CREDIT CARD FRAUD DETECTION

By:
Varun Varma

University of CINCINNATI

INTRODUCTION

Credit card fraud is an inclusive term for fraud committed using a payment card, such as a credit card or debit card. This means a fraud when someone uses your credit card or credit account to make a purchase you didn't authorize. This activity can happen in different ways: If you lose your credit card or have it stolen, it can be used to make purchases or other transactions, either in person or online.

It is important that credit card companies be able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase. Thus we build a machine learning model to predict whether an anonymized credit card transactions is labeled as fraudulent or genuine.

DATA SOURCE

The dataset is obtained from Kaggle: Credit Card Fraud Detection. Link can be found here: https://www.kaggle.com/mlg-ulb/creditcardfraud

The dataset has been collected and analysed during a research collaboration of Worldline and the Machine Learning Group (http://mlg.ulb.ac.be) of ULB (Université Libre de Bruxelles) on big data mining and fraud detection.

DATA CONTENT

The datasets contain transactions made by credit cards in September 2013 by European cardholders.

This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-senstive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

EXPLORATORY DATA ANALYSIS

We start with checking the summary statistic of the given dataset. The following observations obtained are:

1. All variables are numeric except the Class variable which is Factor as desired.
2. Class variable has two values: 0 and 1 which are highly unbalanced.

3. Amount and Time are the only two variables which are not scaled, compared to rest of the V variables. Hence, we scale them before proceeding with further analysis.

We plot the histogram of the various variables to see the distribution. They are as follow:

