**Santander Customer Transaction Prediction**

*Snehal somvanshi*

*10th July 2020*

Contents

[Chapter 1 4](#_Toc46746465)

[Introduction 4](#_Toc46746466)

[1.1 Problem statement 4](#_Toc46746467)

[1.2 Data 4](#_Toc46746468)

[1.3 Software Used: 4](#_Toc46746469)

[Chapter2 5](#_Toc46746470)

[Methodology 5](#_Toc46746471)

[2.1 Pre Processing: 5](#_Toc46746472)

[2.2 Exploratory Data Analysis: 5](#_Toc46746473)

[Chapter 3 7](#_Toc46746474)

[3.1 Missing Value Analysis 7](#_Toc46746475)

[3.2 Outlier Analysis 7](#_Toc46746476)

[Chapter 4 8](#_Toc46746477)

[4.1 Feature Selection 8](#_Toc46746478)

[4.2 Feature Engineering: 8](#_Toc46746479)

[Chapter 5 9](#_Toc46746480)

[Modelling 9](#_Toc46746481)

[5.1 Model Selection: 9](#_Toc46746482)

[5.2 Logistic regression: 10](#_Toc46746483)

[5.3 SMOTE(Synthetic Minority Oversampling Technique) 15](#_Toc46746485)

[5.4 LightGBM 16](#_Toc46746486)

[Chapter 6 17](#_Toc46746487)

[Conclusion: 17](#_Toc46746488)

[Chapter 7 17](#_Toc46746489)

[Code: 17](#_Toc46746490)

[7.1 Python Code: 17](#_Toc46746491)

[7.2 R-Code 17](#_Toc46746492)

# Chapter 1

# Introduction

## 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.

**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 whichproducts 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 accuratelyidentify new ways to solve our most common challenge, binary classification problemssuch as: is a customer satisfied? Will a customer buy this product? Can a customer paythis loan?

## Data

Train & Test dataset

Number of attributes:

You are provided with an anonymized dataset containing numeric feature variables, the

binary target column, and a string ID\_code column. The task is to predict the value

of target column in the test set.

## 1.3 Software Used:

1. Anaconda 3
2. Jupyter Notebook
3. R
4. RStudio

# Chapter2

# Methodology

## 2.1 Pre Processing:

Before running a model we need to understand the data. So data pre-processing contain data mining, Rename the column name as per understanding, Typecasting the numerical and category variable.

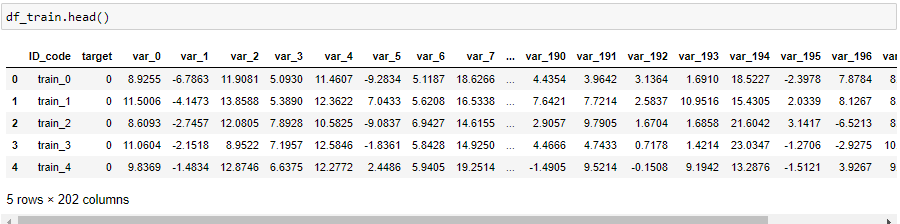
In this project, our task is to build classification models which would be used to predict whic

customers will make a specific transaction in the future. Given below is a sample of the

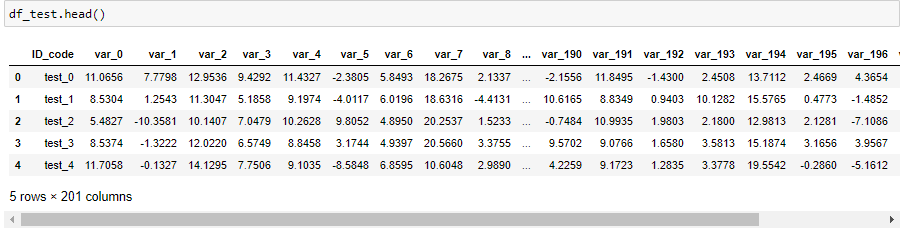
Santander customer transaction dataset:

Cleaning the data and s visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis. To start this process, we will first try and look at Variable details.

Train Dataset:



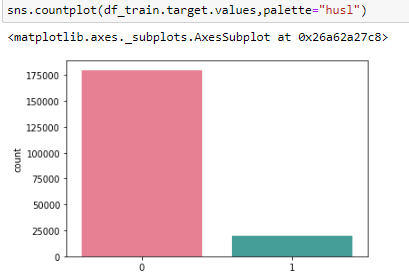
Test Dataset:



## 2.2 Exploratory Data Analysis:

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations. It is a good practice to understand the data first and try to gather as many insights from it. EDA is all about making sense of data in hand, before getting them dirty with it.

Target classes count:



**Observation from plot:**

 We have an unbalanced data, where 90% of the data is the number of customers those

will not make a transaction and 10% of the data is those who will make a transaction.

 Look at the violin plots seems that there is no relationship between the targets with the

index of the train data frame. This is more dominated by the zero targets then for the

ones.

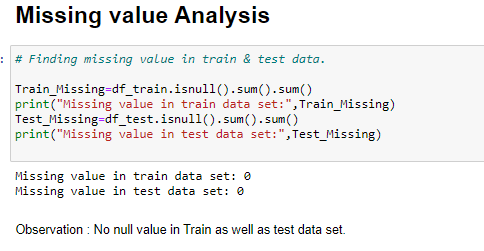
 Look at the jitter plots with violin plots. We can observe that targets look uniformly

distributed over the index of the data frame.

# Chapter 3

## 3.1 Missing Value Analysis

Find out the missing values in data. If missing value present in dataset then replace with mean/median value /KNN or drop this value according to problem statement.



## 3.2 Outlier Analysis

In this project, we haven’t performed outlier analysis due to the data is imbalanced and also

not required for imbalanced data.

