Project Report On

Santander Customer Transaction Prediction

Created By :- Sushil Kumar

INDEX

CHAPTER 1

INTRODUCTION :-

1.1 Problem Statement

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

1.2 Data Understanding

The Understanding of data should be the first and important step in the process of finding solution of any business problem. Here in our case our company has provided a below data set with following features :-

We have 202 features :-

ID_code, target,var_0,var_1,var_2……………………………………………var_199

ID_code :- This feature is consists of codes like a serial number , so it is not of much use for us .

target :- This is the dependent variable which consists if binary classification. i.e 0 and 1

var_0 to var_199 :- These are rest of the variables we have in the dataset for these we do not have much knowledge because these are encrypted variables and we do not know the names of these variables.

Size of Dataset Provided: - 200000 rows, 202 Columns (including dependent

variable)

Size of test Dataset Provided: - 200000 rows, 201 Columns

Missing Values: Yes

Outliers Presented: Yes

CHAPTER 2

Methodology

➢ Pre-Processing

When we required to build a predictive model, we require to look and manipulate the data before we start modelling which includes multiple preprocessing steps such as exploring the data, cleaning the data as well as visualizing the data through graph and plots, all these steps is combined under one shed which is **Exploratory Data Analysis**, which includes following steps:

- Data exploration and Cleaning
- Missing values treatment
- Outlier Analysis
- Feature Selection
- Features Scaling
- Visualization

➢ Modelling

Once all the Pre-Processing steps has been done on our data set, we will now further move to our next step which is modelling. Modelling plays an important role to find out the good inferences from the data. Choice of models depends upon the problem statement and data set. As per our problem statement and dataset, we will try some models on our preprocessed data and post comparing the output results we will select the best suitable model for our problem. As per our data set following models need to be tested as our business problem is a classification problem.

- Logistic regression
- Random forest
- Light GBM

➢ Model Selection

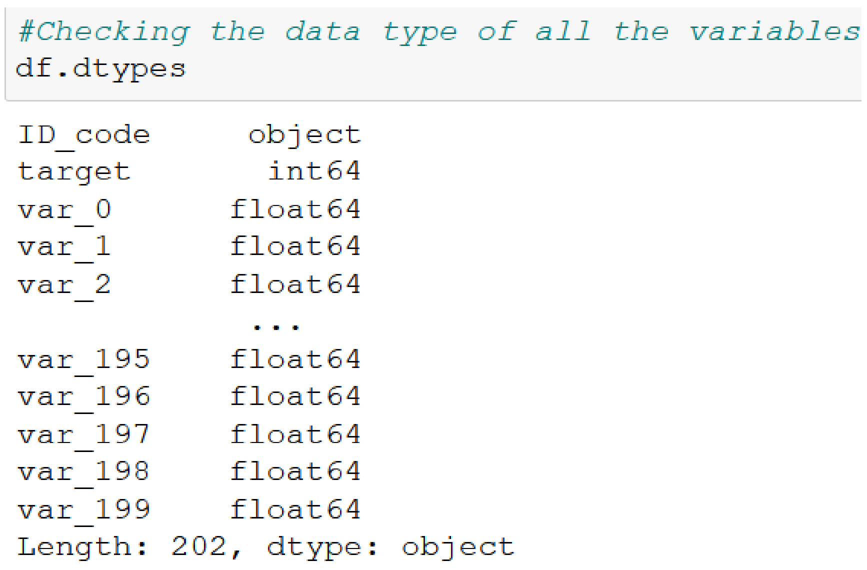
The final step of our methodology will be the selection of the model based on the different output and results shown by different models. We have multiple parameters which we will study further in our report to test whether the model is suitable for our problem statement or not.

CHAPTER 3

Pre-Processing

3.1 Data exploration and cleaning

In the given project, we have training data set of Santander Customer Transaction Prediction. The data which we have is unorganized in nature so, here we need to spend more time for data understanding, data cleaning, and data visualization to figure out features that will be the better predictors of cab fare**.**



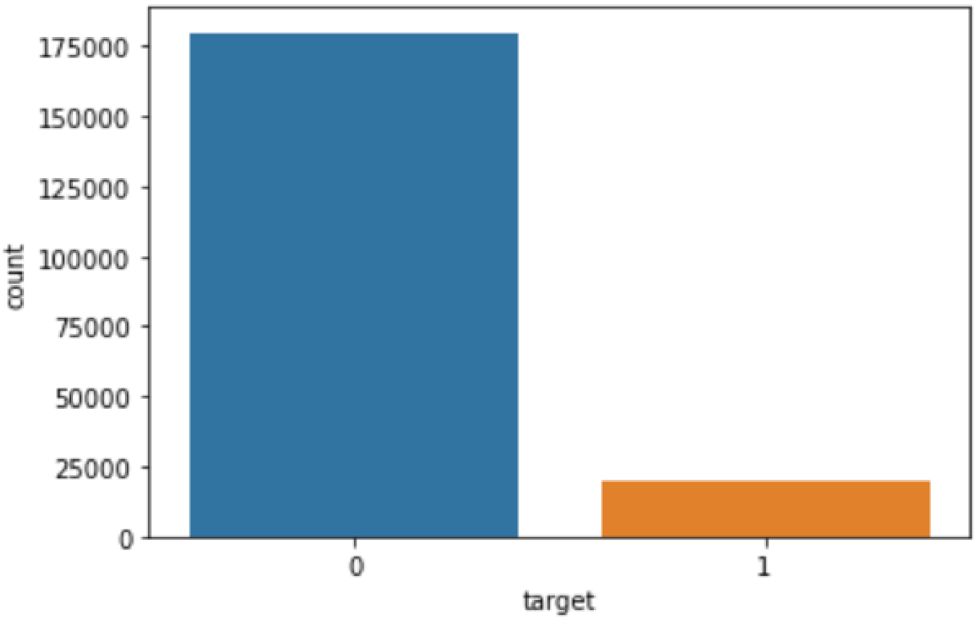
```
#Checking the data type of all the variables
df.dtypes
```

```
ID_code      object
target        int64
var_0       float64
var_1       float64
var_2       float64
             ...
var_195     float64
var_196     float64
var_197     float64
var_198     float64
var_199     float64
Length: 202, dtype: object
```

In the above Fig. we can see the data-types of the given attributes. We have all the variables from var_0 to var_99 are of floating type, that means we do not need to change those data types .Target variable is of int type that is also good to proceed.

Some important features :-

- ➔ **Target Variable** :- This feature is our dependent feature which consists of 2 classes 0 and 1 which is a binary classification. Lets check the count of 0's and 1's present in the dataset:-



Observation: -

We are having a unbalanced data, where 90% of the data is no. of customers who will not make a transaction & 10 % of the data are those who will make a transaction. total datapoints are 200000 on which 1's is only 20098 and 0's is 179902.

- ➔ **var_0 – var_199**:- These are the features we have in our dataset which are encrypted or for which we have not provided the exact feature names.