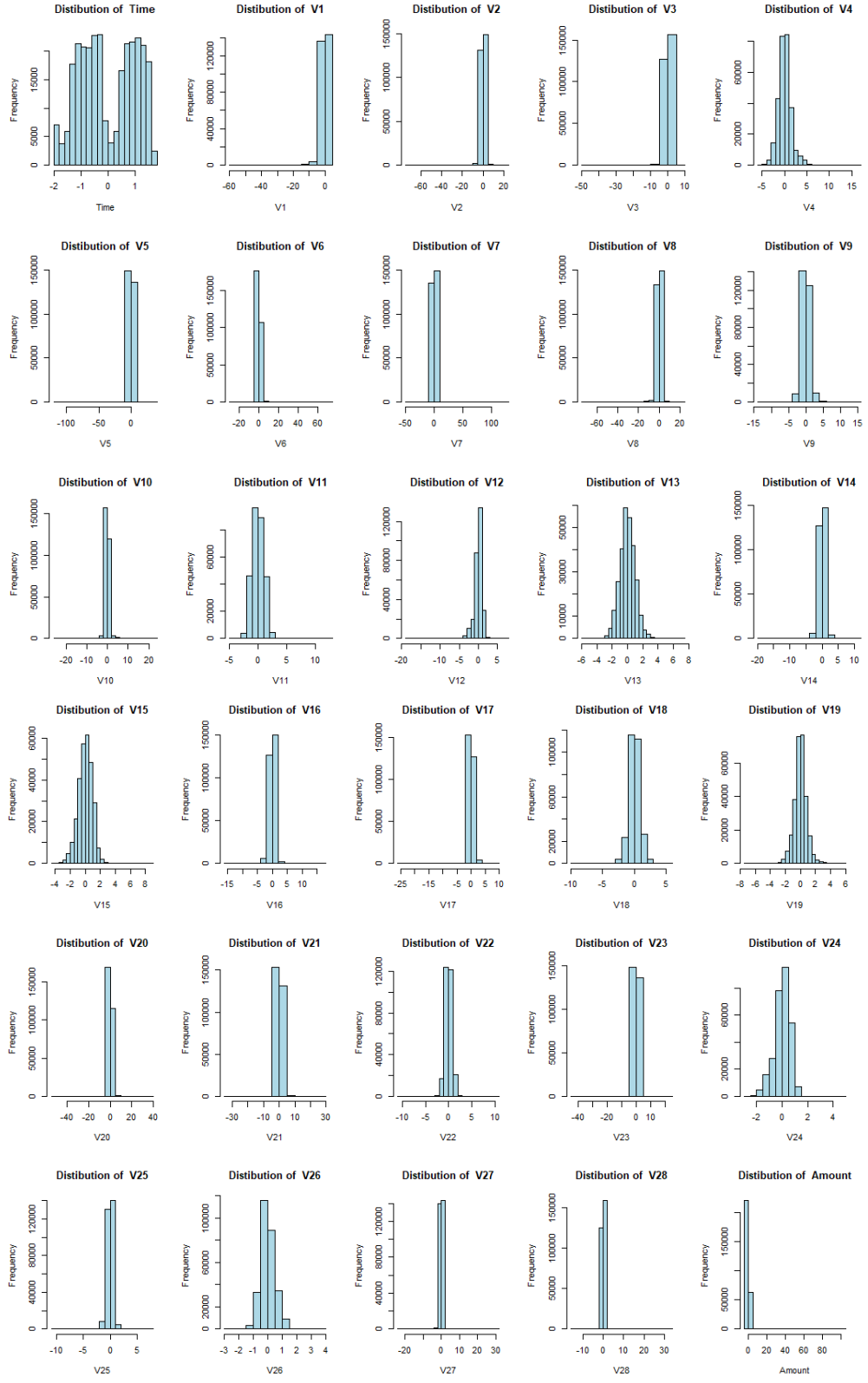

Figure 1: Histograms of Variables

As we see from the above figures, nothing much can be deduce about the variables. Most of the variables seems to follow a normal distribution. Since the data is highly unbalanced, it is not feasible to get a trend of change of the above parameters with fraud and not fraud cases.

We now check into the distribution of not fraud and fraud credit card transaction.

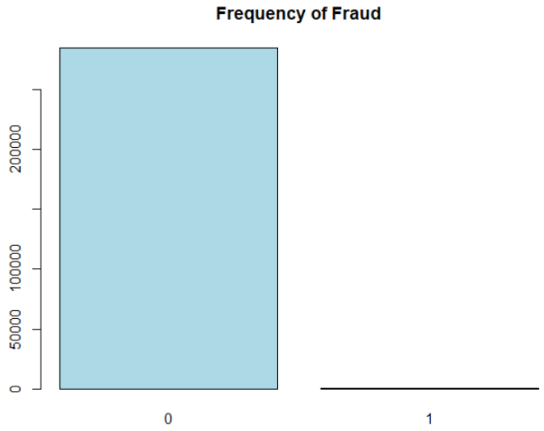

Figure 2: Frequency of Fraud vs Not Fraud

From the above graph, we see that there are 492 cases out of 284,807 transactions. Thus, it signifies dataset is highly unbalanced as the positive class (frauds) account for 0.172% of all transactions. This is a clear indication that we need to balance the dataset before applying any models to get more accurate results.

Now we see how these variables are correlated with each other.

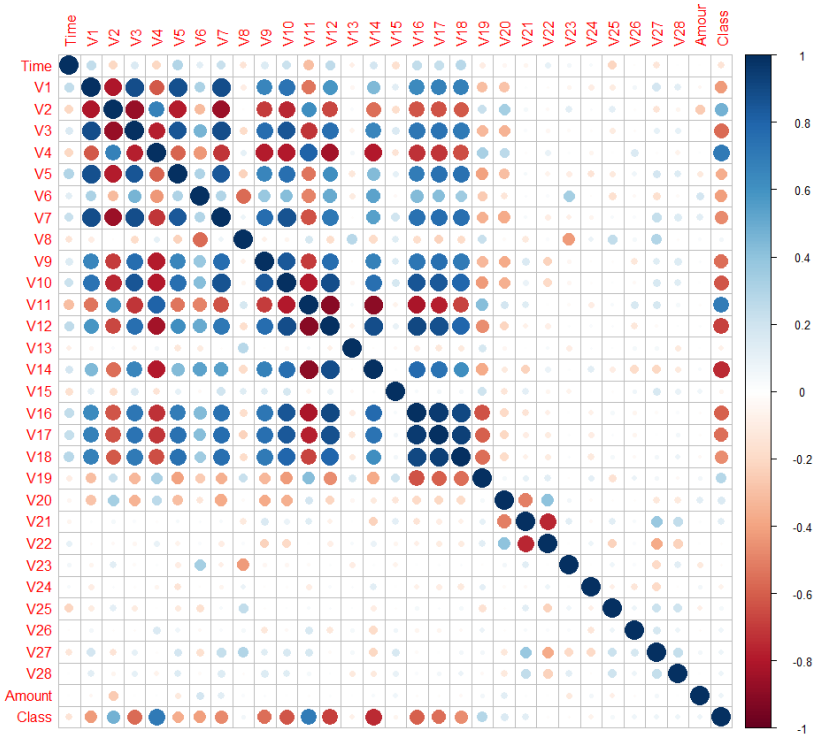

Figure 3: Correlation Plot

From the above chart, we see the correlation between different variables. It is also interesting to note that for class, which is our response variable, how different variables are correlated. V4 and V11 are highly positively correlated with Class while V12, V14 are highly negatively correlated with Class. Some variables like V22 and V23 do not seem to be correlated with Class.

