**SANTANDER CUSTOMER TRANSACTION PREDICTION**

**-SAFA MULANI**

**12/07/2020**

Contents

1 Introduction

1.1 Problem Statement

1.2 Data Understanding

2 DATA PRE-PROCESSING

2.1 Missing Value Analysis

2.1.1 Outlier Analysis

2.1.2 Data Visualization

2.1.2 Feature Selection

2.1.2 Feature Scaling

2.2 Modelling

2.2.1 Logistic Regression

2.2.2 Naïve Bayes Algorithm

2.2.3 Decision Tree

2.2.4 XGBOOST model

2.2.5 CATBOOST model

3 Conclusion

3.1 Model Evaluation

3.1.1 Confusion Matrix

3.1.1.1 Accuracy

3.1.1.2 Precision

3.1.1.3 Recall

3.1.1.4 f1 score

3.1.2 ROC – AUC score

3.1.3 Hyper-Parameter Tuning

3.2 Model Selection

Appendix A – Python Code

References

**Chapter 1: Introduction**

* 1. **Problem Statement**

The objective of this problem statement is to predict which customers would make a particular/specific transaction in the future course of time.

By the end of this project, we shall be able to predict the customers that would perform transactions in future and this would hence help us enhance our business-model in a more precise manner.

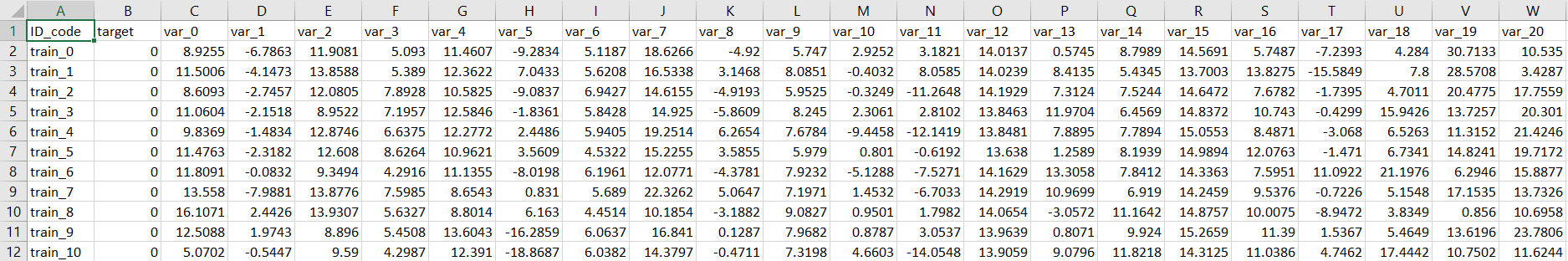
Now, let us understand the historical data i.e. the dataset being used to solve this problem statement.

* 1. **DATA Understanding (At a Glance!)**

The data set comprises of Training and Testing data set as mentioned below –-

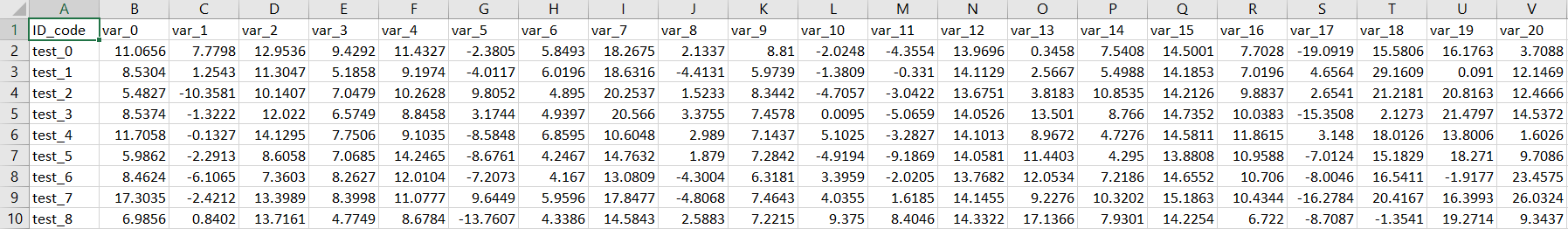
* Training data set contains 2,00,000 data rows and 202 columns [ 200 numeric variables, 1 binary target column, and a string ID\_code column.]

Given below is the dataset (rows 1-10), (columns 1-20) which we will be using to predict the customers for future transaction--



* Testing data set contains 2,00,000 data rows and 201 columns [ 200 numeric variables and a string ID\_code column.]

Given below is the dataset (rows 1-10), (columns 1-20) which we will be using to test the trained data for the customers for future transaction—



**Chapter 2 Methodology**

**2.1** **Pre-Processing**

In the world of predictive analysis and model building, the first and foremost step is the ANALYSIS of DATA. By analysis, we mean to explore the data at the raw level i.e. before transforming and modelling the data.

**2.1.1 EDA**

In the initial stages of pre-processing, **Exploratory Data Analysis (EDA)** plays a very important role in understanding the flow of the dataset provided.

So, for this problem statement, too, we have conducted EDA -- Exploratory Data Analysis as follows:

1. **From the dataset, we analysed the distribution of the data.**
2. **Further, we analysed the data type of every data column in the dataset. Moreover, we found out that all the independent data columns were continuous in nature.**
3. **From the data, we found out that the data set has nearly same values for mean and median. Moreover, the data happens to have large standard deviation.**
4. **The same trends and statistical analysis was found in the training as well as testing data sets.**

Having understood this, we do not feel the need to convert the datatypes of the variables.

From the data columns, we have found that the column ID\_code does not contribute at all to the data prediction and decide to drop those data columns.

# **Insights from the above EDA--**

#### The independent variable ‘ID\_code’ and the dependent variable ‘target’ happen to have NO relationship between them. Thus, we can drop the data column ‘ID\_code’ from the dataset.