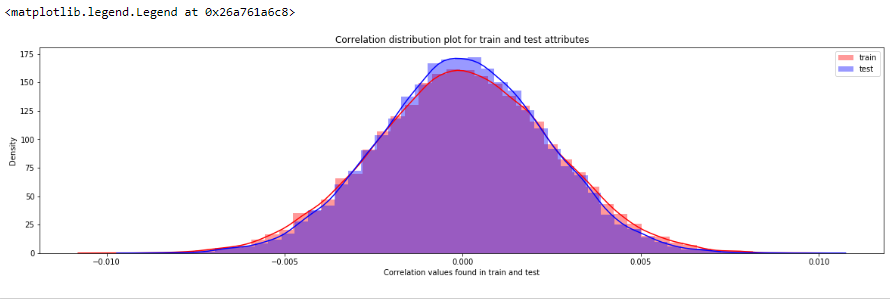
# Chapter 4

## 4.1 Feature Selection

Feature selection is very important for modelling the dataset. Every dataset has good and

unwanted features. The unwanted features would affect on performance of model, so we have to delete those features. We have to select best features by using ANOVA, Chi-Square test and correlation matrix statistical techniques and so on. In this, we are selecting best features by using Correlation matrix.

**Correlation matrix:**

Correlation matrix tells us about linear relationship between attributes and help us to build better models.From correlation distribution plot, we can observe that correlation between both train and test attributes are very small. It means that all both train and test attributes are independent to each other. .

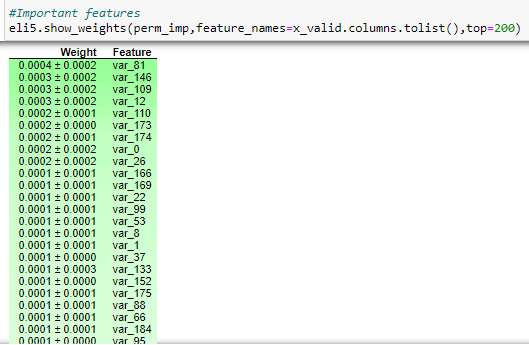
## 4.2 Feature Engineering:

Let us do some feature engineering by using

 Permutation importance

**Permutation importance:** Permutation variable importance measure in a random forest for classification and regression. The variables which are mostly contributed to predict the model.

feature importance can be measured by looking at how much the score (accuracy, F1, R^2, etc. - any score we’re interested in) decreases when a feature is not available.



# Chapter 5

## Modelling

## 5.1 Model Selection:

After all early stages of preprocessing, then model the data. So, we have to select best model for this project with the help of some metrics.

The dependent variable can fall in either of the four categories:

1. Nominal

2. Ordinal

3. Interval

4. Ratio

If the dependent variable is Nominal the only predictive analysis that we can perform is **Classification**, and if the dependent variable is Interval or Ratio like this project, the normal method is to do a **Regression** analysis, or classification after binning.

**Handling of imbalance data**

Now we are going to explore 5 different approaches for dealing with imbalanced datasets.

 Change the performance metric

 Oversample minority class

 Under sample majority class

 Synthetic Minority Oversampling Technique (SMOTE) in Python or Random

Oversampling Examples (ROSE) in R

 Change the algorithm

We always start model building from the simplest to more complex.

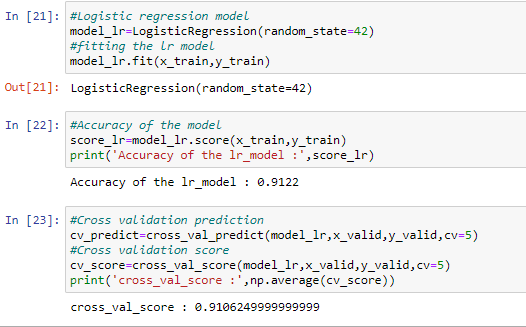
Stratified K-fold Cross Validation:

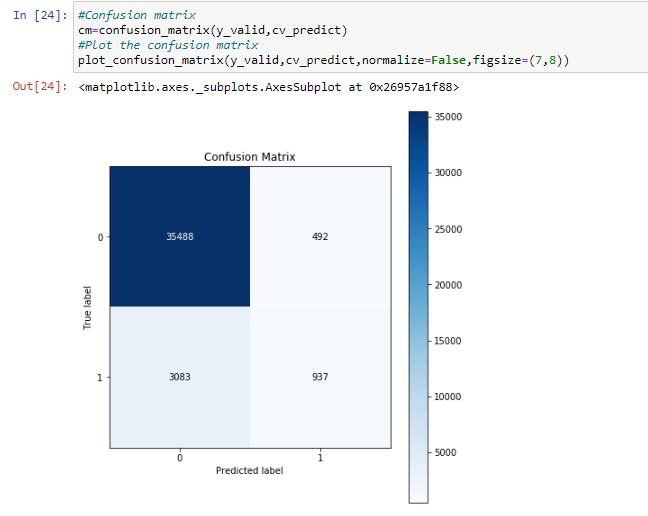
The splitting of data into folds may be governed by criteria such as ensuring that each fold has the same proportion of observations with a given categorical value, such as the class outcome value. This is called stratified cross-validation

As train & test data are imbalanced data set we need to use Stratified K-cross validator

## 5.2 Logistic regression:

We will use a Logistic Regression to predict the values of our target variable.

