Santander Customer Transaction Prediction

By:-

Name: Chinmay Kumar Prusty

Contents

1. Introduction

1.1 Company Background:

At Santander, their mission is to help people and businesses prosper. They are always looking for ways to help their customers understand their financial health and identify which products and services might help them achieve their monetary goals. Their data science team is continually challenging machine learning algorithms, working with the global data science community to make sure that they can more accurately identify new ways to solve our most common challenge, binary classification problems such as: **1.** Is a customer satisfied? **2.** Will a customer buy this product? **3.** Can a customer pay this loan?

1.2 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.

So basically this falls under Binary Classification Problem, where 0 means that the customer will not make a specific transaction and 1 means that the customer will make a specific transaction.

1.3 Dataset Information:

We 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.The training data has 200000 rows and 202 columns(200 feature variables , one target column and ID column).

```
In [4]: #train = train.iloc[0:80000,:]
        my_train.head()
```

Out[4]:

	ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	...	var_190	var_191	var_192	var_193	var_194	var_195	var_196	va
0	train_0	0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187	18.6266	...	4.4354	3.9642	3.1364	1.6910	18.5227	-2.3978	7.8784	8
1	train_1	0	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208	16.5338	...	7.6421	7.7214	2.5837	10.9516	15.4305	2.0339	8.1267	8
2	train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155	...	2.9057	9.7905	1.6704	1.6858	21.6042	3.1417	-6.5213	8
3	train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.9250	...	4.4666	4.7433	0.7178	1.4214	23.0347	-1.2706	-2.9275	10
4	train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405	19.2514	...	-1.4905	9.5214	-0.1508	9.1942	13.2876	-1.5121	3.9267	9

5 rows × 202 columns

```
In [6]: my_train.describe()
```

Out[6]:

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

8 rows × 201 columns

2. Data Pre-processing

2.1 Missing Value Analysis:

No missing values were detected in the dataset.

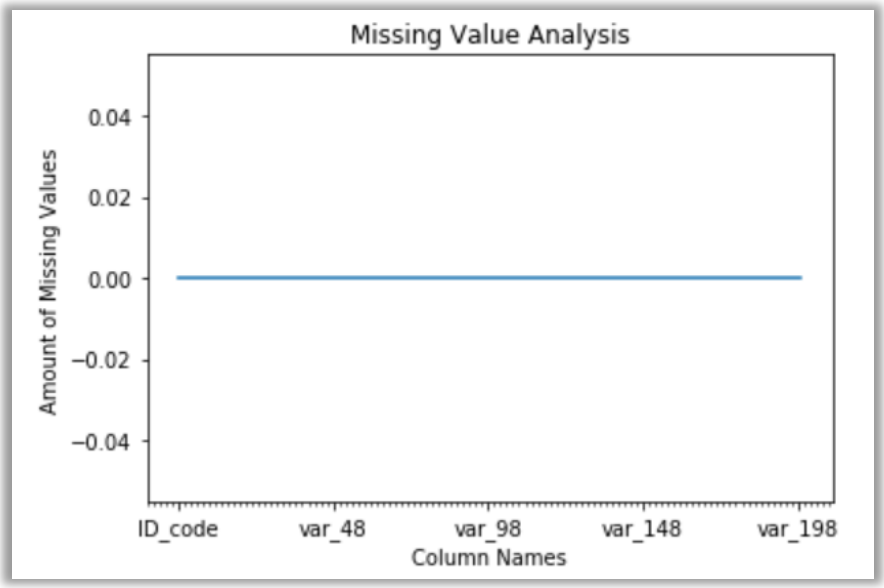


Fig 2.1: Missing Values

2.2 Outlier Analysis:

Below are the boxplots of the feature variables.

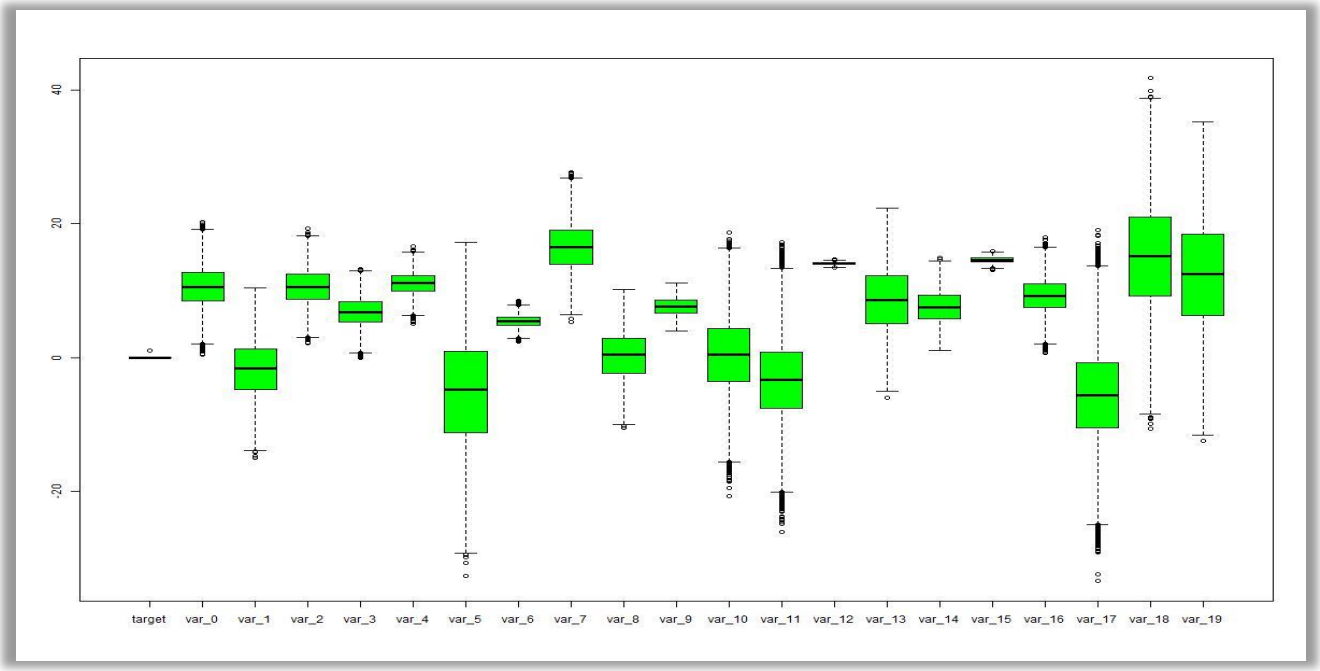


Fig 2.2.1: Boxplot (var 0 – var 19)

