Santander Customer Transaction Prediction

SUBMITTED BY:-

NIKHIL KUMAR

Contents

INTRODUCTION

At Santander their mission is to help people and businesses prosper. They 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.

1.1 Problem Statement

This is a binary classification problem. The aim of this project is to identify which customers will make a specific transaction in the future, irrespective of the amount of money transacted.

1.2 Data

This is a classification problem & we have to build classification models which will predict whether customer will make transactions or not with the given data. As the data is clueless because, there are not specific names of the variables (i.e. var_0, var_1, var_2, ------, var_199). There are two datasets train and test.

PROBLEM DEFINITION

	ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6
1	train_0	0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187
2	train_1	0	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208
3	train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427
4	train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428
5	train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405
6	train_5	0	11.4763	-2.3182	12.6080	8.6264	10.9621	3.5609	4.5322

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

2.1 Problem Feature

1. train.csv= Train data have **(200000 obs. Of 202 variables)**.Train data have ***target variable.***

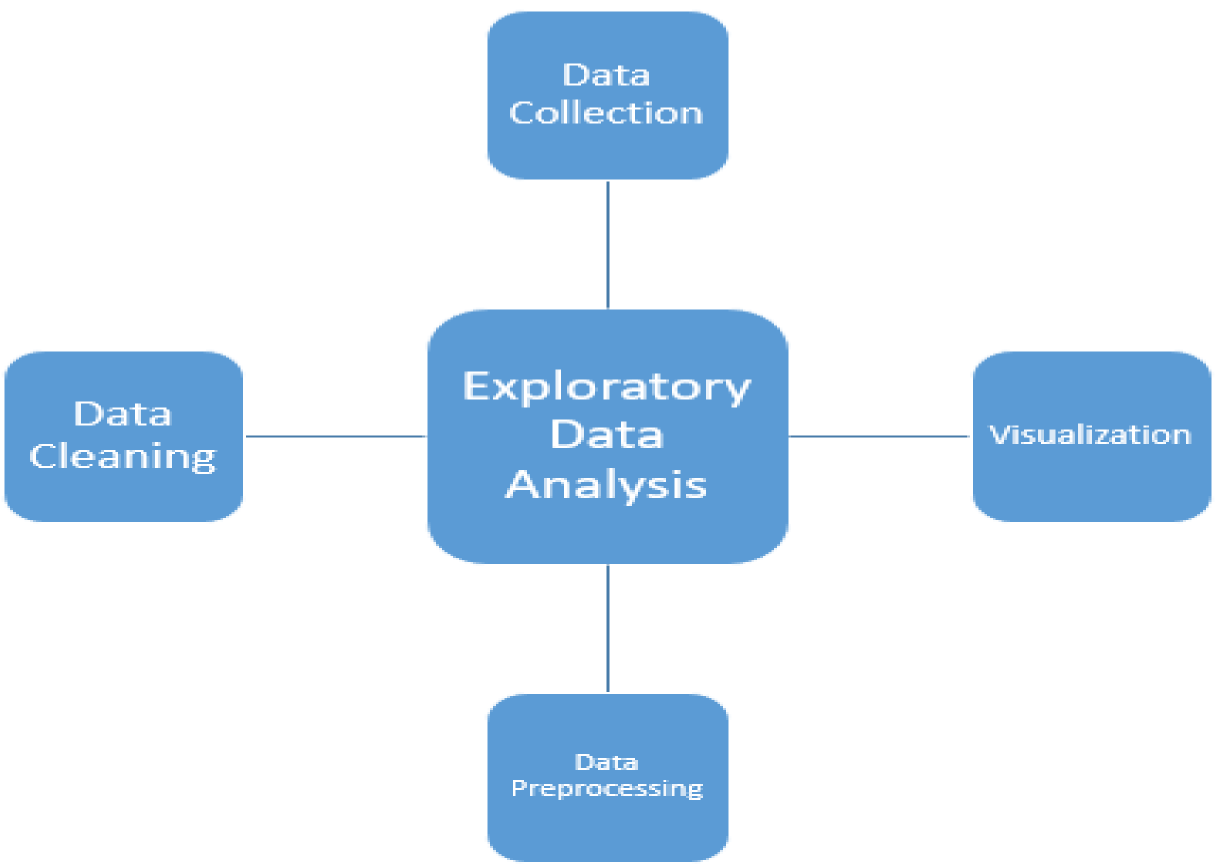
2. test.csv= Test data have **(200000 obs. Of 201 variables)**.

Target variable having two categories 0 and 1.

2.2 Aim

In this project, the task is to predict the value of **target** column. The datasets have numeric data fields.

EXPLORATORY DATA ANALYSIS (EDA)



In this section, we'll analysis how to use graphical and numerical techniques to begin uncovering the structure of your data. A predictive model requires that we look at the data before we start to create a model. However, in data mining, looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is known as Exploratory Data Analysis.

3.1 Data Collection: - In this we acquire all necessary information or overview about our data.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Columns: 202 entries, ID_code to var_199
dtypes: float64(200), int64(1), object(1)
memory usage: 308.2+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Columns: 201 entries, ID_code to var_199
dtypes: float64(200), object(1)
memory usage: 306.7+ MB
```

	target	var_0	var_1	var_2	var_3	var_4
count	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000
mean	0.100490	10.679914	-1.627622	10.715192	6.796529	11.078333
std	0.300653	3.040051	4.050044	2.640894	2.043319	1.623150
min	0.000000	0.408400	-15.043400	2.117100	-0.040200	5.074800
25%	0.000000	8.453850	-4.740025	8.722475	5.254075	9.883175
50%	0.000000	10.524750	-1.608050	10.580000	6.825000	11.108250
75%	0.000000	12.758200	1.358625	12.516700	8.324100	12.261125
max	1.000000	20.315000	10.376800	19.353000	13.188300	16.671400

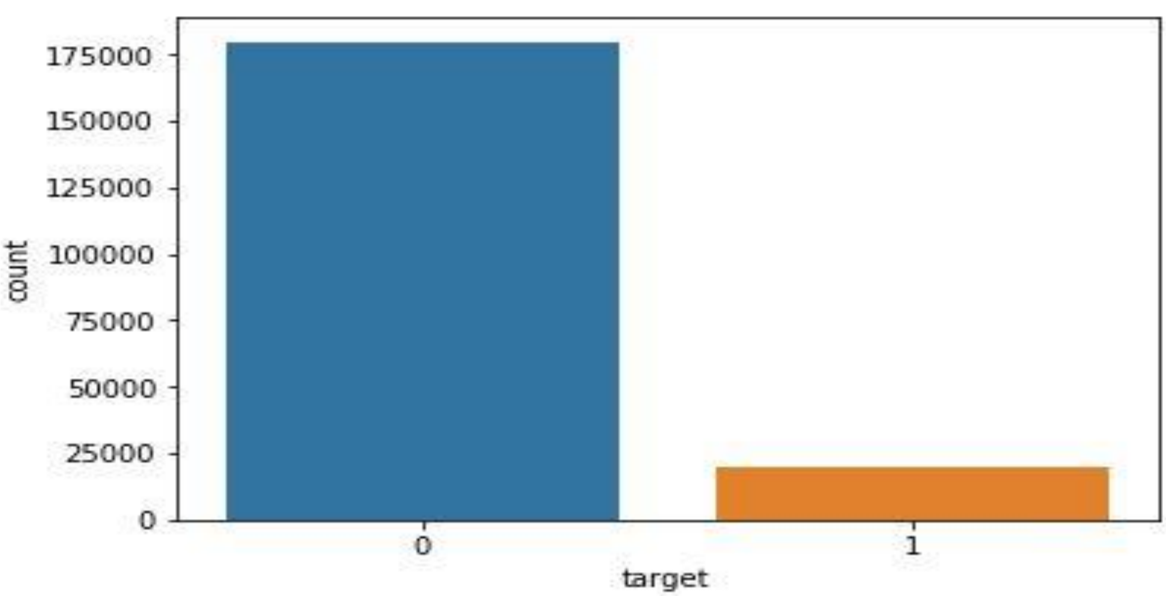
Here we can see; mean, count, standard deviation, min & max values, etc. of each column respectively.

We know that our target column is of two categories 0 and 1, so we can see distribution of our target variable on basis of target category.

```
target
0    179902
1     20098
Name: ID_code, dtype: int64
```

3.2 Visualization

Now in this section we see graphs and plots related to our data and analyze the behavior of data.



Here above we can see the distribution of our target column dividing among categories (factor 0 and 1). We can clearly see that our **data is imbalanced.**

We have **an unbalanced data**, where 90% of the data is the data of number of customers those will not make a transaction and 10% of the data is those who will make a transaction.

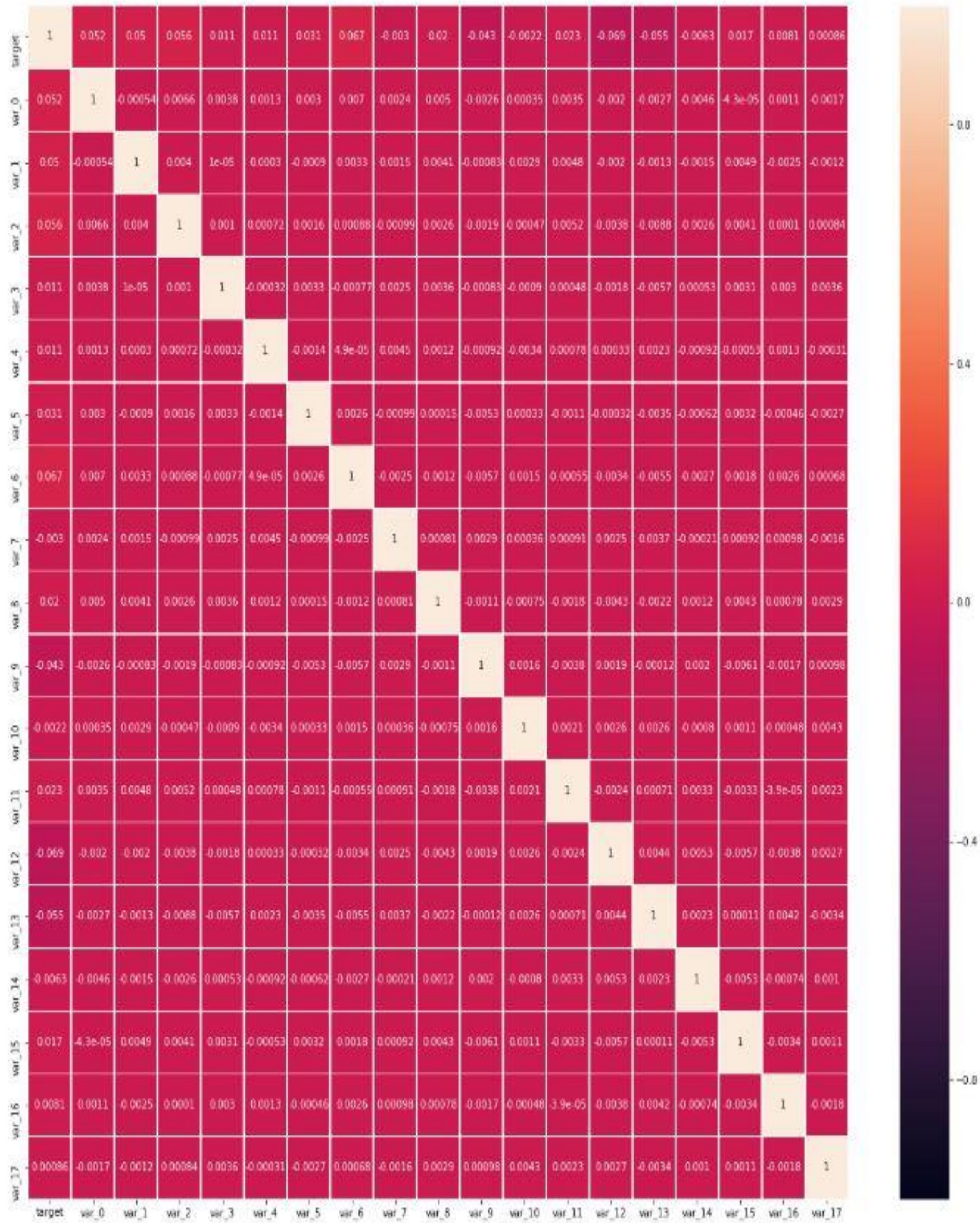
3.2.1 Correlation Plot (Collinearity)

Now we will see correlation plot. Correlation is usually defined as a measure of the linear relationship between two quantitative variables. Through this we can easily find those variables which are linearly correlated with each other.