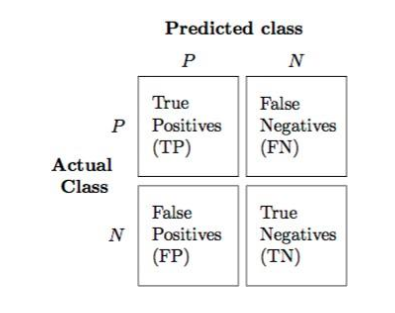


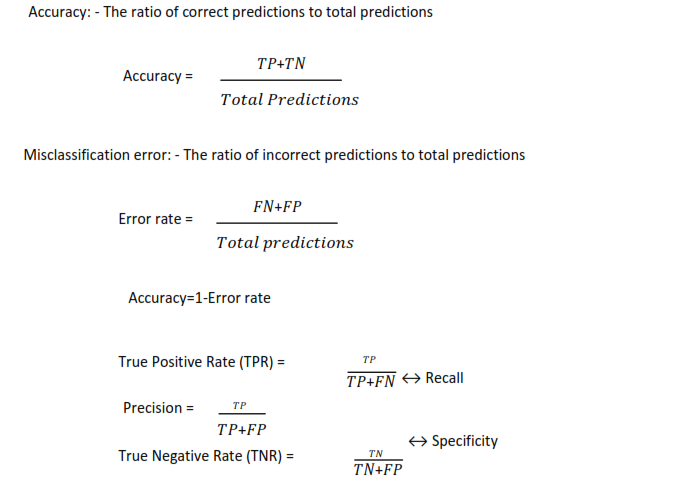


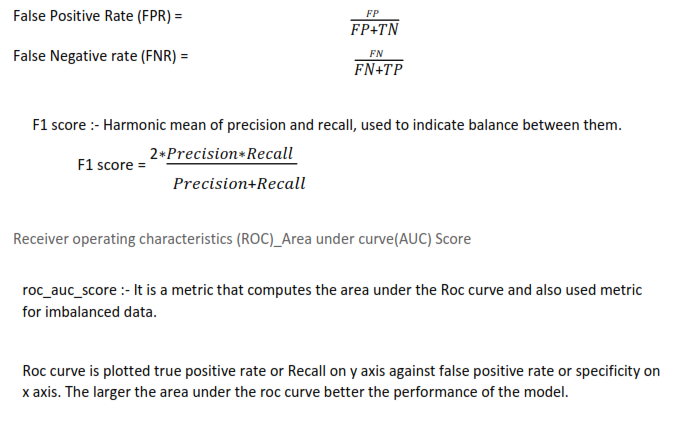
**Confusion Matrix: -** It is a technique for summarizing the performance of a classification

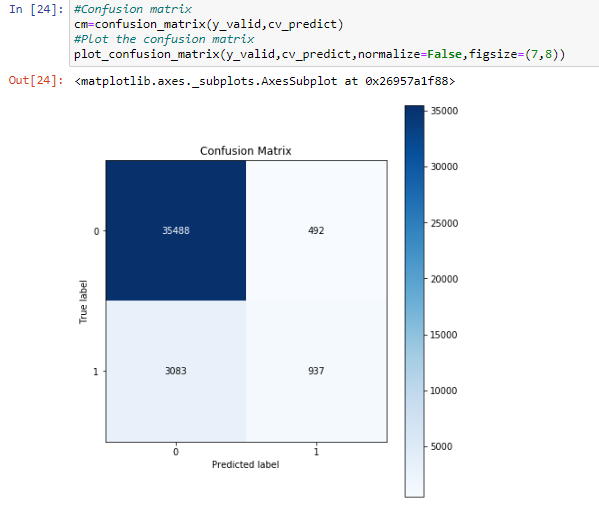
algorithm.

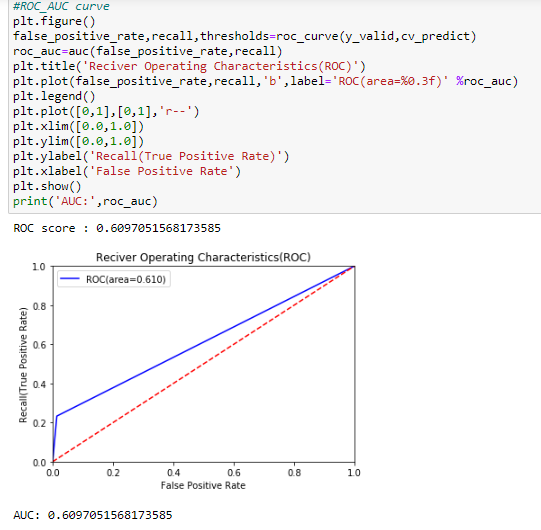
The number of correct predictions and incorrect predictions are summarized with count values and broken down by each class.











When we compare the roc\_auc\_score and cross validation score, conclude that model is

not performing well on imbalanced data.

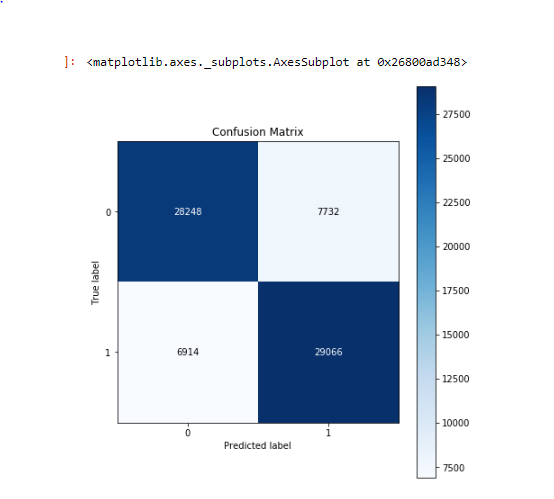
# 

We can observe that f1 score is high for number of customers those who will not make a

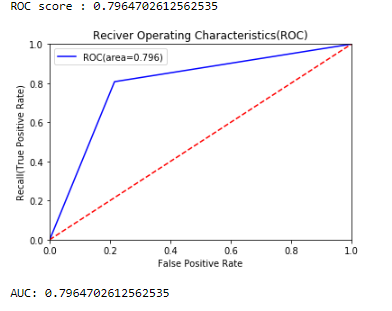
transaction then who will make a transaction. So, we are going to change the algorithm.