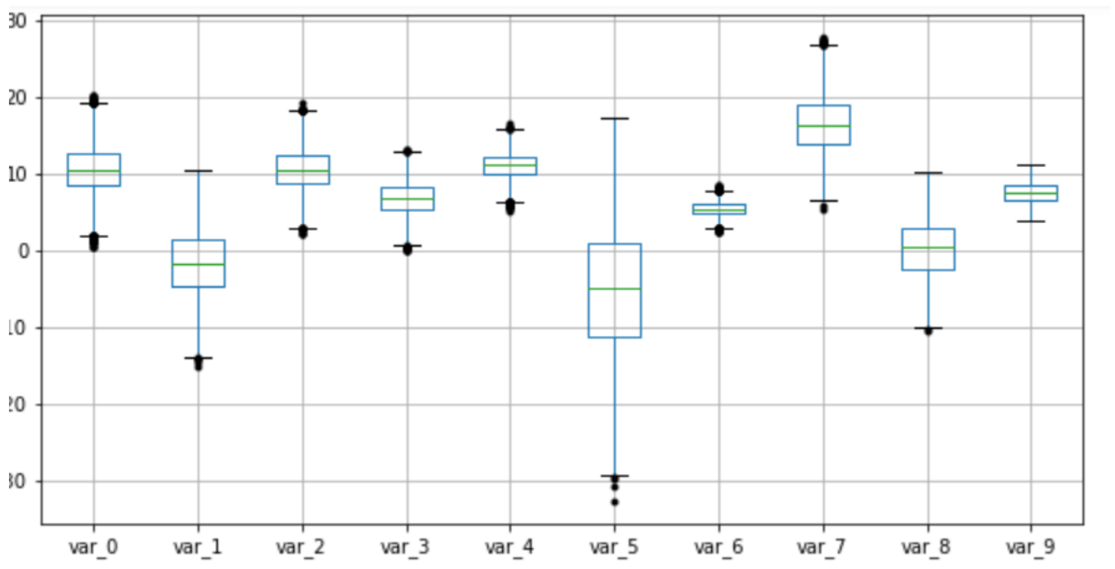
3.2 Missing Value and Outlier Analysis :-

Missing value analysis plays a vital role in data preparing. There are many reasons to occur missing values. In statistics while calculating missing values, if it is more than 30% we just drop the particular attribute because it does not carry much information to predict our target variables. We can impute the missing value with the help of central statistic method i.e. median.

- ➔ No missing values found neither in train nor in test datasets.

Outliers:-

Outlier is the observation which is inconsistent related with all data set. The values of Outliers are the accurate but it is far away from the set of actual values and it heavily impact on the mean so that we consider it as an outlier. Here we have used box plot method to detect outliers graphically as shown below Figs(var_0-var_9)e have checked for all the variables below is the example.



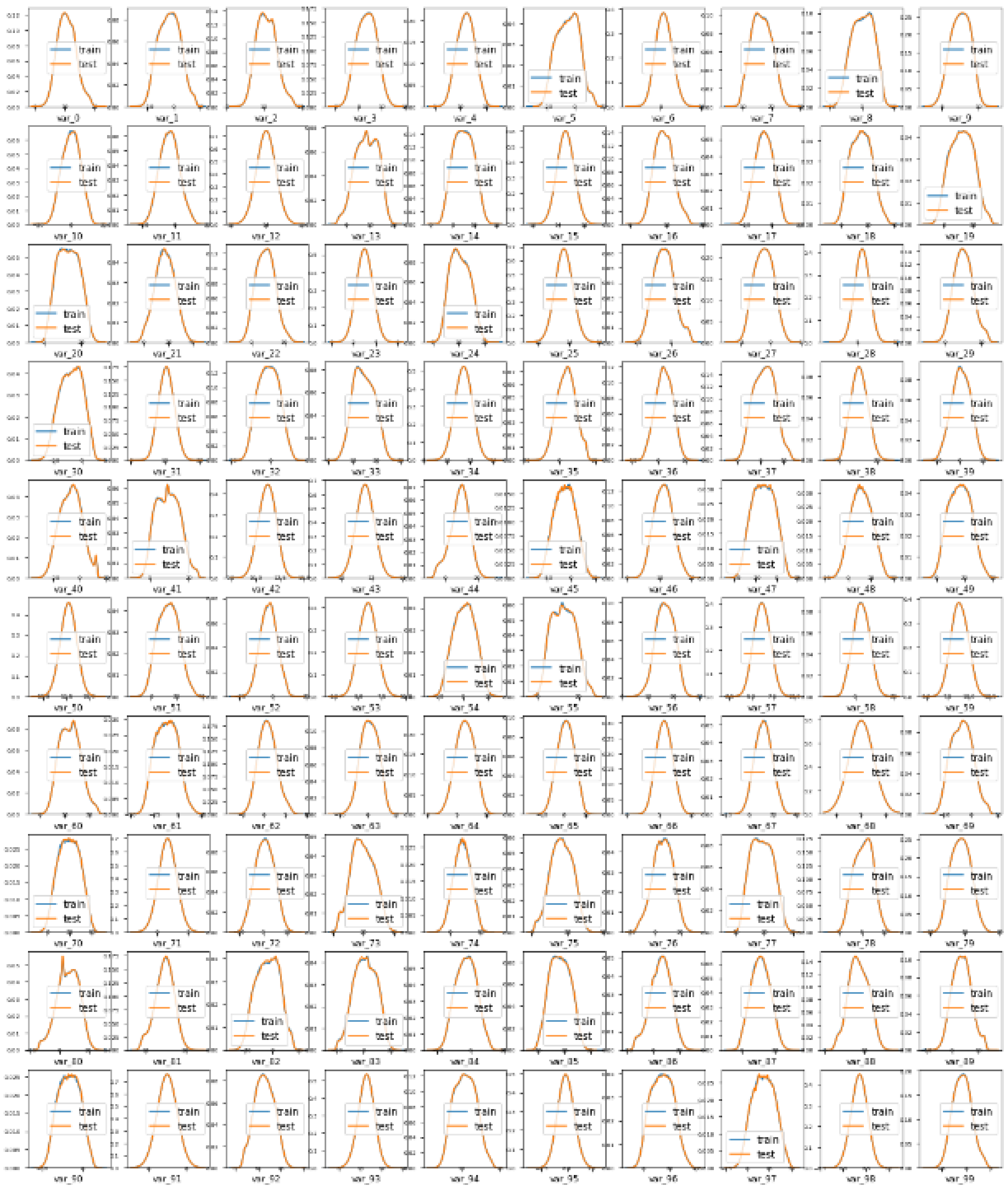
To remove the outliers we have created a function which will calculate the boundaries and remove those point which are out of boundaries , we have used mean+-3std concept to set the boundaries because the variables were normally distributed ,3 standard deviation will include 99% of the data.