```
No variable from the 200 input variables has collinearity problem.

The linear correlation coefficients ranges between:
min correlation ( var_91 ~ var_72 ):  -4.920909e-07
max correlation ( var_187 ~ var_4 ):  0.05669826
```

We have checked multi-collinearity of variables above and find that we didn’t have collinearity problem among variables. We can clearly see that min correlation is up to -20.32 and max correlation is of 0.0566.

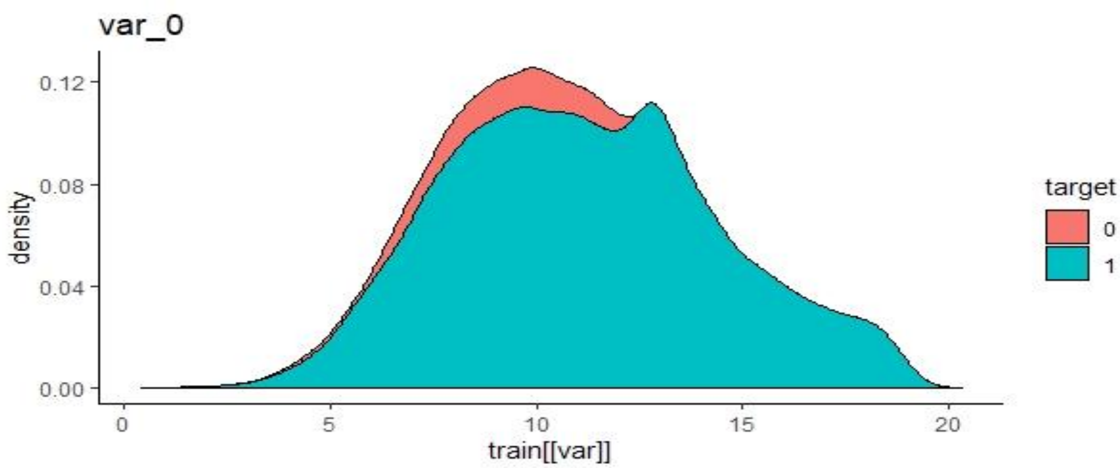
3.2.2 Visualization of Correlation Matrix

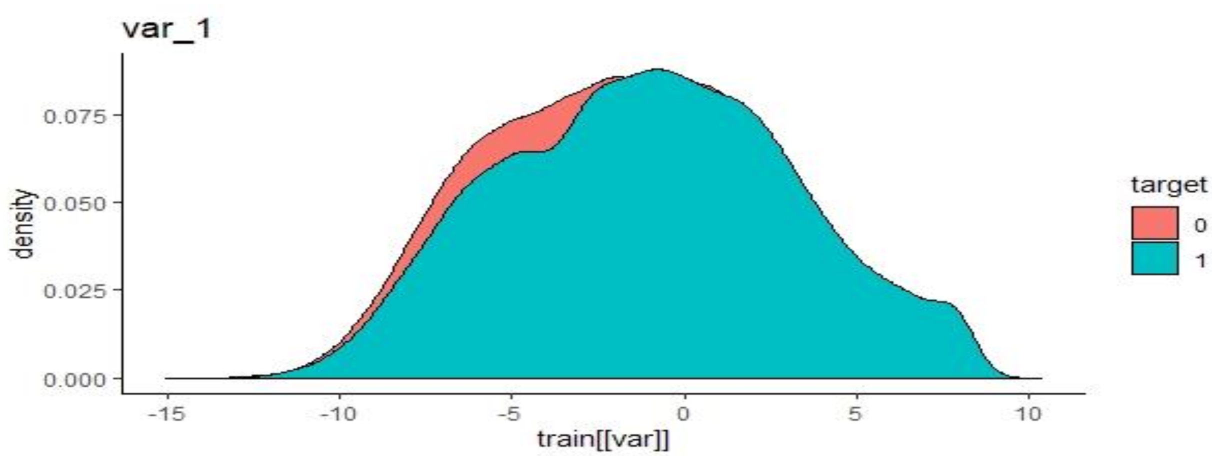


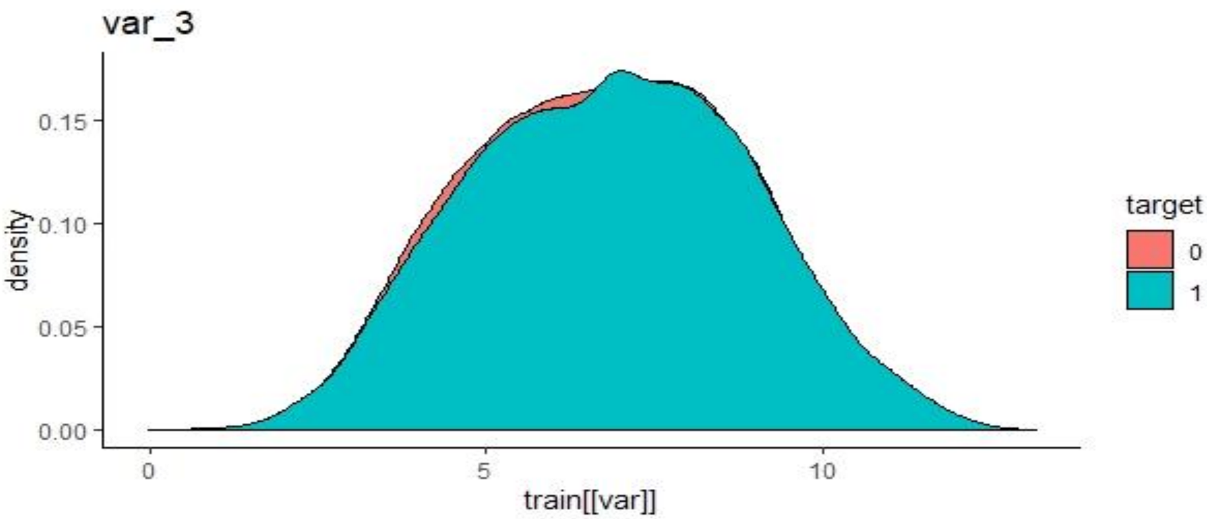
Due to multi-collinearity problem we have to delete some variables those found to be highly correlated, because these both variables contribute same information in explaining our target variable. Negatively correlated variation decreases while doing normalization.

