Detecting credit card fraud using random forest machine learning approach

Chukwuemerie Okechukwu Okoli

2022-09-14

#LIBRARY #————————–

#This are the libraries to run the research project. #The library that are not installed on the R program, we use #the install.packages(name) to install it. #————————————————–

library(dplyr) # for data manipulation

## Warning: package 'dplyr' was built under R version 4.1.3

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(stringr) # for data manipulation  
library(caret) # for sampling

## Warning: package 'caret' was built under R version 4.1.3

## Loading required package: ggplot2

## Loading required package: lattice

library(caTools) # for train/test split

## Warning: package 'caTools' was built under R version 4.1.3

library(ggplot2) # for data visualization  
library(corrplot) # for correlations

## Warning: package 'corrplot' was built under R version 4.1.3

## corrplot 0.92 loaded

library(Rtsne) # for tsne plotting

## Warning: package 'Rtsne' was built under R version 4.1.3

library(ROSE)# for ROSE sampling

## Warning: package 'ROSE' was built under R version 4.1.3

## Loaded ROSE 0.0-4

library(rpart)  
library(DMwR) # for SMOTE FUNCTION

## Loading required package: grid

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

library(Rborist)# for random forest model

## Warning: package 'Rborist' was built under R version 4.1.3

## Rborist 0.2-3

## Type RboristNews() to see new features/changes/bug fixes.

library(xgboost) # for xgboost model

## Warning: package 'xgboost' was built under R version 4.1.3

##   
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':  
##   
## slice

library("randomForest")

## Warning: package 'randomForest' was built under R version 4.1.3

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

## The following object is masked from 'package:dplyr':  
##   
## combine

#SETTING THE PLOT #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# The code below enable us to

# to set plot height and width

fig <- function(width, heigth){  
 options(repr.plot.width = width, repr.plot.height = heigth)  
}

#DATA PREPARATION #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#before we start our , we need to load the dataset # to our system. The Dataset for this research is the Credit #card dataset # loading the data

creditcard = read.csv('creditcard.csv')

#DATA EXPLORATION #—————————

#We start with exploring the data

head(creditcard)

## 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

# we need to see how the dataset is structured

str(creditcard)

## '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 ...

#We need to see the summary description of the dataset

summary(creditcard)