3.3 Data Visualization :-

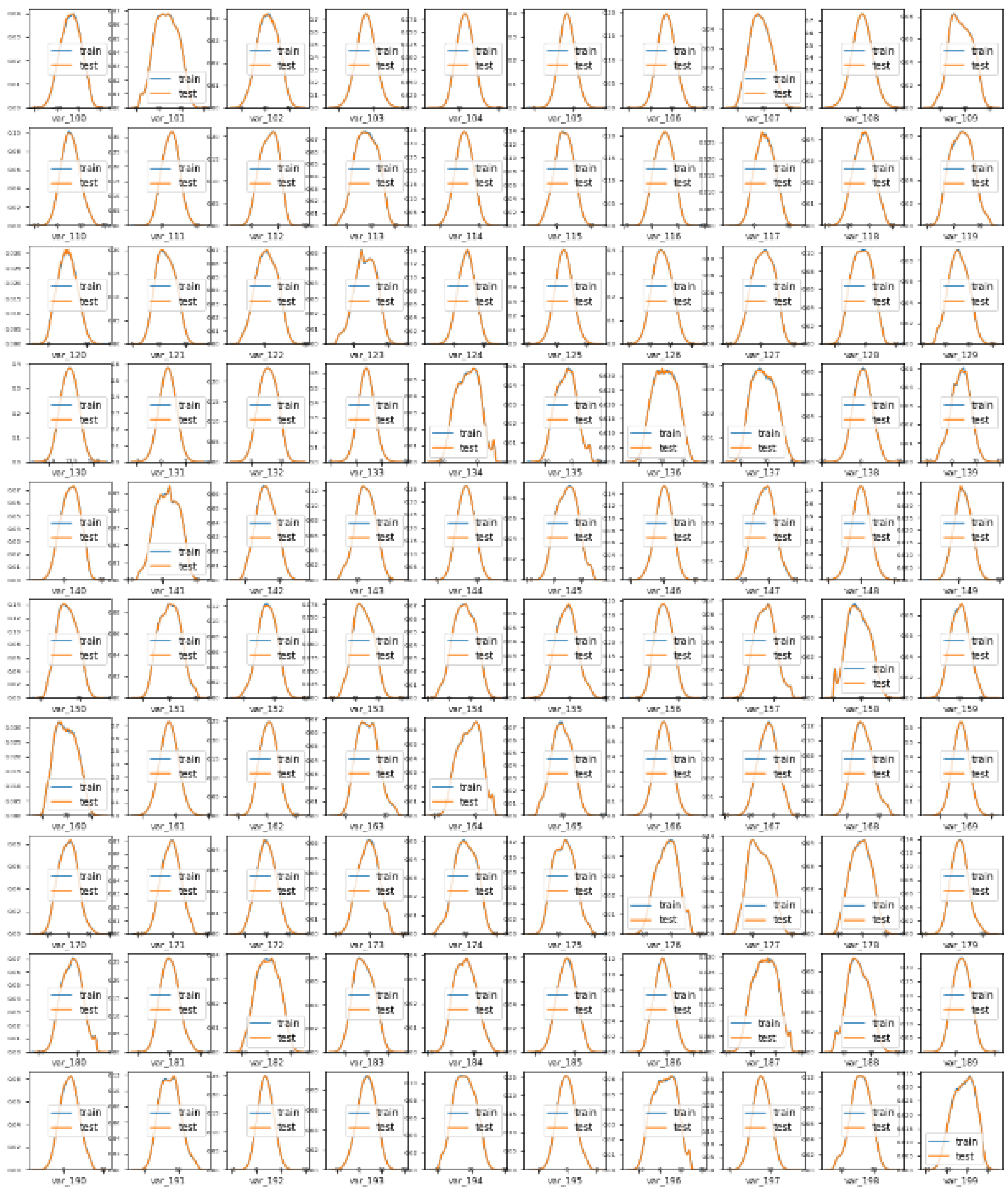
➔ **Attributes distributions and trends:-**

Distribution of train and test attributes :-

Let us look distribution of train and test attributes from var_0 to var_99



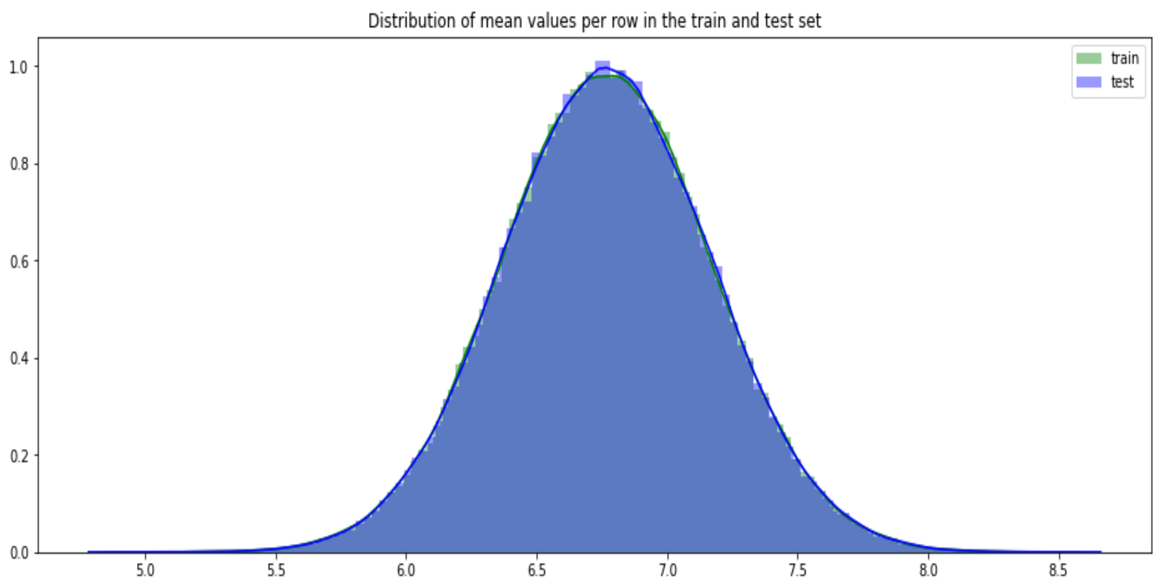
Let us look distribution of train and test attributes from var_100 to var_199