Here earlier, we plot visualization of correlation matrix of 20 variables. Side color bar shows intensity of correlation among variables. Values inside box depicts correlation value.

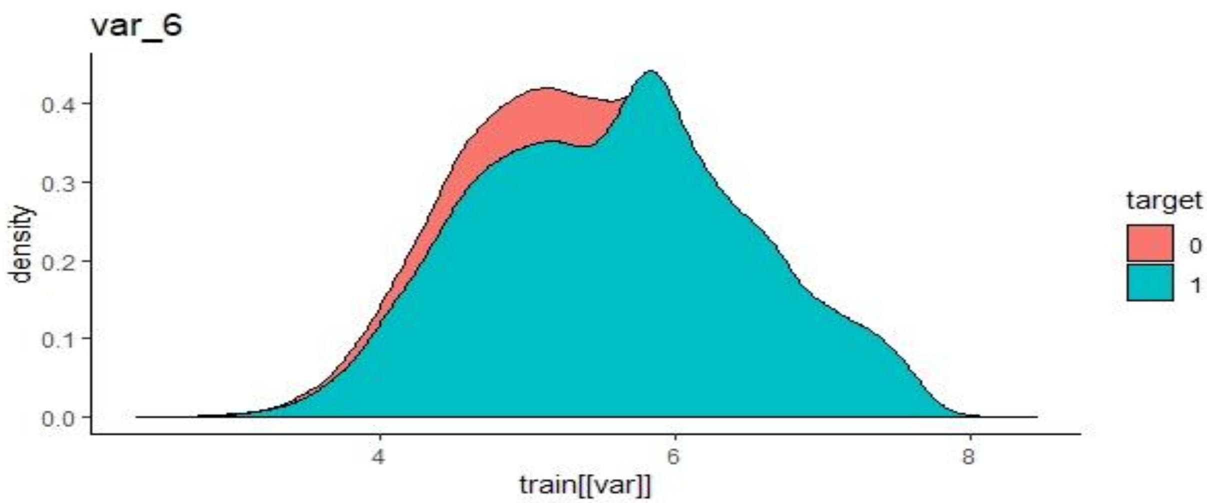
3.2.3 Distribution of train attributes



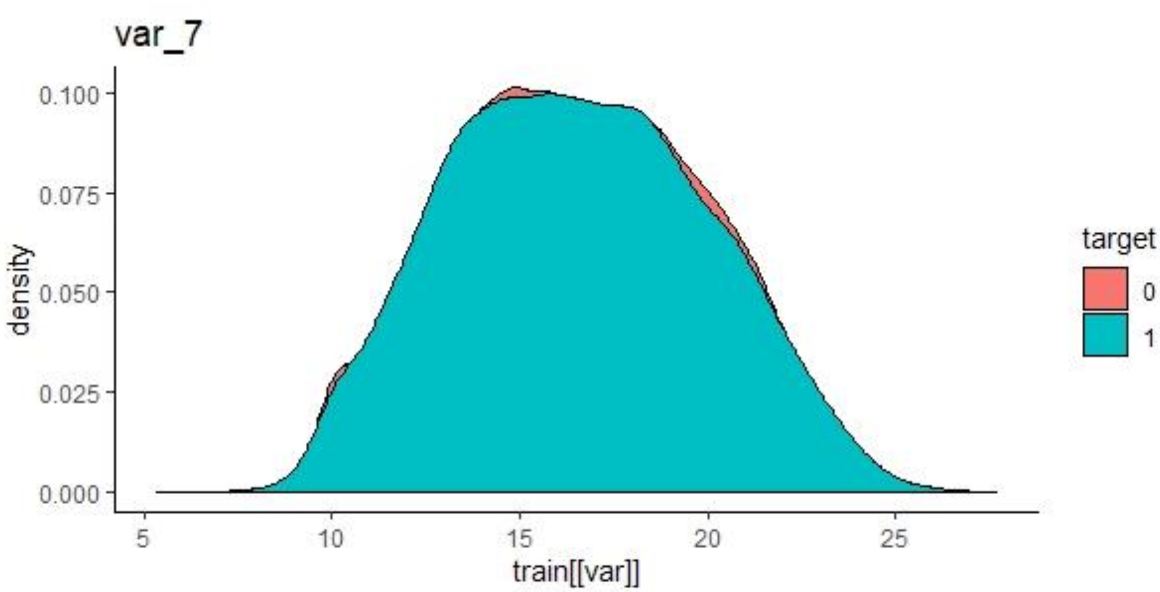




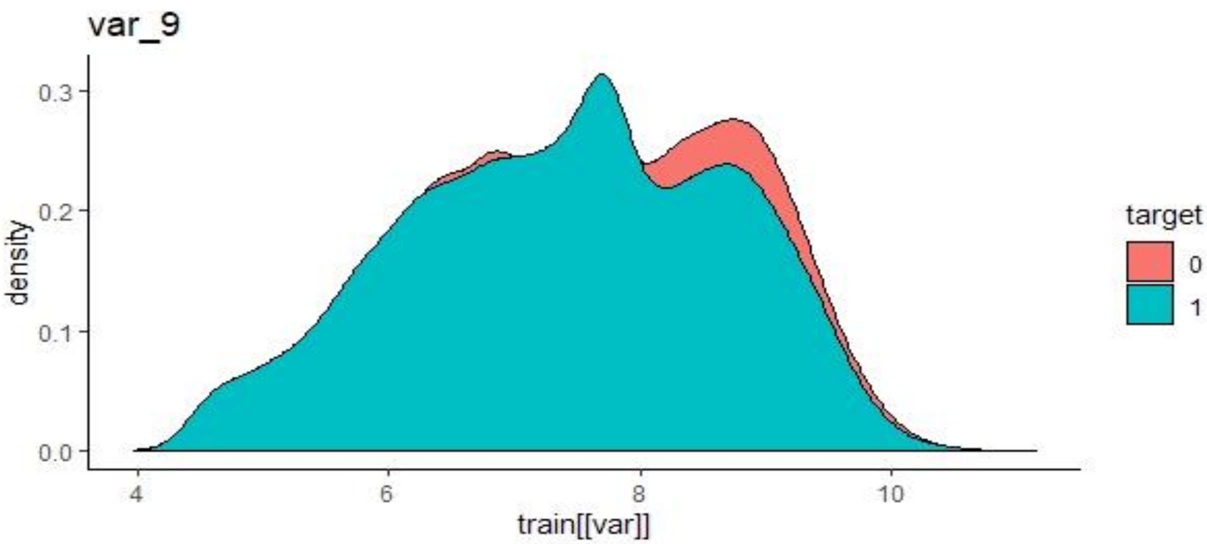
var_3
0.15
0.10
0.05
0.00
density
0
5
10
train[[var]]
target
0
1

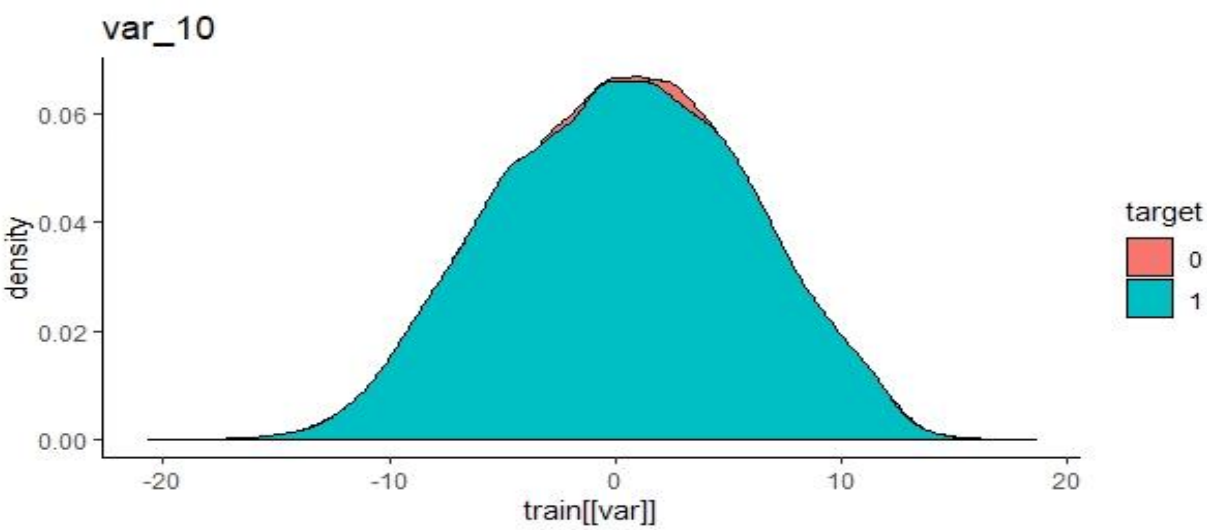


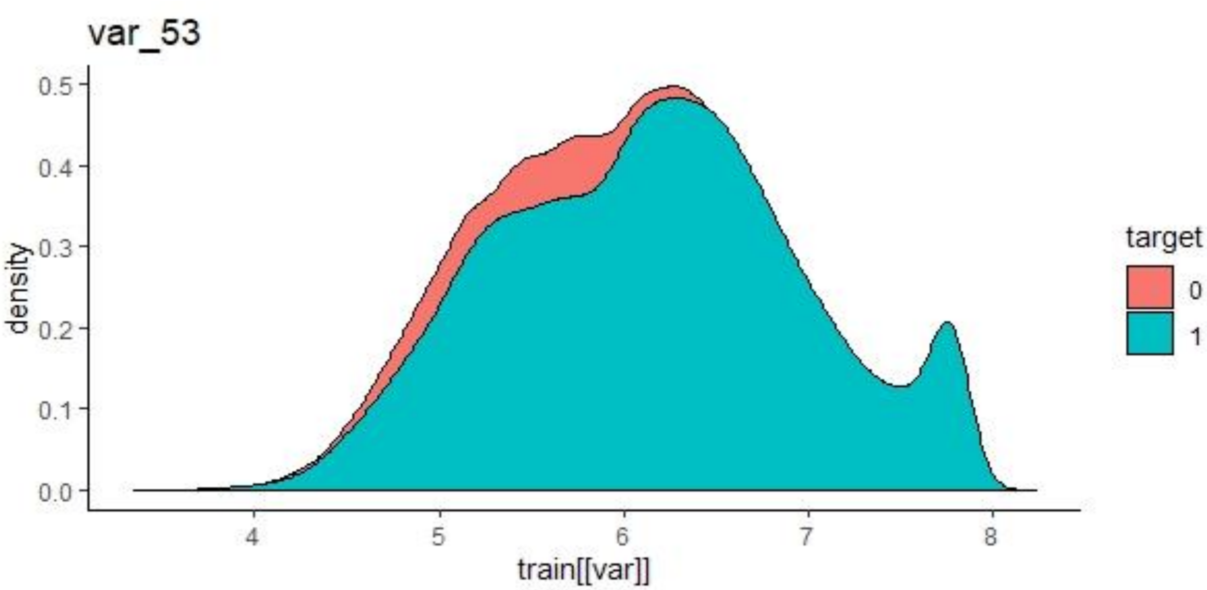
var_6
0.4
0.3
0.2
0.1
0.0
density
4
6
8
train[[var]]
target
0
1

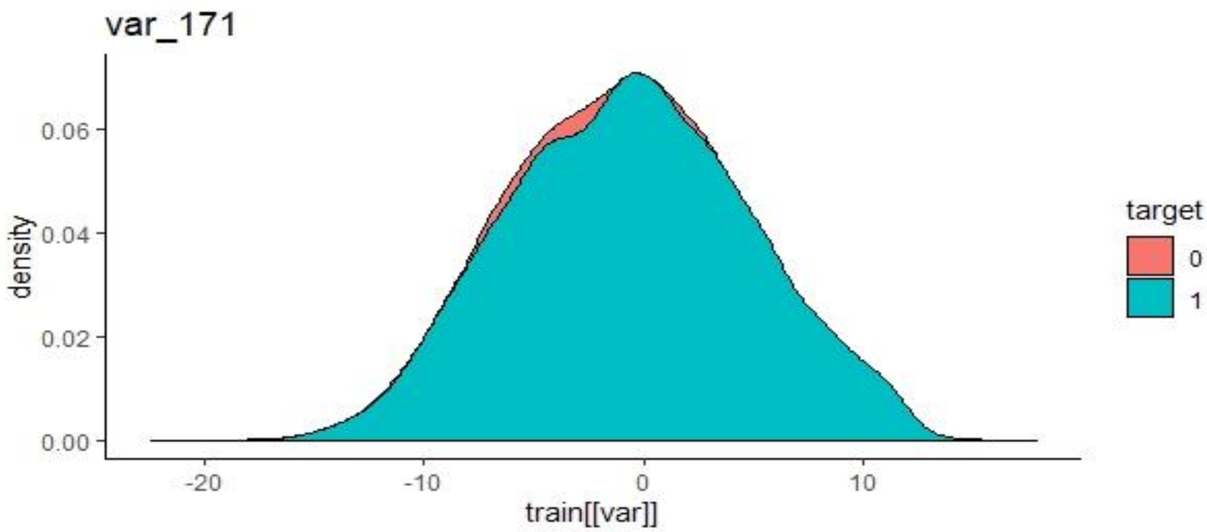


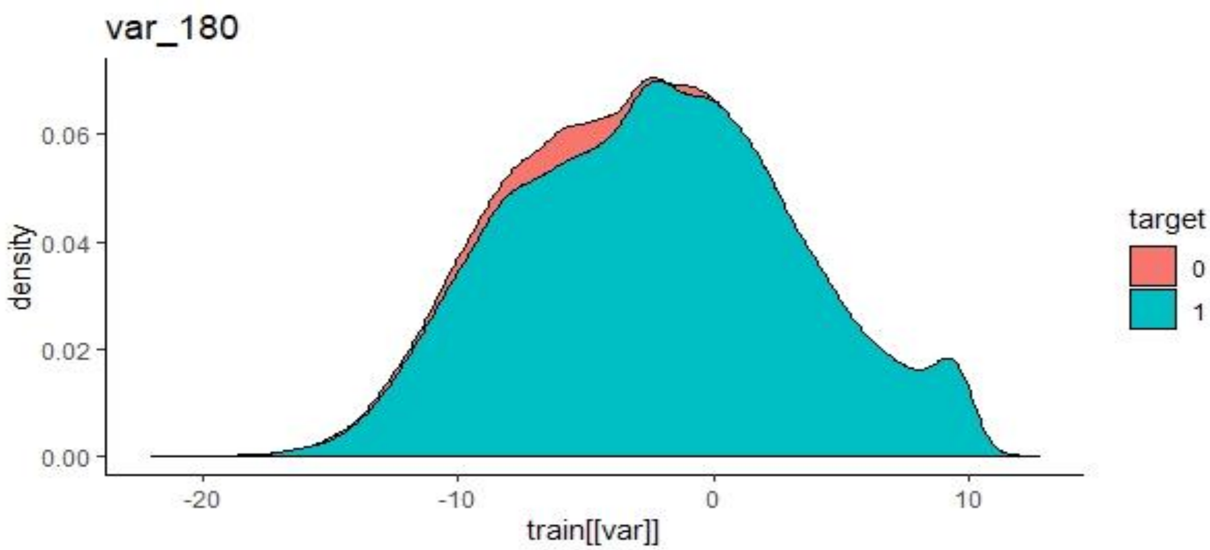
var_7
0.100
0.075
0.050
0.025
0.000
density
5
10
15
20
25
train[[var]]
target
0
1

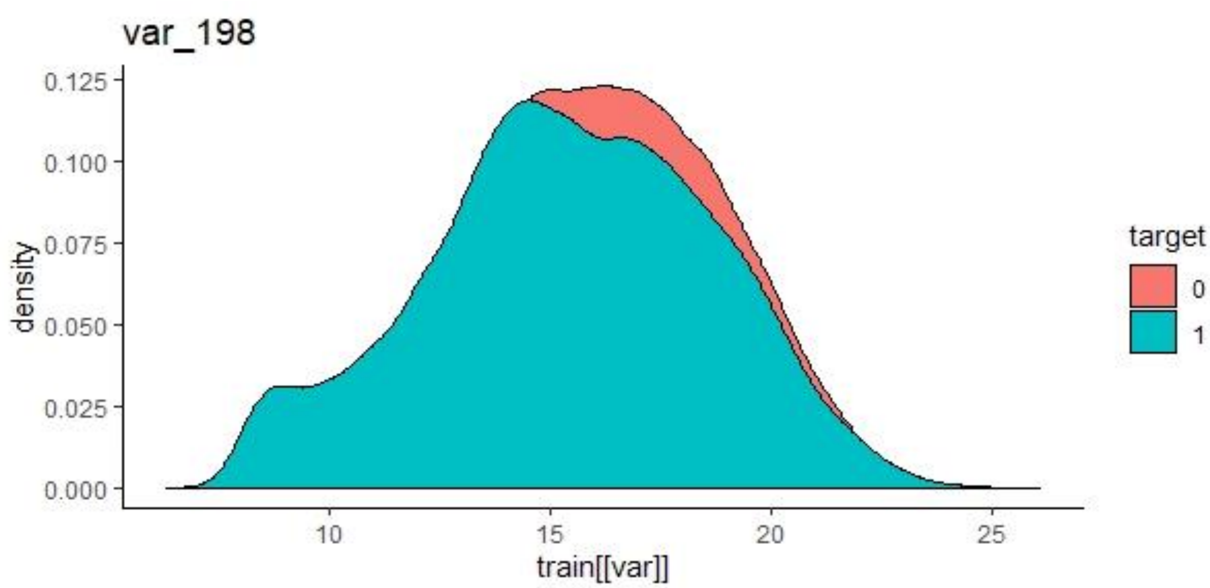

var_9
density
0.0
0.1
0.2
0.3
4
6
8
10
train[[var]]
target
0
1


var_10
density
0.00