## Time V1 V2 V3   
## Min. : 0 Min. :-56.40751 Min. :-72.71573 Min. :-48.3256   
## 1st Qu.: 54202 1st Qu.: -0.92037 1st Qu.: -0.59855 1st Qu.: -0.8904   
## Median : 84692 Median : 0.01811 Median : 0.06549 Median : 0.1799   
## Mean : 94814 Mean : 0.00000 Mean : 0.00000 Mean : 0.0000   
## 3rd Qu.:139321 3rd Qu.: 1.31564 3rd Qu.: 0.80372 3rd Qu.: 1.0272   
## Max. :172792 Max. : 2.45493 Max. : 22.05773 Max. : 9.3826   
## V4 V5 V6 V7   
## Min. :-5.68317 Min. :-113.74331 Min. :-26.1605 Min. :-43.5572   
## 1st Qu.:-0.84864 1st Qu.: -0.69160 1st Qu.: -0.7683 1st Qu.: -0.5541   
## Median :-0.01985 Median : -0.05434 Median : -0.2742 Median : 0.0401   
## Mean : 0.00000 Mean : 0.00000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.74334 3rd Qu.: 0.61193 3rd Qu.: 0.3986 3rd Qu.: 0.5704   
## Max. :16.87534 Max. : 34.80167 Max. : 73.3016 Max. :120.5895   
## V8 V9 V10 V11   
## Min. :-73.21672 Min. :-13.43407 Min. :-24.58826 Min. :-4.79747   
## 1st Qu.: -0.20863 1st Qu.: -0.64310 1st Qu.: -0.53543 1st Qu.:-0.76249   
## Median : 0.02236 Median : -0.05143 Median : -0.09292 Median :-0.03276   
## Mean : 0.00000 Mean : 0.00000 Mean : 0.00000 Mean : 0.00000   
## 3rd Qu.: 0.32735 3rd Qu.: 0.59714 3rd Qu.: 0.45392 3rd Qu.: 0.73959   
## Max. : 20.00721 Max. : 15.59500 Max. : 23.74514 Max. :12.01891   
## V12 V13 V14 V15   
## Min. :-18.6837 Min. :-5.79188 Min. :-19.2143 Min. :-4.49894   
## 1st Qu.: -0.4056 1st Qu.:-0.64854 1st Qu.: -0.4256 1st Qu.:-0.58288   
## Median : 0.1400 Median :-0.01357 Median : 0.0506 Median : 0.04807   
## Mean : 0.0000 Mean : 0.00000 Mean : 0.0000 Mean : 0.00000   
## 3rd Qu.: 0.6182 3rd Qu.: 0.66251 3rd Qu.: 0.4931 3rd Qu.: 0.64882   
## Max. : 7.8484 Max. : 7.12688 Max. : 10.5268 Max. : 8.87774   
## V16 V17 V18   
## Min. :-14.12985 Min. :-25.16280 Min. :-9.498746   
## 1st Qu.: -0.46804 1st Qu.: -0.48375 1st Qu.:-0.498850   
## Median : 0.06641 Median : -0.06568 Median :-0.003636   
## Mean : 0.00000 Mean : 0.00000 Mean : 0.000000   
## 3rd Qu.: 0.52330 3rd Qu.: 0.39968 3rd Qu.: 0.500807   
## Max. : 17.31511 Max. : 9.25353 Max. : 5.041069   
## V19 V20 V21   
## Min. :-7.213527 Min. :-54.49772 Min. :-34.83038   
## 1st Qu.:-0.456299 1st Qu.: -0.21172 1st Qu.: -0.22839   
## Median : 0.003735 Median : -0.06248 Median : -0.02945   
## Mean : 0.000000 Mean : 0.00000 Mean : 0.00000   
## 3rd Qu.: 0.458949 3rd Qu.: 0.13304 3rd Qu.: 0.18638   
## Max. : 5.591971 Max. : 39.42090 Max. : 27.20284   
## V22 V23 V24   
## Min. :-10.933144 Min. :-44.80774 Min. :-2.83663   
## 1st Qu.: -0.542350 1st Qu.: -0.16185 1st Qu.:-0.35459   
## Median : 0.006782 Median : -0.01119 Median : 0.04098   
## Mean : 0.000000 Mean : 0.00000 Mean : 0.00000   
## 3rd Qu.: 0.528554 3rd Qu.: 0.14764 3rd Qu.: 0.43953   
## Max. : 10.503090 Max. : 22.52841 Max. : 4.58455   
## V25 V26 V27   
## Min. :-10.29540 Min. :-2.60455 Min. :-22.565679   
## 1st Qu.: -0.31715 1st Qu.:-0.32698 1st Qu.: -0.070840   
## Median : 0.01659 Median :-0.05214 Median : 0.001342   
## Mean : 0.00000 Mean : 0.00000 Mean : 0.000000   
## 3rd Qu.: 0.35072 3rd Qu.: 0.24095 3rd Qu.: 0.091045   
## Max. : 7.51959 Max. : 3.51735 Max. : 31.612198   
## V28 Amount Class   
## Min. :-15.43008 Min. : 0.00 Min. :0.000000   
## 1st Qu.: -0.05296 1st Qu.: 5.60 1st Qu.:0.000000   
## Median : 0.01124 Median : 22.00 Median :0.000000   
## Mean : 0.00000 Mean : 88.35 Mean :0.001728   
## 3rd Qu.: 0.07828 3rd Qu.: 77.17 3rd Qu.:0.000000   
## Max. : 33.84781 Max. :25691.16 Max. :1.000000