**2.1.1 MISSING VALUE ANALYSIS**

After having explored the data, the next step is to detect and operate on the missing values in the dataset.

Presence of missing values leads to erroneous outcome of the predictions. Thus, it is very important to detect and treat the missing values.

From the dataset, we have understood that **the training as well as testing data set contains NO missing values**.

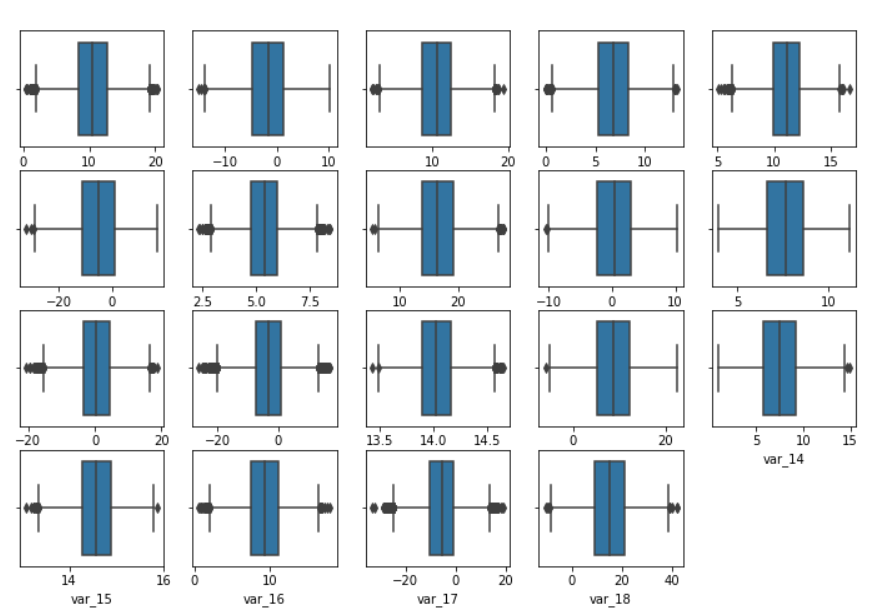
**2.1.2 OUTLIER ANALYSIS**

Outliers in the data are actually the data points that do not fall under the standard distribution of data and thus create a negative impact on the distribution of data as well as modelling. Thus, it makes us to mandate the removal of outliers.

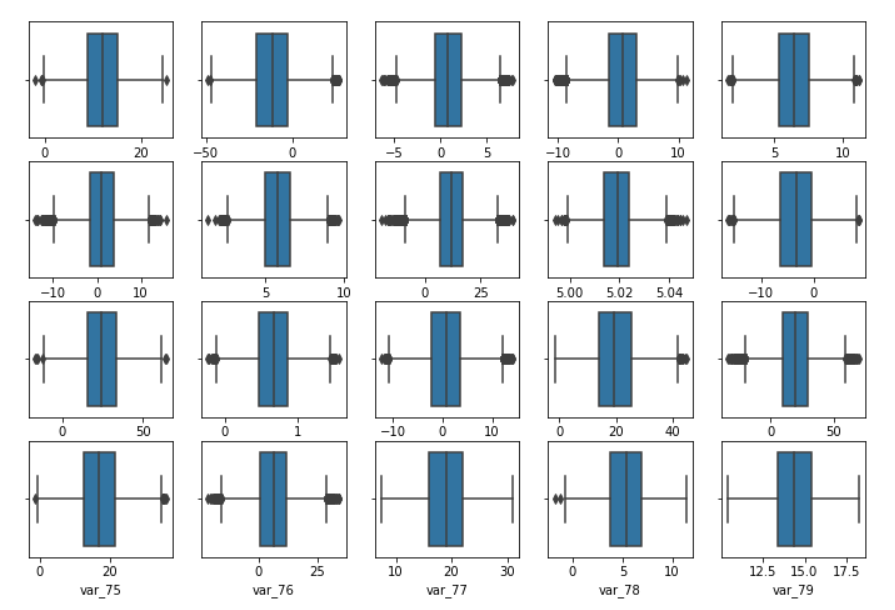
DETECTION of OUTLIERS:

We use Boxplot to detect the presence of outliers in the numeric/continuous data columns –

The data values present above the upper fence and below the lower fence of each boxplot can be considered as the outlier data values for that particular data column.



**Boxplot of variable 0 – 18**

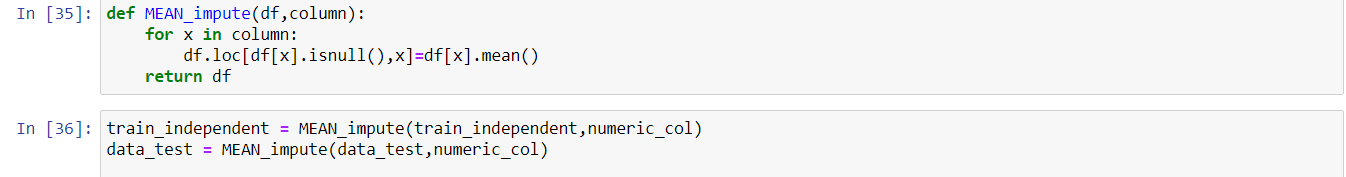


**Boxplot of variable 60 – 79**

From the above boxplot visualization, it is clear that the training as well as testing dataset contains outliers.

REMOVAL of OUTLIERS:

Now, we will replace the detected outlier values with NULL. Further, we will impute the NULL values with the MEAN of the data columns.

