Capstone_self project

George Joseph

20/08/2020

INTRODUCTION:

This project is submitted as part of final capstone project towards completion of data science certification by HBX. The data set used for this project is 'Credit card fraud detection' from kaggle. Below are few details on the content as described by kaggle:

The datasets contains 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 only 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, the original features and more background information about the data is not provided. 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.

[Courtesy: Kaggle]

OBJECTIVE:

The objective of this project is to identified fraudulent credit card transactions. It is important for credit card companies to be able to identify fraudulent credit card transactions so that customers are not charged for the items they did not purchase.

APPROACH:

The constraints we would soon discover on the data set is that it is an highly imbalanced data set. This means that an higher accuracy number doesn't necessarily translate to a good model. We would need to manipulate the data in such a way that the model generated can help us with our goal of identifying fraudulent transactions. Further, we have used the area under curve as a metric to determine the best model.

The initial step is to load the required libraries and download the dataset. This is followed by exploratory analysis where we learn more on the dataset and its features. During this step we observed the high unbalance that exists in the data set. Hence, before proceeding further with model building exercise, the step undertaken was to incorporate oversampling and undersampling methodologies to convert the dataset to a more balanced one. Later we proceed with building familiar data models that was learnt through this course. Finally, the 'area under curve' is used to determine the best performing mode

STEP 1: INSTALLING THE REQUIRED LIBRARIES

```
knitr::opts_chunk$set(echo = TRUE)

#Include library installations
```

```r
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project")
if(!require(ROSE)) install.packages("ROSE", repos = "http://cran.us.r-project")
if(!require(pROC)) install.packages("pROC", repos = "http://cran.us.r-project")
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project")
if(!require(dplyr)) install.packages("rpart", repos = "http://cran.us.r-project")
if(!require(tibble)) install.packages("rpart", repos = "http://cran.us.r-project")
library(caret)
library(pROC)
library(ROSE)
library(data.table)
library(rpart.plot)
library(randomForest)
library(tidyverse)
library(dplyr)
library(tibble)
```

STEP 2: IMPORTING THE DATA SET

In this section we are loading the dataset directly from online data source.

```r
#Loading data set from online source

temp <- tempfile()
download.file("https://storage.googleapis.com/download.tensorflow.org/data/creditcard.csv", temp)
credit_data <- read.csv(temp) #dataset assigned to credit_data
unlink(temp)
```

STEP 3: EXPLORATORY DATA ANALYSIS

Before building any data science model it is key to build an understanding of the dataset. In the section we try to explore the key parameters of the data set including the number of observations, features etc.

The dataset has overall 284807 observations and 31 variables.We observe that there are 30 variables and one class variables which is an intege. Class variable is Zero for a fair/true or legit transaction and one for fraudulant transaction.As per the dataset, the features V1 to V28 are principle components obtained with PCA. The only features which are not transformed are time and amount (PCA is a linear combination of actual variables done to keep confidentiality of the data)

```r
head(credit_data) #View  the first few observations of the data set
```

```
##   Time         V1          V2        V3         V4          V5          V6
## 1    0 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778
## 2    0  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081
## 3    1 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938
## 4    1 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317
## 5    2 -1.1582331  0.87773675 1.5487178  0.4030339 -0.40719338  0.09592146
## 6    2 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755
##            V7          V8         V9         V10        V11         V12
## 1  0.23959855  0.09869790  0.3637870  0.09079417 -0.5515995 -0.61780086
## 2 -0.07880298  0.08510165 -0.2554251 -0.16697441  1.6127267  1.06523531
## 3  0.79146096  0.24767579 -1.5146543  0.20764287  0.6245015  0.06608369
```

```
## 4  0.23760894  0.37743587 -1.3870241 -0.05495192 -0.2264873  0.17822823
## 5  0.59294075 -0.27053268  0.8177393  0.75307443 -0.8228429  0.53819555
## 6  0.47620095  0.26031433 -0.5686714 -0.37140720  1.3412620  0.35989384
##          V13        V14        V15        V16         V17         V18
## 1 -0.9913898 -0.3111694  1.4681770 -0.4704005  0.20797124  0.02579058
## 2  0.4890950 -0.1437723  0.6355581  0.4639170 -0.11480466 -0.18336127
## 3  0.7172927 -0.1659459  2.3458649 -2.8900832  1.10996938 -0.12135931
## 4  0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.68409279  1.96577500
## 5  1.3458516 -1.1196698  0.1751211 -0.4514492 -0.23703324 -0.03819479
## 6 -0.3580907 -0.1371337  0.5176168  0.4017259 -0.05813282  0.06865315
##           V19         V20          V21          V22         V23         V24
## 1  0.40399296  0.25141210 -0.018306778  0.277837576 -0.11047391  0.06692807
## 2 -0.14578304 -0.06908314 -0.225775248 -0.638671953  0.10128802 -0.33984648
## 3 -2.26185710  0.52497973  0.247998153  0.771679402  0.90941226 -0.68928096
## 4 -1.23262197 -0.20803778 -0.108300452  0.005273597 -0.19032052 -1.17557533
## 5  0.80348692  0.40854236 -0.009430697  0.798278495 -0.13745808  0.14126698
## 6 -0.03319379  0.08496767 -0.208253515 -0.559824796 -0.02639767 -0.37142658
##          V25        V26          V27         V28 Amount Class
## 1  0.1285394 -0.1891148  0.133558377 -0.02105305 149.62     0
## 2  0.1671704  0.1258945 -0.008983099  0.01472417   2.69     0
## 3 -0.3276418 -0.1390966 -0.055352794 -0.05975184 378.66     0
## 4  0.6473760 -0.2219288  0.062722849  0.06145763 123.50     0
## 5 -0.2060096  0.5022922  0.219422230  0.21515315  69.99     0
## 6 -0.2327938  0.1059148  0.253844225  0.08108026   3.67     0
```

```
dim(credit_data)  #Dimensions of the dataset
```

```
## [1] 284807     31
```

```
str(credit_data) #Structure of the data set
```

```
## 'data.frame':    284807 obs. of  31 variables:
##  $ Time  : num  0 0 1 1 2 2 4 7 7 9 ...
##  $ V1    : num  -1.36 1.192 -1.358 -0.966 -1.158 ...
##  $ V2    : num  -0.0728 0.2662 -1.3402 -0.1852 0.8777 ...
##  $ V3    : num  2.536 0.166 1.773 1.793 1.549 ...
##  $ V4    : num  1.378 0.448 0.38 -0.863 0.403 ...
##  $ V5    : num  -0.3383 0.06 -0.5032 -0.0103 -0.4072 ...
##  $ V6    : num  0.4624 -0.0824 1.8005 1.2472 0.0959 ...
##  $ V7    : num  0.2396 -0.0788 0.7915 0.2376 0.5929 ...
##  $ V8    : num  0.0987 0.0851 0.2477 0.3774 -0.2705 ...
##  $ V9    : num  0.364 -0.255 -1.515 -1.387 0.818 ...
##  $ V10   : num  0.0908 -0.167 0.2076 -0.055 0.7531 ...
##  $ V11   : num  -0.552 1.613 0.625 -0.226 -0.823 ...
##  $ V12   : num  -0.6178 1.0652 0.0661 0.1782 0.5382 ...
##  $ V13   : num  -0.991 0.489 0.717 0.508 1.346 ...
##  $ V14   : num  -0.311 -0.144 -0.166 -0.288 -1.12 ...
##  $ V15   : num  1.468 0.636 2.346 -0.631 0.175 ...
##  $ V16   : num  -0.47 0.464 -2.89 -1.06 -0.451 ...
##  $ V17   : num  0.208 -0.115 1.11 -0.684 -0.237 ...
##  $ V18   : num  0.0258 -0.1834 -0.1214 1.9658 -0.0382 ...
##  $ V19   : num  0.404 -0.146 -2.262 -1.233 0.803 ...
##  $ V20   : num  0.2514 -0.0691 0.525 -0.208 0.4085 ...
```

```
##  $ V21   : num  -0.01831 -0.22578 0.248 -0.1083 -0.00943 ...
##  $ V22   : num  0.27784 -0.63867 0.77168 0.00527 0.79828 ...
##  $ V23   : num  -0.11 0.101 0.909 -0.19 -0.137 ...
##  $ V24   : num  0.0669 -0.3398 -0.6893 -1.1756 0.1413 ...
##  $ V25   : num  0.129 0.167 -0.328 0.647 -0.206 ...
##  $ V26   : num  -0.189 0.126 -0.139 -0.222 0.502 ...
##  $ V27   : num  0.13356 -0.00898 -0.05535 0.06272 0.21942 ...
##  $ V28   : num  -0.0211 0.0147 -0.0598 0.0615 0.2152 ...
##  $ Amount: num  149.62 2.69 378.66 123.5 69.99 ...
##  $ Class : int  0 0 0 0 0 0 0 0 0 0 ...
```