0.02
0.04
0.06
-20
-10
0
10
20
train[[var]]
target
0
1


var_53
density
0.0
0.1
0.2
0.3
0.4
0.5
4
5
6
7
8
train[[var]]
target
0
1


var_171
density
0.06
0.04
0.02
0.00
-20
-10
0
10
train[[var]]
target
0
1

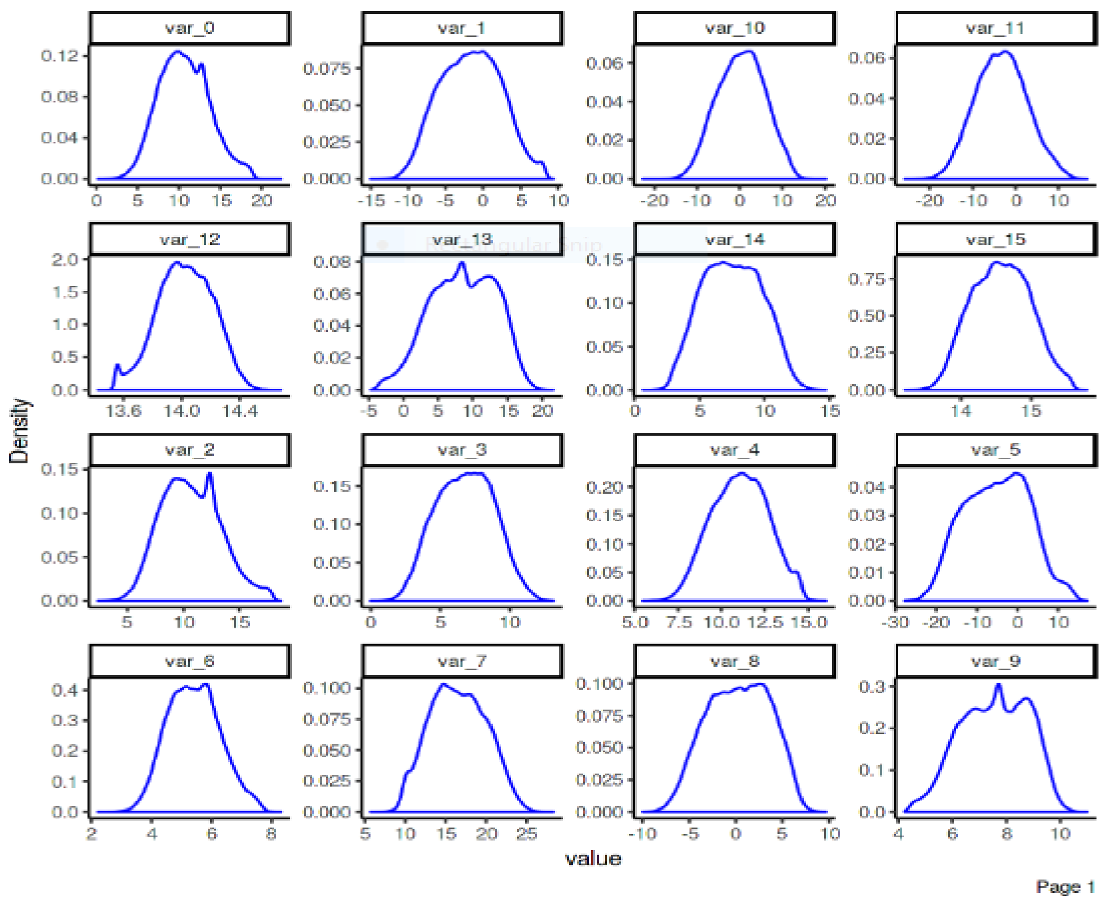

var_180
density
0.06
0.04
0.02
0.00
-20
-10
0
10
train[[var]]
target
0
1

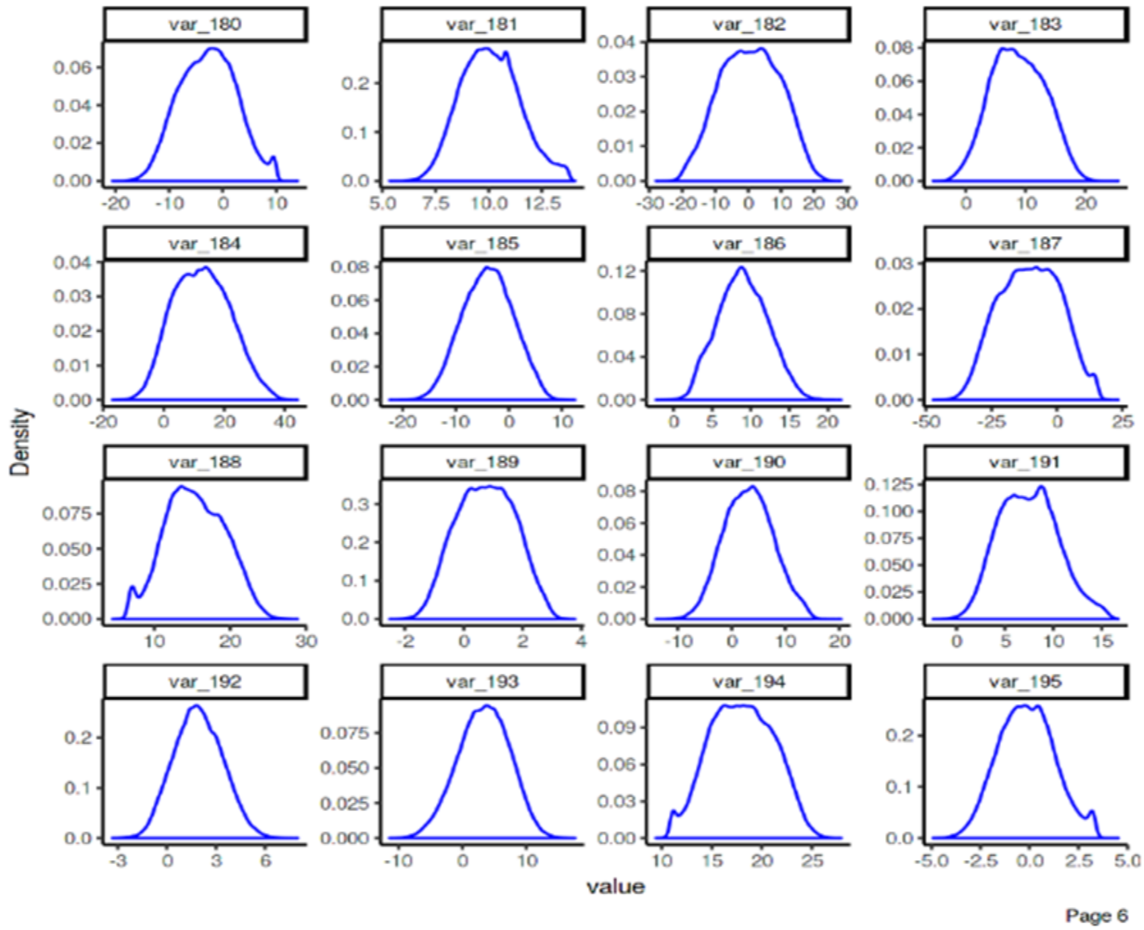

var_198
density
0.125
0.100
0.075
0.050
0.025
0.000
10
15
20
25
train[[var]]
target
0
1

What we observed:-

- We can observed that there is a considerable number of features which have significantly different distributions for two target variables. For example like var_0, var_1, var_9, var_198, var_180 etc.
- We can observed that there is a considerable number of features which have significantly same distributions for two target variables. For example like var_3, var_7, var_10, var_171, var_185 etc.