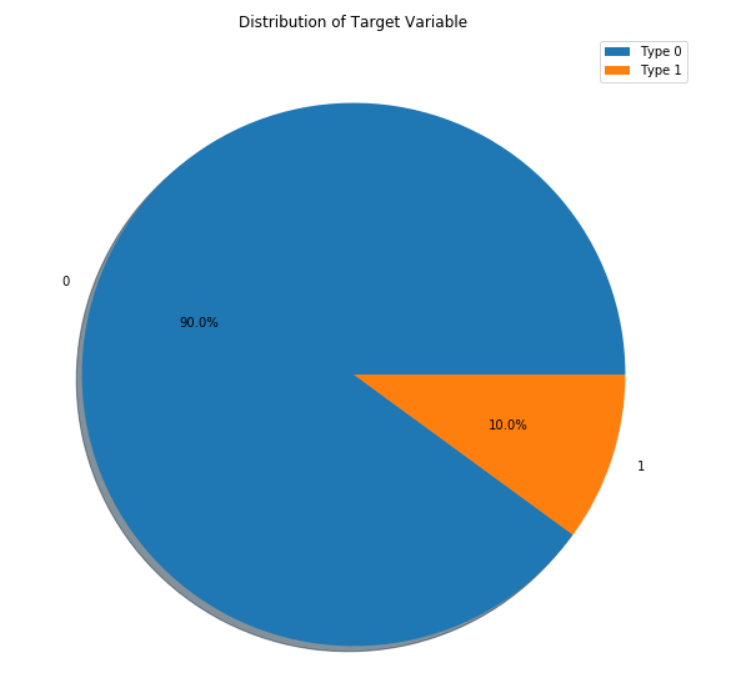


**2.1.3 DATA VISUALIZATION**

Data visualization helps us understand the effect, importance and contribution of each independent variable towards the dependent variable.

# **Visualizing the Target variable of the dataset—**

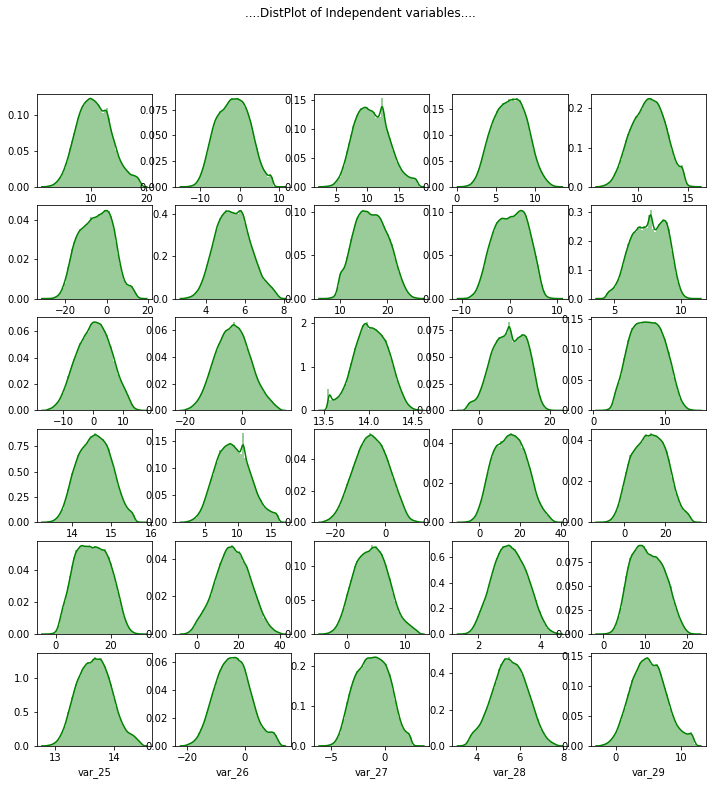
We have used **PIE PLOT** to server the purpose.



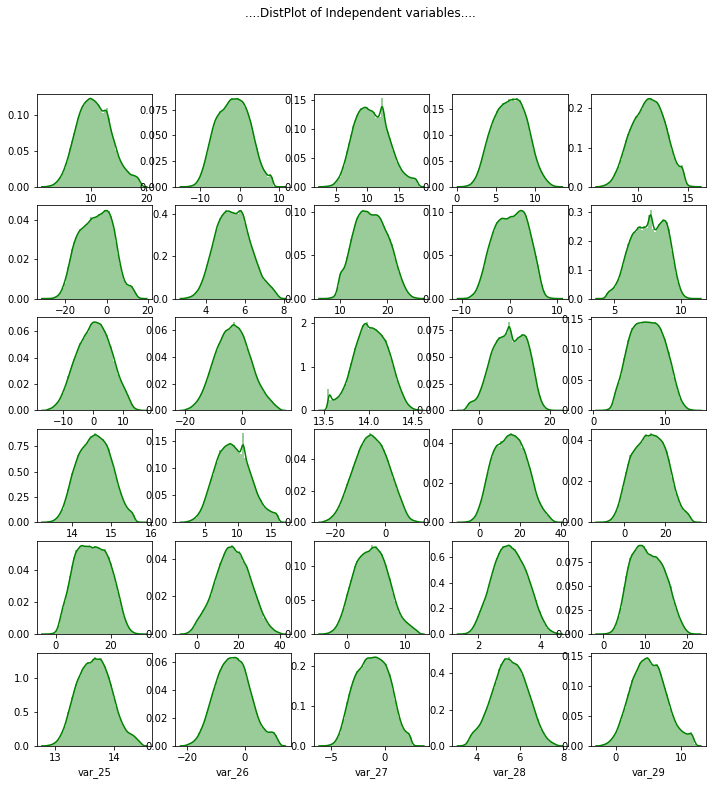
**PIE plot of the target variable**

1. **Visualizing the independent variables of the training dataset--**

We have made the use of Distribution Density Plots to visualize the continuous independent variables.