#We need to check if the dataset has a missing value in any #of the variables # checking missing values

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ #Fortunately for us the output shows that the dataset has # no missing values in any of the variables #The next step is to check if the dataset is balance

colSums(is.na(creditcard))

## Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10   
## 0 0 0 0 0 0 0 0 0 0 0   
## V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21   
## 0 0 0 0 0 0 0 0 0 0 0   
## V22 V23 V24 V25 V26 V27 V28 Amount Class   
## 0 0 0 0 0 0 0 0 0

#CHECKING THE IMBALANCE DATASET #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# checking class imbalance

table(creditcard$Class)

##   
## 0 1   
## 284315 492

# class imbalance in percentage

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ #Looking at the output for the code above, we can see that #the dataset is highly imbalance with 284315 non fraud #transactions and 492 fraud transactions which gives a # percentage of 99% for non fraud and 1% for fraud transactions #and this will surely lead # to biased prediction using the Random forest algorithm #We need to look for a way to balance the dataset.

prop.table(table(creditcard$Class))

##   
## 0 1   
## 0.998272514 0.001727486

#GRAPH VISUALIZATION #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

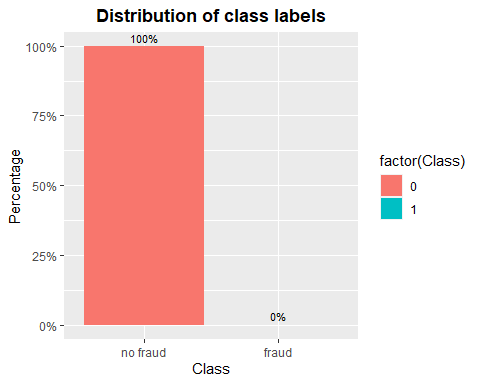
#————————— #We have to still look at the dataset in a visual form #The ggplot library is used to show the graphical representation # of how the class are categorized

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#The graph below clearly shows # that the dataset is highly imbalanced # A simple measure like accuracy is not appropriate here #as even a classifier which labels all transactions #as non-fraudulent will have over 99% accuracy. #An appropriate measure of model performance #here would be AUC (Area Under the Precision-Recall Curve)

#—————————————————

fig(12, 8)  
common\_theme <- theme(plot.title = element\_text(hjust = 0.5, face = "bold"))  
  
ggplot(data = creditcard, aes(x = factor(Class),   
 y = prop.table(stat(count)), fill = factor(Class),  
 label = scales::percent(prop.table(stat(count))))) +  
 geom\_bar(position = "dodge") +   
 geom\_text(stat = 'count',  
 position = position\_dodge(.9),   
 vjust = -0.5,   
 size = 3) +   
 scale\_x\_discrete(labels = c("no fraud", "fraud"))+  
 scale\_y\_continuous(labels = scales::percent)+  
 labs(x = 'Class', y = 'Percentage') +  
 ggtitle("Distribution of class labels") +  
 common\_theme



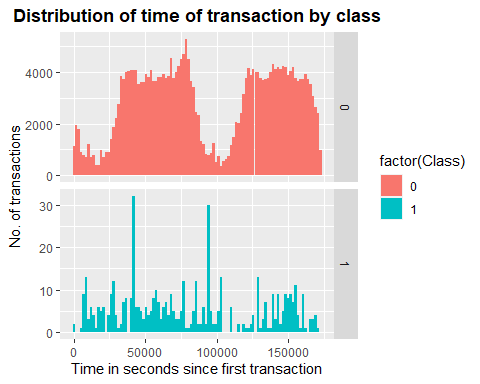
#DATA VISUALIZATION FOR DEPENDENT VARIABLE BY INDEPENDENT VARIABLE #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#We will look at more data visualization between the #independent variable and the dependent variables #we need to see how their relationship