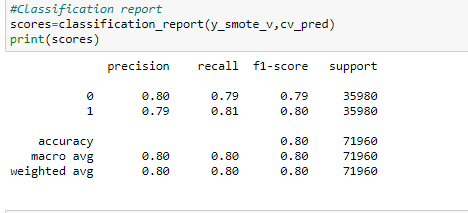
# 5.3 SMOTE(Synthetic Minority Oversampling Technique)

Classification using class-imbalanced data is biased in favor of the majority class. The bias is even larger for high-dimensional data, where the number of variables greatly exceeds the number of samples. The problem can be attenuated by undersampling or oversampling, which produce class-balanced data. Generally undersampling is helpful, while random oversampling is not. Synthetic Minority Oversampling TEchnique (SMOTE) is a very popular oversampling method that was proposed to improve random oversampling but its behavior on high-dimensional data has not been thoroughly investigated. In this paper we investigate the properties of SMOTE from a theoretical and empirical point of view, using simulated and real high-dimensional data.



**Reciever operating characteristics (ROC)-Area under curve (AUC) score and curve**





We can observe that smote model is performing well on imbalance data compare to

baseline logistic regression.

I tried different ways to get good accuracy like changing count of one target class

variable. Finally got area under ROC curve is 1 but this may not be possible.

# 5.4 LightGBM

When we compare scores of areas under the ROC curve of all the models for an imbalanced

data. We could conclude that below points as follow,

1. Logistic regression model is not performed well on imbalanced data.

2. We balance the imbalanced data using resampling techniques like SMOTE in python

and ROSE in R.

3. Baseline logistic regression model is performed well on balanced data.

4. LightGBM model performed well on imbalanced data.

Finally, LightGBM is best choice for identifying which customers will make a specific

transaction in the future, irrespective of the amount of money transacted.

# Chapter 6

# Conclusion:

When we compare scores of area under the ROC curve of all the models for an imbalanced data.

We could conclude that below points as follow:

1. Logistic Regression model is not performed well on imbalanced data.
2. We balanced data with imbalanced data using resampling techniques like SMOTE in python & ROSE in R.
3. Baseline logistic regression model is performed well on balanced data.
4. LightGBM model performed well on imbalanced data.

Finally, LightGBM is best choice for identifying which customers will make a specific transaction in the future, irrespective of the amount of money transacted.

# Chapter 7

# Code:

# 7.1 Python Code:

# Project Name – Santander Customer Transaction Prediction

## Problem Statement -

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

## Number of attributes:

You are provided with an anonymized dataset containing numeric feature variables, thebinary target column, and a string ID\_code column. The task is to predict the valueof target column in the test set.

#loading importtant libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

import os

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import RandomizedSearchCV

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split,cross\_val\_predict,cross\_val\_score

from sklearn.metrics import roc\_auc\_score,confusion\_matrix,make\_scorer,classification\_report,roc\_curve,auc

from sklearn.model\_selection import StratifiedKFold

from imblearn.over\_sampling import SMOTE

import lightgbm as lgb

import eli5

from eli5.sklearn import PermutationImportance

from scikitplot.metrics import plot\_confusion\_matrix,plot\_precision\_recall\_curve

import warnings

warnings.filterwarnings('ignore')

df\_train=pd.read\_csv("train.csv") ##Importing train data set

df\_train.head()

df\_train.shape

df\_train.describe()

Observation : Train Dataset contains 200000 values and 202 variables in which traget variable is available.

# Dataset balance checking & Visualization

target\_class=df\_train['target'].value\_counts() ### target classes values

print("Target values:\n",target\_class)

per\_target\_class=target\_class/len(df\_train)\*100 ##percentage of target\_class

print("Percentage of target class:\n",per\_target\_class)

sns.countplot(df\_train.target.values,palette="husl") #distribution of target

## Observation : we can say that train data is totally biased toward target label 0.

90% - Target label 0

10% - Target label 1

Hence we can say that data is totally imbalnced which leads to overfitting of the model.

df\_test=pd.read\_csv("test.csv") ##Importing test data set

df\_test.head()

df\_test.shape

df\_test.describe()

Observation : we dont have target label as train data set. we have to train data to predict target label.

# Missing value Analysis

# Finding missing value in train & test data.

Train\_Missing=df\_train.isnull().sum().sum()

print("Missing value in train data set:",Train\_Missing)

Test\_Missing=df\_test.isnull().sum().sum()

print("Missing value in test data set:",Test\_Missing)

Observation : No null value in Train as well as test data set.

# Correlation in train & test data set

#Correlations in train data

train\_attributes=df\_train.columns.values[2:202]

correlations\_train=df\_train[train\_attributes].corr()

correlations\_train=correlations\_train.values.flatten()

correlations\_train=correlations\_train[correlations\_train!=1]

#Correlations in test data

test\_attributes=df\_test.columns.values[1:201]

correlations\_test=df\_test[test\_attributes].corr()

correlations\_test=correlations\_test.values.flatten()

correlations\_test=correlations\_test[correlations\_test!=1]

plt.figure(figsize=(20,5))

#Distribution plot for correlations in train data

sns.distplot(correlations\_train, color="Red", label="train")

#Distribution plot for correlations in test data

sns.distplot(correlations\_test, color="Blue", label="test")

plt.xlabel("Correlation values found in train and test")

plt.ylabel("Density")

plt.title("Correlation distribution plot for train and test attributes")

plt.legend()

x=df\_train.iloc[:,2:]

x.head()

x.shape

y=df\_train.iloc[:,1]

y.head()

y.shape

x\_train,x\_valid,y\_train,y\_valid=train\_test\_split(x,y,random\_state=42)

print('Shape of X\_train :',x\_train.shape)

print('Shape of X\_valid :',x\_valid.shape)

print('Shape of y\_train :',y\_train.shape)

print('Shape of y\_valid :',y\_valid.shape)