3.2.4 Distribution of test attributes





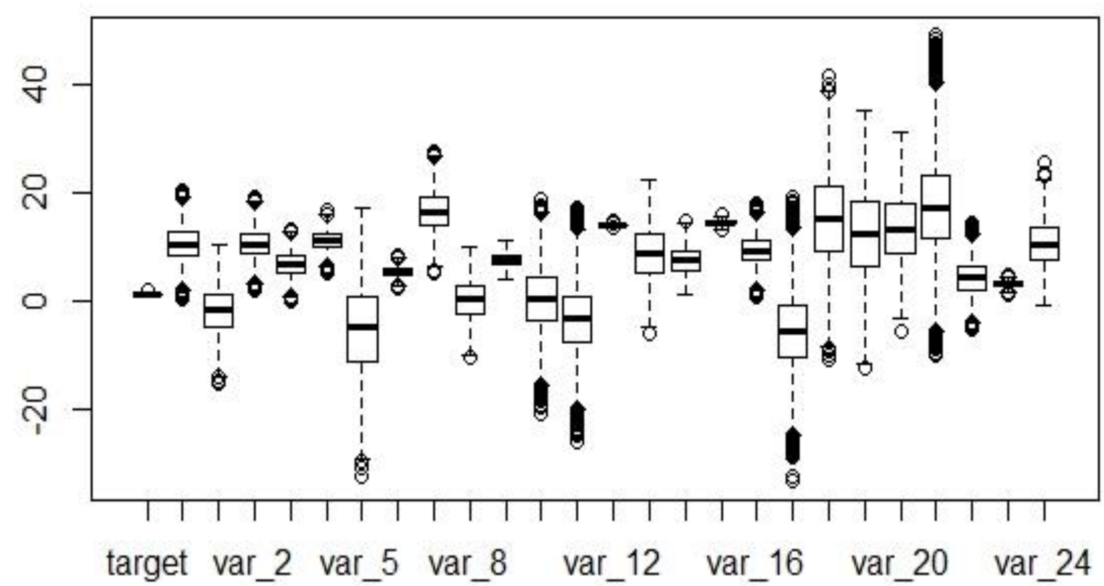
What we observed:-

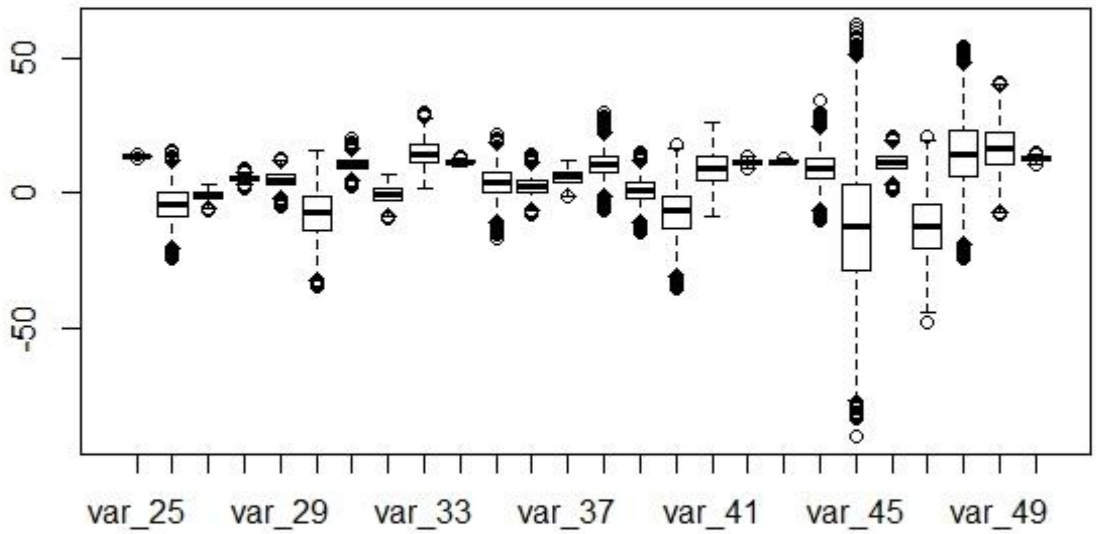
- We can observed that there is a considerable number of features which have significantly different distributions. For example like var_0, var_1, var_9, var_180, var_198 etc.
- We can observed that there is a considerable number of features which have significantly same distributions. For example like var_3, var_7, var_10, var_171, var_185, var_192 etc.

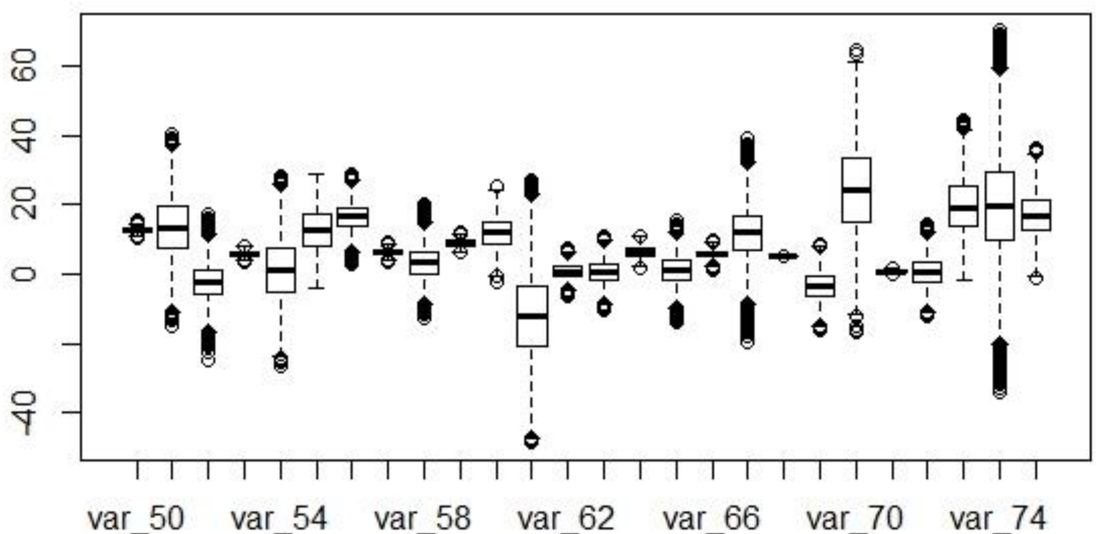
3.3 Data Pre-processing

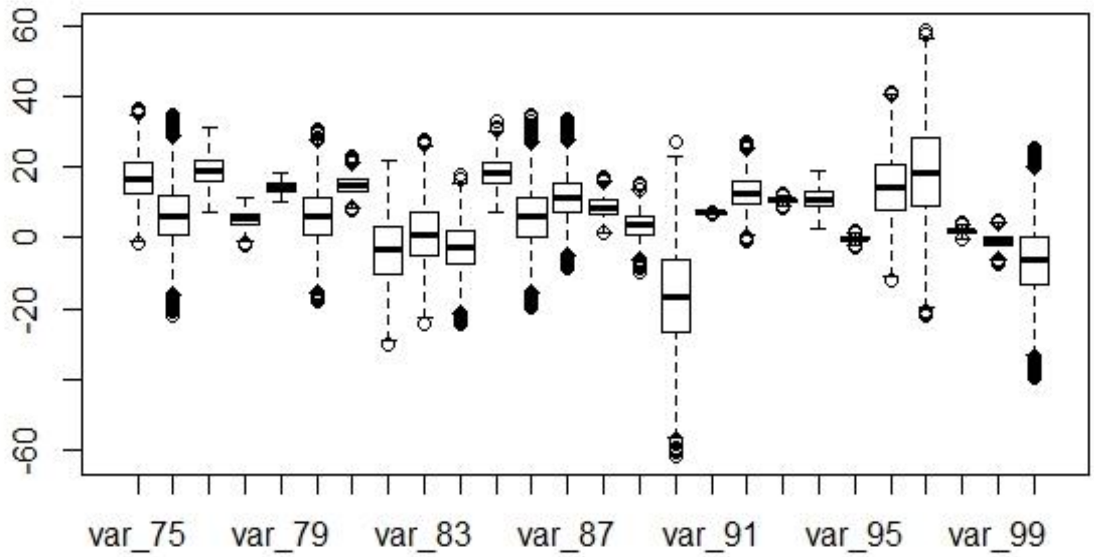
On analyzing data, we can see that our data is imbalanced and have some impurities in it. **On analyzing I came to know that the data is outlined and have outliers. This is the data of transactional/computational history and in these type of data some