1. **Visualizing the variables of the testing dataset**—



From the above data visualization, the following insights can be drawn from the data—

* The training as well as the testing data set is NORMALLY distributed.
* The data distribution in the testing as well as training data set is similar.
* The target variable contains data in an imbalanced form i.e. 90% of negative outcome (0) and 10% of positive outcome (1).
* Thus, we can say that the problem statement contains imbalanced data for prediction.
  + 1. **FEATURE SELECTION**

Feature/Variable selection plays a very crucial role in model building and prediction of data values.

For our problem statement, we have chosen the following techniques to select more important variables that have a huge impact on the dependent variable—

1. **CORRELATION ANALYSIS** – Numeric variable selection technique

The correlation analysis helps us figure out the correlation between the variables within a range of – 1 to +1.

**The independent variables must be least correlated and there must be a high correlation between the dependent and the independent variables.**

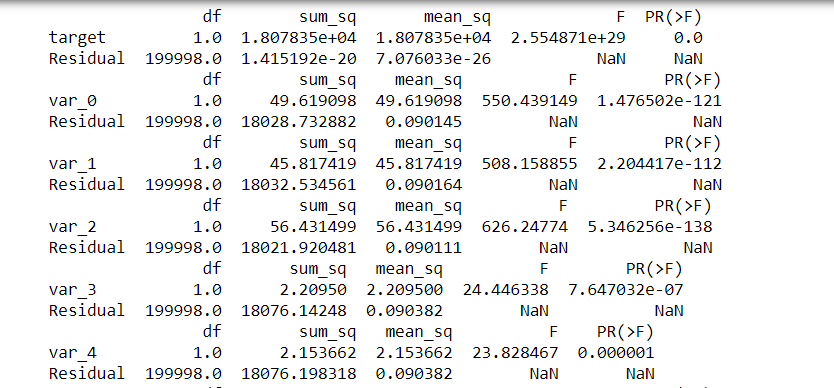
#### From the correlation analysis, we can say that the independent variables are not at all correlated to each other. That is, every variable serves some different information about the prediction to be performed.

Moreover, we have performed VIF test to check for the presence of multi-collinearity amongst the independent variables. As a result, all the data columns were found to have a VIF = 1.

Thus, we can now assume that the data variables are not correlated and do not possess multi-collinearity.

**ANOVA test** – Check the mean and variance distribution between the independent variables.

We performed ANOVA test, to understand the relationship between the mean values of the independent data input.



And, we got to know that all the data variables depict different information and no groups represented the same mean. Thus, we reject the NULL hypothesis and assume that every variable contributes to the prediction of target value.

* + 1. **FEATURE SCALING**

Feature Scaling is essential to convert the numeric data values to a standard and scalable format and make the data values completely scale free.

In our dataset, from the data visualization it is pretty clear that the training as well as testing dataset has normal distribution of data.

Thus, we decide to apply the process of STANDARDIZATION to scale the data variables of testing as well as training dataset.

**STANDARDIZATION—**

This method is applicable only to normalized data.

It converts the numeric data values to have values between the range of 0 to infinity.

Formula:

Z-score = (X – mean)/standard deviation

**CHAPTER 3: MODELLING**

From the Data Pre-processing performed on the training as well as testing dataset, we draw the following conclusions –

1. The dependent variable of the dataset is ‘target’, which is a binary categorical data variable [0,1].

In Data Modelling, we need to identify the type of Problem statement.

There are 4 types of Problem statement—

1. Predictive: The dependent variable needs to be of type continuous.
2. Classification: The dependent variable needs to be of type categorical.
3. Optimization
4. Unsupervised Learning

Thus, we conclude that **Santander Customer Transaction Prediction is a Classification Problem**.

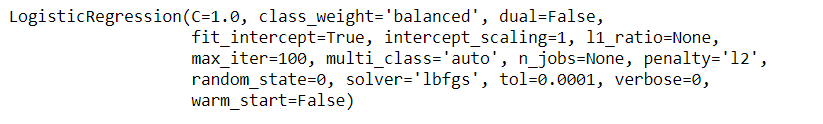
We will be using the following Machine Learning Algorithms to analyse and build our model:

* LOGISTIC REGRESSION
* DECISION TREE
* NAIVE BAYES
* XGBOOST MODEL
* CATBOOST MODEL
  1. **LOGISTIC REGRESSION**

The logistic regression model explains the relationship between the categorical dependent variable and the independent variables (continuous or categorical).

We have applied Binary logistic regression to find the best fit prediction values for the model.

The following values of the parameters were formulated using Logistic Regression—



We have specified class\_weight as balanced so that it will handle the imbalanced data.

Metrics derived from the model—

**Precision value of the model: 0.28493852272933656**

**Accuracy of the model: 0.781095**

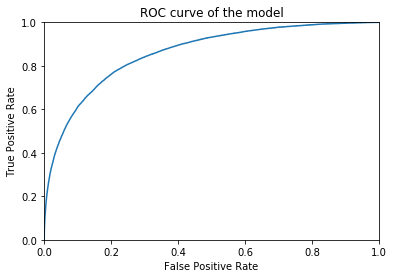
**Recall value of the model: 0.7806249378047567**

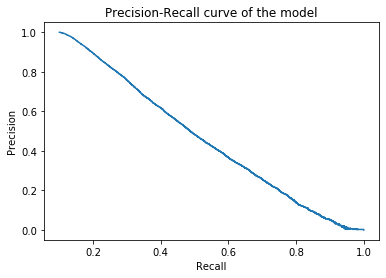
**Specificity of the model: 0.7811475136463185**

**False Positive rate of the model: 0.21885248635368146**

**False Negative rate of the model: 0.2193750621952433**

**f1 score of the model: 0.41748825822589447**

****

****

* 1. **NAÏVE BAYES**

Further, we have applied **NAÏVE BAYES** to build our model for classification. Naïve Bayes works on the probability theorem to find the best fit values for the target variable.