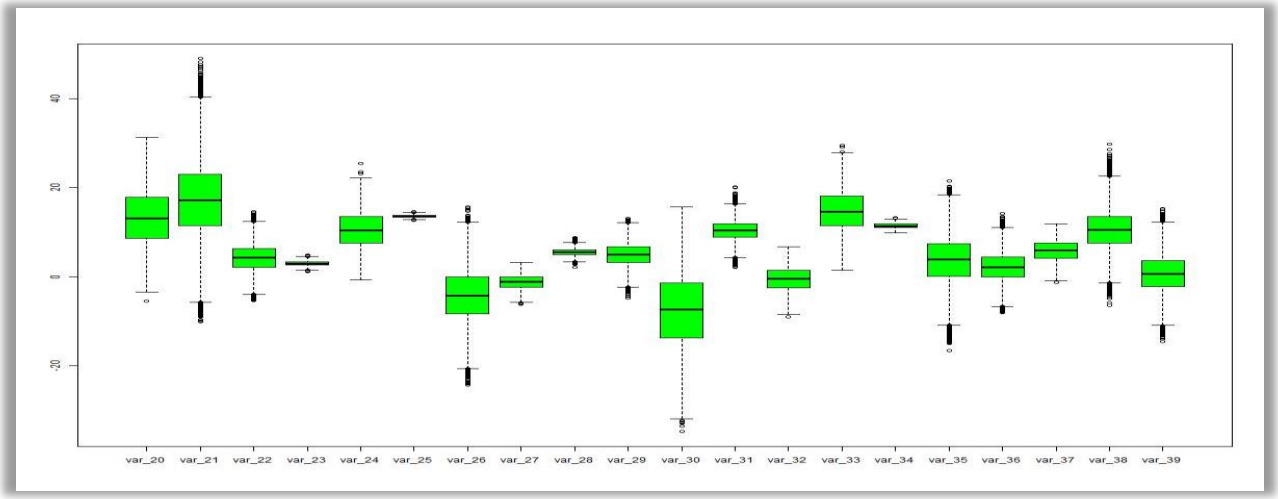


Fig 2.2.2: Boxplot (var 20 – var 39)

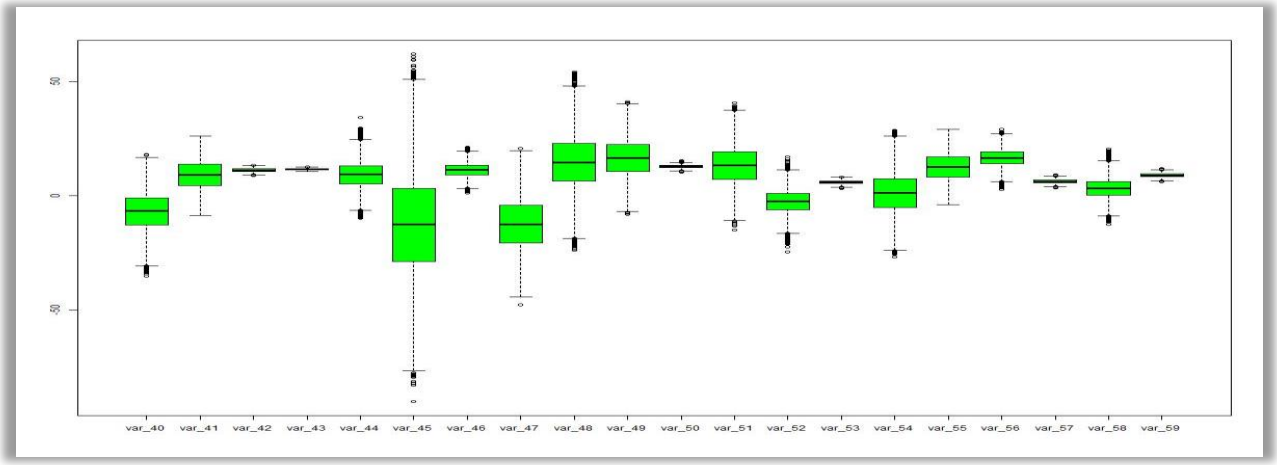


Fig 2.2.3: Boxplot (var 40 – var 59)

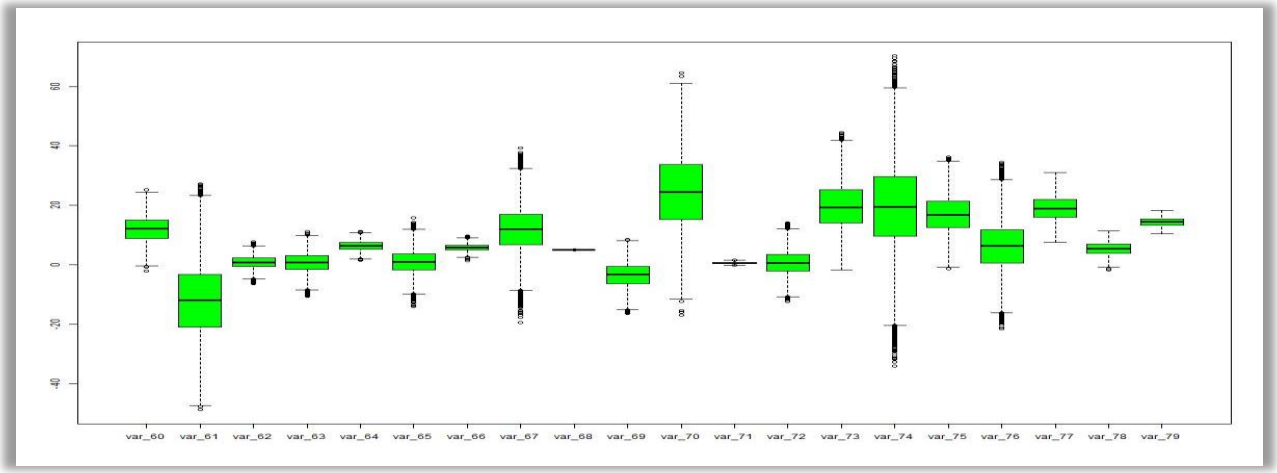


Fig 2.2.4: Boxplot (var 60 – var 79)