values might seem to be an outlier but contribute some values in model developing. So I decided not to remove outliers and proceed** with what data I was given.

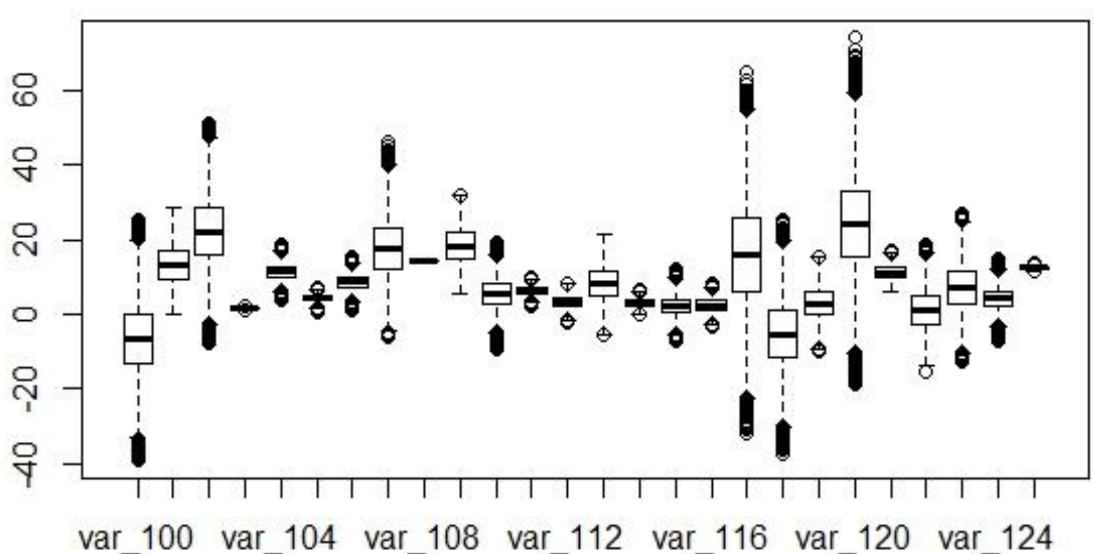
3.3.1 Outliers

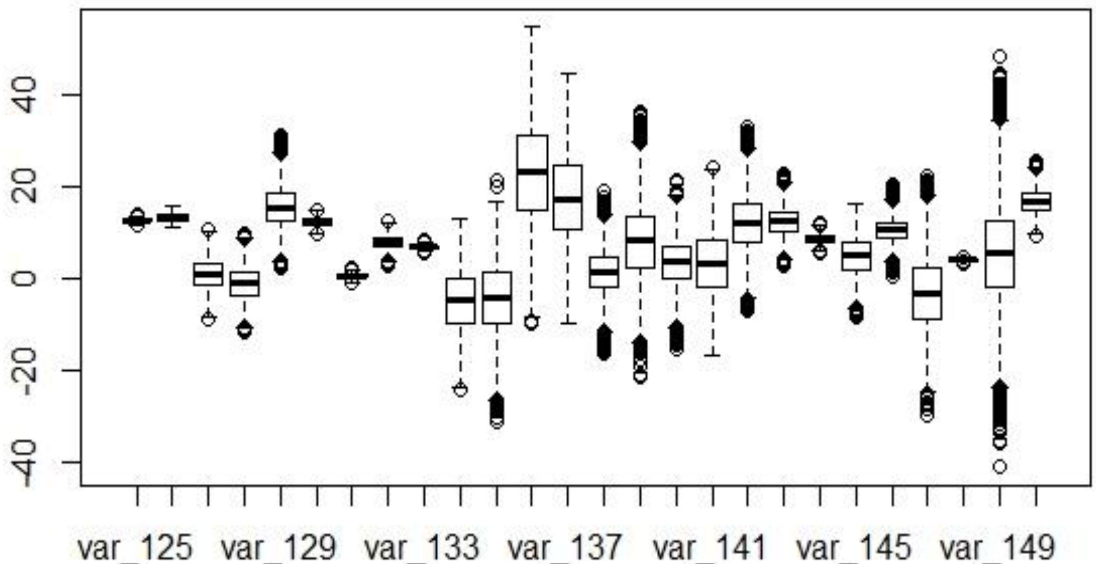


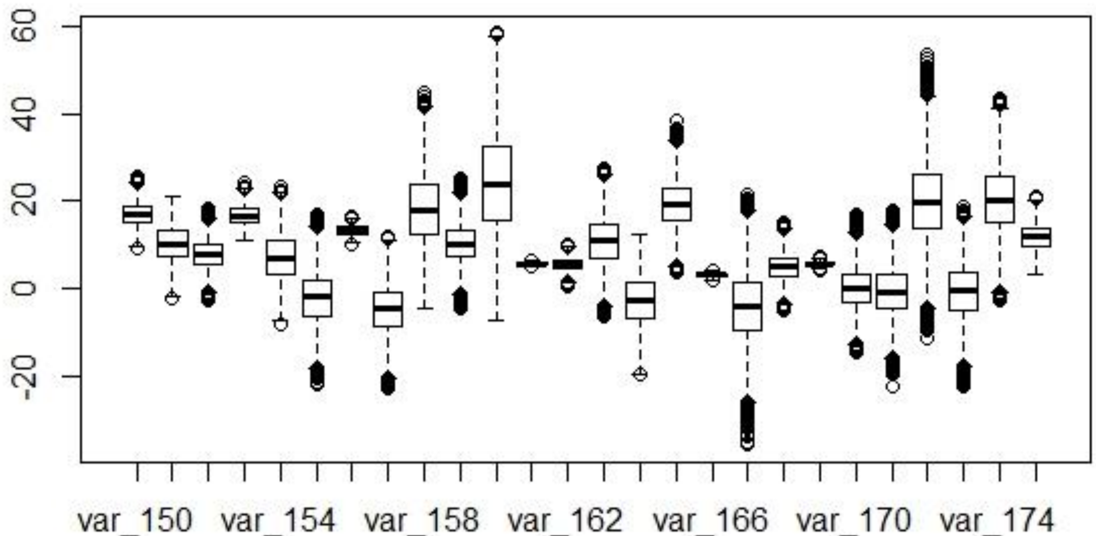

50
0
-50
var_25
var_29
var_33
var_37
var_41
var_45
var_49

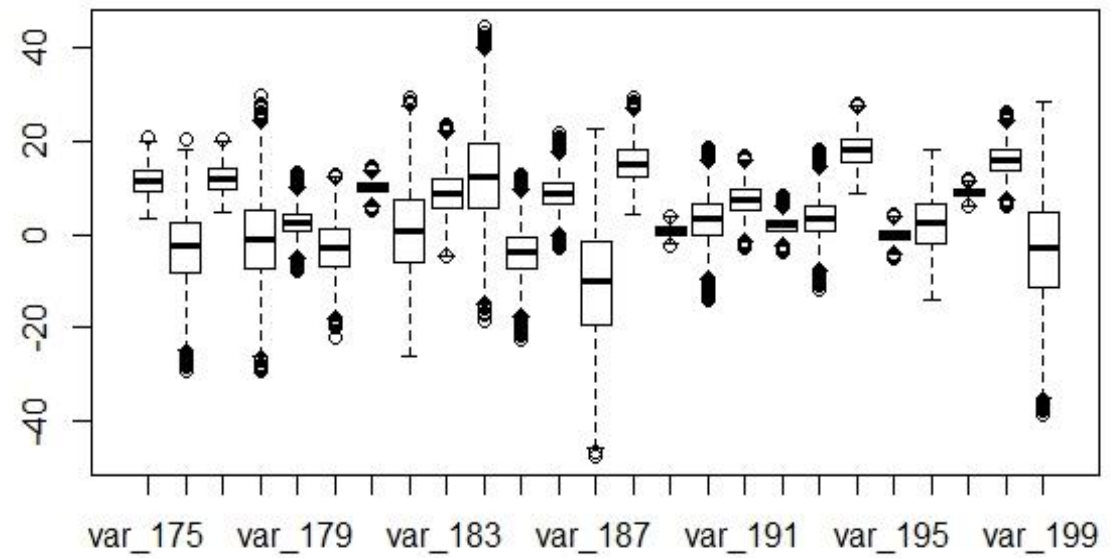

60
40
20
0
-40
var_50
var_54
var_58
var_62
var_66
var_70
var_74


60
40
20
0
-20
-60
var_75
var_79
var_83
var_87
var_91
var_95
var_99


60
40
20
0
-20
-40
var_100
var_104
var_108
var_112
var_116
var_120
var_124


40
20
0
-20
-40
var_125
var_129
var_133
var_137
var_141
var_145
var_149


60
40
20
0
-20
var_150
var_154
var_158
var_162
var_166
var_170
var_174



3.3.2 Missing Value Analysis

On computation we see that, there is no missing value in our datasets.