Metrics derived from the model—

**Precision value of the model: 0.7181307305530908**

**Accuracy of the model: 0.921635**

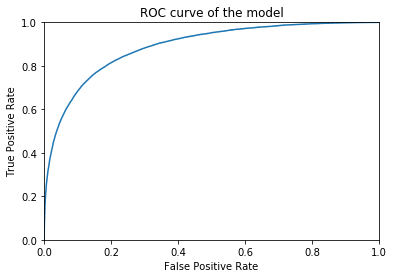
**Recall value of the model: 0.36242412180316447**

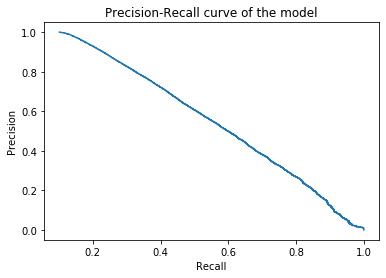
**Specificity of the model: 0.9841080143633756**

**False Positive rate of the model: 0.015891985636624386**

**False Negative rate of the model: 0.6375758781968355**

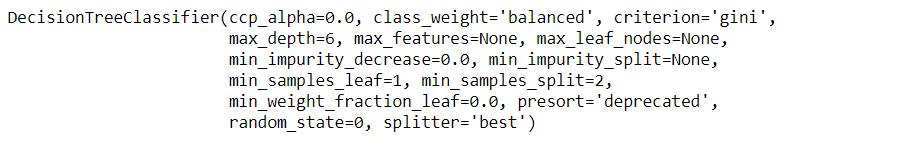
**f1 score of the model: 0.48173010151780693**





* 1. **DECISION TREE**

In order to work even more with the parameters, we applied DECISION Tree model that uses the bagging technique to select the variables and tune up the parameters and the following was observed—



Metrics derived from the model—

**Precision value of the model: 0.1751485791339959**

**Accuracy of the model: 0.704375**

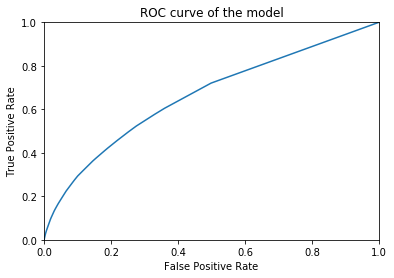
**Recall value of the model: 0.5234849238730221**

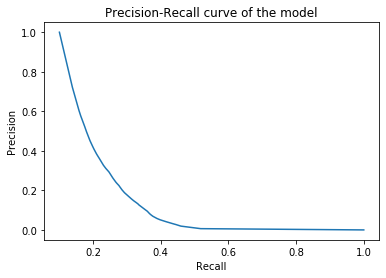
**Specificity of the model: 0.7245833842870008**

**False Positive rate of the model: 0.2754166157129993**

**False Negative rate of the model: 0.4765150761269778**

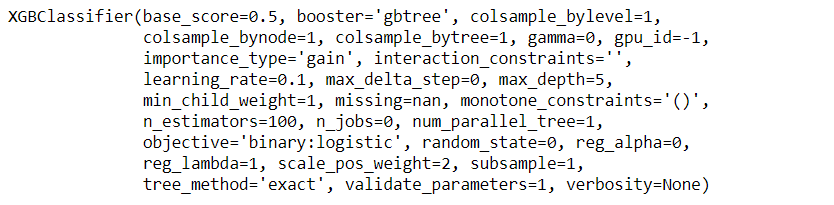
**f1 score of the model: 0.2624770790973842**





* 1. **XGBOOST model**

XGBOOST model is actually the extended gradient boosted model that works efficiently on complex datasets and provides results in a fast manner with boosting.



Metrics derived from the model—

**Precision value of the model: 0.912396990326048**

**Accuracy of the model: 0.92253**

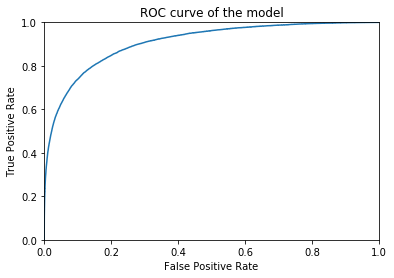
**Recall value of the model: 0.253408299333267**

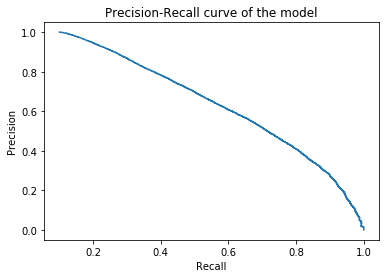
**Specificity of the model: 0.9972818534535469**

**False Positive rate of the model: 0.002718146546453069**

**False Negative rate of the model: 0.746591700666733**

**f1 score of the model: 0.3966510903426791**

****



* 1. **CATBOOST model**

CATBOOST models have been applied to increase the efficiency of the model. CATBOOST model works well and executes in a very less time as compared to XGBOOST model.

Metrics derived from the model—

**Precision value of the model: 0.8983860342555995**

**Accuracy of the model: 0.94789**

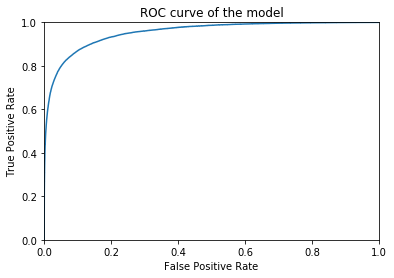
**Recall value of the model: 0.542840083590407**

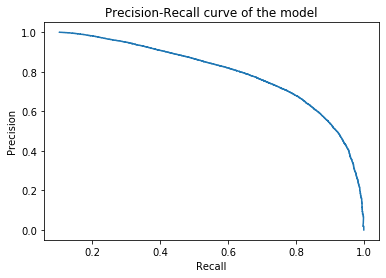
**Specificity of the model: 0.9931407099420796**

**False Positive rate of the model: 0.006859290057920423**

**False Negative rate of the model: 0.45715991640959297**

**f1 score of the model: 0.6767570249984491**





**CHAPTER 3: CONCLUSION**

Perhaps, after building the model, it is very important for us to analyse the accuracy rate of every model through various metrics and choose the best fit model.