CONVERTING UNBALANCED DATA TO BALANCED

Unbalanced classification problems cause problems to many learning algorithms. These problems are characterized by the uneven proportion of cases that are available for each class of the problem.

We have used SMOTE (Chawla et. al. 2002) technique to fight this problem. The general idea of this method is to artificially generate new examples of the minority class using the nearest neighbors of these cases. Furthermore, the majority class examples are also under-sampled, leading to a more balanced dataset.

The parameters perc.over and perc.under control the amount of over-sampling of the minority class and under-sampling of the majority classes, respectively. After applying SMOTE technique, we get a more balanced dataset. The distribution of Class is as follow:

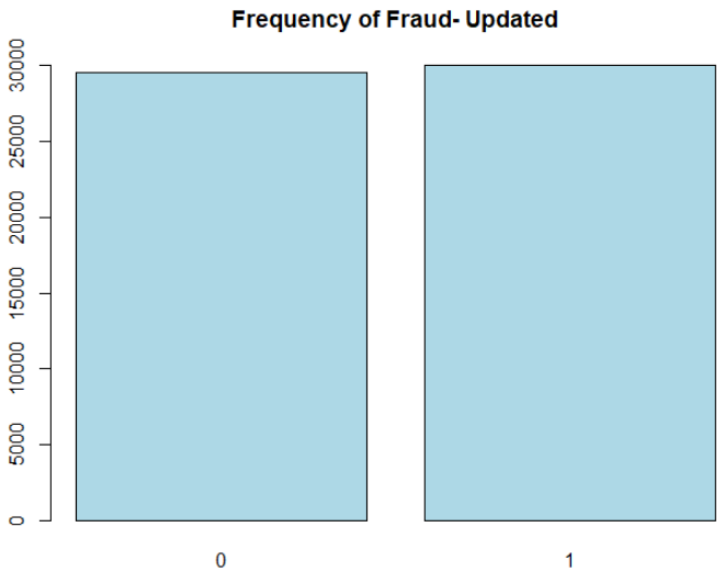

Figure 4: Frequency of Fraud- Updated

From the above graph, we see that the number of fraud cases increased to around 30000 cases out of around 60000 transactions thus making the dataset almost balanced and ready for analysis.

We not split the data into two parts: training (70%) and testing(30%). The training dataset is ready to build various machine learning model and evaluate their performance with the help of testing dataset.

K-NEAREST NEIGHBOURS(KNN)

With multiple trail and error, we fix the number of nearest neighbour(k) in the algorithm to be 5 and train the model to evaluate the accuracy (1- misclassification rate) on both training and testing data.

The results are as follow:

Parameters	Value
In Sample Accuracy	0.996
Out of Sample Accuracy	0.994
F1 Score on Test Data	0.994

Table 1: Model Parameters of KNN Model

The confusion matrix obtained from the test data is shown below which is used to calculate F1score and out of sample accuracy:

	PREDICTED	
TRUE	**0**	**1**
0	8740	94
1	1	9025

Table 2: Confusion Matrix of KNN Model

LOGISTIC REGRESSIONS

We fix the asymmetrical cost as 10:1. One of the main task is to reduce the false negative values as we rather classify not fraud cases as fraud than classifying the fraud cases as not fraud.

First we do a variable selection technique using step function (AIC and BIC with forward direction) and lasso(with nfolds as 10 and lamda as 1 SE). Based on the least residual deviance obtained from all the above methods, we select the model obtained by AIC as our final model.

```
glm(formula = Class ~ V14 + V4 + V10 + V12 + V8 + V13 + V11 +
    V26 + V6 + V7 + V22 + V23 + V21 + V16 + V9 + V1 + V2 + V17 +
    Time + V5 + V27 + V18 + V28 + V20 + Amount + V3 + V19 + V15,
    family = binomial, data = credit.train)
```

We decide the cut off probability via grid search method by plotting the cost(MR) and probability.