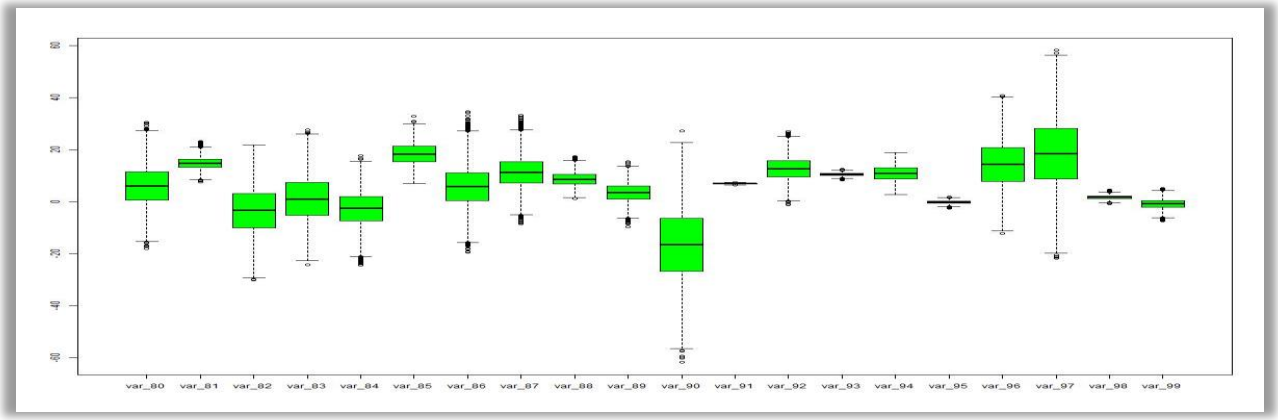


Fig 2.2.5: Boxplot (var 80 – var 99)

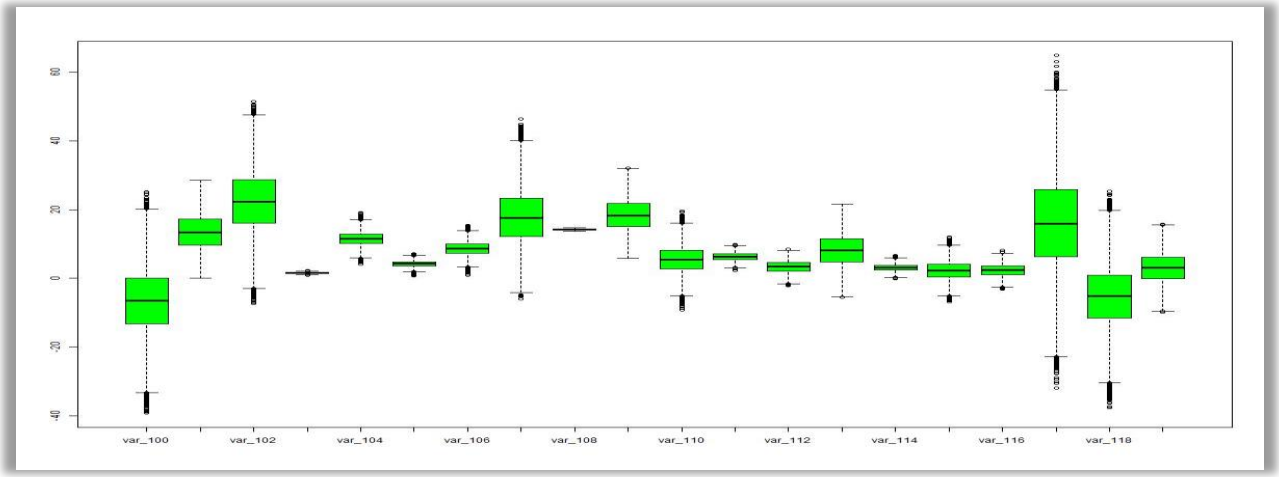


Fig 2.2.6: Boxplot (var 100 – var 119)

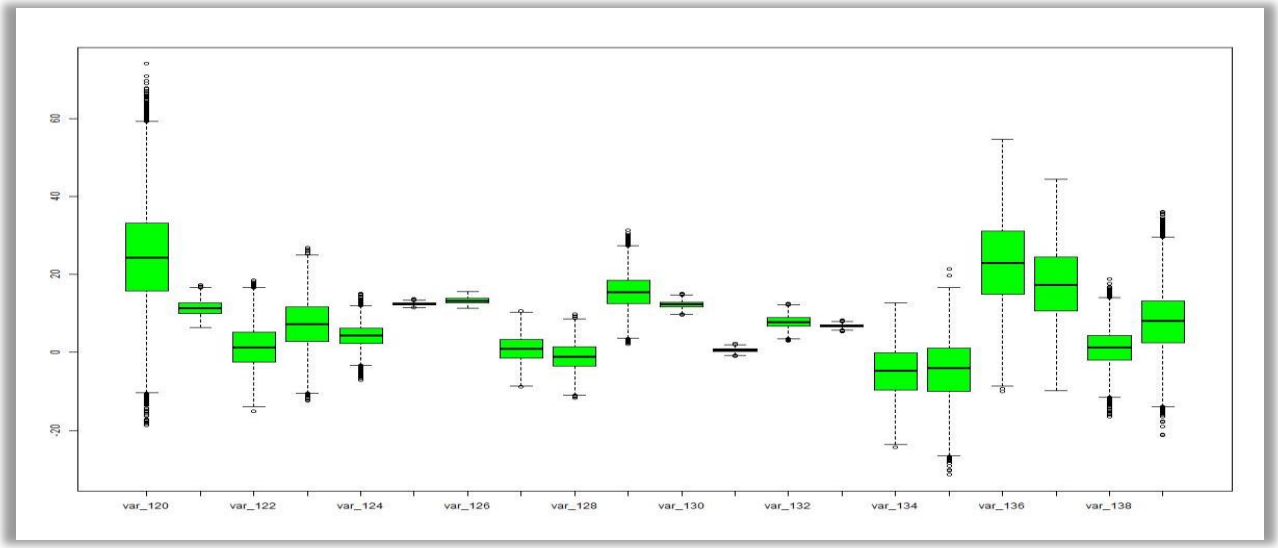


Fig 2.2.7: Boxplot (var 120 – var 139)

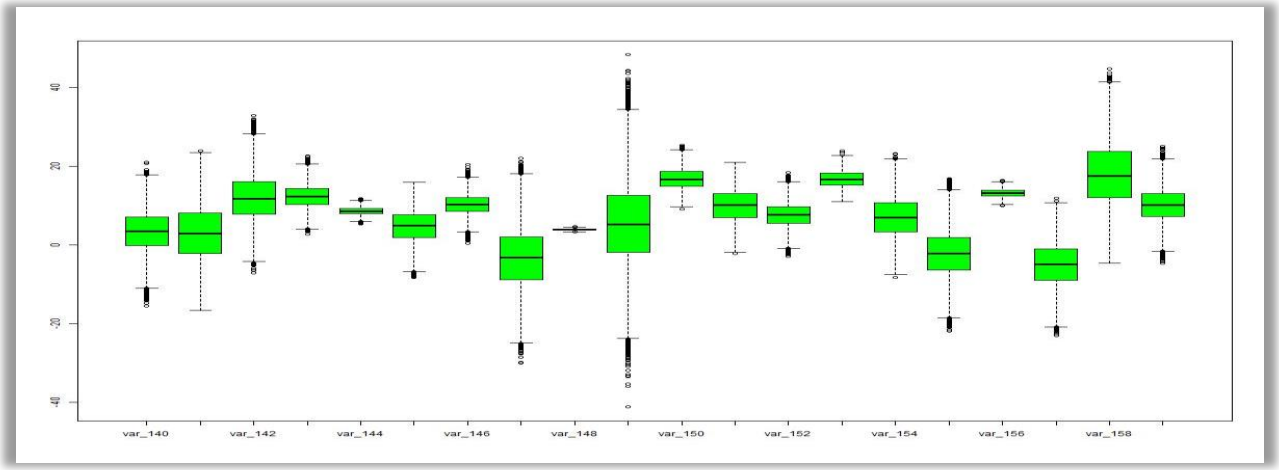


Fig 2.2.8: Boxplot (var 140 – var 159)