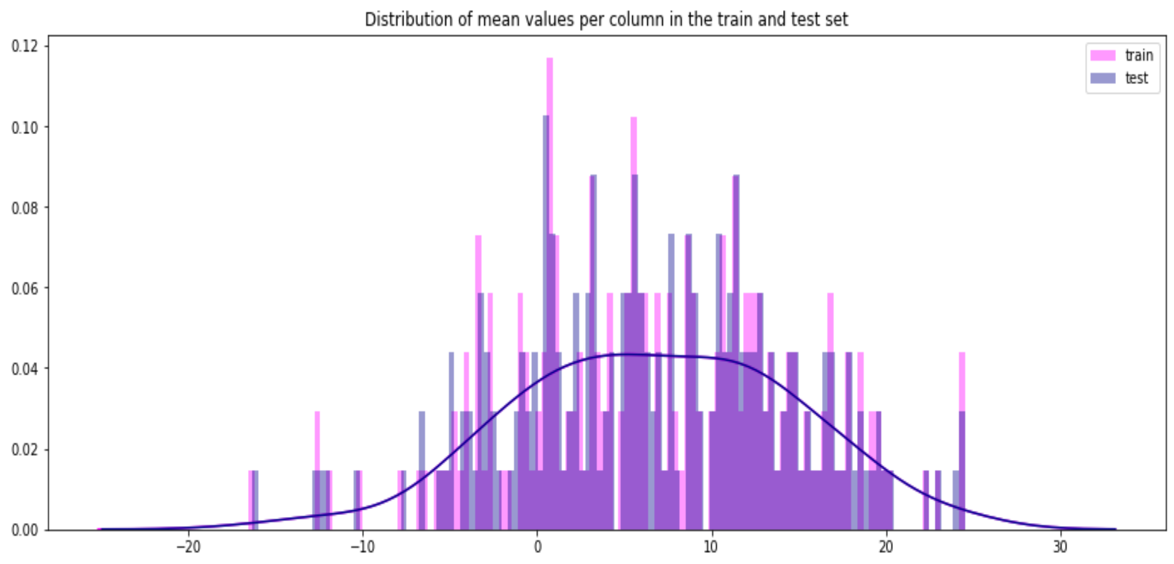
We can see that all the 200 Numerical variable is Normally distributed

➔ Distribution of mean and std

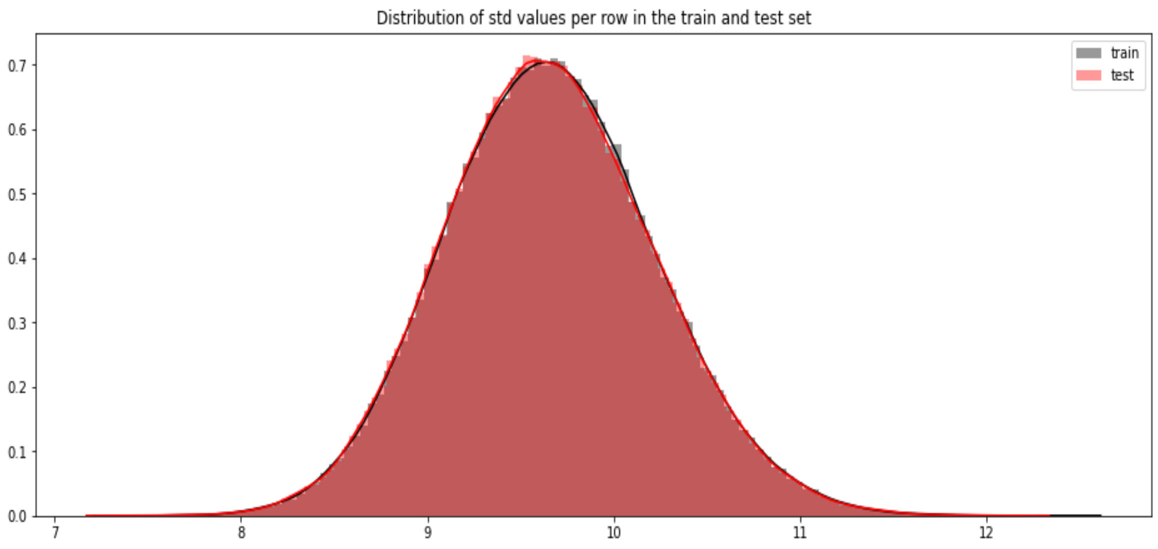
Distribution of the mean values per row in the train and test set :-



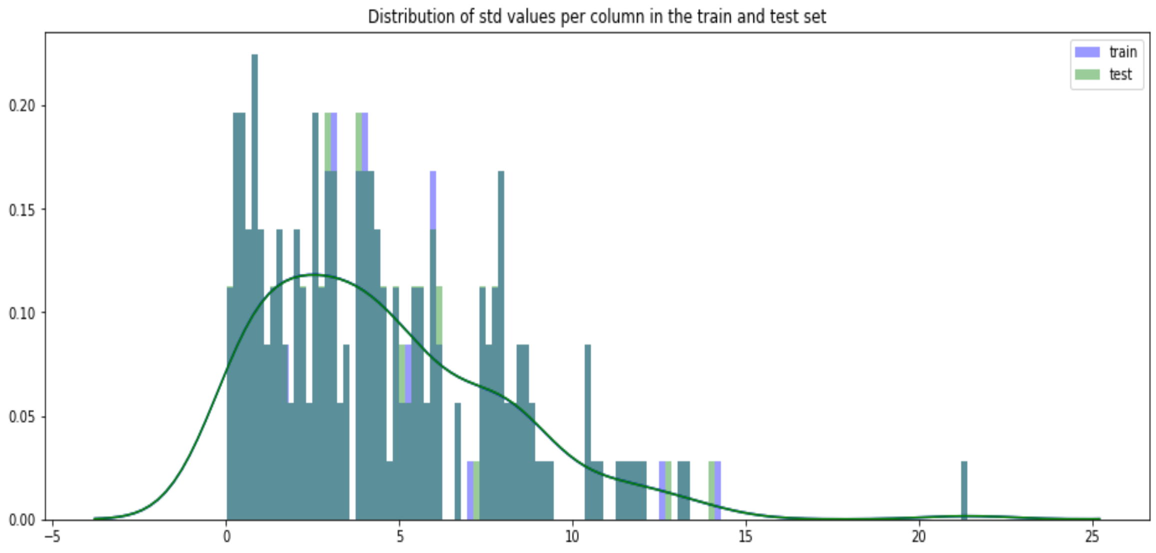
Distribution of mean values per column in the train and test set :-



Distribution of std values per row in the train and test set :-



Distribution of std values per column in the train and test set :-



Observations:

Mean values are distributed over a large range Standard deviation is relatively large for both train and test variable data

3.4 Feature Selection :-

Feature selection is very important for modelling the dataset. The every dataset have good and unwanted features. The unwanted features would effect 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 and PCA.

➔ Correlation matrix :-

Correlation matrix, it tells about linear relationship between attributes and help us to build better models.