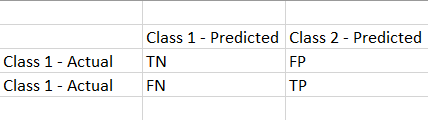
* 1. **MODEL EVALUATION**

Depending upon the type of model, there are various metrics to judge the performance of a model.

We need to evaluate the model based on the accuracy level and the evaluation metrics.

* + 1. **CONFUSION MATRIX**

It represents the summary of the predictions performed by the classification models.



* TRUE NEGATIVE (TN): The values which are actually negative and also predicted as negative.
* FALSE NEGATIVE (FN): The values which are actually positive but predicted as negative.
* FALSE POSITIVE (FP): The values which are actually negative but predicted as positive.
* TRUE POSITIVE (TP): The values which are actually positive and predicted as positive too.

In this classification problem statement of imbalanced data, we are more interested to build and choose such as model that would have a comparatively lower FP and FN values.

* + - 1. **ACCURACY of the Model**

Formula—

Accuracy = TP + TN / (TP + TN + FP + FN)

* + - 1. **Precision score**

By precision, we mean to understand that the positive values are indeed predicted as positive.

Formula—

Precision = TP / (TP + FP)

* + - 1. **Recall score**

By recall, we mean to understand that the particular class of samples is correctly predicted.

Formula—

Recall = TP / (TP +FN)

* + - 1. **F1 score**

F1 score helps us rate the accuracy and efficiency of the model when the data is imbalanced. It is actually the harmonic mean of Precision and Recall scores.

Formula—

F1 = 2\*(Recall \* Precision) / (Recall + Precision)

* + 1. **ROC-AUC score**

ROC curve represents the performance measurement of the model. The AUC score actually represents the capability of the model to differentiate and predict the different classes or categories.

* + 1. **Hyper-parameter tuning of the models**

Hyper parameters are those parameters that are provided at the time of model building. By performing tuning of hyper parameters, we can improve the accuracy and optimize the model even more.

1. Hyper parameter tuning on Logistic Regression--

**Precision value of the model: 0.28594968691640**

**Accuracy of the model: 0.7826**

**Recall value of the model: 0.7770922479848741**

**Specificity of the model: 0.7832153061111049**

**False Positive rate of the model: 0.2167846938**

**False Negative rate of the model: 0.2229077520**

**f1 score of the model: 0.4180630654745972**

**AUC value of the model: 0.86**

1. Hyper parameter tuning on Naïve Bayes--

**Precision value of the model: 0.71837379119794**

**Accuracy of the model: 0.92164**

**Recall value of the model: 0.36222509702457956**

**Specificity of the model: 0.9841358072728486**

**False Positive rate of the model: 0.0158641927**

**False Negative rate of the model: 0.6377749029**

**f1 score of the model: 0.4816088912410691**

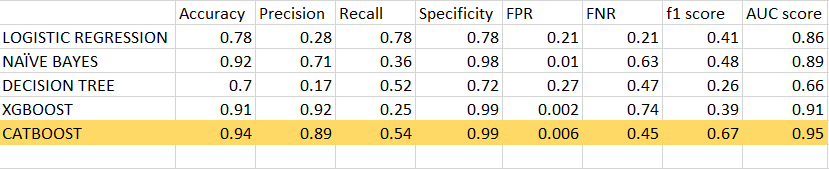
**AUC value of the model: 0.89**

* + 1. **MODEL SELECTION**

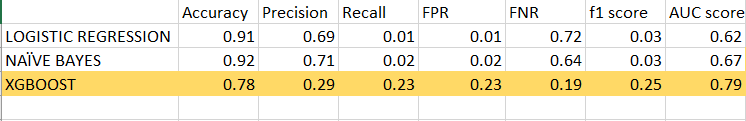
We will select a model fulfilling the below conditions:

* High f1 score
* Low False positive rate
* Low False negative rate
* High AUC score
* High level of Precision and Recall
* High Accuracy

Model Evaluation – PYTHON



Model Evaluation – R



Thus, we select CATBOOST model in Python and XGBOOST model in R as the best fit models for this classification problem.

**APPENDIX A – Python code**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from fancyimpute import KNN

import seaborn as sn

import os

import statsmodels.api as sm

from statsmodels.formula.api import ols

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import roc\_curve,auc

from sklearn.metrics import roc\_auc\_score,precision\_recall\_curve

from sklearn.linear\_model import LogisticRegression

from sklearn.naive\_bayes import GaussianNB

from sklearn.tree import DecisionTreeClassifier

from xgboost import XGBClassifier

from sklearn.model\_selection import RandomizedSearchCV

from catboost import CatBoostClassifier

#Changing the working directory to the specified path as mentioned below--

os.chdir("D:/Edwisor\_Project - Santander Prediction") os.getcwd()

data\_train = pd.read\_csv("train.csv") # training dataset

data\_train.shape

data\_train.describe()

data\_train.dtypes

data\_test = pd.read\_csv("test.csv") # testing dataset

data\_test.describe()

data\_train = data\_train.drop(['ID\_code'],axis=1)

data\_test = data\_test.drop(['ID\_code'],axis=1)

print(data\_train.isnull().sum())

print(data\_train.isnull().sum())

numeric\_col = [x for x in data\_train.columns.values if x!='target']

train\_independent = data\_train[numeric\_col]

train\_dependent = data\_train['target']

def plot\_box(data,begin,end):

fig = plt.figure(figsize=(12,12))

if(begin >=0 and end<len(numeric\_col)):

for i,z in enumerate(numeric\_col[begin:end],1):

ax = fig.add\_subplot(6,5,i)

sn.boxplot(x=train\_independent.loc[:,z],ax=ax)

plt.suptitle("....Outlier Analysis using boxplot....")