STEP 4: VISUALISATION AND INSIGHTS

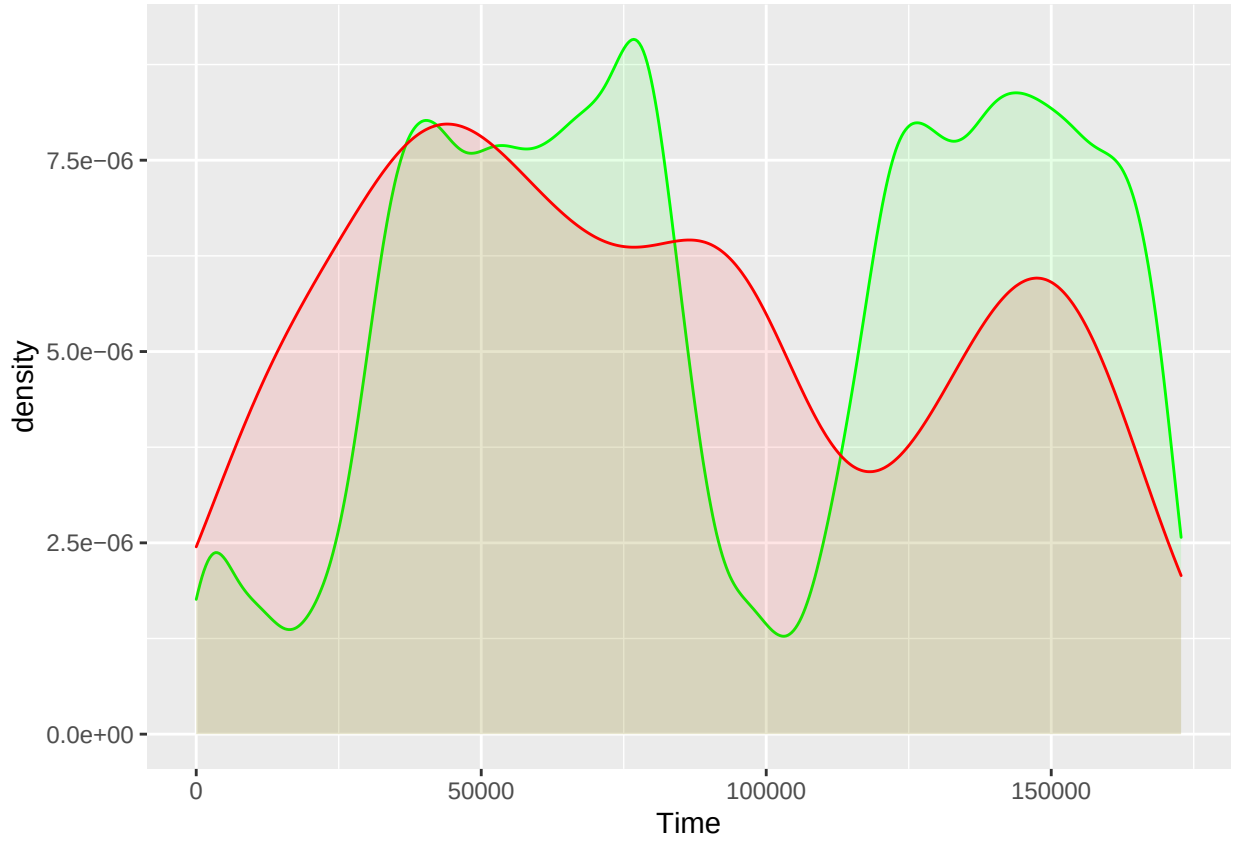
In this section we dive deep into the data set and use visualiosation techniques to generate insights which would later help us in building data models.

4.1: It would be interesting to look at how different features vary on the legitimate (Class = "0") and fraud (Class = "1") transactions. Lets do this first for two physically meaningfull features Time and amount.

Fraud/Legit transactions v/s Time:

From the plaots we observe fall in density of true transation in the beginning of both months (From the dataset its know that the overall period is for 2 months), probably because of the pay/wage/salary being credited.

```
# Splitting the dataset to true/legit and false/fraud transactions

credit_data.true <- credit_data[credit_data$Class ==  0, ]
credit_data.false <- credit_data[credit_data$Class ==  1, ]

#Plotting both true and false observations v/s time

library(ggplot2)
ggplot() +
  geom_density(data = credit_data.true,aes(x=Time), color = "green", fill = "green", alpha = 0.1) +
  geom_density(data = credit_data.false,aes(x=Time), color = "red", fill = "red", alpha = 0.1)
```

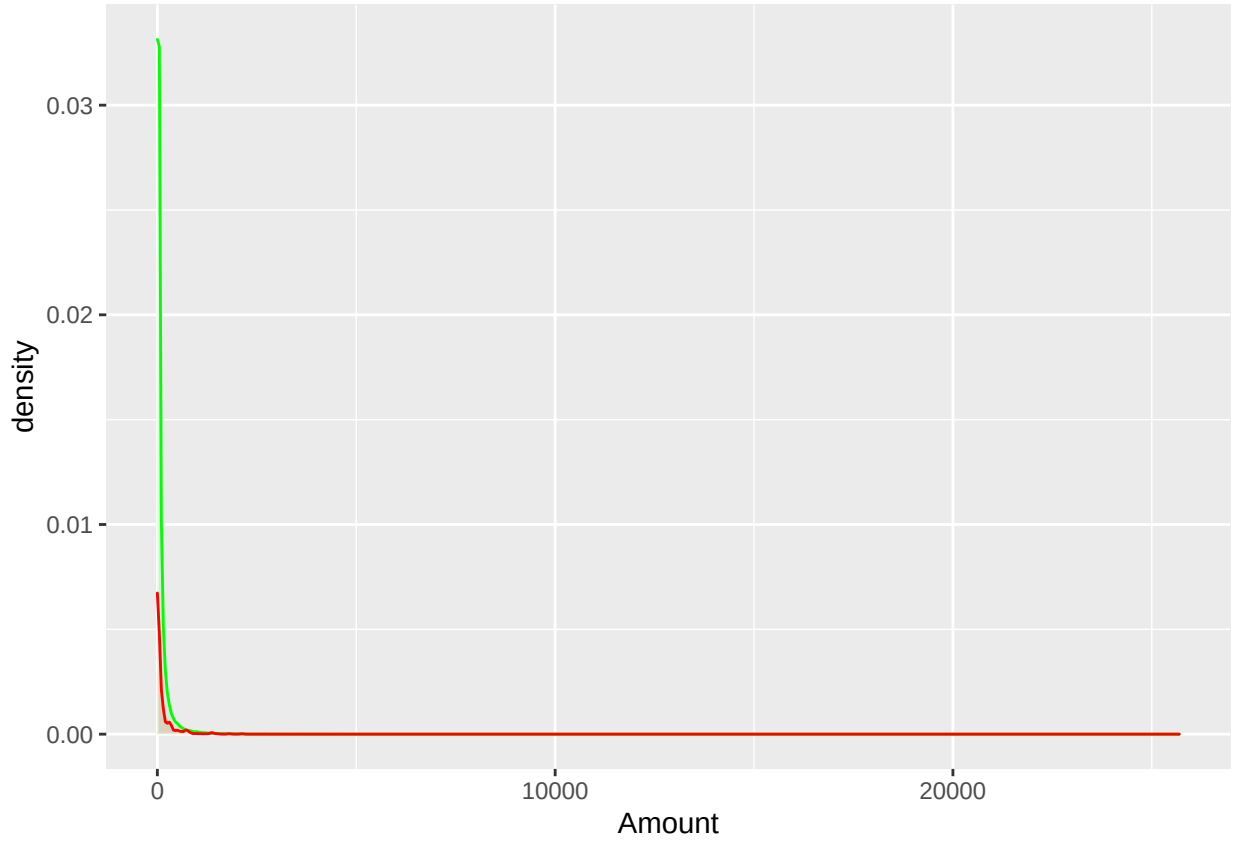



```
#From the dataset its know that the overall period is for 2 months. We see fall in density of true tran
```

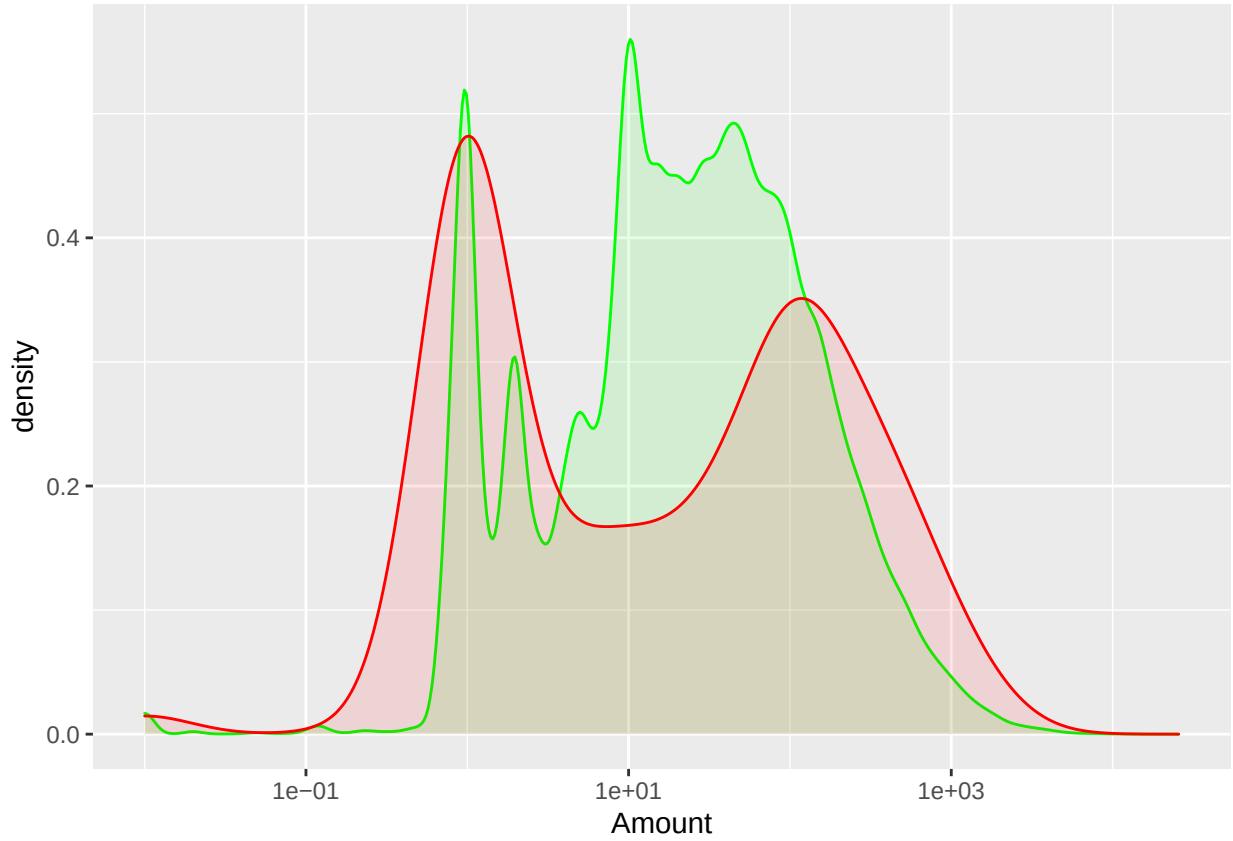
Fraud/Legit transactions v/s Amount

From the plots its observed that the fraudent transactions are seen till higher amounts whereas true transactions drops considerably towards higher amount making the average fradulant transation amount higher than average true transaction

```
#Plotting both true and false observations v/s amount

ggplot() +
  geom_density(data = credit_data.true,aes(x=Amount), color = "green", fill = "green", alpha = 0.1) +
  geom_density(data = credit_data.false,aes(x=Amount), color = "red", fill = "red", alpha = 0.1)
```




```
#Very skewed plot, we may have to change the scale to logarithmic for better insights

ggplot() +
  geom_density(data = credit_data.true,aes(x=Amount), color = "green", fill = "green", alpha = 0.1) +
  geom_density(data = credit_data.false,aes(x=Amount), color = "red", fill = "red", alpha = 0.1) +
  scale_x_continuous(trans = 'log10')
```



