 ##################
#[1] "accuracy 0.78"
#[2] "precision 0.28"
#[3] "recall 0.22"
#[4] "fpr 0.22"
#[5] "fnr 0.22"
#[6] "f1 0.25"
#Area under the curve: 0.8596
```

NAIVE BAYES:

It is classification and practical learning algorithm.it classifies based on the probability. It mainly works on Bayes theorem to predict the probability of the dataset.

IN PYTHON:

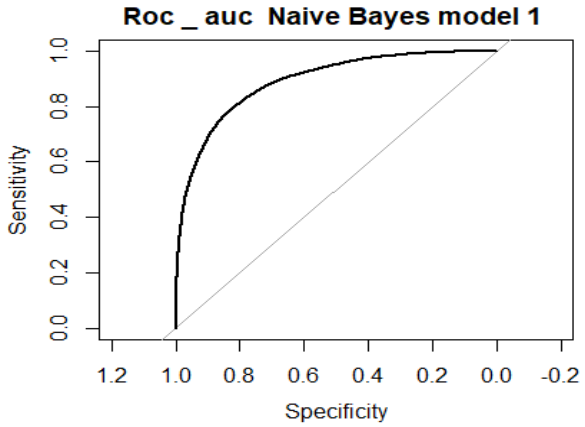
```
FOR NAIVE BAYES MODEL
Accuracy = 80.15
FNR = 20.75
Recall = 0.79
precision = 0.80
AUC CURVE = 0.89
```



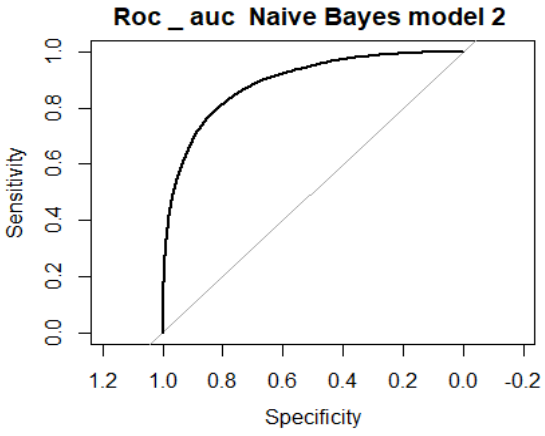
IN R SCRIPT:

MODEL 1 – STRATIFIED SAMPLING METHOD

```
################  MODEL 1 ##############
# [1] "accuracy 0.92"
# [1] "precision 0.72"
# [1] "recall 0.02"
# [1] "fpr 0.02"
# [1] "fnr 0.63"
# [1] "f1 0.03"
#Area under the curve: 0.8894
```



MODEL2 – OVER SAMPLING METHOD:




```
###############   MODEL 2 ##############
# [1] "accuracy 0.81"
# [1] "precision 0.32"
# [1] "recall 0.19"
# [1] "fpr 0.19"
# [1] "fnr 0.21"
# [1] "f1 0.24"
# Area under the curve: 0.8893
```

Below mentioned models are used only in python only

RANDOM FOREST:

Random forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. In our dataset we got,

```
FOR RANDOM FOREST MODEL
with n =500
Accuracy = 75.94
FNR = 20.55
Recall = 0.79
precision = 0.73

with n =100
Accuracy = 89.93
FNR = 99.93
```

K-NEAREST NEIGHBOUR(KNN):

KNN is simple algorithm that stores all available cases and classifies new cases based on a similarity measure. It is a lazy algorithm because it takes more time to train the model. In our data set we got,

```
FOR K NEAREST NEIGHBOUR MODEL
Accuracy = 56.4
FNR = 83
Recall = 0.17
precision = 0.18
```

SGD CLASSIFIER:

Stochastic Gradient Descent trains the model by randomly taking set of data in the whole dataset for each iteration. By using that information, it trains our model.it is very simple and efficient to use. In our dataset we got,

```
FOR SGDClssifier MODEL
Accuracy = 60.31
FNR = 1.94
Recall = 0.98
precision = 0.55
```

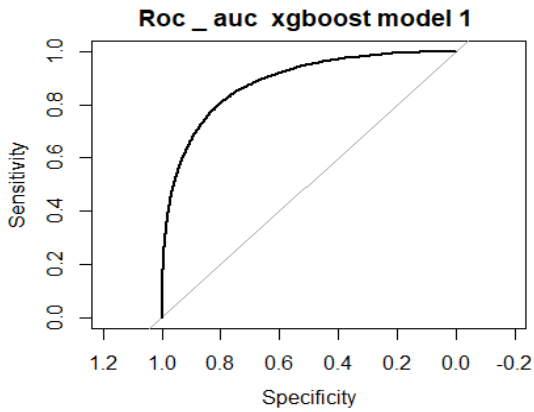
Below mentioned model is used only in R

XGBOOST:

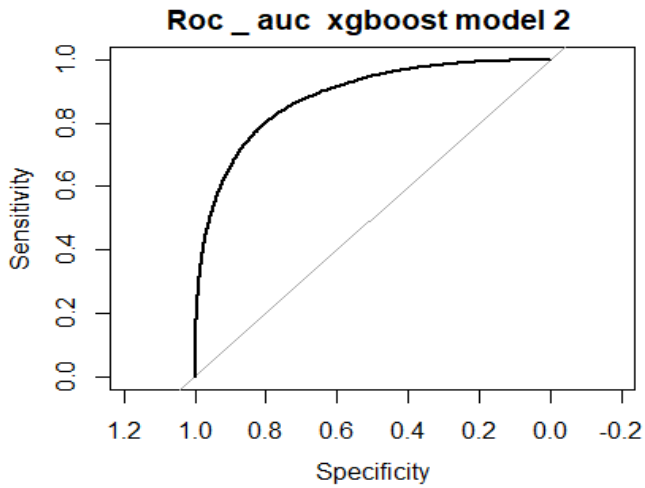
XGBOOST is the decision tree-based ensemble algorithm which provides gradient boosting framework.

MODEL 1 – STRATIFIED SAMPLING METHOD