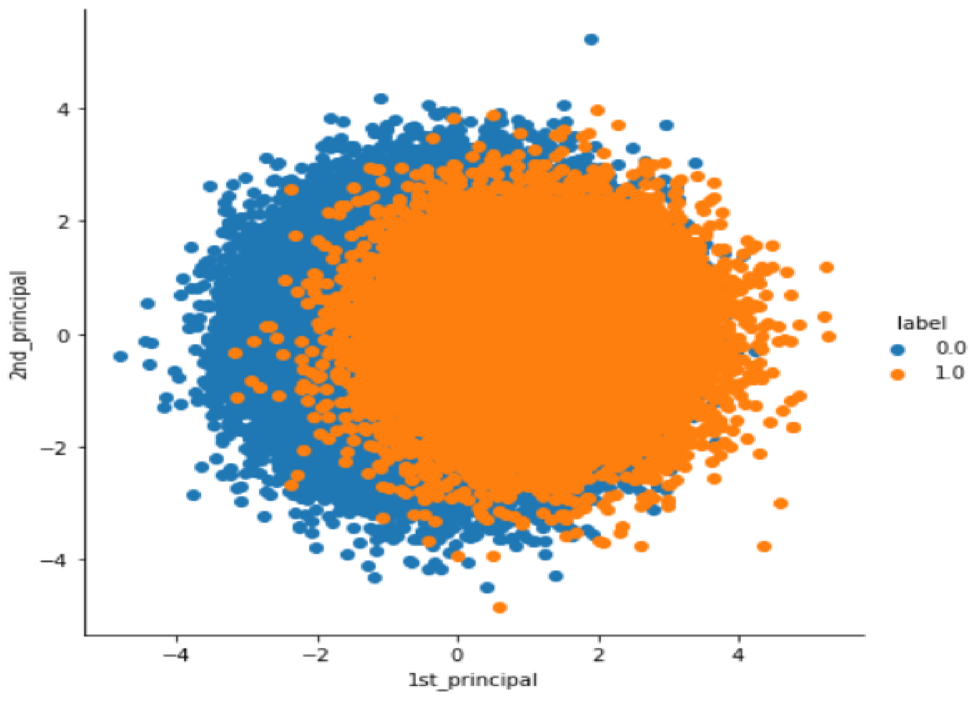
we can observed that correlation between both train and test attributes are very small. It means that all both train and test attributes are independent to each other.

1. correlation between train attributes is very small. Ranges between 0.001 to 0.009
2. correlation between test attributes is very small. Ranges between 0.001 to 0.009

➔ DIMENSIONALITY REDUCTION :-

Using Principal Component Analysis(PCA) :-

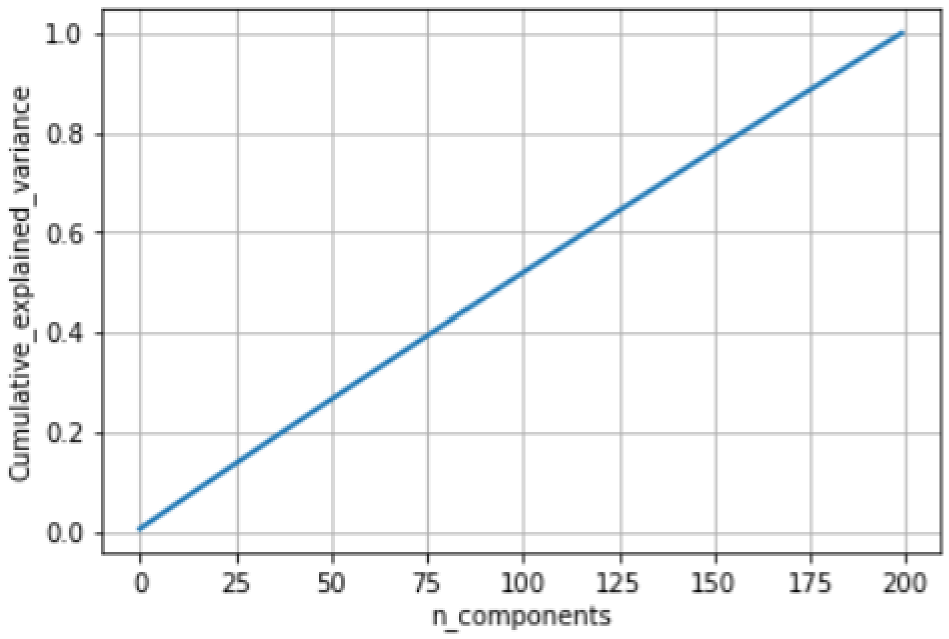
When we are not getting any dimentionality reduction option by using correlation then I have tried PCA below is the visualization .



Here, we clearly see that both the targets(0's & 1's) are overlapping.

Both the classes are not linearly separable.

we found that observations of different class labels are not well separated, indicating that classification cannot be implemented on low-dimensional space.



1. Here we have used PCA for feature importance.
2. We can clearly see that we are getting a straight line which means there is a linear relationship between no. of components and cumulative explained variance.
3. 100% of the cumulative explained variance is explained by all the 200 dimensions only.
4. That's why we are considering all the 200 dimensions.

CHAPTER 4

Modelling:-

After data cleaning and exploratory data analysis phase, we finally arrived at the model building phase. In this chapter we will apply multiple machine learning algorithm to predict the test case. In Santander Customer Transaction Prediction project our target variable is categorical (classification type of problem) so that here we are using classification models on structure data to predict the class of test case.

The next step is to differentiate the train data into 2 parts i.e. train and test. The splitting of train data into 2 parts is very important factor to verify the model performance and to understand the problem of over-fitting and under-fitting. Over-fitting is the term where training error is low and testing error is high and under-fitting is the term where both training and testing error is high. Those are the common problem of complex model.

In this analysis, since we are classifying the test class which is the categorical. So, we come to know that, our problem statement is classification type. So, what we can do is we will apply supervise machine learning algorithms to claasify our target variable. As we know our target variable is categorical in nature so, here we will build classification models.

Handling of imbalance data:-

To handel the Imbalanced data we are using the below technique to split the train and test dataset :-

Stratified K-Folds cross-validator

It Provides train/test indices to split data in train/test sets.

This cross-validation object is a variation of KFold that returns stratified folds. The folds are made by preserving the percentage of samples for each class.

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

4.1 Logistic Regression:-

Logistic Regression is a statistical model used to determine if an independent variable has an effect on a binary dependent variable. This means that there are only two potential outcomes given an input.

After applying the Logistic Regression we are getting the below accuracy and AUC :-

Before SMOTE :-

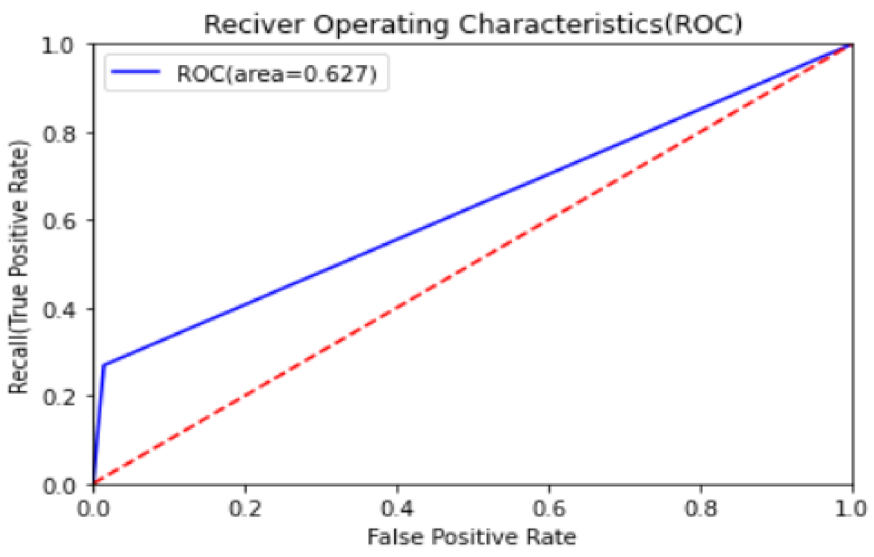
Accuracy :-

```
logit_score = logit.score(X_train,y_train)
print("The accuracy of Logit is", logit_score)
```

```
The accuracy of Logit is 0.913975
```

Accuracy of the model is not the best metric to use for classification problems while evaluating the imbalanced datasets as it may be misleading. We are going to change the performance metric.