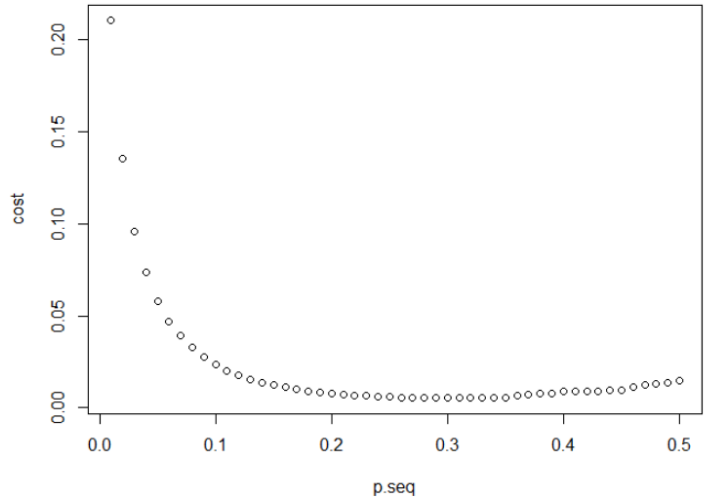

Figure 5: Error vs pcut graph

From the above graph, we see that the optimal cut off probability as 0.29.

Now, we train the model on training data using cut off probability as 0.29 and obtain the ROC graph on the testing data as follow.

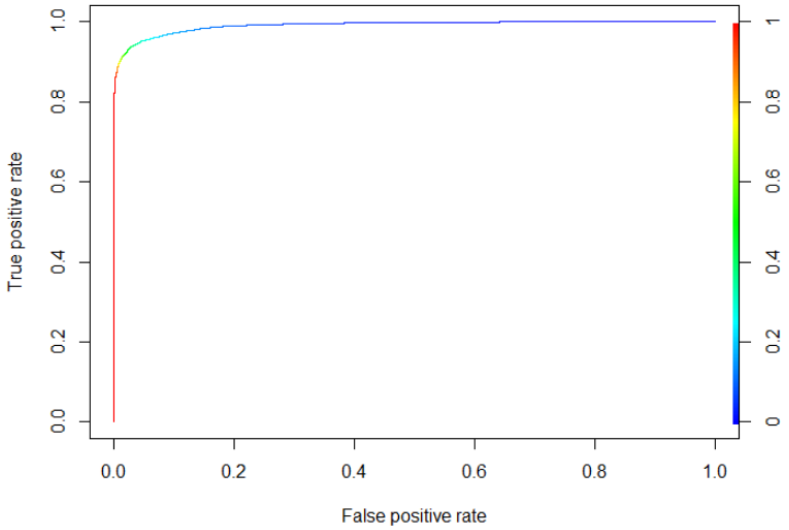

Figure 6: ROC Curve for Test Data

We obtain the results as follow:

Parameters	Value
In Sample Accuracy	0.901
Out of Sample Accuracy	0.899
F1 Score on Test Data	0.889
AUC for Test Data	0.990

Table 3: Model Parameters of Logistic Regression

The confusion matrix obtained from the test data is shown below which is used to calculate F1score and out of sample accuracy:

	PREDICTED	
TRUE	**0**	**1**
0	7147	1687
1	103	8923

Table 3: Confusion Matrix of Logistic Regression Model

CLASSIFICATION DECISION TREE

We model a classification decision tree with the asymmetrical cost as 10:1 and cp value as 0.01 on the training data. Since the tree obtained is short, there is no need of pruning.