plt.show()

plot\_box(train\_independent ,0,19)

def replace\_outlier(df ,numerci\_col): # replacing the outlier values with NULL

for x in numeric\_col:

q75,q25 = np.percentile(df.loc[:,x],[75,25])

intr\_qr = q75-q25

max = q75+(1.5\*intr\_qr)

min = q25-(1.5\*intr\_qr)

df.loc[df[x] < min,x] = np.nan

df.loc[df[x] > max,x] = np.nan

return df

train\_independent = replace\_outlier(train\_independent,numeric\_col)

data\_test = replace\_outlier(data\_test,numeric\_col)

def MEAN\_impute(df,column):

for x in column:

df.loc[df[x].isnull(),x]=df[x].mean()

return df

train\_independent = MEAN\_impute(train\_independent,numeric\_col)

data\_test = MEAN\_impute(data\_test,numeric\_col)

## Data Distribution

plt.figure(figsize=(10,10))

plt.pie(train\_dependent.value\_counts() ,labels=[0,1] ,autopct='%1.1f%%',shadow=True)

plt.title("Distribution of Target Variable")

plt.legend(['Type 0','Type 1'])

plt.show()

def plot\_dist(data,begin,end):

fig = plt.figure(figsize=(12,12))

if(begin >=0 and end<len(numeric\_col)):

for i,z in enumerate(numeric\_col[begin:end],1):

ax = fig.add\_subplot(6,5,i)

sn.distplot(train\_independent.loc[:,z],color='green',ax=ax)

plt.suptitle("....DistPlot of Independent variables....")

plt.show()

plot\_dist(train\_independent,0,30)

plot\_dist(train\_independent,30,60)

plot\_dist(train\_independent,60,90)

plot\_dist(train\_independent,90,120)

plot\_dist(train\_independent,120,150)

plot\_dist(train\_independent,150,180)

plot\_dist(data\_test,0,30)

plot\_dist(data\_test,30,60)

plot\_dist(data\_test,60,90)

plot\_dist(data\_test,90,120)

plot\_dist(data\_test,120,150)

plot\_dist(data\_test,150,180)

#Using Correlation analysis to depict the relationship between the numeric/continuous data variables

corr\_matrix = train\_independent.loc[:,numeric\_col].corr()

print(corr\_matrix)

for x in col:

model = ols('target' + '~' + x, data = data\_train).fit() #Oridnary least square method

result\_anova = sm.stats.anova\_lm(model) # ANOVA Test

print(result\_anova)

def scale\_data (data):

obj =StandardScaler()

res\_data = pd.DataFrame(obj.fit\_transform(data) ,columns=data.columns)

return res\_data

train\_independent = scale\_data(train\_independent)

data\_test = scale\_data(data\_test)

# Defining Error metrics

def error\_metric(CM):

TN = CM.iloc[0,0]

FN = CM.iloc[1,0]

TP = CM.iloc[1,1]

FP = CM.iloc[0,1]

precision =(TP)/(TP+FP)

accuracy\_model =(TP+TN)/(TP+TN+FP+FN)

recall\_score =(TP)/(TP+FN)

specificity\_value =(TN)/(TN + FP)

False\_positive\_rate =(FP)/(FP+TN)

False\_negative\_rate =(FN)/(FN+TP)

f1\_score =2\*(( precision \* recall\_score)/( precision + recall\_score))

print("Precision value of the model: ",precision)

print("Accuracy of the model: ",accuracy\_model)

print("Recall value of the model: ",recall\_score)

print("Specificity of the model: ",specificity\_value)

print("False Positive rate of the model: ",False\_positive\_rate)

print("False Negative rate of the model: ",False\_negative\_rate)

print("f1 score of the model: ",f1\_score)

def ROC\_curve(y\_act ,y\_pred) :

FPR,TPR,thresholds=roc\_curve(y\_act ,y\_pred)

plt.plot(FPR,TPR)

plt.title('ROC curve of the model')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.0])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.show()

AUC = auc(FPR,TPR)

print("AUC value of the model: %.2f"%AUC)

def Precision\_recall\_curve(y\_act ,y\_pred) :

recall,precision,thresholds=precision\_recall\_curve(y\_act ,y\_pred)

plt.plot(recall,precision)

plt.title('Precision-Recall curve of the model')

plt.xlabel('Recall')

plt.ylabel('Precision')

plt.show()

# Logistic Regression

logistic\_model = LogisticRegression(class\_weight='balanced' , random\_state=0).fit(train\_independent,train\_dependent)

target\_predict = logistic\_model.predict(train\_independent)

targetclass\_prob = logistic\_model.predict\_proba(train\_independent)[:, 1]

confusion\_matrix = pd.crosstab(train\_dependent,target\_predict)

error\_metric(confusion\_matrix)

ROC\_curve(train\_dependent,targetclass\_prob)

Precision\_recall\_curve(train\_dependent,targetclass\_prob)

# Naive Baiyes Algorithm

Naive\_model = GaussianNB().fit(train\_independent,train\_dependent)

target\_predict = Naive\_model.predict(train\_independent)

targetclass\_prob = Naive\_model.predict\_proba(train\_independent)[:, 1]

confusion\_matrix = pd.crosstab(train\_dependent,target\_predict)

error\_metric(confusion\_matrix)