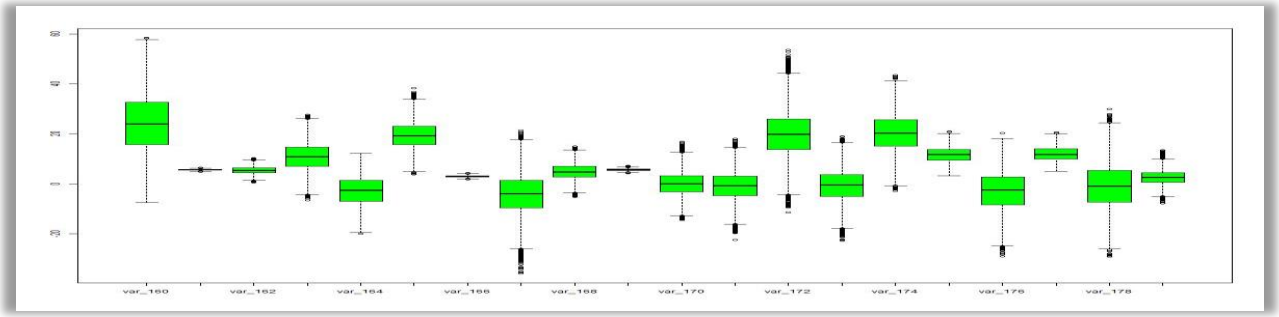


Fig 2.2.9: Boxplot (var 160 – var 179)

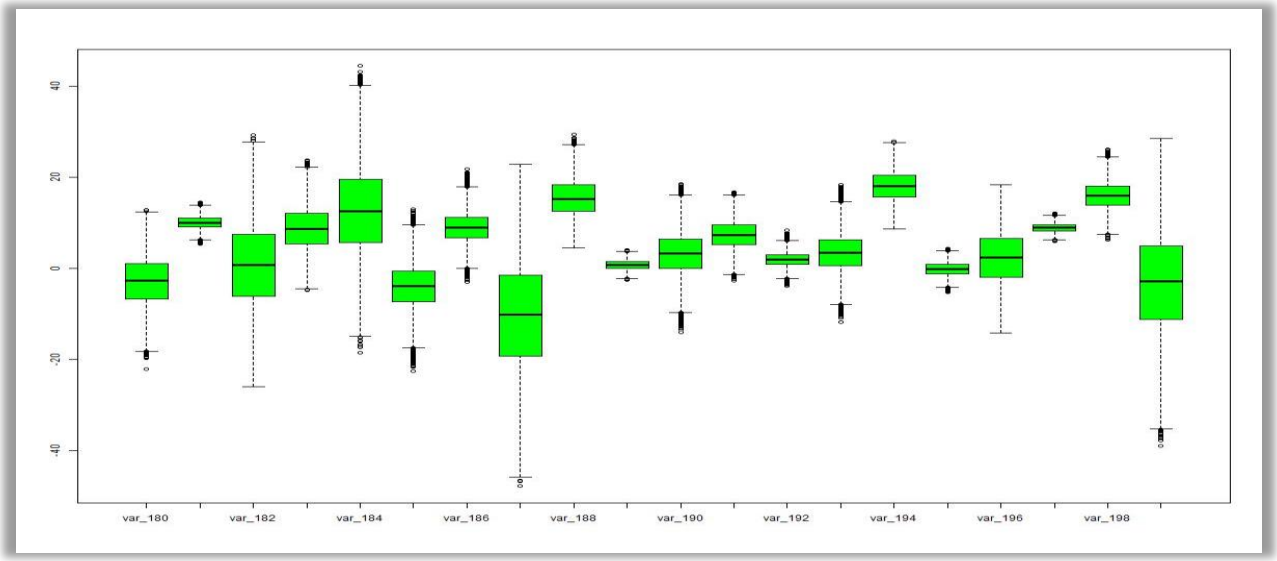


Fig 2.2.10: Boxplot (var 180 – var 199)

Observation: It's clear that almost all the variables are having some outliers.

What is an Outlier?

In statistics, an outlier is a data point that differs significantly from other observations. An outlier may be due to variability in the measurement or it may indicate experimental error. An outlier can cause serious problem while training the predictive model.

In this project, I have followed two approaches to solve outlier problem:

1. Remove the rows having outliers. (Cleaned_Df)

Total no. of rows after outlier rows have been removed = 175073

Percentage of data removed = ((200000 – 175073)/200000) = 12.46%

Since only 12.46% of data is removed which is less than 30% we can consider removing the outliers.

2. Replace outliers with NA and impute them. (Imputed_DF)

In Python IQR method was used to detect outliers while in R boxplot method was used.

2.3 Feature Scaling:

Feature scaling is a method used to normalize/standardize the range of independent variables or features of data.

It is helpful for ML models that uses Euclidean distance.

Formula used for Standardizing the data:

Z = (Actual value – Mean)/ (st.dev)

```
df_temp.describe()
```

	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	.
count	175073.000000	1.750730e+05	1.750730e+05	1.750730e+05	1.750730e+05	1.750730e+05	1.750730e+05	1.750730e+05	1.750730e+05	1.750730e+05	.
mean	0.097691	-8.168176e-14	-1.582155e-15	-8.185415e-15	-8.232897e-15	-2.498801e-14	-5.532915e-16	1.425031e-14	-1.078399e-13	1.117257e-15	.
std	0.296897	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	.
min	0.000000	-2.861413e+00	-3.021002e+00	-2.912535e+00	-2.997106e+00	-2.937773e+00	-3.065591e+00	-2.881573e+00	-2.949753e+00	-3.085728e+00	.
25%	0.000000	-7.337079e-01	-7.695495e-01	-7.549617e-01	-7.562071e-01	-7.376945e-01	-7.803419e-01	-7.405248e-01	-7.627033e-01	-7.804151e-01	.
50%	0.000000	-4.967068e-02	4.752408e-03	-5.111243e-02	1.337537e-02	1.726536e-02	2.785047e-02	-2.788347e-02	-2.663459e-02	3.472570e-02	.
75%	0.000000	6.849932e-01	7.363998e-01	6.825603e-01	7.483973e-01	7.287161e-01	7.622796e-01	6.870717e-01	7.496408e-01	7.954497e-01	.
max	1.000000	2.814414e+00	2.966982e+00	2.843928e+00	3.000409e+00	2.918242e+00	2.837489e+00	2.830086e+00	2.989041e+00	2.959964e+00	.