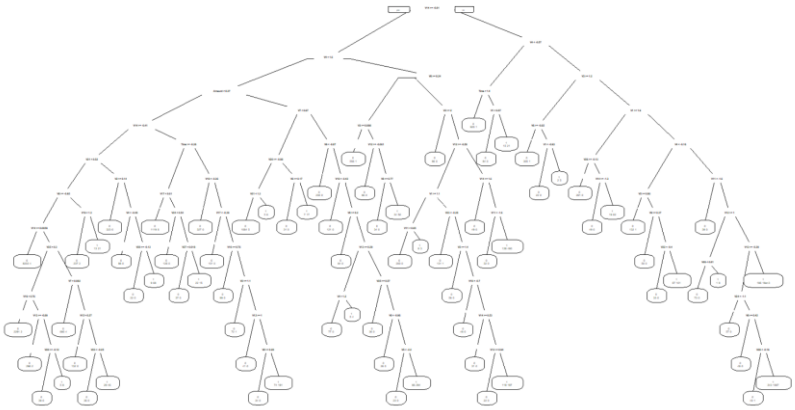

Figure 7: Classification Decision Tree

We use these tuning parameters to train the model and find the accuracy on the training and testing data along with ROC curve on testing data.

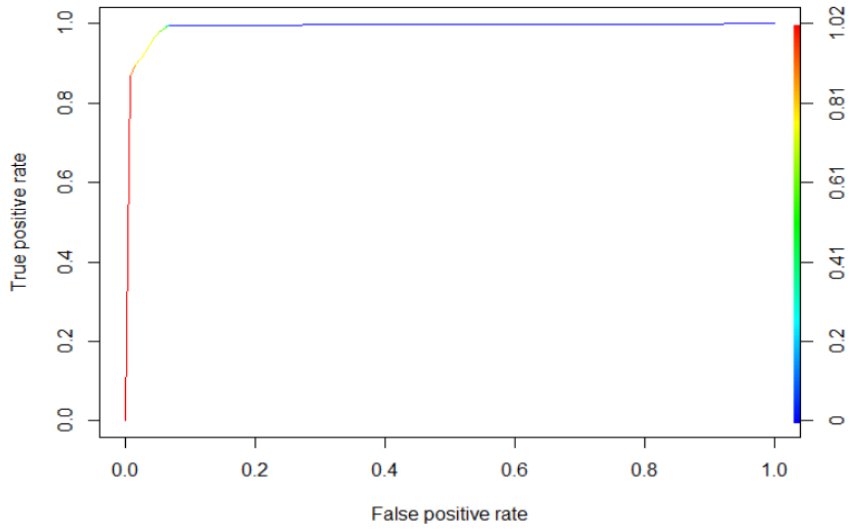

Figure 6: ROC Curve for Test Data

The results are as follow:

Parameters	Value
In Sample Accuracy	0.971
Out of Sample Accuracy	0.965
F1 Score on Test Data	0.963
AUC for Test Data	0.990

Table 5: Model Parameters of Classification Tree Model

The confusion matrix obtained from the test data is shown below which is used to calculate F1score and out of sample accuracy:

	PREDICTED	
TRUE	**0**	**1**
0	8267	567
1	53	8973

Table 6: Confusion Matrix of Classification Tree Model

RANDOM FOREST

We develop a random forest model with number of tress as 500 and the cut off probability obtained via grid search method (0.28 in this case) by plotting the cost(MR) and probability.

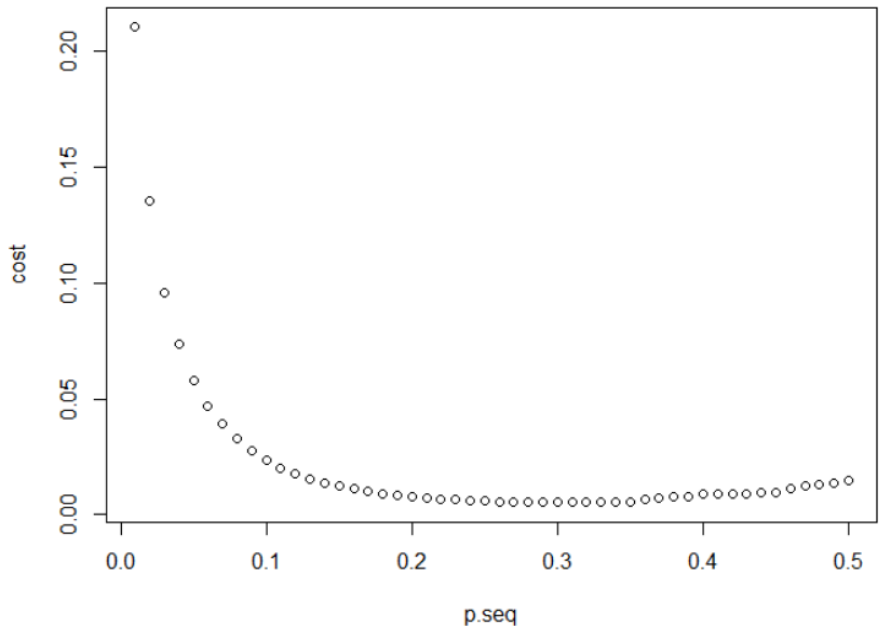

Figure 7: Error vs pcut graph

Now, we train the model on training and obtain the ROC graph on the testing data as follow.