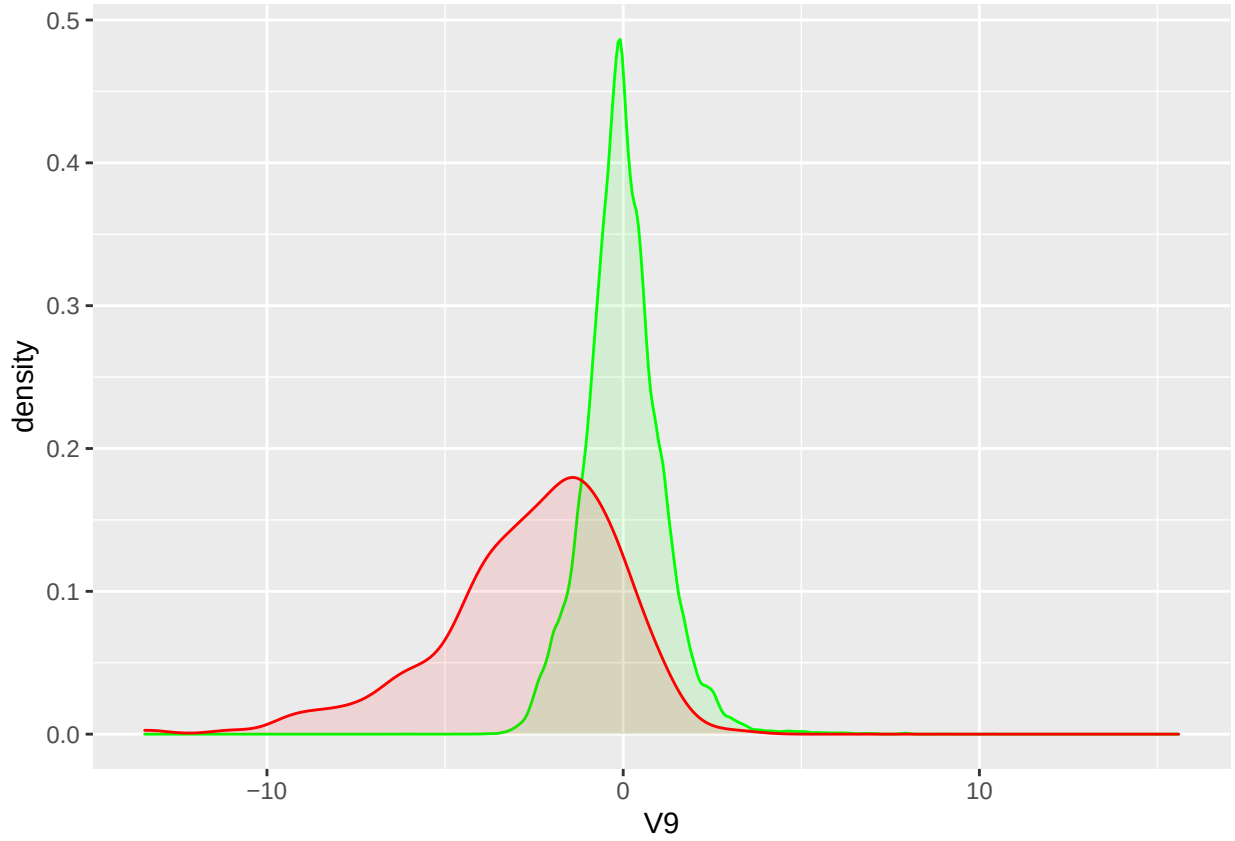

```
#After logrithemic transformation of the x axis, it is interesting to observe that the fraud transactio
```

4.2: Density function of few other features for fraud and legit transactions

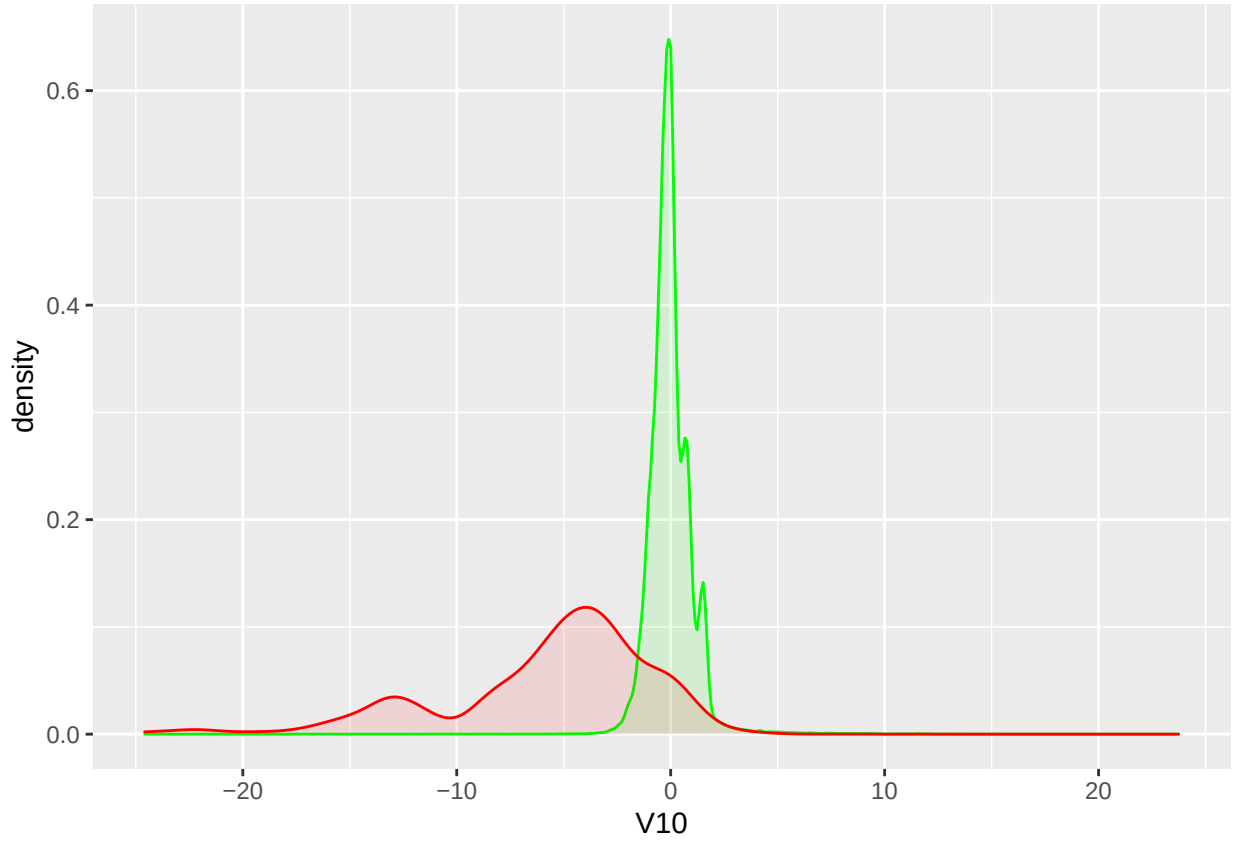
After plotting similar density plots for other features, few important observations from some of these features are being analysed below. Please note that only selected features that provided interesting inputs are being included to keep the analysis brief

1) If the V9 value is less than ~-5 the transation has high probablity of being fraud
2) From the below plots, similar observation as above is seen for featurs V10 and V14
3) The true and fraud density plots for V15 and V25 overlap each other indicating that these features might not much value in the prediction of true/fraud transactions