#Random forest classifier

model\_rf=RandomForestClassifier(n\_estimators=10,random\_state=42)

#fitting the model

model\_rf.fit(x\_train,y\_train)

#Permutation importance

from eli5.sklearn import PermutationImportance

perm\_imp=PermutationImportance(model\_rf,random\_state=42)

#fitting the model

perm\_imp.fit(x\_valid,y\_valid)

#Important features

eli5.show\_weights(perm\_imp,feature\_names=x\_valid.columns.tolist(),top=200)

#Stratified cross validator

cv=StratifiedKFold(n\_splits=5,random\_state=42,shuffle=True)

for train\_index,valid\_index in cv.split(x,y):

x\_train, x\_valid=x.iloc[train\_index], x.iloc[valid\_index]

y\_train, y\_valid=y.iloc[train\_index], y.iloc[valid\_index]

print('Shape of X\_train :',x\_train.shape)

print('Shape of X\_valid :',x\_valid.shape)

print('Shape of y\_train :',y\_train.shape)

print('Shape of y\_valid :',y\_valid.shape)

#Logistic regression model

model\_lr=LogisticRegression(random\_state=42)

#fitting the lr model

model\_lr.fit(x\_train,y\_train)

#Accuracy of the model

score\_lr=model\_lr.score(x\_train,y\_train)

print('Accuracy of the lr\_model :',score\_lr)

#Cross validation prediction

cv\_predict=cross\_val\_predict(model\_lr,x\_valid,y\_valid,cv=5)

#Cross validation score

cv\_score=cross\_val\_score(model\_lr,x\_valid,y\_valid,cv=5)

print('cross\_val\_score :',np.average(cv\_score))

#Confusion matrix

cm=confusion\_matrix(y\_valid,cv\_predict)

#Plot the confusion matrix

plot\_confusion\_matrix(y\_valid,cv\_predict,normalize=False,figsize=(7,8))

#ROC\_AUC score

roc\_score=roc\_auc\_score(y\_valid,cv\_predict)

print('ROC score :',roc\_score)

#ROC\_AUC curve

plt.figure()

false\_positive\_rate,recall,thresholds=roc\_curve(y\_valid,cv\_predict)

roc\_auc=auc(false\_positive\_rate,recall)

plt.title('Reciver Operating Characteristics(ROC)')

plt.plot(false\_positive\_rate,recall,'b',label='ROC(area=%0.3f)' %roc\_auc)

plt.legend()

plt.plot([0,1],[0,1],'r--')

plt.xlim([0.0,1.0])

plt.ylim([0.0,1.0])

plt.ylabel('Recall(True Positive Rate)')

plt.xlabel('False Positive Rate')

plt.show()

print('AUC:',roc\_auc)

#Classification report

scores=classification\_report(y\_valid,cv\_predict)

print(scores)

#Predicting the model

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

pred\_lr=model\_lr.predict(x\_test)

print(pred\_lr)

from imblearn.over\_sampling import SMOTE

#Synthetic Minority Oversampling Technique

sm = SMOTE(random\_state=42)

#Generating synthetic data points

x\_smote,y\_smote=sm.fit\_sample(x\_train,y\_train)

x\_smote\_v,y\_smote\_v=sm.fit\_sample(x\_valid,y\_valid)

#Logistic regression model for SMOTE

smote=LogisticRegression(random\_state=42)

#fitting the smote model

smote.fit(x\_smote,y\_smote)

#Accuracy of the model

smote\_score=smote.score(x\_smote,y\_smote)

print('Accuracy of the smote\_model :',smote\_score)

#Cross validation prediction

cv\_pred=cross\_val\_predict(smote,x\_smote\_v,y\_smote\_v,cv=5)

#Cross validation score

cv\_score=cross\_val\_score(smote,x\_smote\_v,y\_smote\_v,cv=5)

print('cross\_val\_score :',np.average(cv\_score))

#Confusion matrix

cm=confusion\_matrix(y\_smote\_v,cv\_pred)

#Plot the confusion matrix

plot\_confusion\_matrix(y\_smote\_v,cv\_pred,normalize=False,figsize=(7,8))

#ROC\_AUC score

roc\_score=roc\_auc\_score(y\_smote\_v,cv\_pred)

print('ROC score :',roc\_score)

#ROC\_AUC curve

plt.figure()

false\_positive\_rate,recall,thresholds=roc\_curve(y\_smote\_v,cv\_pred)

roc\_auc=auc(false\_positive\_rate,recall)

plt.title('Reciver Operating Characteristics(ROC)')

plt.plot(false\_positive\_rate,recall,'b',label='ROC(area=%0.3f)' %roc\_auc)

plt.legend()

plt.plot([0,1],[0,1],'r--')

plt.xlim([0.0,1.0])

plt.ylim([0.0,1.0])

plt.ylabel('Recall(True Positive Rate)')

plt.xlabel('False Positive Rate')

plt.show()

print('AUC:',roc\_auc)

#Classification report

scores=classification\_report(y\_smote\_v,cv\_pred)

print(scores)

#Predicting the model

smote\_pred=smote.predict(x\_test)

print(smote\_pred)

#Training the model

#training data

lgb\_train=lgb.Dataset(x\_train,label=y\_train)

#validation data

lgb\_valid=lgb.Dataset(x\_valid,label=y\_valid)

params={'boosting\_type': 'gbdt',

'max\_depth' : -1, #no limit for max\_depth if <0

'objective': 'binary',

'boost\_from\_average':False,

'nthread': 20,

'metric':'auc',

'num\_leaves': 50,

'learning\_rate': 0.01,

'max\_bin': 100, #default 255

'subsample\_for\_bin': 100,

'subsample': 1,

'subsample\_freq': 1,

'colsample\_bytree': 0.8,

'bagging\_fraction':0.5,

'bagging\_freq':5,

'feature\_fraction':0.08,

'min\_split\_gain': 0.45, #>0

'min\_child\_weight': 1,

'min\_child\_samples': 5,

'is\_unbalance':True,

}

num\_rounds=10000

lgbm= lgb.train(params,lgb\_train,num\_rounds,valid\_sets=[lgb\_train,lgb\_valid],verbose\_eval=1000,early\_stopping\_rounds = 5000)

lgbm

#predict the model

#probability predictions

lgbm\_predict\_prob=lgbm.predict(x\_test,random\_state=42,num\_iteration=lgbm.best\_iteration)