```
missing_val_1=pd.DataFrame(train.isnull().sum())
missing_val_2=pd.DataFrame(test.isnull().sum())

sum(missing_val_1),sum(missing_val_2)
```

```
(0, 0)
```

But if we consider outliers and apply operations for outlier removal, then we have NA’s (missing values).But that is not our case, so I proceed without NA’s.

```
> sum(is.na(train))
[1] 27735
> sum(is.na(test))
[1] 28389
```

FEATURE ENGINEERING

In this section, I want to extract insights from models with the help of model development. The Goal behind of feature engineering for Santander is:

1. All features are senseless named (var_0, var_1, var2,...) but certainly the importance of each one is different.
2. Extract insights from models.
3. Find the important feature in models.
4. Effect of each feature on the model's predictions.

4.1 Permutation Importance

What features have the biggest impact on predictions?

This concept is called **feature importance**. There are multiple ways to measure feature importance. Here we discuss permutation importance. Compared to most other approaches, permutation importance is:

- Fast to calculate
- Widely used and understood
- Consistent with properties we would want a feature importance measure to have

How it works?

Permutation importance uses models differently than anything other method. It shuffle the data and then remove different input variables to see what relative change results in the calculating the training model. It measures how much the outcome goes up or down given the input variable, thus calculating their impact on the results.

Permutation importance is calculated after a model has been fitted.

We use eli5 library, from eli5.sklearn import PermutationImportance. This will show the variables ranked up and down with their participated weights in the model prediction.

Weight	Feature
0 ± 0.0000	var_124
0 ± 0.0000	var_155
0 ± 0.0000	var_153
0 ± 0.0000	var_152
0 ± 0.0000	var_37
0 ± 0.0000	var_38
0 ± 0.0000	var_39
0 ± 0.0000	var_147
0 ± 0.0000	var_40
0 ± 0.0000	var_145
0 ± 0.0000	var_42
0 ± 0.0000	var_142
0 ± 0.0000	var_43
0 ± 0.0000	var_140
0 ± 0.0000	var_138
0 ± 0.0000	var_156
0 ± 0.0000	var_137
0 ± 0.0000	var_50
0 ± 0.0000	var_51
0 ± 0.0000	var_131
0 ± 0.0000	var_54
0 ± 0.0000	var_129
0 ± 0.0000	var_127
0 ± 0.0000	var_126
0 ± 0.0000	var_57
0 ± 0.0000	var_198
0 ± 0.0000	var_123
0 ± 0.0000	var_122
0 ± 0.0000	var_121
0 ± 0.0000	var_60
0 ± 0.0000	var_119
0 ± 0.0000	var_47
0 ± 0.0000	var_61

-0.0000 ± 0.0000	var_193
-0.0000 ± 0.0000	var_186
-0.0000 ± 0.0000	var_103
-0.0000 ± 0.0000	var_102
-0.0000 ± 0.0000	var_106
-0.0000 ± 0.0000	var_2
-0.0000 ± 0.0000	var_13
-0.0000 ± 0.0000	var_120
-0.0000 ± 0.0000	var_99
-0.0000 ± 0.0000	var_160
-0.0000 ± 0.0000	var_65
-0.0000 ± 0.0000	var_66
-0.0000 ± 0.0000	var_67
-0.0000 ± 0.0000	var_56
-0.0000 ± 0.0000	var_52
-0.0000 ± 0.0000	var_46
-0.0000 ± 0.0000	var_44
-0.0000 ± 0.0000	var_74
-0.0000 ± 0.0000	var_41
-0.0000 ± 0.0000	var_78
-0.0000 ± 0.0000	var_164
-0.0000 ± 0.0000	var_35
-0.0000 ± 0.0000	var_84
-0.0000 ± 0.0000	var_151
-0.0000 ± 0.0000	var_133
-0.0000 ± 0.0000	var_132
-0.0000 ± 0.0000	var_30
-0.0000 ± 0.0000	var_28
-0.0000 ± 0.0000	var_154
-0.0000 ± 0.0000	var_130
-0.0000 ± 0.0000	var_157
-0.0000 ± 0.0000	var_128
-0.0000 ± 0.0000	var_25
-0.0000 ± 0.0000	var_34
-0.0000 ± 0.0000	var_14
-0.0000 ± 0.0000	var_0
-0.0000 ± 0.0000	var_29
… 50 more …	

Here left side bar shows variables having positive weights and ranked above while variables shown on right side (in red color bar) having weights in negative quantity ranked below.

Variable Importance based on Mean Gini Index

We implement variable importance after developing random forest model on base of mean gini index.

Importance(x, type=NULL, class=NULL, scale=TRUE,)

Arguments:

x = an object of class (model used i.e. randomForest)

type = either 1 or 2 (1 = mean decrease in accuracy, 2 = mean decrease in node impurity)

class = for classification problem, which class- specific measure to return)

scale = for permutation based measures

```
        MeanDecreaseGini
var_0          182.99282
var_1          171.28617
var_2          184.00243
var_3          112.19883
var_4          116.91288
var_5          146.11268
var_6          201.00502
var_7          107.52188
var_8          112.65563
var_9          163.64656
var_10         102.57936
var_11         115.22529
var_12         274.27533
var_13         186.92434
var_14         103.14343
var_15         114.28493
var_16         114.13036
var_17         105.34615
var_18         158.21214
var_19         111.93889
var_20         115.91428
```

```
var_180        144.36509
var_181        108.81400
var_182        109.74257
var_183        107.37021
var_184        156.48210
var_185        102.63720
var_186        114.40558
var_187        111.27304
var_188        144.54885
var_189        105.49889
var_190        165.19751
var_191        158.23979
var_192        124.56205
var_193        111.63502
var_194        122.80724
var_195        120.92841
var_196        119.49852
var_197        140.96359
var_198        192.74122
var_199        106.36831
>
```

MODEL DEVELOPMENT

As I told you earlier that, it is a classification problem, so according to the instructions mentioned, I used logistic regression first followed by other algorithms both in R and Python to derive the outcomes.

#LOGISTIC REGRESSION

Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

NOTE: I use random sampling for all model development and in the ratio of 80:20 (train: test).

I develop model, test its predictions with test dataset to have confusion matrix.

R	**Python**
Accuracy = 91.40%	Accuracy = 91.56%
FNR = 29.81%	FNR = 72.35%
Precision = 27.07%	Precision = 68.47%
Recall (Sensitivity) = 70.18%	Recall = 27.64%
AUC score = 0.6138	AUC score = 0.6312

```
Confusion Matrix and Statistics

   LGR_Predictions
        0     1
  0 35454   469
  1  2973  1104

               Accuracy : 0.914
                 95% CI : (0.9112, 0.9167)
    No Information Rate : 0.9607
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3541

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9226
            Specificity : 0.7018
         Pos Pred Value : 0.9869
         Neg Pred Value : 0.2708
             Prevalence : 0.9607
         Detection Rate : 0.8863
   Detection Prevalence : 0.8981
      Balanced Accuracy : 0.8122

       'Positive' Class : 0
```

R

In R, I used 3 algorithms named Random Forest, Naïve Bayes and SVM respectively.

1) RANDOM FOREST

Random Forest is a trademark term for an ensemble of decision trees. In Random Forest, we’ve collection of decision trees (so known as “Forest”). To classify a new object based on attributes, each tree gives a classification and we say the tree “votes” for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

I chose Random Forest because it is an ensemble method of constructing multiple Decision Trees at one place. The results that I will get in constructing single DT models, I will have those results collectively by Random Forest. The average votes of several DT’s is inherently less noisy and less susceptible to outliers than a single DT output.

R	**Python**
Accuracy = 89.98%	Accuracy = 90.05%
FNR = 0%	FNR = 99.97%
Precision = 0.019%	Precision = 100%
Recall = ∞	Recall = 2.51
AUC Score = 0.4986	AUC Score = 50.01

```
Confusion Matrix and Statistics

   RF_Predictions
        0     1
  0 44988     0
  1  5011     1

               Accuracy : 0.8998
                 95% CI : (0.8971, 0.9024)
    No Information Rate : 1
    P-Value [Acc > NIR] : 1

                  Kappa : 4e-04

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8997780
            Specificity : 1.0000000
         Pos Pred Value : 1.0000000
         Neg Pred Value : 0.0001995
             Prevalence : 0.9999800
         Detection Rate : 0.8997600
   Detection Prevalence : 0.8997600
      Balanced Accuracy : 0.9498890

       'Positive' Class : 0
```

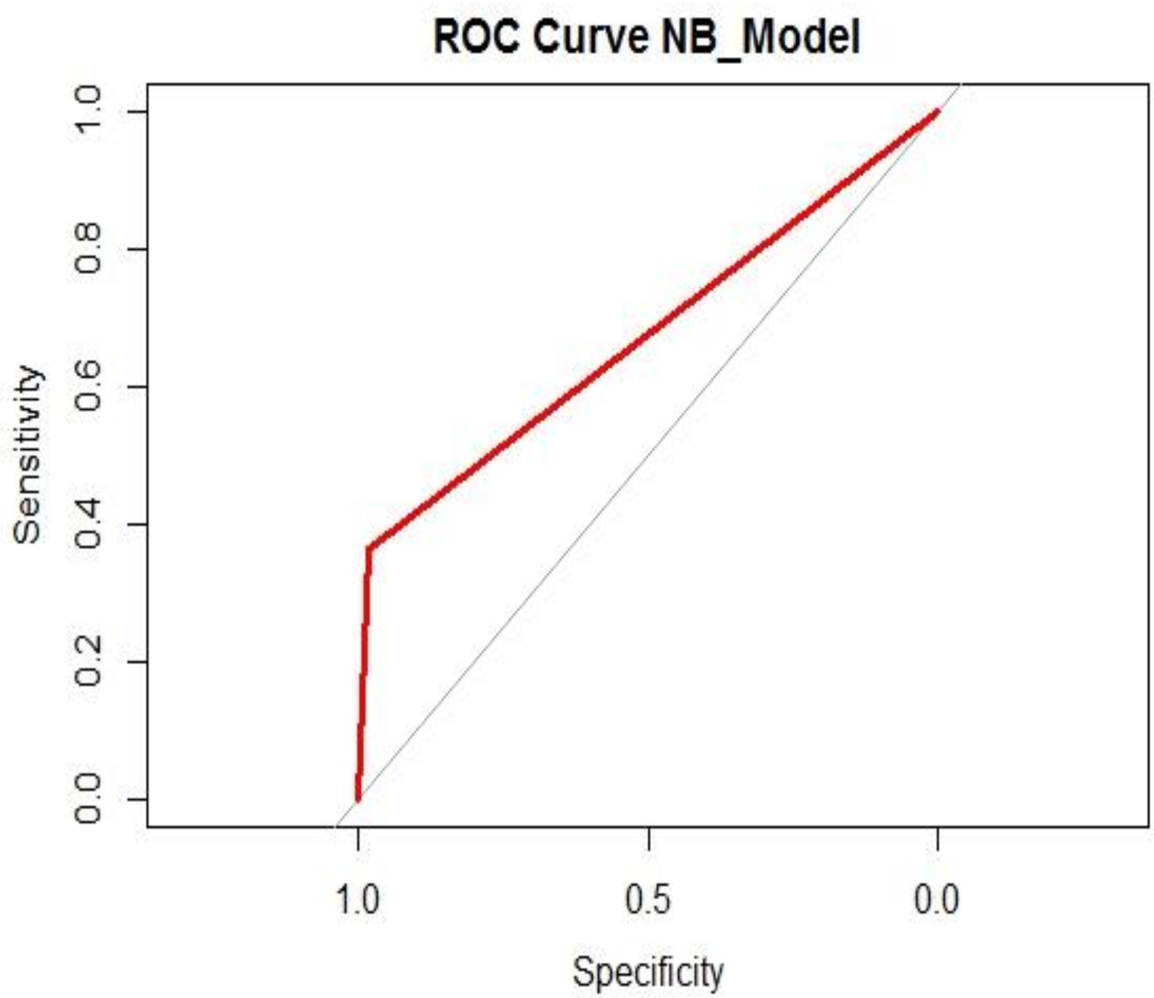
2) NAÏVE BAYES

It is a classification technique based on Bayes' theorem with an assumption of independence between predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, a naive Bayes classifier would consider all of these properties to independently contribute to the probability that this fruit is an apple.

I chose this model because it is easy to build and particularly useful for very large data sets. Along with simplicity, it is known to outperform even highly sophisticated classification methods.