8 rows × 201 columns

Fig 2.3.1: Standardized Dataset

In the above figure we can see that after standardizing the dataset, the range of feature variables does not differ that much.

2.4 Feature Selection:

Here we check for multicollinearity between predictor variables. And if one predictor describes another predictor then we may drop one of them.

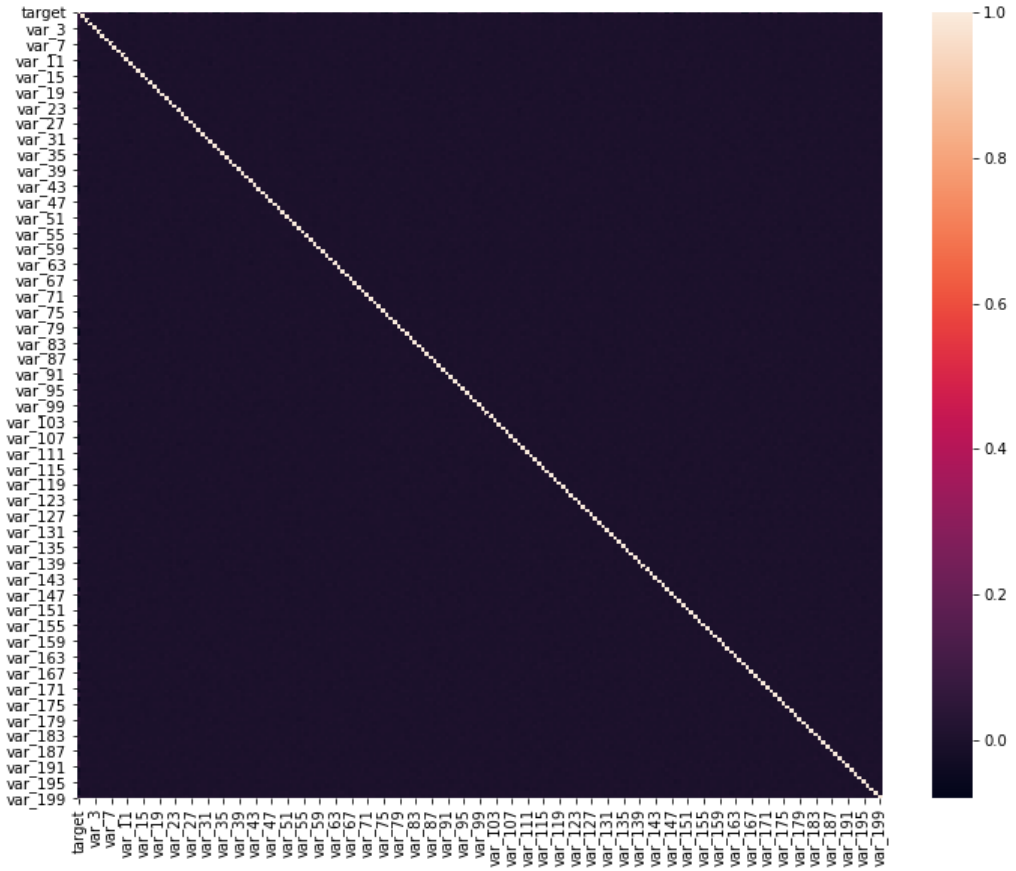


Figure 2.4: Correlogram

All the variables are independent to each other.

Since all the variables are already independent to each other we can't use Principal Component Analysis to reduce the no. of feature variables.

3. Exploratory Data Analysis

3.1 Checking the Distribution of Variables

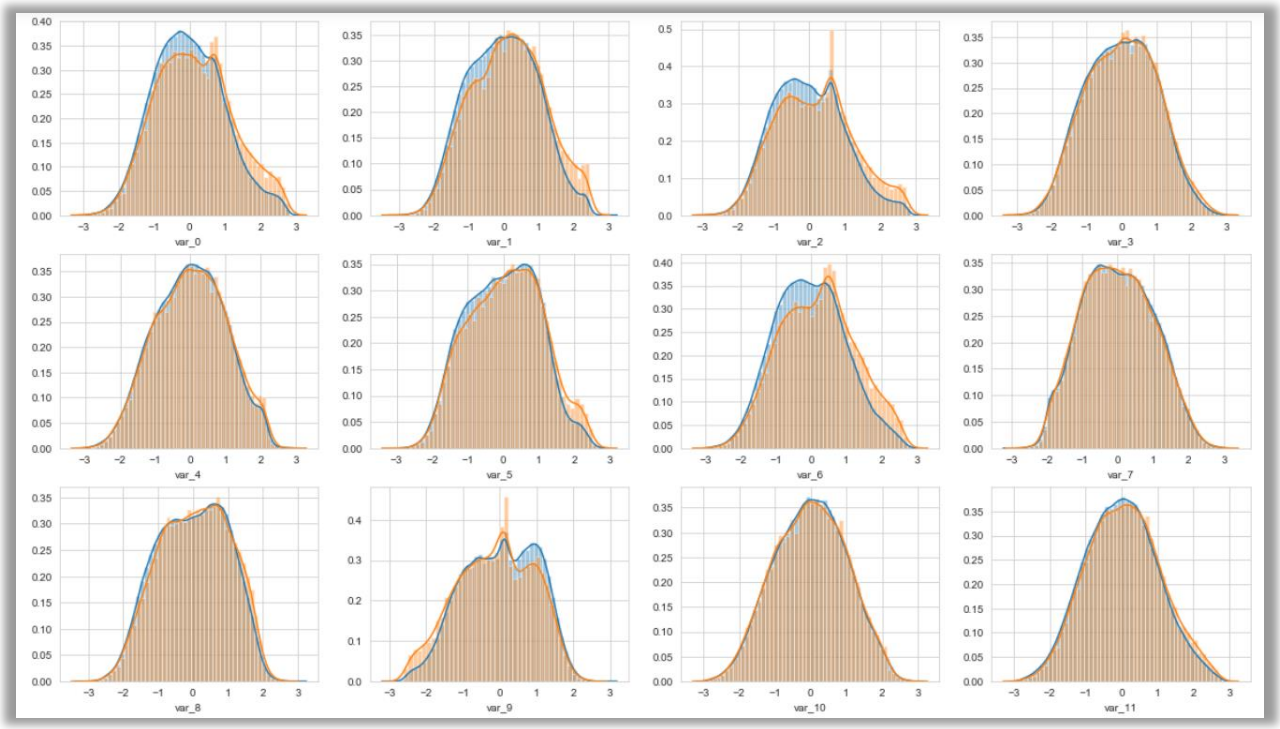


Fig 3.1: Distributions of first 10 variables

Almost all the variables are having Normal Distribution while some of them are also having Bi-modal distribution.