AUC :-



```
AUC:  0.6274752557390922
```

There is big difference between roc_auc_score and model accuracy, model is not performing well on imbalanced data. So, we are going to make the dataset balanced first and try again.

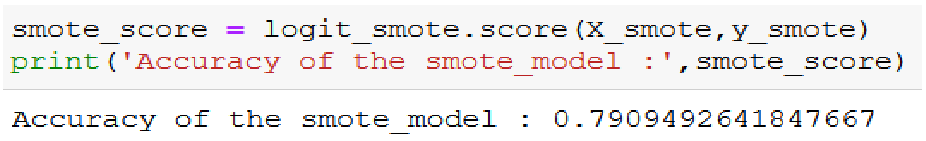
To make the dataset balanced we are using SMOTE here and then we will try Logistic Regression.

Synthetic Minority Oversampling Technique (SMOTE)

SMOTE uses a nearest neighbor's algorithm to generate new and synthetic data to use for training the model. In order to balance imbalanced data we are going to use SMOTE sampling method.

After SMOTE :-

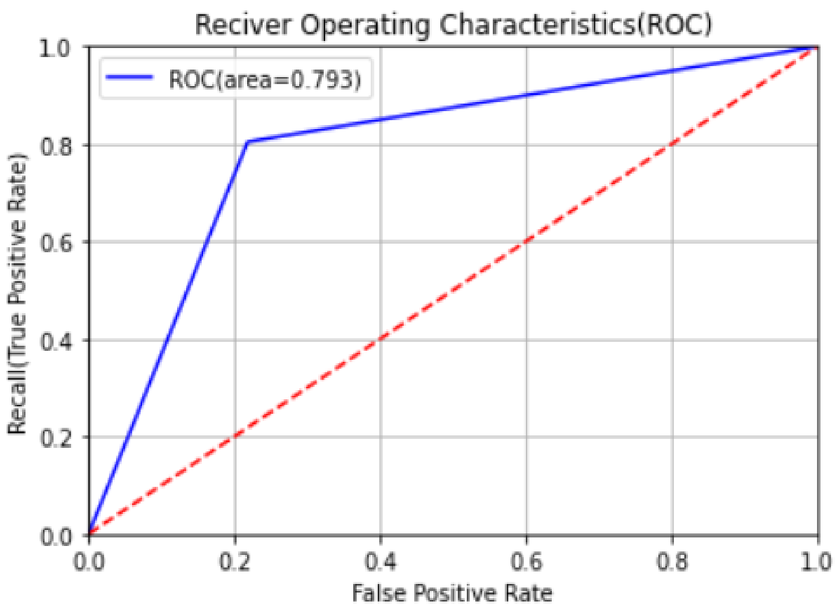
Accuracy :-



```
smote_score = logit_smote.score(X_smote,y_smote)
print('Accuracy of the smote_model :',smote_score)
```

```
Accuracy of the smote_model : 0.7909492641847667
```

AUC :-



Now we can see that the AUC is increased and model is performing better then before SMOTE

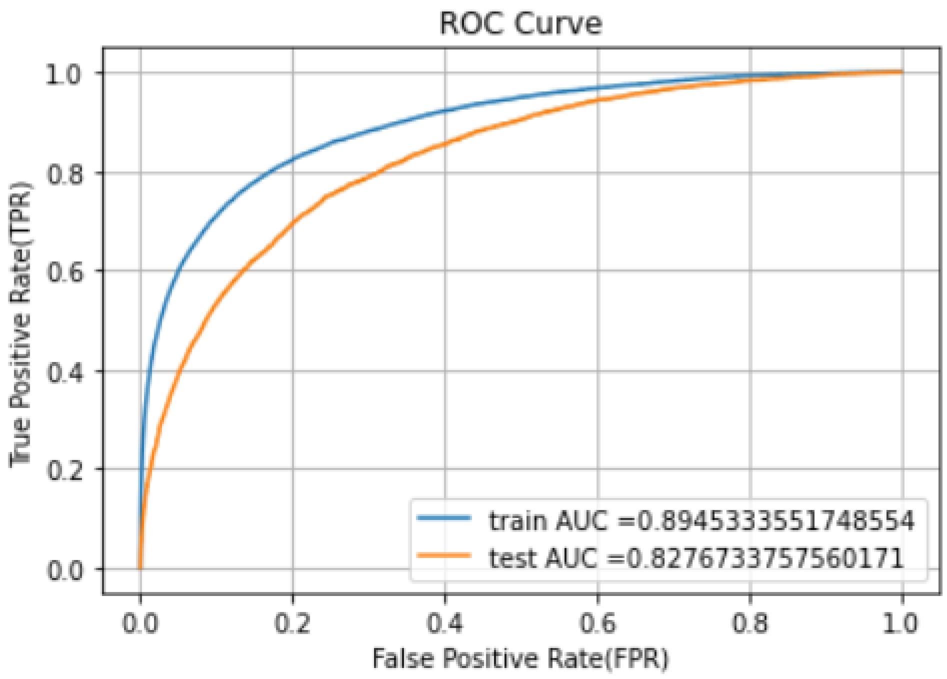
4.2 Random Forest :-

Random forest is the collection of multiple decision trees. In Random Forest, output is the average prediction by each of these trees. For this method to work, the baseline models must have a lower bias. The idea behind Random Forest is to build n number of trees to have more accuracy in dataset. Random Forest uses bagging method for predictions. It can handle large no of independent variables without variable deletion.