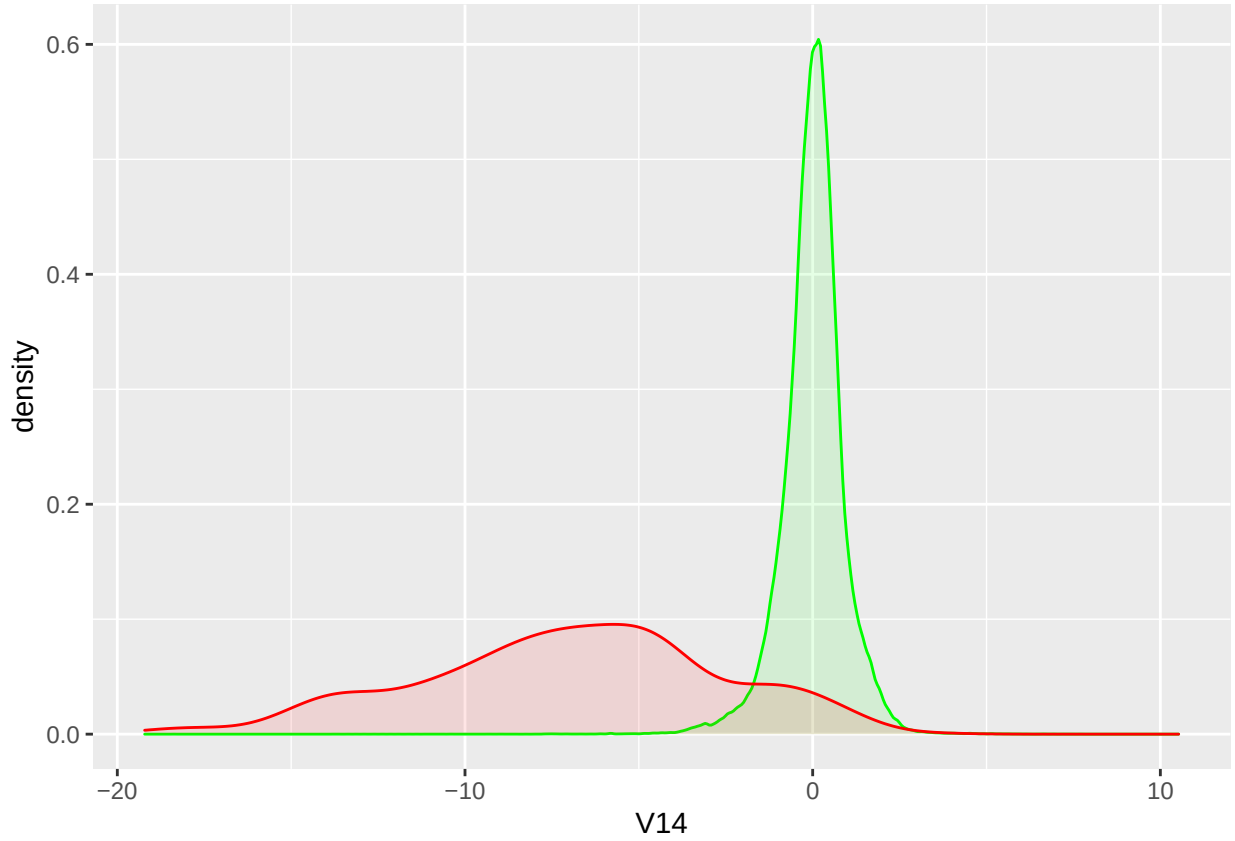
```
#Density plot of Feature V9 for true and fraud transactions

ggplot() +
  geom_density(data = credit_data.true,aes(x=V9), color = "green", fill = "green", alpha = 0.1) +
  geom_density(data = credit_data.false,aes(x=V9), color = "red", fill = "red", alpha = 0.1)
```



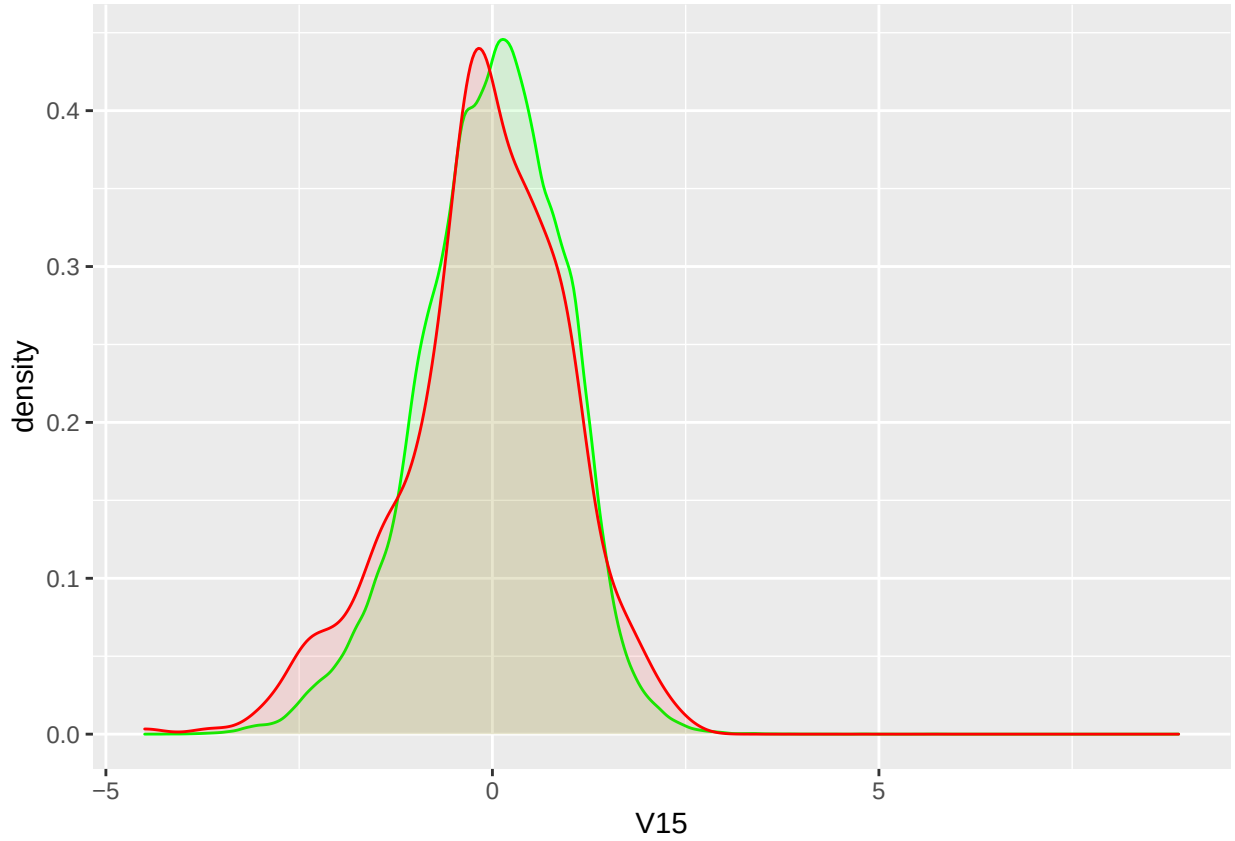

```
#The legitimate transaction has a high peak around 0, whereas most of the fraud transactions are around


#WE OBSERVE SIMILAR BEHAVIOUR FOR FEATURES V10 AND V14

#Density plot of Feature V10 for true and fraud transactions

ggplot() +
  geom_density(data = credit_data.true,aes(x=V10), color = "green", fill = "green", alpha = 0.1) +
  geom_density(data = credit_data.false,aes(x=V10), color = "red", fill = "red", alpha = 0.1)
```



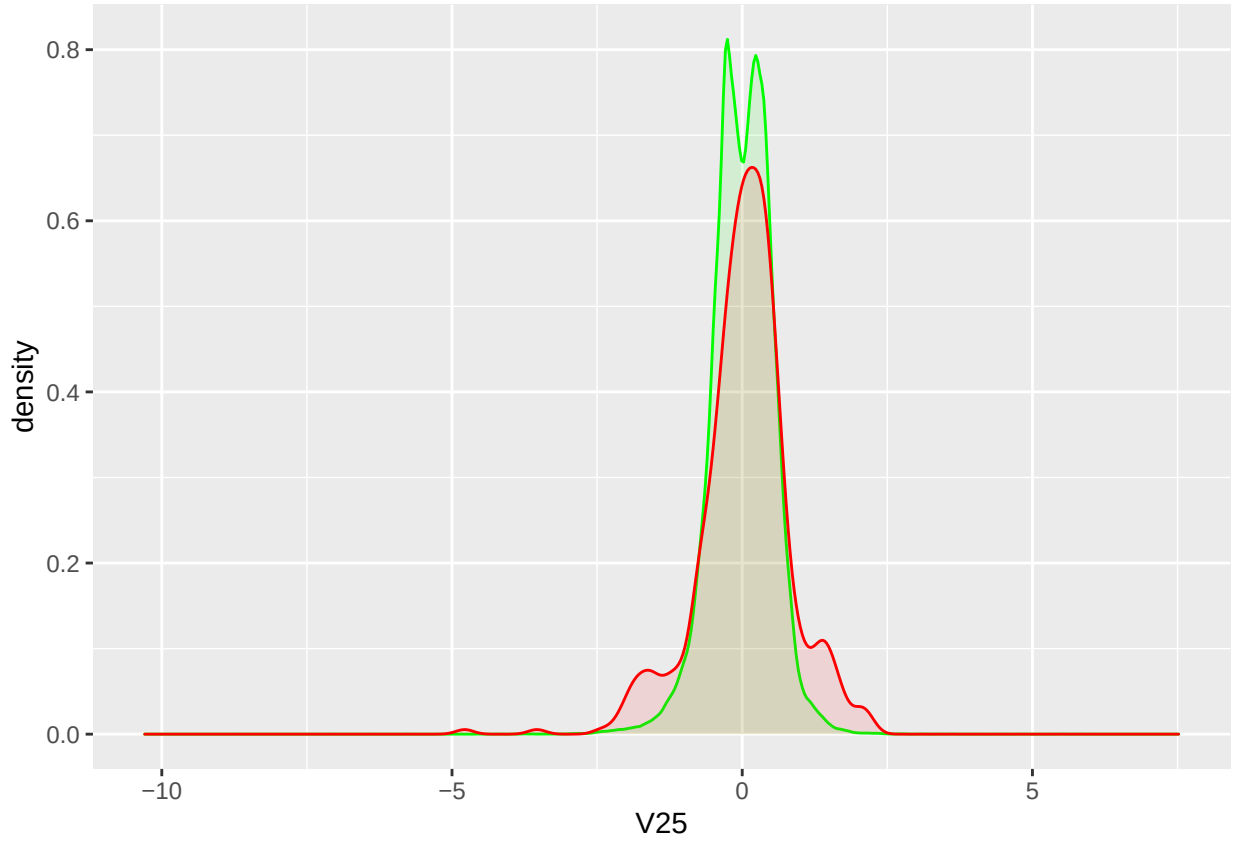

```
#Density plot of Feature V14 for true and fraud transactions

ggplot() +
  geom_density(data = credit_data.true,aes(x=V14), color = "green", fill = "green", alpha = 0.1) +
  geom_density(data = credit_data.false,aes(x=V14), color = "red", fill = "red", alpha = 0.1)
```




```
#ANOTHER INTERESTING OBSERVATION IS SEEN FOR V15 AND V25, LETS PLOT THESE BELOW

#Density plot of Feature V15 for true and fraud transactions

ggplot() +
  geom_density(data = credit_data.true,aes(x=V15), color = "green", fill = "green", alpha = 0.1) +
  geom_density(data = credit_data.false,aes(x=V15), color = "red", fill = "red", alpha = 0.1)
```




```
#Density plot of Feature V25 for true and fraud transactions

ggplot() +
  geom_density(data = credit_data.true,aes(x=V25), color = "green", fill = "green", alpha = 0.1) +
  geom_density(data = credit_data.false,aes(x=V25), color = "red", fill = "red", alpha = 0.1)
```




```
#The graphs for fraud and true transactions overlap considerably for these variables. This indicated th
```