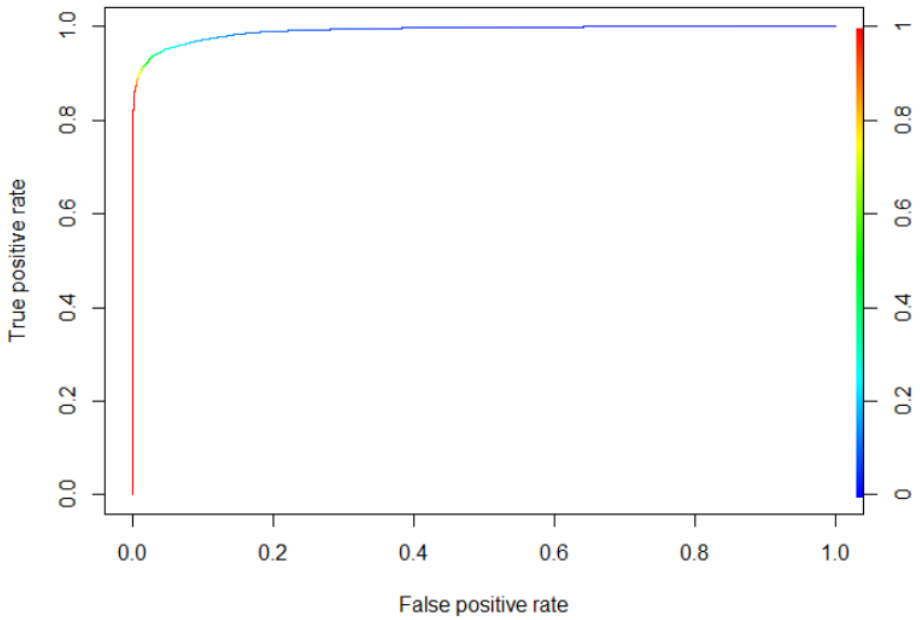

Figure 8: ROC Curve for Test Data

We obtain the results as follow:

Parameters	Value
In Sample Accuracy	0.999
Out of Sample Accuracy	0.994
F1 Score on Test Data	0.989
AUC for Test Data	0.998

Table 7: Model Parameters of Random Forest

The confusion matrix obtained from the test data is shown below which is used to calculate F1score and out of sample accuracy:

	PREDICTED	
TRUE	**0**	**1**
0	8788	46
1	0	9026

Table 8: Confusion Matrix of Random Forest Model

NEURAL NETWORK

We use nnet package of the neural network along with 1 hidden layer and 500 maximum iterations to develop a model on the training data and evaluate its performance on the testing data.

The model converges after 210 iterations to give us the following result:

Parameters	Value
In Sample Accuracy	0.920
Out of Sample Accuracy	0.918
F1 Score on Test Data	0.914

Table 9: Model Parameters of Neural Network Model

The confusion matrix obtained from the test data is shown below which is used to calculate F1score and out of sample accuracy:

	PREDICTED	
TRUE	0	1
0	7455	1379
1	74	8952

Table 10: Confusion Matrix of Neural Network Model

CONCLUSION

After initial analysis of checking the summary of the highly unbalanced dataset and converting it to balanced using SMOTE, the whole dataset was divided into 70%(training) and 30%(testing).

After that we have created 5 different models. We now evaluate the various models created by comparing the model parameters and the results are as follow:

Parameters	KNN Model	Logistic Regression	Classification Tree	Random Forest	Neural Network
In Sample Accuracy	0.996	0.901	0.971	0.999	0.920
Out of Sample Accuracy	0.994	0.899	0.965	0.994	0.918
F1 Score on Test Data	0.994	0.889	0.963	0.996	0.914
AUC for Test Data		0.990	0.990	0.998	

Table 10: Models Summary

From the above results, we see that KNN and Random Forest technique stands out in all analysis compared to the result of the other models.