3.2 Count of Target Variable:

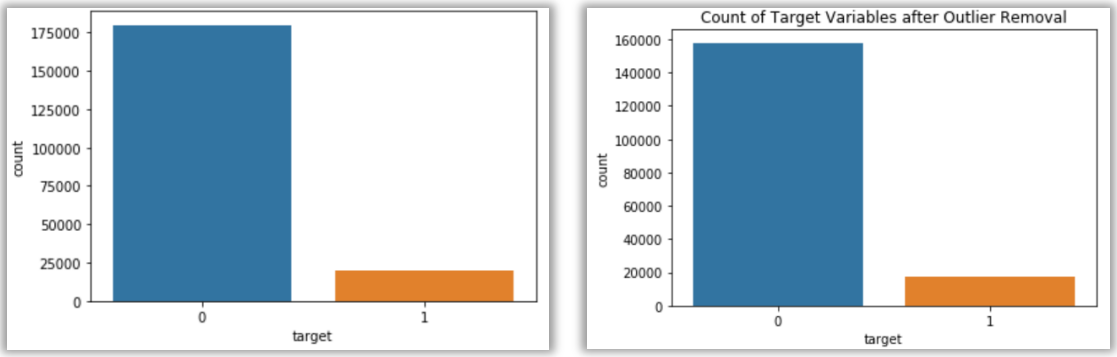


Fig 3.2.1 Count of Target Variable(before outlier removal and after outlier removal)

It is observed that there is a target class imbalance problem. To solve this we can use various sampling techniques: Oversampling , Under sampling, Stratified sampling, etc.

3.3 Count of unique values in each feature variable:

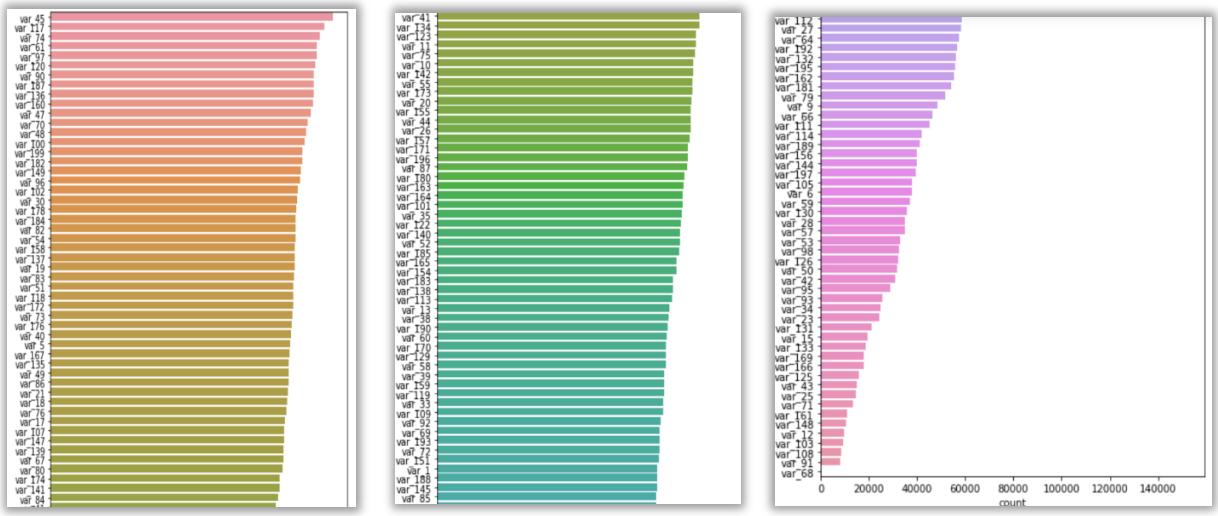


Fig 3.3.1 Bar plot showing count of unique values in each variable

Observation: var_68, var_91, var_108, var_103 and var_12 are having very less unique values. It is possible that they are categorical and we may handle them accordingly.

4. Evaluation Metrics

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Fig 4: Classification Matrix

Accuracy alone can't tell us about how good our model is performing.

So we'll be using various evaluation metrics:

4. 1 Accuracy: (TP+TN)/(TP+FP+FN+TN)

It is the proportion of true results among the total no. of cases examined. And it is pretty easy to understand.

Accuracy is a valid choice of examination when the target class is balanced and not skewed.

4.2 Precision: (TP)/(TP+FP)

Precision tells us what proportion of predicted positives are truly positive.

It is a valid choice of examination when we want to be very sure for our prediction. (For example in this problem , we are more more interested in knowing if the customers are going to make transaction.)

4.3 Recall: (TP)/(TP+FN)

Recall tells us about what proportion of actual positives are truly classified.

It is a valid choice of evaluation metric when we want to capture as many positives as possible.

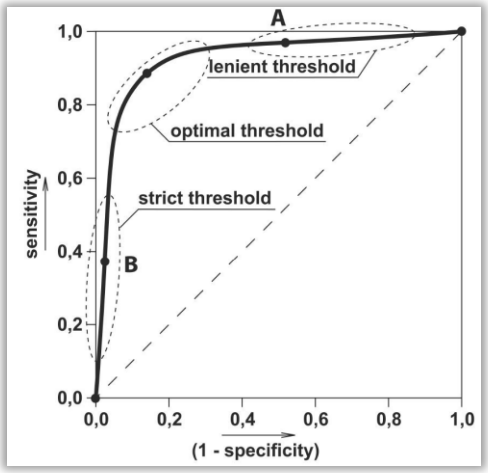
4.4 F1-Score: 2* (precision * recall)/(precision + recall)

It is the harmonic mean of precision and recall.

It sort of maintains the balance between precision and recall.

4.5 AUC:

AUC ROC indicates how well the probabilities from the positive classes are separated from the negative classes



Sensitivty = TPR(True Positive Rate)= Recall = TP/(TP+FN)

1- Specificity = FPR(False Positive Rate)= FP/(TN+FP)

5. Model Development

5.1 logistic Regression

I have developed 4 models of Logistic Regression.

Model 1 : Dataset with removed outliers is used for training the model.

#1 Accuracy: 91.3800000

#2 Precision: 0.7800000

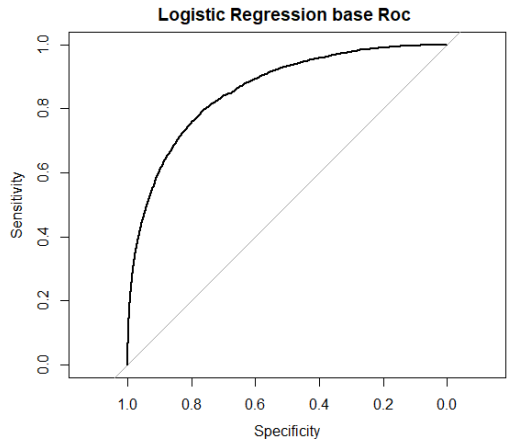
#3 Recall: 0.1700000

#4 False Positive Rate: 0.0100000

#5 False Negative Rate: 0.8300000

#6 F1 Score: 0.2791579

#7 AUC: 0.8599



Interpretation:

Accuracy of the model is very good i.e. 91.3%.