STEP 5: DATA CLEANING AND PREPROCESSING

Data Cleaning comprises of one or more of the following measures to make the data ready for modelling: a) Remove unrelated variables b) Check for unusual entries c) Verify that the variable value makes sense d) Check on missing values and define impute action if needed e) Check if variables need some king of transformation: i)Normalization: Linear scaling of all values between 0 and 1 ii)Logarithmic transformation: taking log of all values iii)Feature selection: choosing a handful of variables to keep the model simmple and transparent None of these transformations like logartimic, normalisation etc are needed here for our dataset, since our model is discrete model which only partiitions data

Checking for class imbalance:

We see that the number of fraud transactions is only approx 1 in 600! Very typical imbalanced dataset. This needs to be taken care else would impact our classification.

The dataset we are dealing with is an imbalanced dataset since the number of fraud transactions are very less when compared to the number of true transactions. As shown below we have only 492 fraudulent transactions against 284315 true transactions.

```
#finding the number of true and fraud transactions

credit_data$Class <- as.factor(credit_data$Class)
summary(credit_data$Class)
```

```
##      0      1
## 284315    492
```

The data set needs to be balanced and below are the three key ways on how we can do it:

Balancing data:

1. Undersampling : Removing many observations of the majority class
2. Oversampling: Create more minority samples
3. Combination of the above two

Splitting dataset to train and test:

```
# Train-Test split of 80:20

set.seed(42, sample.kind = 'Rounding')
test_index <- createDataPartition(y = credit_data$Class, times = 1,
                                  p = 0.2, list = FALSE)
train_set <- credit_data[-test_index,]
test_set <- credit_data[test_index,]
nrow(train_set) #Number of observations in training data set
```

```
## [1] 227845
```

```
nrow(test_set) #Number of observations in test data set
```

```
## [1] 56962
```

Working on the above mentioned methods to balance the data before building the model

Method1: Undersampling : Removing many observations of the majority class. We will reduce the number of legitimate cases and keep it equal to that of fraud cases. However, the disadvantage here is that we end up loosing lots of data points.

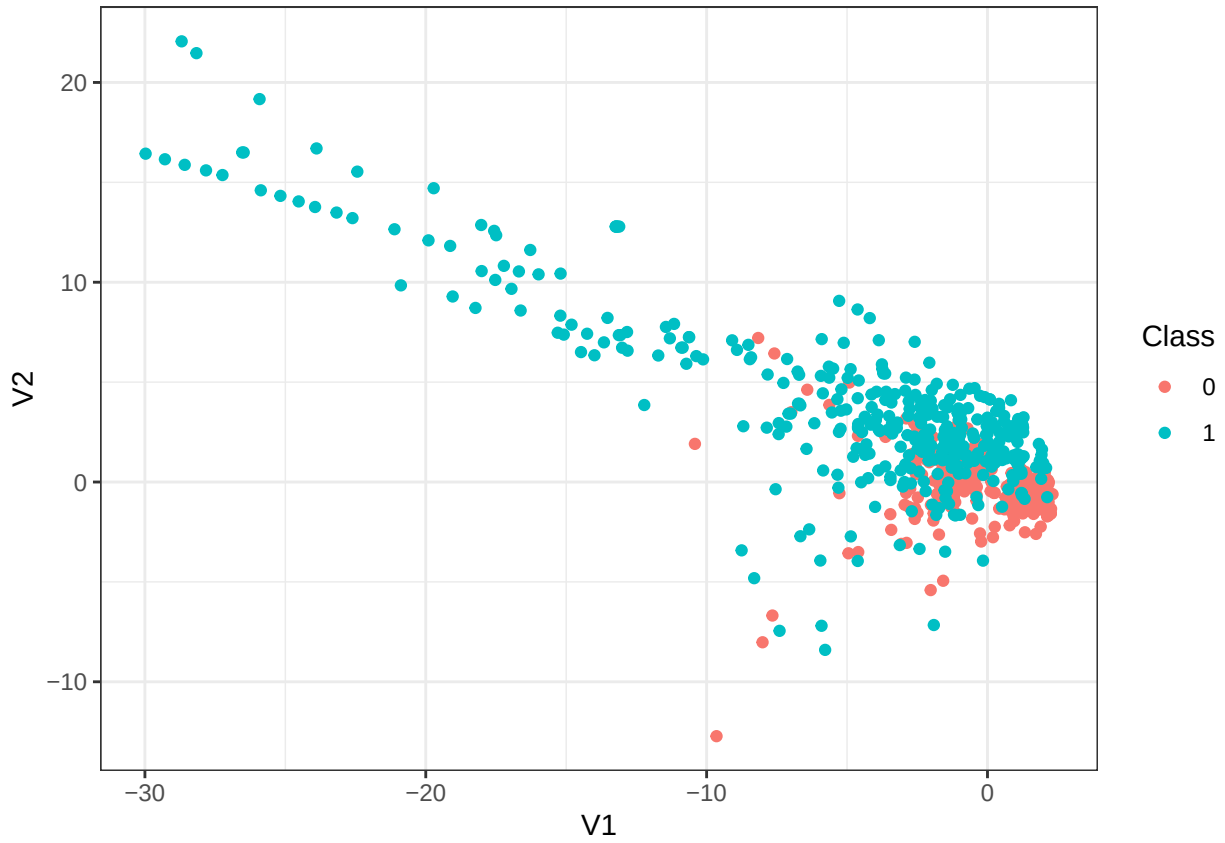
```
summary(train_set$Class) #We observe that there are 393 observations of fraudulent transactions
```

```
##      0      1
## 227452    393
```

```
n_fraud <- 393 #Number of true cases in the train dataset
new_frac_legit <-0.50 #What percentage of the training set do we need true cases to be
new_n_total <- n_fraud/new_frac_legit
undersampling_result <- ovun.sample(Class ~., data = train_set, method ="under", N = new_n_total,seed=2
undersampled_credit <- undersampling_result$data
table(undersampled_credit$Class)  #We now have equal number of true and false (legit and fraud) cases m
```

```
##
##   0   1
## 393 393
```

```
#Lets visualise these points in the newly created undersampled data set

ggplot(data=undersampled_credit, aes(x=V1,y=V2,col=Class))+
  geom_point(position =position_jitter(width=0.1))+
  theme_bw()
```