```
########## MODEL 1 ###############
  # [1] "accuracy 0.81"
  # [1] "precision 0.32"
  # [1] "recall 0.18"
  # [1] "fpr 0.18"
  # [1] "fnr 0.21"
  # [1] "f1 0.23"
  # Area under the curve: 0.8845
```



MODEL 2 – OVER SAMPLING METHOD




```
#########  MODEL 2 ############
# [1] "accuracy 0.71"
# [1] "precision 0.24"
# [1] "recall 0.3"
# [1] "fpr 0.3"
# [1] "fnr 0.13"
# [1] "f1 0.27"
# Area under the curve: 0.8822
```

MODEL EVALUATION

Evaluation of model can be different for classification and regression problems.

For classification problems, the error metrics are

1. *ACCURACY*
2. *PRECISION*
3. *RECALL*
4. *FALSE NEGATIVE RATE*
5. *FALSE POSTIVE RATE*

For regression problems ,the error metrics are

1. *MSE(Mean Square Error)*
2. *RMSE(Root Mean Square Error)*
3. *MAPE(Mean Absolute Percentage Error)*

In our dataset, we are going to predict that whether the customer makes transaction or not. For that apart from accuracy our important error metrics must be false negative rate. Because if our model predicted potential customers wrongly means the client going to lose the customers.

I had mentioned all the predicted test cases in a tabluar format in both Python and R

By comparing all the models in the tabular format,we came with final decision with the help of error metrics.

EVALUATION OF MODEL USING OUR ALGORITHM				
ALGORITHM	EVALUATION METRICS	IN PYTHON	IN R SCRIPT	
			MODEL 1	MODEL 2
DECISION TREE	*ACCURACY*	0.5871	0.65	0.55
	PRECISION	0.58	0.14	0.14
	RECALL	0.58	0.33	0.47
	FALSE NEGATIVE RATE	41.625	0.51	0.33
	F1 SCORE	NA	0.2	0.21
	AREA UNDER THE CURVE	NA	0.5843	0.6124
LOGISTIC REGRESSION	*ACCURACY*	0.7698	0.91	0.78
	PRECISION	0.76	0.69	0.28
	RECALL	0.77	0.01	0.22
	FALSE NEGATIVE RATE	0.233	0.73	0.22
	F1 SCORE	NA	0.03	0.25
	AREA UNDER THE CURVE	0.86	0.86	0.8597
NAÏVE BAYES	*ACCURACY*	0.8015	0.92	0.81
	PRECISION	0.8	0.72	0.32
	RECALL	0.79	0.02	0.19
	FALSE NEGATIVE RATE	80.15	0.63	0.21
	F1 SCORE	NA	0.03	0.24
	AREA UNDER THE CURVE	0.89	0.8894	0.8893
EAREST NEIGHBOUR(K	*ACCURACY*	0.564	NA	NA
	PRECISION	0.18	NA	NA
	RECALL	0.17	NA	NA
	FALSE NEGATIVE RATE	83	NA	NA
	F1 SCORE	NA	NA	NA
	AREA UNDER THE CURVE	NA	NA	NA
RANDOM FOREST	*ACCURACY*	0.7594	NA	NA
	PRECISION	0.73	NA	NA
	RECALL	0.79	NA	NA
	FALSE NEGATIVE RATE	0.2055	NA	NA
	F1 SCORE	NA	NA	NA
	AREA UNDER THE CURVE	NA	NA	NA
SGB CLASSIFIER	*ACCURACY*	0.6031	NA	NA
	PRECISION	0.55	NA	NA
	RECALL	0.98	NA	NA
	FALSE NEGATIVE RATE	0.194	NA	NA
	F1 SCORE	NA	NA	NA
	AREA UNDER THE CURVE	NA	NA	NA
XG BOOST	*ACCURACY*	NA	0.81	0.71
	PRECISION	NA	0.32	0.24
	RECALL	NA	0.18	0.3
	FALSE NEGATIVE RATE	NA	0.21	0.13
	F1 SCORE	NA	0.23	0.27
	AREA UNDER THE CURVE	NA	0.8845	0.8822

We need to select the model with low FNR and high auc curve with acceptable accuracy.

IN PYTHON

I am selecting **LOGISTIC REGRESSION** algorithm because it has low FNR value and acceptable accuracy.

Error metrics values are mentioned below:

- Accuracy = 76.98
- FNR = 23.03
- Recall = 0.77
- precision = 0.76
- AUC curve = 0.86

IN R SCRIPT

I am selecting **NAIVE BAYES** algorithm because it has low FNR value, high AUC value, high f1 score and acceptable accuracy.

Error metrics values are mentioned below:

- Accuracy = 0.81
- Precision = 0.32
- Recall = 0.18
- FNR = 0.21
- F1 = 0.23
- AUC curve = 0.88

REFERENCES:

- Edwisor (learning.edwisor.com)
- Kaggle (Kaggle.com)
- GitHub(github.com)
- Google (rbloggers, towardsdatascience, mediam and Wikipedia