#Convert to binary output 1 or 0

lgbm\_predict=np.where(lgbm\_predict\_prob>=0.5,1,0)

print(lgbm\_predict\_prob)

print(lgbm\_predict)

#plot the important features

lgb.plot\_importance(lgbm,max\_num\_features=50,importance\_type="split",figsize=(20,50))

#final prediction

predict\_df=pd.DataFrame({'ID\_code':df\_test['ID\_code'].values})

predict\_df['lgbm\_predict\_prob']=lgbm\_predict\_prob

predict\_df['lgbm\_predict']=lgbm\_predict

predict\_df.to\_csv('test\_predict.csv',index=False)

predict\_df.head()

# 7.2 R-Code

library(tidyverse)

library(moments)

library(DataExplorer)

library(caret)

library(Matrix)

library(pdp)

library(mlbench)

library(caTools)

library(randomForest)

library(glmnet)

library(mlr)

library(vita)

library(rBayesianOptimization)

library(lightgbm)

library(pROC)

library(DMwR)

library(ROSE)

library(yardstick)

list.files(path = "C:/Users/rkocherlakota/Desktop/ds\_projects/Santander Customer Transaction")

#loading the train data

train\_df<-read.csv('C:/Users/rkocherlakota/Desktop/ds\_projects/Santander Customer Transaction/train.csv')

head(train\_df)

#Dimension of train data

dim(train\_df)

#Summary of the dataset

str(train\_df)

#convert to factor

train\_df$target<-as.factor(train\_df$target)

require(gridExtra)

#Count of target classes

table(train\_df$target)

#Percenatge counts of target classes

table(train\_df$target)/length(train\_df$target)\*100

#Bar plot for count of target classes

plot1<-ggplot(train\_df,aes(target))+theme\_bw()+geom\_bar(stat='count',fill='lightgreen')

#Violin with jitter plots for target classes

plot2<-ggplot(train\_df,aes(x=target,y=1:nrow(train\_df)))+theme\_bw()+geom\_violin(fill='lightblue')+

facet\_grid(train\_df$target)+geom\_jitter(width=0.02)+labs(y='Index')

grid.arrange(plot1,plot2, ncol=2)

#Distribution of train attributes from 3 to 102

for (var in names(train\_df)[c(3:102)]){

target<-train\_df$target

plot<-ggplot(train\_df, aes(x=train\_df[[var]],fill=target)) +

geom\_density(kernel='gaussian') + ggtitle(var)+theme\_classic()

print(plot)

}

#Distribution of train attributes from 103 to 202

for (var in names(train\_df)[c(103:202)]){

target<-train\_df$target

plot<-ggplot(train\_df, aes(x=train\_df[[var]], fill=target)) +

geom\_density(kernel='gaussian') + ggtitle(var)+theme\_classic()

print(plot)

}

#loading test data

test\_df<-read.csv('C:/Users/rkocherlakota/Desktop/ds\_projects/Santander Customer Transaction/test.csv')

head(test\_df)

#Dimension of test dataset

dim(test\_df)

#Distribution of test attributes from 2 to 101

plot\_density(test\_df[,c(2:101)], ggtheme = theme\_classic(),geom\_density\_args = list(color='blue'))

#Distribution of test attributes from 102 to 201

plot\_density(test\_df[,c(102:201)], ggtheme = theme\_classic(),geom\_density\_args = list(color='blue'))

#Applying the function to find mean values per row in train and test data.

train\_mean<-apply(train\_df[,-c(1,2)],MARGIN=1,FUN=mean)

test\_mean<-apply(test\_df[,-c(1)],MARGIN=1,FUN=mean)

ggplot()+

#Distribution of mean values per row in train data

geom\_density(data=train\_df[,-c(1,2)],aes(x=train\_mean),kernel='gaussian',show.legend=TRUE,color='blue')+theme\_classic()+

#Distribution of mean values per row in test data

geom\_density(data=test\_df[,-c(1)],aes(x=test\_mean),kernel='gaussian',show.legend=TRUE,color='green')+

labs(x='mean values per row',title="Distribution of mean values per row in train and test dataset")

#Applying the function to find mean values per column in train and test data.

train\_mean<-apply(train\_df[,-c(1,2)],MARGIN=2,FUN=mean)

test\_mean<-apply(test\_df[,-c(1)],MARGIN=2,FUN=mean)

ggplot()+

#Distribution of mean values per column in train data

geom\_density(aes(x=train\_mean),kernel='gaussian',show.legend=TRUE,color='blue')+theme\_classic()+

#Distribution of mean values per column in test data

geom\_density(aes(x=test\_mean),kernel='gaussian',show.legend=TRUE,color='green')+

labs(x='mean values per column',title="Distribution of mean values per row in train and test dataset")

#Applying the function to find standard deviation values per row in train and test data.

train\_sd<-apply(train\_df[,-c(1,2)],MARGIN=1,FUN=sd)

test\_sd<-apply(test\_df[,-c(1)],MARGIN=1,FUN=sd)

ggplot()+

#Distribution of sd values per row in train data

geom\_density(data=train\_df[,-c(1,2)],aes(x=train\_sd),kernel='gaussian',show.legend=TRUE,color='red')+theme\_classic()+