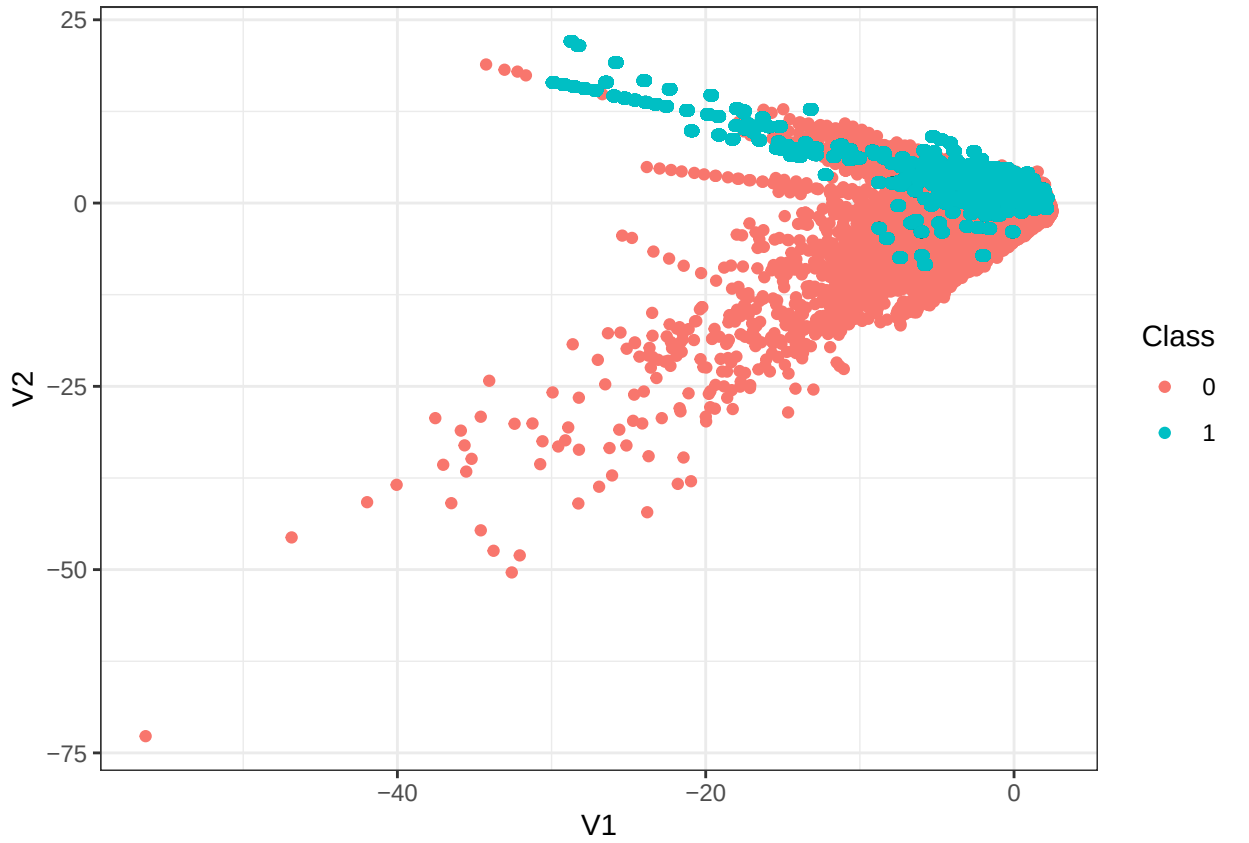
Method2: Oversampling of minority class

Though we have oversampled to have equal number of legit and fraud observations, in the plot we see more number of legit observations.This is because in oversampling, duplicates have been created for fraud cases. Hence the duplicate points would coincide with the original in the plot, hence giving a visual feel of lesser number of points. We have used the position argument to show the duplicate points on the plot

```
n_legit <- 227452 #Number of true cases in the train dataset as seen above
new_frac_legit <-0.50 #What percentage of the training set do we need true cases to be
new_n_total <- n_legit/new_frac_legit
oversampling_result <- ovun.sample(Class ~., data = train_set, method ="over", N = new_n_total,seed=2020
oversampled_credit <- oversampling_result$data
table(oversampled_credit$Class)  #We now have equal number of true and false (legit and fraud) cases ma
```

```
##
##      0      1
## 227452 227452
```

```
#Lets visualise these points

ggplot(data=oversampled_credit, aes(x=V1,y=V2,col=Class))+
  geom_point(position =position_jitter(width=0.1))+
  theme_bw()
```



Method3: Both oversampling and undersampling

This method ensures that we do not loose on data points at the same time ensures we have similiar proportion (50% each) of legit and fraudulent cases in the train data set. Please note that we see the number of fraud cases are overlapping each other since they are duplicates

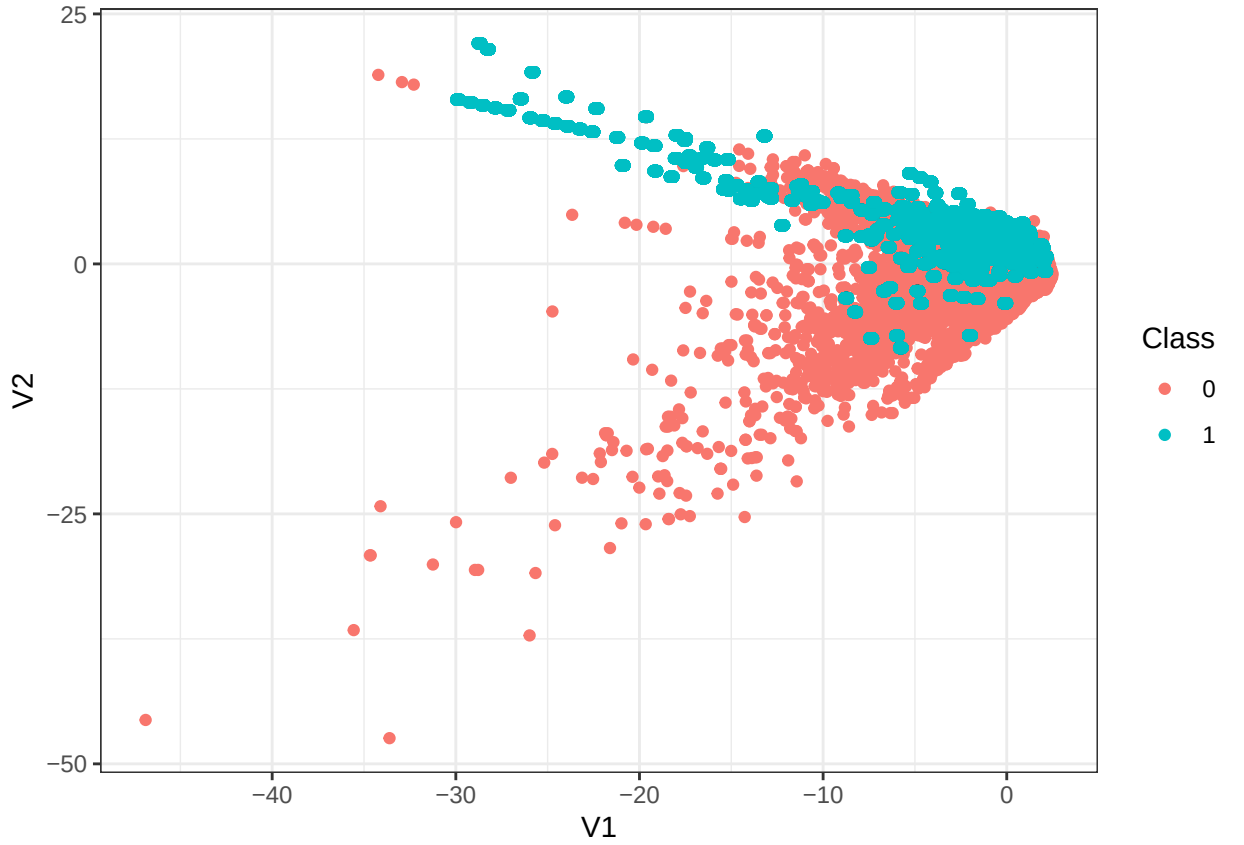
```
new_n <- nrow(train_set)
fraction_fraud_new <-0.50 #What percentage of the training set do we need fraud cases to be
sampling_result <- ovun.sample(Class ~., data = train_set, method ="both", N = new_n, p= fraction_fraud
train_set <- sampling_result$data
dim(train_set) #approx equal number of fraud and legit transactions
```

```
## [1] 227845     31
```

```
prop.table(table(train_set$Class))  #Approx 50% each
```

```
##
##         0         1
## 0.5016876 0.4983124
```

```
#Lets visualise these points

ggplot(data=train_set, aes(x=V1,y=V2,col=Class))+
  geom_point(position =position_jitter(width=0.1))+
  theme_bw()
```



STEP 6: Data modelling

We have accounted for the imbalance existing in the data set and have built a balanced data set that can now be used for building data models.Further the data set has been split into training and test in the previous section.

As described earlier the objective of modelling is to predict fraudulent transactions from the data set. In order to achieve this objective we would be building four models : Logistic Regression, KNN, Decision tree and Random forrest.

Further, we would be using area under curve as the metric to pick the best model that can be applied to solve this data science problem

MODEL1: LOGISTIC REGRESSION

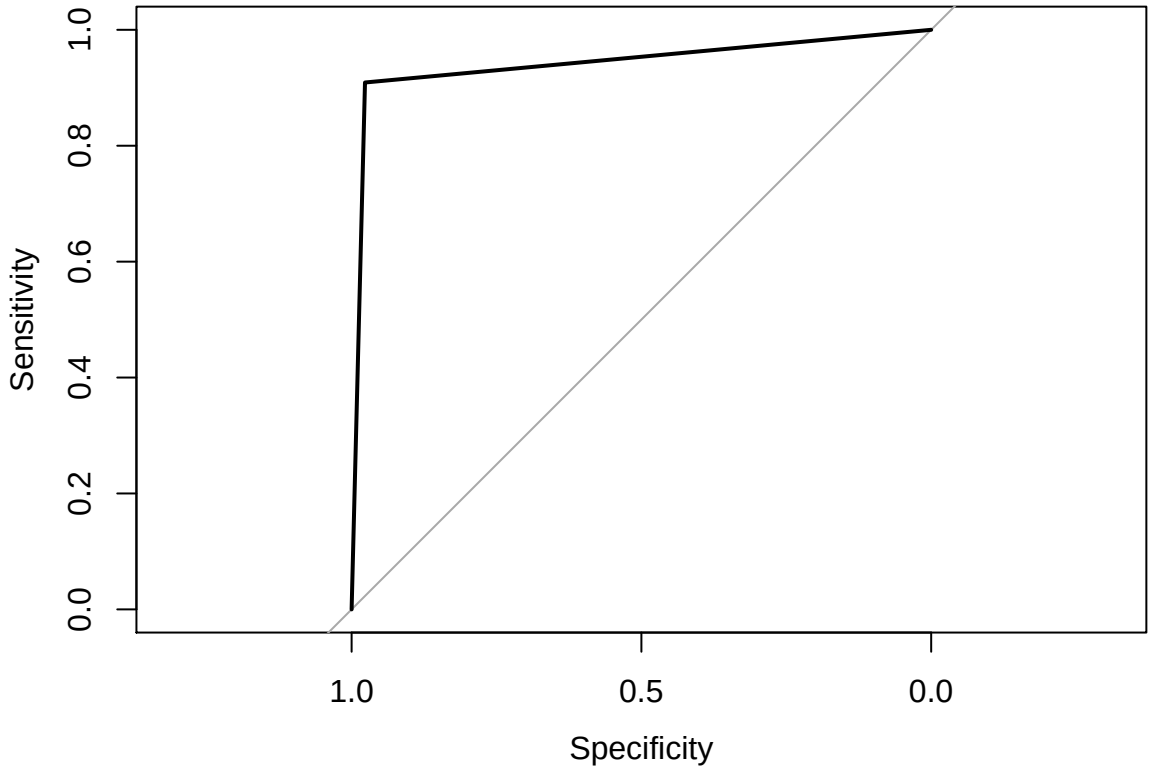
As we know the primary objective of a classification model is to predict the probablity of an observation belonging to a class. Logistic regression is a statistical model in which the response variable takes a discrete value and the explanatory variable can be either continuous or discrete.

```
train_glm <- train(Class ~ ., method = "glm", data = train_set)  #glm: Generalised Linear model. The le
Class_hat_glm <- predict(train_glm, test_set, type = "prob") #Predicting the probablities
test_set$pred<- 0L
test_set$pred[Class_hat_glm[,2]>0.5] <- 1L  # prob>0.5 is assigned to 1 class
test_set$pred <-factor(test_set$pred )
confusionMatrix(data = test_set$pred, reference = test_set$Class)  #building confusion matrix
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction     0     1
##          0 55551     9
##          1  1312    90
##
##                Accuracy : 0.9768
##                  95% CI : (0.9755, 0.978)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1171
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.97693
##             Specificity : 0.90909
##          Pos Pred Value : 0.99984
##          Neg Pred Value : 0.06419
##              Prevalence : 0.99826
##          Detection Rate : 0.97523
##    Detection Prevalence : 0.97539
##       Balanced Accuracy : 0.94301
##
##        'Positive' Class : 0
##
```

```
test_set$Class <- ordered(test_set$Class,levels = c("0","1"))
test_set$pred <- ordered(test_set$pred,levels = c("0","1"))
roc <- roc(test_set$Class, test_set$pred) #ROC curve is a plot between sesitivity (true positive rate)
plot(roc)
```




```
auc(roc) #Returns the area under the curve (auc). Model with higher AUC is preferred
```

```
## Area under the curve: 0.943
```

MODEL2: KNN (K - Nearest Neighbours)

K-Nearest Neighbors (KNN) algorithm is a non-parametric, learning algorithm used for regression or classification problems. It is called a non-parametric model since it does not make any assumptions on the underlying data distribution. KNN memorizes the data and classifies new observations by comnparing the training data.