Model has low recall 17% only.

F1-score is low i.e. 27.9%.

AUC ROC is also good 85.9%.

Poor model since recall is very less.

Model 2: Dataset with removed outliers and over sampled train data is used.

#1 Accuracy 86.8900

#2 Precision: 0.3900

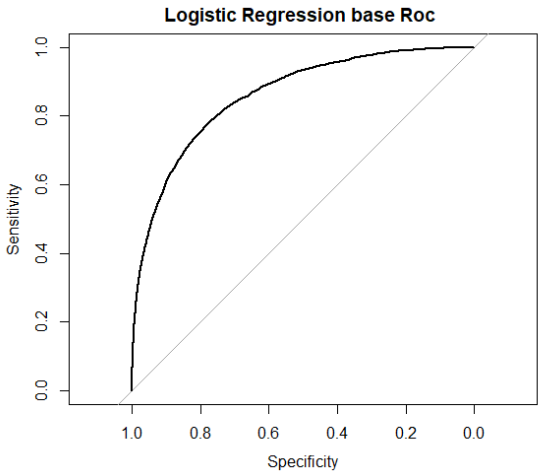
#3 Recall: 0.6100

#4 False Positive Rate: 0.1000

#5 False Negative Rate 0.3900

#6 F1 Score: 0.4758

#7 AUC: 0.8595



Interpretation:

Accuracy of model is ok 86.8%

Recall is also good 61%.

F1-score is better than model-1, 47.5%.

AUC is same as before.

False Negative Rate is also ok.

Overall a better model than the previous model.

Model 3: Dataset with imputed outliers is used.

#1 Accuracy 91.44000000

#2 Precision: 0.76000000

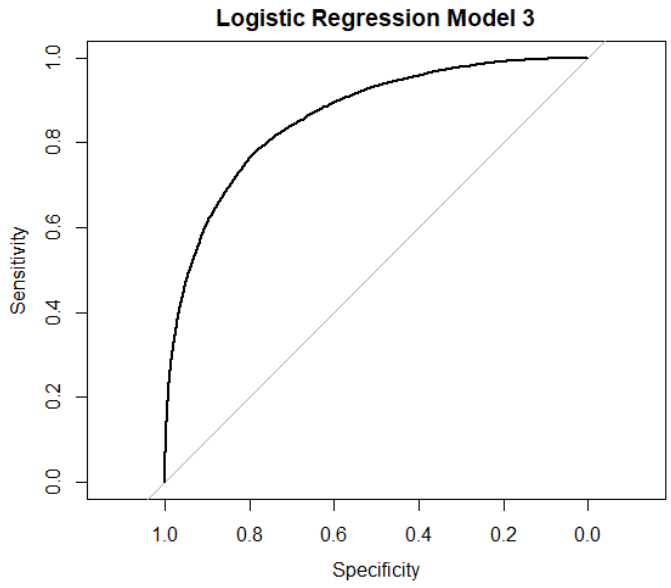
#3 Recall: 0.01000000

#4 False Positive Rate: 0.01000000

#5 False Negative Rate 0.82000000

#6 F1 Score: 0.01974026

#7 AUC: 0.8615



Interpretation:

Accuracy is very good 91%

Recall is very low 1% only.

FNR is very high 82%.

F1-score is also very low 1%

Overall very poor model.

Model 4: Outliers have been replaced with NA and then imputed with mean values.

Oversampled data is used.

#1 Accuracy 88.9500000

#2 Precision: 0.4500000

#3 Recall: 0.0700000

#4 False Positive Rate: 0.0700000

#5 False Negative Rate 0.4700000

#6 F1 Score: 0.1211538

#7 AUC: 0.8615

Interpretation:

Accuracy is good 89%.

Recall is low 7%.

False Negative is also high 47%.

F1-score is very low 12% only.

Observing all the 4 models of logistic regression , I have noticed that Model performs better if rows having outliers have been removed and also the training data is over sampled to solve target class imbalance problem.

So, out of all the 4 logistic regression models, model 2 is the best.

And for the upcoming models we will use dataset where outliers have been removed and oversampled training data.

5.2 Decision Tree

Model 1 – Oversampled cleaned data**.**

#1 Accuracy 61.9400000

#2 Precision: 0.1300000

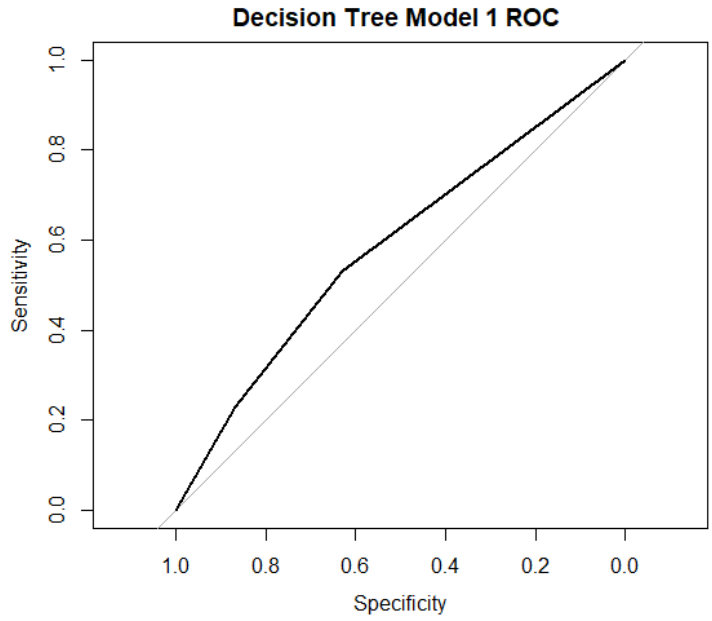
#3 Recall: 0.5300000

#4 False Positive Rate: 0.3700000

#5 False Negative Rate 0.4700000

#6 F1 Score: 0.2087879

#7 AUC: 0.5881



Interpretation:

Accuracy, f1-score and AUC are low.

Recall is good but precision is low.

Overall a poor model.

Model 2: Model parameters have been tuned

Minsplit and max depth have been set to 5 in this model to avoid overfitting.