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ #Looking at the graph below # we can see that The ‘Time’ feature looks pretty #similar across both types of transactions. #One could argue that fraudulent transactions #are more uniformly distributed, #while normal transactions have a cyclical distribution

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

fig(14, 8)  
creditcard %>%  
 ggplot(aes(x = Time, fill = factor(Class))) + geom\_histogram(bins = 100)+  
 labs(x = 'Time in seconds since first transaction', y = 'No. of transactions') +  
 ggtitle('Distribution of time of transaction by class') +  
 facet\_grid(Class ~ ., scales = 'free\_y') + common\_theme

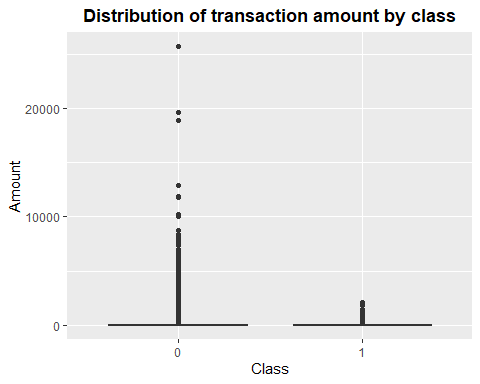


#Lets look at the relationship between amount by Class

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

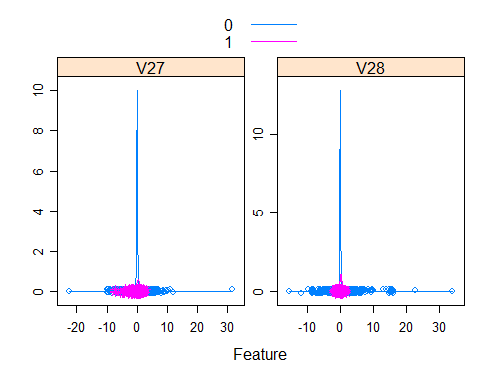
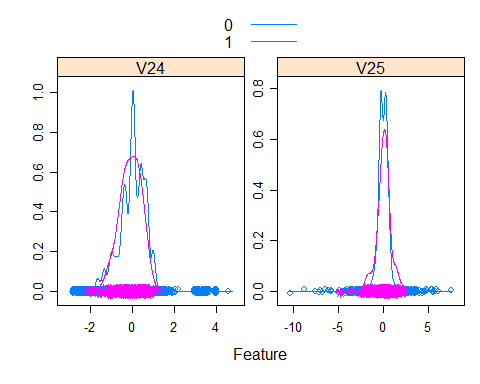
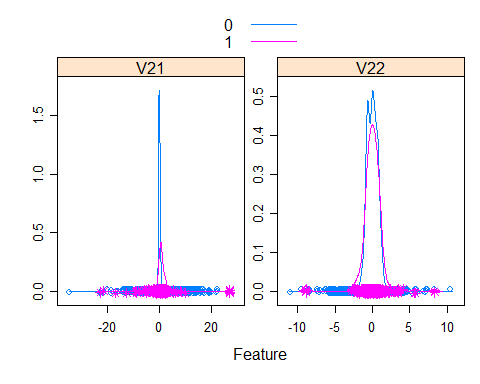
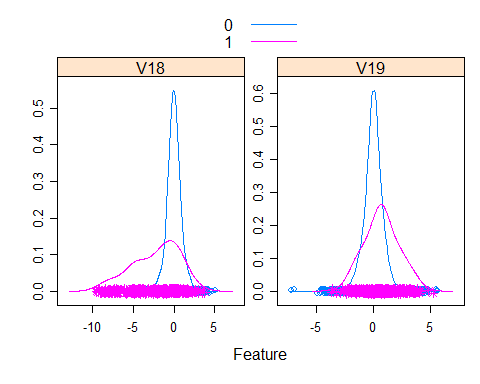
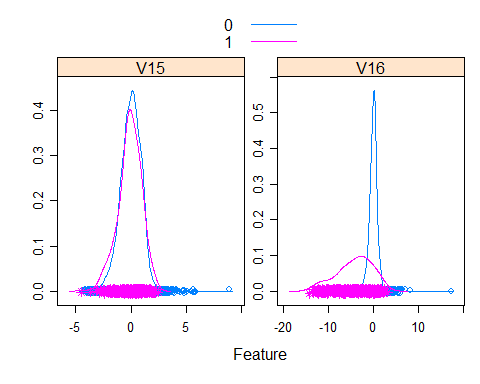
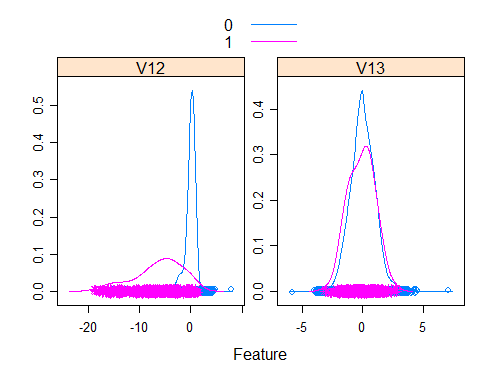
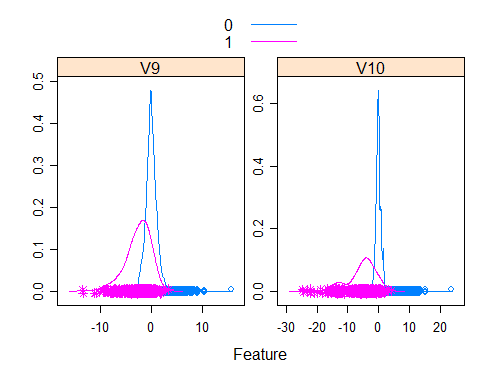
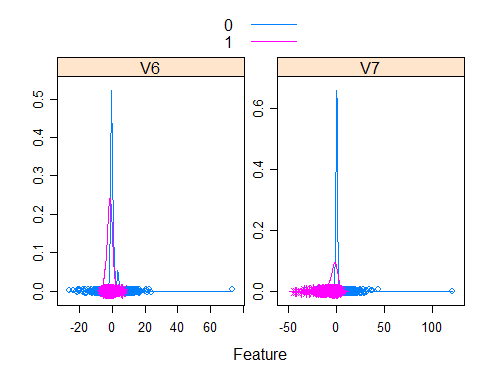
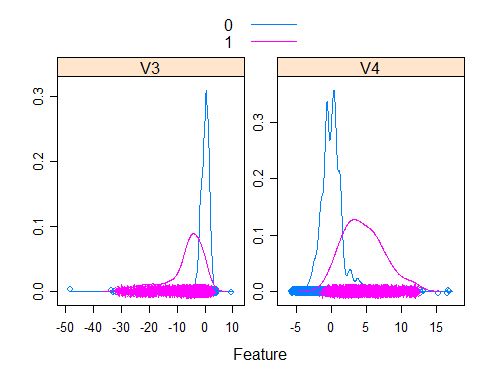
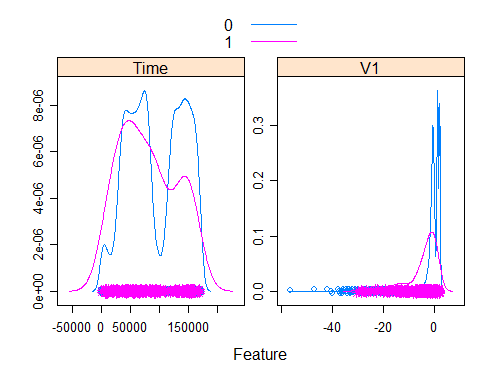
#Looking at the graph for the relationship between #Amount by Class,there is clearly a lot more #variability in the transaction values for non-fraudulent #transactions.