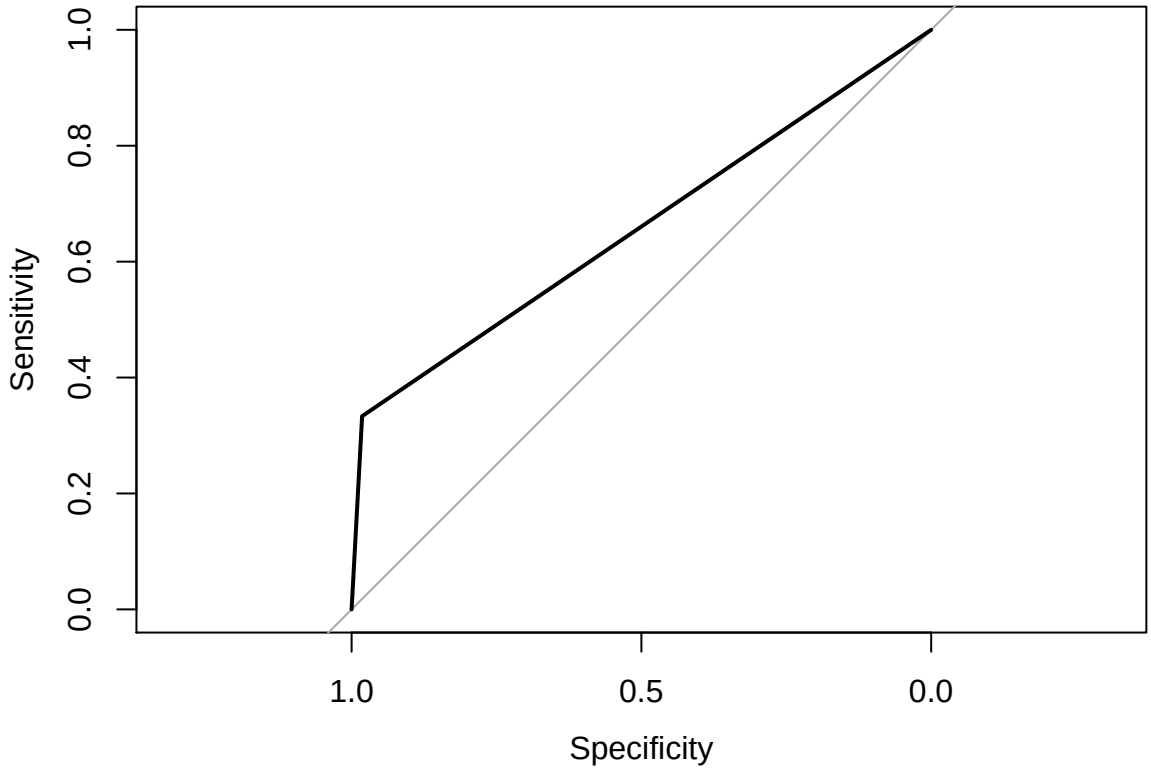
Since the computation time is high when using the entire data set, we have scaled the training set to be only 50% of the original train set. We would be loosing on data points however ensures computation is done in a reasonable time frame.

```
scaled_train_set <- train_set %>% sample_frac(0.5) #Due to higher computation time we are using only 50
scaled_test_set <- test_set %>% sample_frac(1.0)
fit_knn<- knn3(Class ~.,data = scaled_train_set) #Model fitting
pred_knn <- predict(fit_knn, scaled_test_set, type = "prob") #predicting probablities
scaled_test_set$pred<- 0L
scaled_test_set$pred[pred_knn[,2]>0.5] <- 1L # prob>0.5 is assigned to 1 class
scaled_test_set$pred <-factor(scaled_test_set$pred )
confusionMatrix(data = scaled_test_set$pred, reference = scaled_test_set$Class) #Building confusion mat
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction     0     1
##          0 55833    66
##          1  1030    33
##
##                Accuracy : 0.9808
##                  95% CI : (0.9796, 0.9819)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0538
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.98189
##             Specificity : 0.33333
##          Pos Pred Value : 0.99882
##          Neg Pred Value : 0.03104
##              Prevalence : 0.99826
##          Detection Rate : 0.98018
##    Detection Prevalence : 0.98134
##       Balanced Accuracy : 0.65761
##
##        'Positive' Class : 0
##
```

```
scaled_test_set$Class <- ordered(scaled_test_set$Class,levels = c("0","1"))
scaled_test_set$pred <- ordered(scaled_test_set$pred,levels = c("0","1"))
roc <- roc(scaled_test_set$Class, scaled_test_set$pred) #ROC curve is a plot between sesitivity (true p
plot(roc)
```




```
auc(roc) #Returns the area under the curve (auc). Model with higher AUC is preferred
```

```
## Area under the curve: 0.6576
```

MODEL3: DECISION TREE

A decision tree is one of the simmplest yet effective models for classification. The best thing about the model is its transprency/interpretability. The domain opens up logical interpretation when a DT is built. The decision on branching is decided by Gini index or entropy. The R package rpart is used below for building decision tree

```
fit_rpart <- rpart(Class ~ .,
                   data=train_set,
                   method = "class") #Model fitting

Class_hat_rpart <- predict(fit_rpart,test_set) #predicting probablities
test_set$pred<- 0L
test_set$pred[Class_hat_rpart[,2]>0.5] <- 1L # prob>0.5 is assigned to 1 class
test_set$pred <-factor(test_set$pred )
confusionMatrix(data = test_set$pred, reference = test_set$Class) #Building confusion matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
```

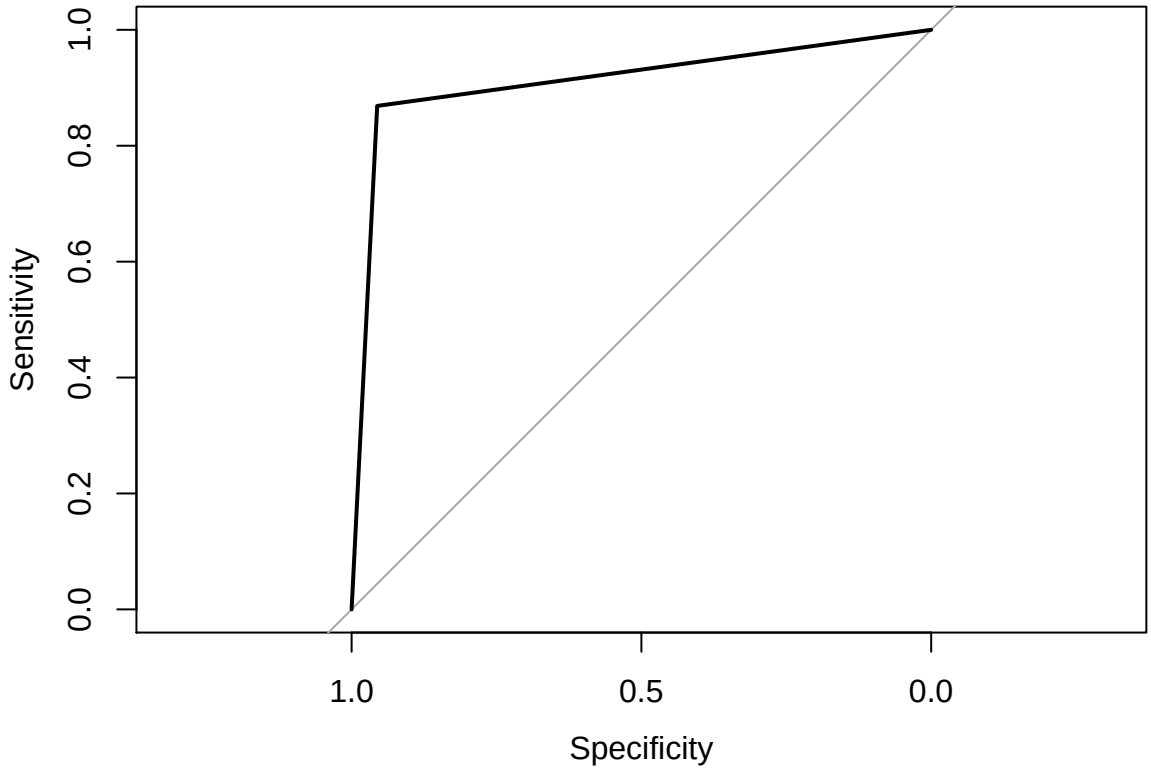
```
##          0 54354    13
##          1  2509    86
##
##                Accuracy : 0.9557
##                  95% CI : (0.954, 0.9574)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0607
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.95588
##             Specificity : 0.86869
##          Pos Pred Value : 0.99976
##          Neg Pred Value : 0.03314
##              Prevalence : 0.99826
##          Detection Rate : 0.95422
##    Detection Prevalence : 0.95444
##       Balanced Accuracy : 0.91228
##
##        'Positive' Class : 0
##
```

```
test_set$Class <- ordered(test_set$Class,levels = c("0","1"))
test_set$pred <- ordered(test_set$pred,levels = c("0","1"))
roc <- roc(test_set$Class, test_set$pred) #ROC curve is a plot between sesitivity (true positive rate)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

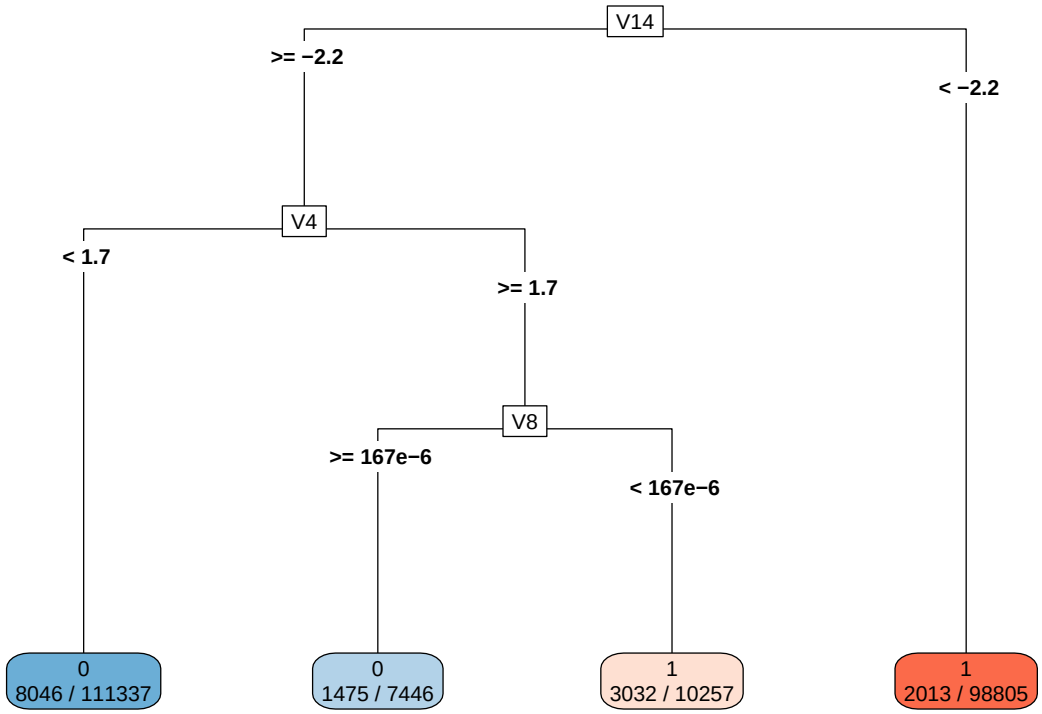
```
plot(roc)
```