AUC :-

```
==============================================
Test accuracy score : 0.8995
==============================================
AUC score on train data : 0.8945333551748554
AUC score on test data : 0.8276733757560171
```

ROC Graph :-



4.3 Light GBM:-

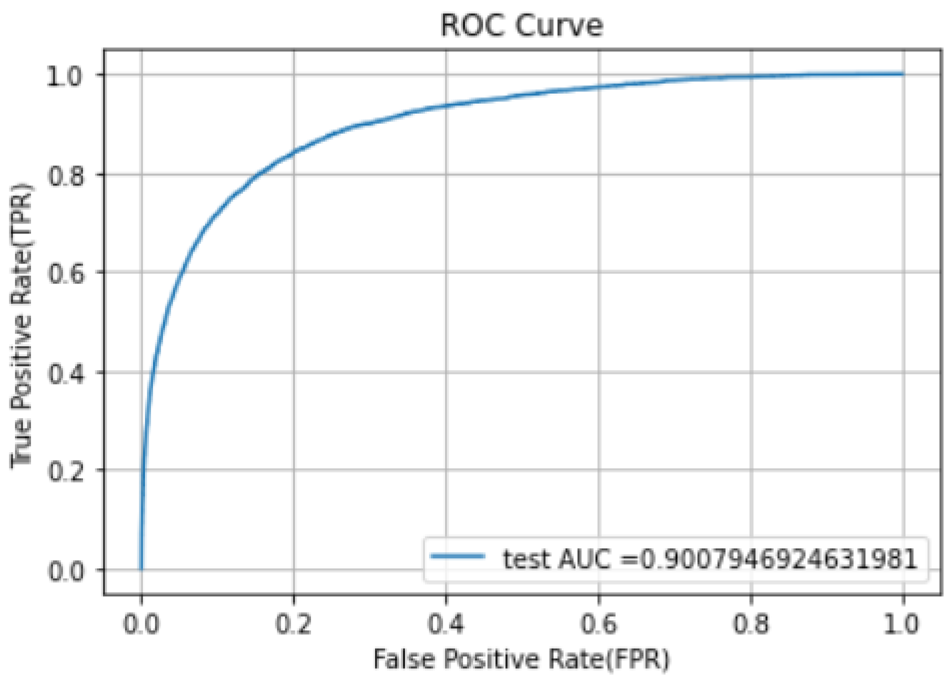
Light GBM is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for classification and many other machine learning tasks.

Since it is based on decision tree algorithms, it splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise. So when growing on the same leaf in Light GBM, the leaf-wise algorithm can reduce more loss than the level-wise algorithm and hence results in much better accuracy which can rarely be achieved by any of the existing boosting algorithms. Also, it is surprisingly very fast, hence the word 'Light'.

AUC :-

```
Training until validation scores don't improve for 5000 rounds
[1000]  training's auc: 0.870831        valid_1's auc: 0.859831
[2000]  training's auc: 0.899956        valid_1's auc: 0.884114
[3000]  training's auc: 0.912861        valid_1's auc: 0.893401
[4000]  training's auc: 0.920855        valid_1's auc: 0.897634
[5000]  training's auc: 0.926323        valid_1's auc: 0.899636
[6000]  training's auc: 0.930799        valid_1's auc: 0.90044
[7000]  training's auc: 0.934878        valid_1's auc: 0.90096
[8000]  training's auc: 0.938866        valid_1's auc: 0.9009
[9000]  training's auc: 0.942638        valid_1's auc: 0.900891
[10000] training's auc: 0.946282        valid_1's auc: 0.900795
Did not meet early stopping. Best iteration is:
[10000] training's auc: 0.946282        valid_1's auc: 0.900795
```

ROC GRAPH :-



CHAPTER 5

Conclusion

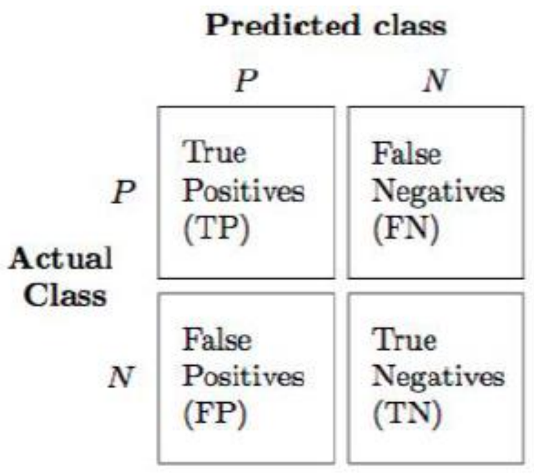
In above chapters we applied multiple preprocessing to frame our data into the structural format and different machine learning algorithm to check the performance of model. In this chapter we finalize one of them.

5.1 Model Evaluation

Now, we have a three models for predicting the target variable, but we need to decide which model better for this project. There are many metrics used for model evaluation.
Classification accuracy may be misleading if we have an imbalanced dataset or if we have more than two classes in dataset.
For classification problems, the confusion matrix used for evaluation. But, in our case the data is imbalanced. So, roc_auc_score is used for evaluation.

In this project, we are using two metrics for model evaluation as follows,:

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.

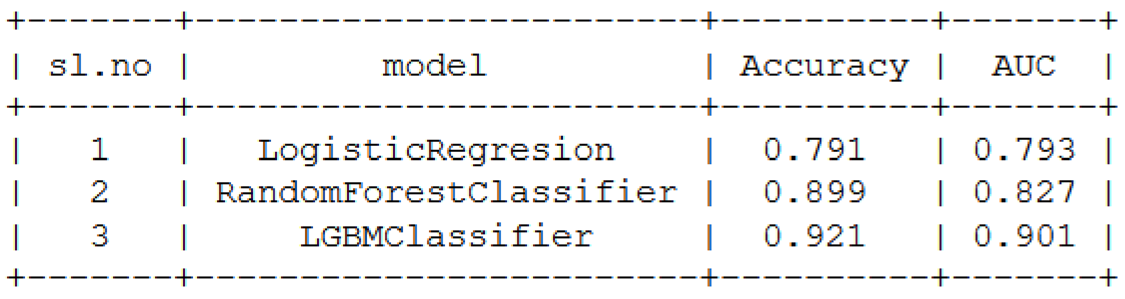


Accuracy: - The ratio of correct predictions to total predictions
Accuracy = $(TP+TN)/Total\ Predictions$
Misclassification error: - The ratio of incorrect predictions to total predictions
Error rate = $(FN+FP)/Total\ predictions$
Accuracy=1-Error rate
True Positive Rate (TPR) = $TP/(TP+FN)$ ↔ Recall
Precision = $TP/(TP+FP)$
True Negative Rate (TNR) = $TN/(TN+FP)$ ↔ Specificity
False Positive Rate (FPR) = $FP/(FP+TN)$
False Negative rate (FNR) = $FN/(FN+TP)$
F1 score :- Harmonic mean of precision and recall, used to indicate balance between them. 33

F1 score = $2*Precision*RecallPrecision+Recall$

Receiver operating characteristics (ROC)_Area under curve(AUC) Score

roc_auc_score :- It is a metric that computes the area under the Roc curve and also used metric for imbalanced data.
Roc curve is plotted true positive rate or Recall on y axis against false positive rate or specificity on x axis. The larger the area under the roc curve better the performance of the model.



```
+-------+------------------------+----------+-------+
| sl.no |         model          | Accuracy |  AUC  |
+-------+------------------------+----------+-------+
|   1   |   LogisticRegresion    |  0.791   | 0.793 |
|   2   | RandomForestClassifier |  0.899   | 0.827 |
|   3   |     LGBMClassifier     |  0.921   | 0.901 |
+-------+------------------------+----------+-------+
```

Here we can see clearly that Light GBM is performing well as compared to other models.