ROC\_curve(train\_dependent,targetclass\_prob)

Precision\_recall\_curve(train\_dependent,targetclass\_prob)

#Decision Trees

decision\_model = DecisionTreeClassifier(max\_depth= 6,class\_weight='balanced' ,random\_state =0).fit(train\_independent,train\_dependent)

target\_predict = decision\_model.predict(train\_independent)

targetclass\_prob = decision\_model.predict\_proba(train\_independent)[:, 1]

confusion\_matrix = pd.crosstab(train\_dependent,target\_predict)

error\_metric(confusion\_matrix)

ROC\_curve(train\_dependent,targetclass\_prob)

Precision\_recall\_curve(train\_dependent,targetclass\_prob)

xgb\_model = XGBClassifier(max\_depth=5,scale\_pos\_weight=2,learning\_rate=0.1)

xgb\_model.fit(train\_independent,train\_dependent)

target\_predict = xgb\_model.predict(train\_independent)

targetclass\_prob = xgb\_model.predict\_proba(train\_independent)[:, 1]

confusion\_matrix = pd.crosstab(train\_dependent,target\_predict)

error\_metric(confusion\_matrix)

ROC\_curve(train\_dependent,targetclass\_prob)

Precision\_recall\_curve(train\_dependent,targetclass\_prob)

# Hyper parameter tuning of Logistic Regression

logit\_HP = LogisticRegression(class\_weight='balanced' , random\_state=0 ,penalty= 'l2')

max\_iter=[5,10,15,20,25]

C=np.logspace(-5, 3, 20)

# Creating a random grid of Hyper Parameters

logit\_GRID = {'max\_iter': max\_iter,'C':C}

# Building a RandomSearch Cross Validated model over Logistic Regression

logit\_CV = RandomizedSearchCV(logit\_HP, param\_distributions = logit\_GRID, n\_iter = 10, cv = 10, random\_state=0,scoring='roc\_auc').fit(train\_independent,train\_dependent)

# Best Parameters of the LOGSITIC REGRESSION Model

Best\_Logit\_param = logit\_CV.best\_params\_

print("BEST PARAMETERS OF LOGISTIC REGRESSION MODEL:",Best\_Logit\_param)

Best\_Logit\_Estimator = logit\_CV.best\_estimator\_

print("BEST ESTIMATORS OF LOGISTIC REGRESSION MODEL:",Best\_Logit\_Estimator)

# Prediction on the test dataset

Logit\_predict = Best\_Logit\_Estimator.predict(train\_independent)

targetclass\_prob = Best\_Logit\_Estimator.predict\_proba(train\_independent)[:, 1]

confusion\_matrix = pd.crosstab(train\_dependent,Logit\_predict)

error\_metric(confusion\_matrix)

ROC\_curve(train\_dependent,targetclass\_prob)

Precision\_recall\_curve(train\_dependent,targetclass\_prob)

#Hyper Parameter tuning on Naive Bayes

Naive\_HP = GaussianNB()

var\_smooth =np.logspace(-4,-3, num=20)

Naive\_GRID = {'var\_smoothing': var\_smooth}

# Building a RandomSearch Cross Validated model over Logistic Regression

Naive\_CV = RandomizedSearchCV(Naive\_HP, param\_distributions = Naive\_GRID, n\_iter = 10, cv = 10, random\_state=0,scoring='roc\_auc').fit(train\_independent,train\_dependent)

# Best Parameters of the LOGSITIC REGRESSION Model

Best\_Naive\_param = Naive\_CV.best\_params\_

print("BEST PARAMETERS OF NAIVE BAYES MODEL:",Best\_Naive\_param)

Best\_Naive\_Estimator = Naive\_CV.best\_estimator\_

print("BEST ESTIMATORS OF NAIVE BAYES MODEL:",Best\_Naive\_Estimator)

Naive\_predict = Best\_Naive\_Estimator.predict(train\_independent)

targetclass\_prob = Best\_Naive\_Estimator.predict\_proba(train\_independent)[:, 1]

confusion\_matrix = pd.crosstab(train\_dependent,Naive\_predict)

error\_metric(confusion\_matrix)

ROC\_curve(train\_dependent,targetclass\_prob)

Precision\_recall\_curve(train\_dependent,targetclass\_prob)

cat\_model = CatBoostClassifier(

depth=4,

custom\_loss=['AUC'],

learning\_rate=0.3,

verbose=50,

iterations=None,

od\_type='Iter',

early\_stopping\_rounds=10

).fit(train\_independent,train\_dependent)

target\_predict = cat\_model.predict(train\_independent)

targetclass\_prob = cat\_model.predict\_proba(train\_independent)[:, 1]

confusion\_matrix = pd.crosstab(train\_dependent,target\_predict)

error\_metric(confusion\_matrix)

ROC\_curve(train\_dependent,targetclass\_prob)

Precision\_recall\_curve(train\_dependent,targetclass\_prob)

predict=cat\_model.predict(data\_test)

##Create a dataframe for actual values and predicted values

Result = pd.DataFrame({'Actual\_target': train\_dependent, 'Predicted\_target':predict})

Result.to\_csv("CUSTOMER\_PREDICTION\_PYTHON.csv",index=False)

**REFERENCES**

* StackOverflow
* www.towardsdatascience.com
* www.medium.com

Instructions to execute the Python code—

Load the code (jupyter notebook file) in the jupyter notebook environment and execute the code.

Instructions to execute the R code—

Load the code in the R studio environment and execute the code.

Understanding of how this project will help attain strategic goals—

By this project, we can help the companies plan for future business proposal and offers in the near time.

As this is irrespective of the amount of transaction, the positive customers can be provided with better offers to improve the business of the model.