#1 Accuracy 80.6900000

#2 Precision: 0.1600000

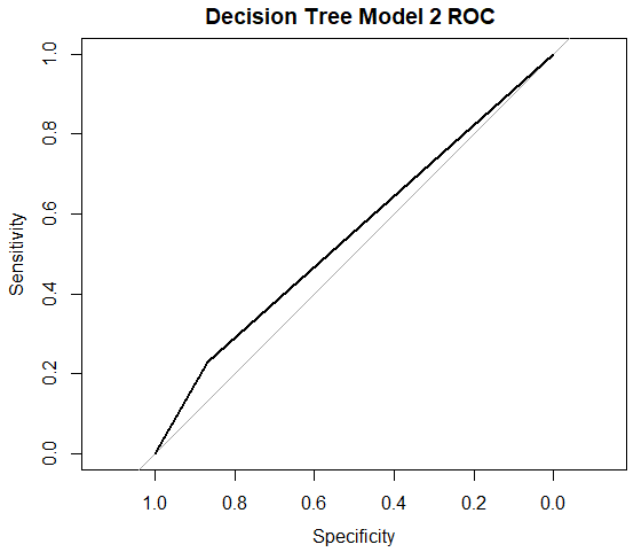
#3 Recall: 0.2300000

#4 False Positive Rate: 0.1300000

#5 False Negative Rate 0.7700000

#6 F1 Score: 0.1887179

#7 AUC: 0.5498



Interpretation:

Accuracy is good 80%.

Recall and Precision are balanced.

F1-score is low and AUC is low.

This is a poor model.

5.3 Naïve Bayes

Model 1: with non over sampled cleaned data

#1 Accuracy 92.2400000

#2 Precision: 0.7100000

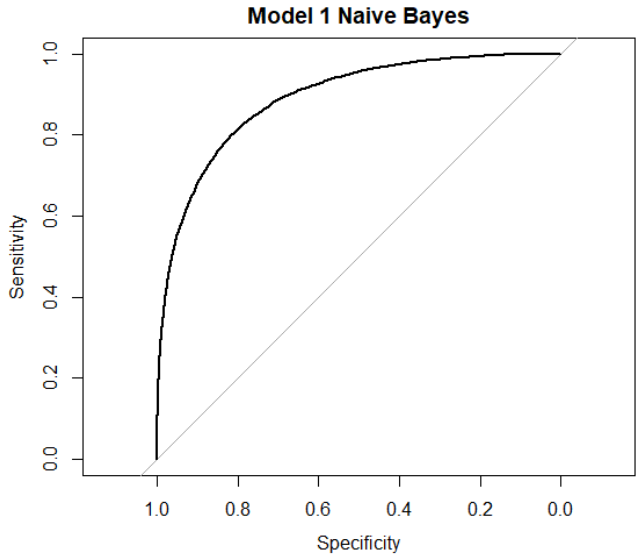
#3 Recall: 0.3500000

#4 False Positive Rate: 0.0200000

#5 False Negative Rate 0.6500000

#6 F1 Score: 0.4688679

#7 AUC: 0.89



Interpretation:

Accuracy , Precision and AUC are quite good.

F1-score is also good 47%.

FNR is more.

Overall this model is ok.

Model 2: with over sampled cleaned data

#1 Accuracy 83.8300000

#2 Precision: 0.3500000

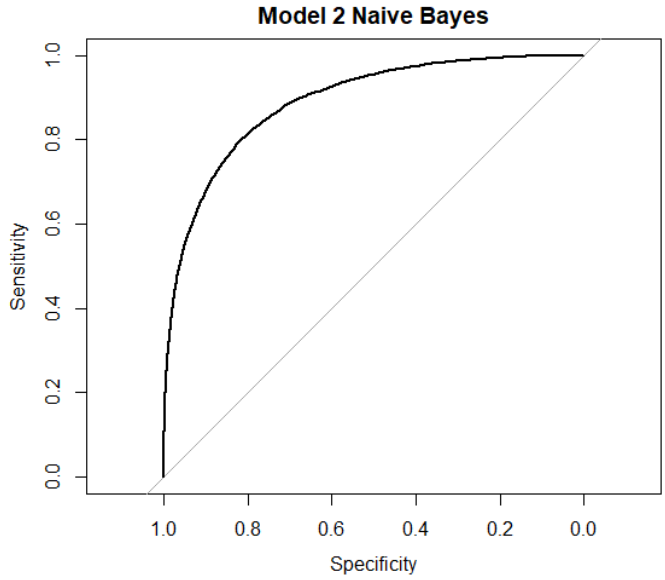
#3 Recall: 0.7600000

#4 False Positive Rate: 0.1500000

#5 False Negative Rate 0.2400000

#6 F1 Score: 0.4792793

#7 AUC: 0.8889



Interpretation:

Accuracy is slightly less but not bad.

Recall is very good 76%.

F1-score is also good 48%.

FNR is low which is good.

AUC is also good 89%.

Model is quite good and is also better than Logistic Regression Model 2.

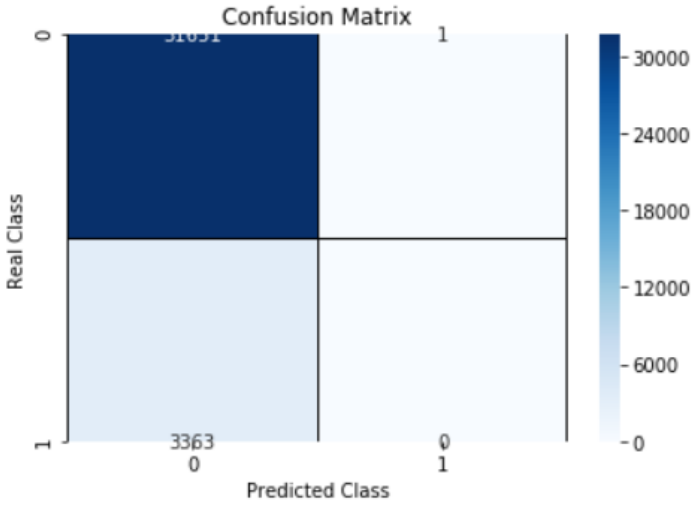
5.4 KNN

True Positive: 0
True Negative: 31651
False Positive: 1
False Negative: 3363

Accuracy Rate: 90.39268884763672
Error Rate: 9.607311152363271

----------Classification Report----------

	precision	recall	f1-score	support
0	0.90	1.00	0.95	31652
1	0.00	0.00	0.00	3363
accuracy			0.90	35015
macro avg	0.45	0.50	0.47	35015
weighted avg	0.82	0.90	0.86	35015



Interpretation:

Accuracy is good 90%.

But Recall is extremely low.

Moreover this model has not predicted a single positive class.

Worst model so far.

6. Summary

Out of all the models we've tried so far, Naïve Bayes Model 2 with over sampled data is the best one.

This model has highest Recall ,f1-score and AUC .

It also has a good balance between Precision and Recall.

7. References

1. https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/
2. https://humansofdata.atlan.com/2018/03/when-delete-outliers-dataset/
3. https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226
4. Github
5. Kaggle