5.2 Model Selection

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 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.

I have applied the same on test case data to classify. Results that I found are attached with my submissions.

Appendix

1. **Python code is attached separately**

2. **R Code :-**

```
#Loading Libraries:-

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)

#Setting Directory:-
setwd("D:/Practice_R")

#Importing the training Data:-
df_train=read.csv("train.csv")
head(df_train)

#Dimension of the train data:-
dim(df_train)

#Summary of the train dataset:-
str(df_train)

#Typecasting the target variable:-
df_train$target=as.factor(df_train$target)

#Target class count in train data:-
table(df_train$target)

#Percentage count of taregt class in train data:-
table(df_train$target)/length(df_train$target)*100
```

```
#Bar plot for count of target classes in train data:-
plot1=ggplot(df_train,aes(target))+theme_bw()+geom_bar(stat='count',fill='lightgreen')

#Violin with jitter plots for target classes
plot2=ggplot(df_train,aes(x=target,y=1:nrow(df_train)))+theme_bw()+geom_violin(fill='lightblue')+
facet_grid(df_train$target)+geom_jitter(width=0.02)+labs(y='Index')

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

#Observation:- We are having a unbalanced data, where 90% of the data is no. of customers who will not make a transaction & 10 % of the data are those who will make a transaction.

```
#Distribution of train attributes from 3 to 102:-
for (var in names(df_train)[c(3:102)]){
target<-df_train$target
plot<-ggplot(df_train, aes(df_train[[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(df_train)[c(103:202)]){
target<-df_train$target
plot<-ggplot(df_train, aes(df_train[[var]],fill=target)) +
geom_density(kernel='gaussian') + ggtitle(var)+theme_classic()
print(plot)
}


#Importing the test data:-
df_test=read.csv("test.csv")
head(df_test)

#Dimension of test dataset:-
dim(df_test)

#Distribution of test attributes from 2 to 101:-
plot_density(df_test[,c(2:101)],ggtheme = theme_classic(),geom_density_args = list(color='red'))


#Distribution of test attributes from 102 to 201:-
plot_density(df_test[,c(102:201)],ggtheme = theme_classic(),geom_density_args = list(color='red'))
```

#Mean value per rows and columns in train & test dataset:-

```
#Applying the function to find mean values per row in train and test data.
train_mean<-apply(df_train[,-c(1,2)],MARGIN=1,FUN=mean)
test_mean<-apply(df_test[,-c(1)],MARGIN=1,FUN=mean)
ggplot()+
#Distribution of mean values per row in train data
geom_density(data=df_train[,-
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=df_test[,-
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(df_train[,-c(1,2)],MARGIN=2,FUN=mean)
test_mean<-apply(df_test[,-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 column in train and test dataset")
```

Standard Deviation Distribution:-

```
#Applying the function to find standard deviation values per row in train and test data.
train_sd<-apply(df_train[,-c(1,2)],MARGIN=1,FUN=sd)
test_sd<-apply(df_test[,-c(1)],MARGIN=1,FUN=sd)
ggplot()+

#Distribution of sd values per row in train data
geom_density(data=df_train[,-
c(1,2)],aes(x=train_sd),kernel='gaussian',show.legend=TRUE,color='red')+theme_classic()+

#Distribution of sd values per row in test data
geom_density(data=df_test[,-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(df_train[,-c(1,2)],MARGIN=2,FUN=sd)
test_sd<-apply(df_test[,-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")
```

```
#Missing Value Analysis:-
#Finding the missing values in train data
missing_val<-data.frame(missing_val=apply(df_train,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(df_test,2,function(x){sum(is.na(x))}))
missing_val<-sum(missing_val)
missing_val


#Correlations in train data:-
#convert factor to int
df_train$target<-as.numeric(df_train$target)
train_correlation<-cor(df_train[,c(2:202)])
train_correlation

#Observation:- We can observe that correlation between train attributes is very small.

#Correlations in test data
test_correlation<-cor(df_test[,c(2:201)])
test_correlation

#Observation:- We can observe that correlation between test attributes is very small.
```

#Handling of imbalanced data- Now we are going to explore 5 different approaches for dealing with imbalanced datasets.

```
#Change the performance metric
#Oversample minority class
#Undersample majority class
#ROSE
#LightGBM


#Logistic Regression Model:-

#Split the data using simple random sampling:-
set.seed(689)
train.index<-sample(1:nrow(df_train),0.8*nrow(df_train))

#train data
train.data<-df_train[train.index,]

#validation data
valid.data<-df_train[-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 and validation dataset
#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(df_test[,-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

#Plotting the missclassification error vs log(lambda) where lambda is regularization parameter
#Minimum lambda
cv_lr$lambda.min

#plot the auc score vs log(lambda)
plot(cv_lr)


#Observation:-We can observed that miss classification error increases as increasing the log(Lambda).

#Model performance on validation dataset
set.seed(5363)
cv_predict.lr<-predict(cv_lr,X_v,s = "lambda.min", type = "class")
cv_predict.lr
```

#Observation:-Accuracy of the model is not the best metric to use when evaluating the imbalanced datasets as it may be misleading. So, we are going to change the performance metric.

```
#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)

#Reciever operating characteristics(ROC)-Area under curve(AUC) score and curve:-
#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)
```

#Oversample Minority Class:-
#-Adding more copies of minority class.
#-It cab be a good option we dont have that much large data to work.
#-Drawback of this process is we are adding info. That can lead to overfitting or poor performance on test data.

#Undersample Mojorityclass:-
#-Removing some copies of majority class.
#-It can be a good option if we have very large amount of data say in millions to work.
#-Drawback of this process is we are removing some valuable info. that can leads to underfitting & poor performance on test data.
#Both Oversampling and undersampling techniques have some drawbacks. So, we are not going to use this models for this problem and also we will use other best algorithms.

#Random Oversampling Examples(ROSE)- It creates a sample of synthetic data by enlarging the features space of minority and majority class examples.

```
#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

#Plotting the missclassification error vs log(lambda) where lambda is regularization parameter:-
#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)
```

#LightGBM:-LightGBM is a gradient boosting framework that uses tree based learning algorithms. We are going to use LightGBM model.

```
#Training and validation dataset
#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(df_test[,-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)

#Choosing best hyperparameters

#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)
```

```
#Training the lgbm model

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)

#Let us plot the important features
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)


#We tried model with logistic regression,ROSE and lightgbm. But,lightgbm is performing well on
imbalanced data compared to other models based on scores of roc_auc_score.

#Final submission
sub_df<-
data.frame(ID_code=df_test$ID_code,lgb_predict_prob=lgbm_pred_prob,lgb_predict=lgbm_pred)
write.csv(sub_df,'submission-R.CSV',row.names=F)
head(sub_df)
```

References

1. Data Cleaning, Model Development and Data Visualization
https://edwisor.com/career-data-scientist

2. For Visualization using seaborn and other topics
Krish Naik Youtube channel.