```
Confusion Matrix and Statistics

        predicted
observed     0     1
       0 35431   580
       1  2509  1480

               Accuracy : 0.9228
                 95% CI : (0.9201, 0.9254)
    No Information Rate : 0.9485
    P-Value [Acc > NIR] : 1

                  Kappa : 0.4521

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9339
            Specificity : 0.7184
         Pos Pred Value : 0.9839
         Neg Pred Value : 0.3710
             Prevalence : 0.9485
         Detection Rate : 0.8858
   Detection Prevalence : 0.9003
      Balanced Accuracy : 0.8262

       'Positive' Class : 0
```

R	**Python**
Accuracy = 92.28%	Accuracy = 92.15%
FNR = 28.15%	FNR = 63.58%
Precision = 37.10%	Precision = 69.52%
Recall = 71.84%	Recall = 36.41%
AUC Score = 0.6726	AUC Score = 0.6736



3) SVM – STATE VECTOR MACHINE

It is a classification method. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.

For example, if we only had two features like Height and Hair length of an individual, we'd first plot these two variables in two dimensional space where each point has two co-ordinates (these co-ordinates are known as **Support Vectors**).

Now, we will find some *line* that splits the data between the two differently classified groups of data. This will be the line such that the distances from the closest point in each of the two groups will be farthest away.

NOTE: Think of this algorithm as playing JezzBall (Microsoft's game) in n-dimensional space.

First I'm thinking of using KNN Imputation, but then I realized that KNN is sensitive to outliers and will not produce satisfactory results. So I chose SVM in place of KNN as it is faster than KNN and produces good results too. It is also effective on large datasets, but one thing to keep in mind is that the data should be normalized or scaled before putting into SVM. SVM is good with outliers as it will only use the most relevant points to find a linear separation (support vectors). It is able to find the linear separation that should exist.

I took 1lac observations from the train data of 2lac obs.

Accuracy = 95.10

FNR = 0.951%

Precision = 51.79%

Recall = 99.04%

AUC Score = 0.7586

```
Confusion Matrix and Statistics

        predicted
observed     0     1
       0 17980    10
       1   969  1041

               Accuracy : 0.951
                 95% CI : (0.948, 0.954)
    No Information Rate : 0.9474
    P-Value [Acc > NIR] : 0.01113

                  Kappa : 0.6565

 Mcnemar's Test P-Value : < 2e-16

            Sensitivity : 0.9489
            Specificity : 0.9905
         Pos Pred Value : 0.9994
         Neg Pred Value : 0.5179
             Prevalence : 0.9475
         Detection Rate : 0.8990
   Detection Prevalence : 0.8995
      Balanced Accuracy : 0.9697

       'Positive' Class : 0
```

PYTHON

In Python, I used 4 algorithms apart from Logistic Regression taking into account. Two algorithms were same as I used earlier in R (i.e. Random Forest & Naïve Bayes)

1) RANDOM FOREST

I tried building both models i.e. CART and C5.0 model in Random Forest, but I prefer CART model because its attributes are good that I got. Attributes mentioned earlier in R section.

2) NAÏVE BAYES

Attributes of Naïve Bayes collected from Python mentioned earlier.

3) CAT BOOST

"CatBoost" name comes from two words "**Cat**egory" and "**Boost**ing". CatBoost is a recently open-sourced machine learning algorithm from Yandex. It can work with diverse data types to help solve a wide range of problems that businesses face today. To top it up, it provides best-in-class accuracy. It is especially powerful in two ways:

- It yields state-of-the-art results without extensive data training typically required by other machine learning methods, and
- Provides powerful out-of-the-box support for the more descriptive data formats that accompany many business problems.

I chose CatBoost because it is easy to use and it does not require conversion of data set to any specific format like XGBoost and LightGBM. It reduces the need for extensive hyper-parameter tuning and lower the chances of over fitting also which leads to more generalized models.

Accuracy = 92.40%

FNR = 65.71%

Precision = 78.70%

Recall = 34.28%

AUC Score = 0.6662

4) DECISION TREE

It is a type of supervised learning algorithm that is mostly used for classification problems. Surprisingly, it works for both categorical and continuous dependent variables. In this algorithm, we split the population into two or more homogeneous sets. This is done based on most significant attributes/ independent variables to make as distinct groups as possible.

Tree based learning algorithms are considered to be one of the best and mostly used supervised learning methods. Tree based methods empower predictive models with high accuracy, stability and ease of interpretation.

I chose DT because the results I got from the Random Forest model were not satisfactory, or I would say not explanatory. Because of two main attributes FNR & Recall, that were not good in Random Forest, so I decided to run the DT Model.

Accuracy = 83.31%

FNR = 79.20%

Precision = 19.87%

Recall = 20.79%

AUC Score = 0.5561

CONCLUSION

Now we have a couple of models for predicting the target variable, we need to decide which one to choose. There are several criteria/parameters that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

In our case of Transaction prediction data, Interpretability and Computational Efficiency don not hold much significance. Therefore, we will use Predictive Performance as the criteria to compare and evaluate models.

Predictive Performance can be measured by comparing Predictions of the models with real values of the target variables, and calculate some average error metrics measures.

I will decide best model on the basis of the attributes shown above with respect to each model evaluation, i.e. Accuracy, FNR, Precision, Recall and AUC Score. We need to take care of model with higher recall value and AUC Score with lower FNR value, we can compromise a little bit in with the values of Accuracy. Our need is to classify customers who will make transactions (i.e. correctly identifying positives).

ROC – AUC

AUC – ROC curve is a performance measurement for classification problem at various thresholds settings. ROC is a probability curve and AUC represents degree or measure of separability. It tells you how much model is capable of distinguishing between classes. Higher the AUC, better the model is.

An excellent model has AUC near to the 1 which means it has good measure of separability. A poor model has AUC near to the 0.

For R

The accuracy standards between Naïve Bayes and SVM is somewhat similar, but SVM gave me extra ordinary results in whole model building, both in R and Python.

So I chose **SVM** in R for model building. As I took sample of 1lac observations out of 2lac and I got these results.

I’m sure when we develop model on whole dataset, it will also outrun as well.

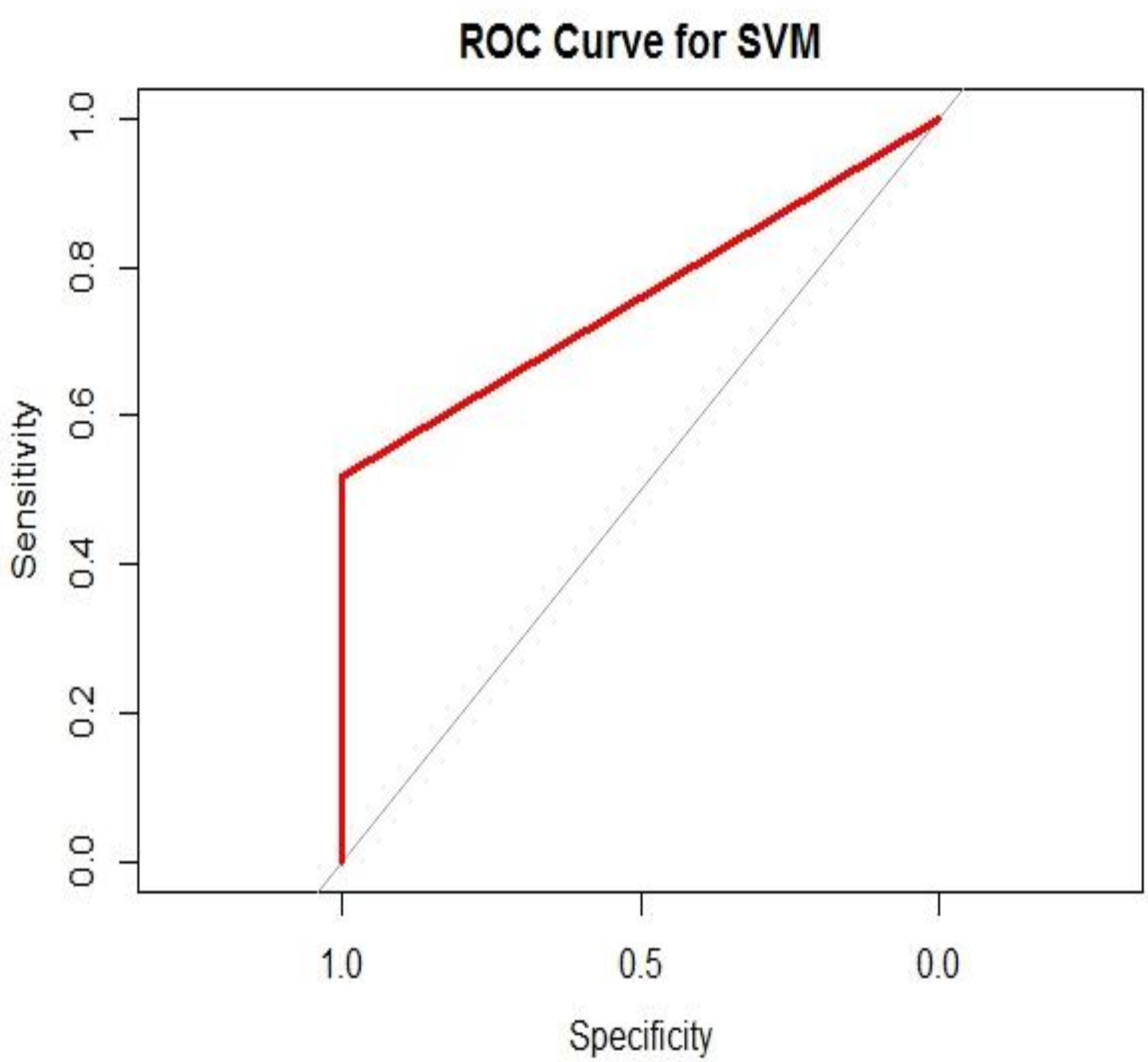
Accuracy = 95.10%

FNR = 0.951%

Precision = 51.79%

Recall = 99.04%

AUC Score = 0.7587



For Python

There is a close call between the Naïve Bayes and CatBoost. By examination of all required values, I chose **NAÏVE BAYES** in python as it outperformed than other models.

Accuracy = 92.15

FNR = 63.58%

Precision = 69.52%

Recall = 36.41%

AUC Score = 0.6736

SUMMARY

By the help of Machine Learning algorithms, we are here able to find the important insights from this model building using Machine Learning.

In the form of Visualizations and through Data Pre-Processing techniques we fetch out the important insights from the data that will help us in order to build our model. We have the insights of variables/factors involved in increasing or decreasing the model compatibility and results.

Confusion Matrix plays a vital role in establishing of a good model and helps in fulfilling the business needs. How accurately model works on given input data shows up in confusion matrix. By some measures through confusion matrix, the company will be able to predict those customers who will make transactions in future and focus on those customers and on those factors which will invoke the transaction of customers.

So in every way, these ML models are helping the companies to establish their goals and focus on the prior important features of the data than rest of the features and invest accordingly. They also reduce human intervention and reduce error rate as much as possible.