#Distribution of mean values per row in test data

geom\_density(data=test\_df[,-c(1)],aes(x=test\_sd),kernel='gaussian',show.legend=TRUE,color='blue')+

labs(x='sd values per row',title="Distribution of sd values per row in train and test dataset")

#Applying the function to find sd values per column in train and test data.

train\_sd<-apply(train\_df[,-c(1,2)],MARGIN=2,FUN=sd)

test\_sd<-apply(test\_df[,-c(1)],MARGIN=2,FUN=sd)

ggplot()+

#Distribution of sd values per column in train data

geom\_density(aes(x=train\_sd),kernel='gaussian',show.legend=TRUE,color='red')+theme\_classic()+

#Distribution of sd values per column in test data

geom\_density(aes(x=test\_sd),kernel='gaussian',show.legend=TRUE,color='blue')+

labs(x='sd values per column',title="Distribution of std values per column in train and test dataset")

#Applying the function to find skewness values per row in train and test data.

train\_skew<-apply(train\_df[,-c(1,2)],MARGIN=1,FUN=skewness)

test\_skew<-apply(test\_df[,-c(1)],MARGIN=1,FUN=skewness)

ggplot()+

#Distribution of skewness values per row in train data

geom\_density(aes(x=train\_skew),kernel='gaussian',show.legend=TRUE,color='green')+theme\_classic()+

#Distribution of skewness values per column in test data

geom\_density(aes(x=test\_skew),kernel='gaussian',show.legend=TRUE,color='blue')+

labs(x='skewness values per row',title="Distribution of skewness values per row in train and test dataset")

#Applying the function to find skewness values per column in train and test data.

train\_skew<-apply(train\_df[,-c(1,2)],MARGIN=2,FUN=skewness)

test\_skew<-apply(test\_df[,-c(1)],MARGIN=2,FUN=skewness)

ggplot()+

#Distribution of skewness values per column in train data

geom\_density(aes(x=train\_skew),kernel='gaussian',show.legend=TRUE,color='green')+theme\_classic()+

#Distribution of skewness values per column in test data

geom\_density(aes(x=test\_skew),kernel='gaussian',show.legend=TRUE,color='blue')+

labs(x='skewness values per column',title="Distribution of skewness values per column in train and test dataset")

#Applying the function to find kurtosis values per row in train and test data.

train\_kurtosis<-apply(train\_df[,-c(1,2)],MARGIN=1,FUN=kurtosis)

test\_kurtosis<-apply(test\_df[,-c(1)],MARGIN=1,FUN=kurtosis)

ggplot()+

#Distribution of sd values per column in train data

geom\_density(aes(x=train\_kurtosis),kernel='gaussian',show.legend=TRUE,color='blue')+theme\_classic()+

#Distribution of sd values per column in test data

geom\_density(aes(x=test\_kurtosis),kernel='gaussian',show.legend=TRUE,color='red')+

labs(x='kurtosis values per row',title="Distribution of kurtosis values per row in train and test dataset")

#Applying the function to find kurtosis values per column in train and test data.

train\_kurtosis<-apply(train\_df[,-c(1,2)],MARGIN=2,FUN=kurtosis)

test\_kurtosis<-apply(test\_df[,-c(1)],MARGIN=2,FUN=kurtosis)

ggplot()+

#Distribution of sd values per column in train data

geom\_density(aes(x=train\_kurtosis),kernel='gaussian',show.legend=TRUE,color='blue')+theme\_classic()+

#Distribution of sd values per column in test data

geom\_density(aes(x=test\_kurtosis),kernel='gaussian',show.legend=TRUE,color='red')+

labs(x='kurtosis values per column',title="Distribution of kurtosis values per column in train and test dataset")

#Finding the missing values in train data

missing\_val<-data.frame(missing\_val=apply(train\_df,2,function(x){sum(is.na(x))}))

missing\_val<-sum(missing\_val)

missing\_val

#Finding the missing values in test data

missing\_val<-data.frame(missing\_val=apply(test\_df,2,function(x){sum(is.na(x))}))

missing\_val<-sum(missing\_val)

missing\_val

#Correlations in train data

#convert factor to int

train\_df$target<-as.numeric(train\_df$target)

train\_correlations<-cor(train\_df[,c(2:202)])

train\_correlations

#Correlations in test data

test\_correlations<-cor(test\_df[,c(2:201)])

test\_correlations

#Split the training data using simple random sampling

train\_index<-sample(1:nrow(train\_df),0.75\*nrow(train\_df))

#train data

train\_data<-train\_df[train\_index,]

#validation data

valid\_data<-train\_df[-train\_index,]

#dimension of train and validation data

dim(train\_data)

dim(valid\_data)

#Training the Random forest classifier

set.seed(2732)

#convert to int to factor

train\_data$target<-as.factor(train\_data$target)

#setting the mtry

mtry<-floor(sqrt(200))

#setting the tunegrid

tuneGrid<-expand.grid(.mtry=mtry)

#fitting the ranndom forest

rf<-randomForest(target~.,train\_data[,-c(1)],mtry=mtry,ntree=10,importance=TRUE)

#Variable importance

VarImp<-importance(rf,type=2)

VarImp

#We will plot "var\_13"

par.var\_13 <- partial(rf, pred.var = c("var\_13"), chull = TRUE)

plot.var\_13 <- autoplot(par.var\_13, contour = TRUE)

plot.var\_13

#We will plot "var\_34"

par.var\_34 <- partial(rf, pred.var = c("var\_34"), chull = TRUE)

plot.var\_34 <- autoplot(par.var\_34, contour = TRUE)

plot.var\_34

#Split the data using CreateDataPartition

set.seed(689)

#train.index<-createDataPartition(train\_df$target,p=0.8,list=FALSE)

train.index<-sample(1:nrow(train\_df),0.8\*nrow(train\_df))

#train data

train.data<-train\_df[train.index,]

#validation data

valid.data<-train\_df[-train.index,]