```
auc(roc) #Returns the area under the curve (auc). Model with higher AUC is preferred
```

```
## Area under the curve: 0.9123
```

```
rpart.plot(fit_rpart,cex=0.66,extra=3,type=5,box.palette="BuRd") #Plotting the decision tree
```



MODEL4: RANDOM FORREST

Random forrest as the name suggest builds a bunch of trees to improve model accuracy. Random forrest can help reduce uncertainty in predictions that may exist in a decision tree. The R package randomForest gives good Random Forests with properly set tuning parameters:

1) ntree = 10 to 100
2) maxnodes = 30 to 70

Randomly picked up values of 39 trees and 44 maxnodes have been taken for building the model. These tuning parameters can be decided through Gridsearch, however given the very high computation time involved, for this project we have picked these values by random.

```
fit_rf <- randomForest(Class ~., data = train_set,  importance = TRUE, ntree=39, maxnodes=44) #fitting
Class_hat_rf <- predict(fit_rf,test_set,type = "prob") #predicting probablities
test_set$pred<- 0L
test_set$pred[Class_hat_rf[,2] >0.5] <- 1L # prob>0.5 is assigned to 1 class
test_set$pred <-factor(test_set$pred )
confusionMatrix(data = test_set$pred, reference = test_set$Class) #Building confusion matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 56751    12
##          1   112    87
```

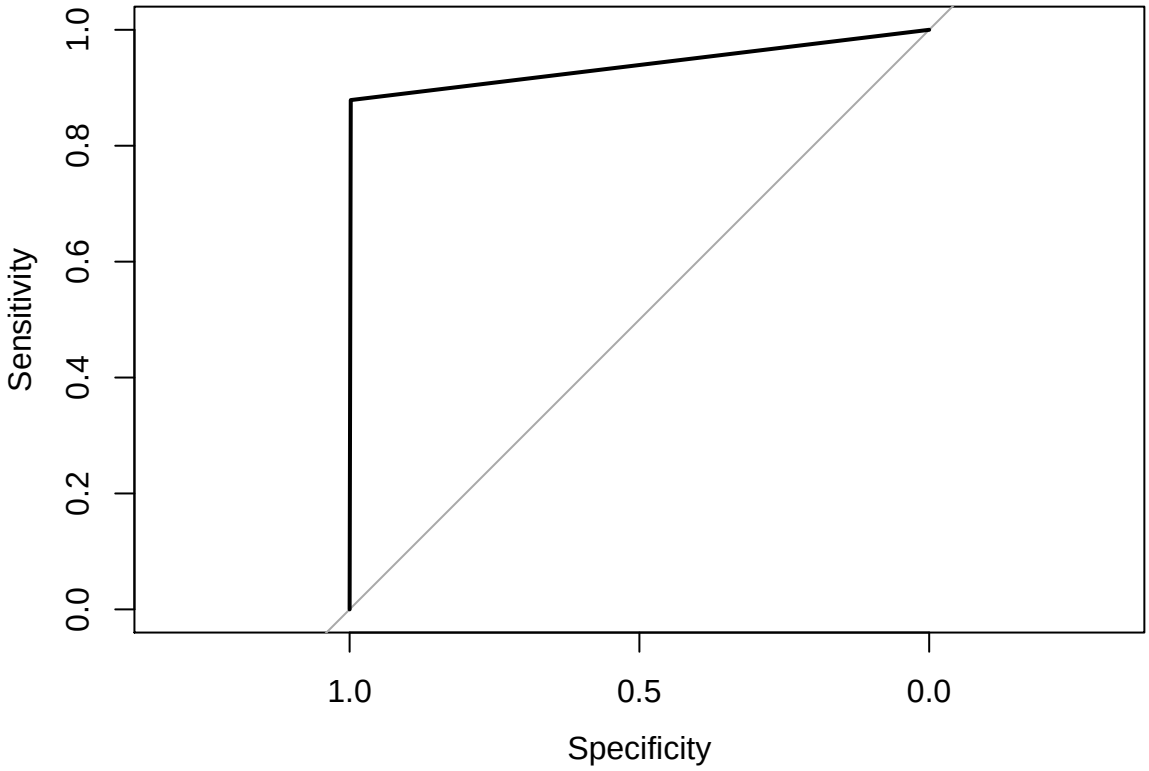
```
##
##                Accuracy : 0.9978
##                  95% CI : (0.9974, 0.9982)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 0.9935
##
##                   Kappa : 0.5829
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9980
##             Specificity : 0.8788
##          Pos Pred Value : 0.9998
##          Neg Pred Value : 0.4372
##              Prevalence : 0.9983
##          Detection Rate : 0.9963
##    Detection Prevalence : 0.9965
##       Balanced Accuracy : 0.9384
##
##        'Positive' Class : 0
##
```

```
test_set$Class <- ordered(test_set$Class,levels = c("0","1"))
test_set$pred <- ordered(test_set$pred,levels = c("0","1"))
roc <- roc(test_set$Class, test_set$pred)  #ROC curve is a plot between sesitivity (true positive rate)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc)
```




```
auc(roc)  #Returns the area under the curve (auc). Model with higher AUC is preferred
```

```
## Area under the curve: 0.9384
```

STEP 6: RESULTS

We build logsitic regression, KNN, decision tree and random forrest models for prediction of fraudulent/legit classes. Given the large size of the dataset some of these models took considerable time to run when the tuning parameters where set through Gridsearch. Hence, for this final report we have directly input the optimal values for few hyperparameters.

The area under curve is used as the metric to evaluate performance of the models. Logistic regression and random forrest provides us with the best auc figures of 0.9429 and 0.9382 respectively. A look at the confusion matrix tells us that we are able to correctly classify about 90.9% and 87.8% respectively of the fraudulent transactions through these models.

STEP 7: CONCLUSION

a) Summary: Through this project we learnt on how to identify fraudulent transaction from a credit card transactions data set. The challenging part was the high imbalance which existing in the data set. However, we were able to solve for it through balancing methodologies and preferred to use a combination of oversampling and undersampling methodologies.

The data exploration and visualisation gave us good insights around certain features that would be good predictors and on some that does not contribute much to any prediction algorithm.

Of the models build, logistic regression and random forrest gave us high value for area under curve and emerged as better predictor algorithms for this project.

b) Potential Impact: With the increase in digital transactions across the globe fraudulent activities are on the rise. This model has great impact particularly for finacial institutions like banks to prevent fraudulent transactions and hence protect their customers. Being able to predict a possible fraudulent transactions realtime and proceeding with either blocking the transaction, contacting the customer to verify legitimacy or even studying the trends and building further system checks to prevent re-occurance in the future are all practical applications of this model

c) Limitations: Given the large size of the dataset there are multiple limitation when building the model particularly with respect to the computation power of local machine. We had to manually set values for the tuning parameters to save on the computation time which is definitely not the best practise

d) Future work: With better computation power we could build much more advanced models like aritificial neural networks, ensemble methodologies, SVM, gradient boosted tree etc that would give us much better results.