#dimension of train data

dim(train.data)

#dimension of validation data

dim(valid.data)

#target classes in train data

table(train.data$target)

#target classes in validation data

table(valid.data$target)

#Training dataset

X\_t<-as.matrix(train.data[,-c(1,2)])

y\_t<-as.matrix(train.data$target)

#validation dataset

X\_v<-as.matrix(valid.data[,-c(1,2)])

y\_v<-as.matrix(valid.data$target)

#test dataset

test<-as.matrix(test\_df[,-c(1)])

#Logistic regression model

set.seed(667) # to reproduce results

lr\_model <-glmnet(X\_t,y\_t, family = "binomial")

summary(lr\_model)

#Cross validation prediction

set.seed(8909)

cv\_lr <- cv.glmnet(X\_t,y\_t,family = "binomial", type.measure = "class")

cv\_lr

#Minimum lambda

cv\_lr$lambda.min

#plot the auc score vs log(lambda)

plot(cv\_lr)

#Model performance on validation dataset

set.seed(5363)

cv\_predict.lr<-predict(cv\_lr,X\_v,s = "lambda.min", type = "class")

cv\_predict.lr

#Confusion matrix

set.seed(689)

#actual target variable

target<-valid.data$target

#convert to factor

target<-as.factor(target)

#predicted target variable

#convert to factor

cv\_predict.lr<-as.factor(cv\_predict.lr)

confusionMatrix(data=cv\_predict.lr,reference=target)

#ROC\_AUC score and curve

set.seed(892)

cv\_predict.lr<-as.numeric(cv\_predict.lr)

roc(data=valid.data[,-c(1,2)],response=target,predictor=cv\_predict.lr,auc=TRUE,plot=TRUE)

#predict the model

#set.seed(763)

#lr\_pred<-predict(lr\_model,test,type='class')

#Random Oversampling Examples(ROSE)

set.seed(699)

train.rose <- ROSE(target~., data =train.data[,-c(1)],seed=32)$data

#target classes in balanced train data

table(train.rose$target)

valid.rose <- ROSE(target~., data =valid.data[,-c(1)],seed=42)$data

#target classes in balanced valid data

table(valid.rose$target)

#Logistic regression model

set.seed(462)

lr\_rose <-glmnet(as.matrix(train.rose),as.matrix(train.rose$target), family = "binomial")

summary(lr\_rose)

#Cross validation prediction

set.seed(473)

cv\_rose = cv.glmnet(as.matrix(valid.rose),as.matrix(valid.rose$target),family = "binomial", type.measure = "class")

cv\_rose

#Minimum lambda

cv\_rose$lambda.min

#plot the auc score vs log(lambda)

plot(cv\_rose)

#Model performance on validation dataset

set.seed(442)

cv\_predict.rose<-predict(cv\_rose,as.matrix(valid.rose),s = "lambda.min", type = "class")

cv\_predict.rose

#Confusion matrix

set.seed(478)

#actual target variable

target<-valid.rose$target

#convert to factor

target<-as.factor(target)

#predicted target variable

#convert to factor

cv\_predict.rose<-as.factor(cv\_predict.rose)

#Confusion matrix

confusionMatrix(data=cv\_predict.rose,reference=target)

#ROC\_AUC score and curve

set.seed(843)

#convert to numeric

cv\_predict.rose<-as.numeric(cv\_predict.rose)

roc(data=valid.rose[,-c(1,2)],response=target,predictor=cv\_predict.rose,auc=TRUE,plot=TRUE)

#predict the model

#set.seed(6543)

#rose\_pred<-predict(lr\_rose,test,type='class')

#Convert data frame to matrix

set.seed(5432)

X\_train<-as.matrix(train.data[,-c(1,2)])

y\_train<-as.matrix(train.data$target)

X\_valid<-as.matrix(valid.data[,-c(1,2)])

y\_valid<-as.matrix(valid.data$target)

test\_data<-as.matrix(test\_df[,-c(1)])

#training data

lgb.train <- lgb.Dataset(data=X\_train, label=y\_train)

#Validation data

lgb.valid <- lgb.Dataset(data=X\_valid,label=y\_valid)

#Selecting best hyperparameters

set.seed(653)

lgb.grid = list(objective = "binary",

metric = "auc",

boost='gbdt',

max\_depth=-1,

boost\_from\_average='false',

min\_sum\_hessian\_in\_leaf = 12,

feature\_fraction = 0.05,

bagging\_fraction = 0.45,

bagging\_freq = 5,

learning\_rate=0.02,

tree\_learner='serial',

num\_leaves=20,

num\_threads=5,

min\_data\_in\_bin=150,

min\_gain\_to\_split = 30,

min\_data\_in\_leaf = 90,

verbosity=-1,

is\_unbalance = TRUE)

set.seed(7663)

lgbm.model <- lgb.train(params = lgb.grid, data = lgb.train, nrounds =10000,eval\_freq =1000,

valids=list(val1=lgb.train,val2=lgb.valid),early\_stopping\_rounds = 5000)

#lgbm model performance on test data

set.seed(6532)

lgbm\_pred\_prob <- predict(lgbm.model,test\_data)

print(lgbm\_pred\_prob)

#Convert to binary output (1 and 0) with threshold 0.5

lgbm\_pred<-ifelse(lgbm\_pred\_prob>0.5,1,0)

print(lgbm\_pred)

set.seed(6521)

#feature importance plot

tree\_imp <- lgb.importance(lgbm.model, percentage = TRUE)

lgb.plot.importance(tree\_imp, top\_n = 50, measure = "Frequency", left\_margin = 10)

sub\_df<-data.frame(ID\_code=test\_df$ID\_code,lgb\_predict\_prob=lgbm\_pred\_prob,lgb\_predict=lgbm\_pred)

write.csv(sub\_df,'Submission.CSV',row.names=F)

head(sub\_df)