fig(14, 8)  
ggplot(creditcard, aes(x = factor(Class), y = Amount)) + geom\_boxplot() +   
 labs(x = 'Class', y = 'Amount') +  
 ggtitle("Distribution of transaction amount by class") + common\_theme



#Lets us look at more of the variable relations between the class

for(i in seq(from=1, to=28, by = 3))  
{  
 show(  
 featurePlot(  
 X <- creditcard[,c(i, i+1)],  
 Y <- as.factor(creditcard$Class),  
 plot = "density",  
 scales = list(x = list(relation="free"), y = list(relation="free")),   
 adjust = 1.5, # Adjusts curve smoothness  
 pch = c(1, 8, 15), # Points charted at the bottom to depict density  
 layout = c(2,1 ),  
 auto.key=TRUE  
 ) # end of feature plot  
 ) # end of show  
} # end of for loop

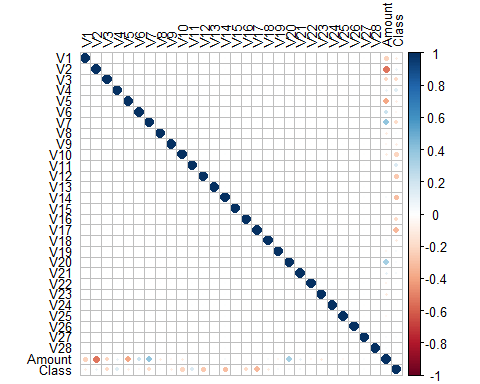


#DATA CORRELATION #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ #Lets see how the data are correlated #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#We observe that most of the data features are not correlated. #This is because before publishing, most of the features were #presented to a Principal Component Analysis (PCA) algorithm. #The features V1 to V28 are most probably the Principal #Components resulted after propagating the real features #through PCA. We do not know if the numbering of the #features reflects the importance of the Principal #Components.

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

fig(14, 8)  
correlations <- cor(creditcard[,-1],method="pearson")  
corrplot(correlations, number.cex = .9, method = "circle", type = "full", tl.cex=0.8,tl.col = "black")

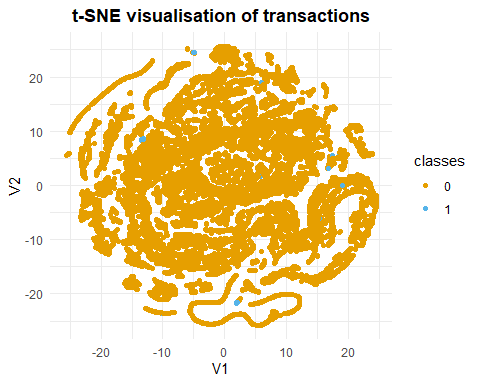


#Let’s visualize the dataset using the #T-Distributed Stochastic #we will try visualizing the data using #t-Distributed Stochastic Neighbour Embedding, #a technique to reduce dimensionality. #To train the model, perplexity was set to 20. #The visualisation should give us a hint as to #whether there exist any “discoverable” patterns #in the data which the model could learn. If there #is no obvious structure in the data, #it is more likely that the model will perform poorly.

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#Let’s just try to see how we can balance the dataset #we will be using the different techniques #before we do that, let’s reorganized the data preparation #Looking at the ‘Time’ feature, #it does not indicate the actual time #of the transaction and is more of listing the #data in chronological order. #Based on the data visualization above, #we assume that ‘Time’ feature has little #or no significance in correctly classifying #a fraud transaction and hence eliminate this #column from further analysis.

fig(16, 10)  
# Use 10% of data to compute t-SNE  
tsne\_subset <- 1:as.integer(0.1\*nrow(creditcard))  
tsne <- Rtsne(creditcard[tsne\_subset,-c(1, 31)], perplexity = 20, theta = 0.5, pca = F, verbose = F, max\_iter = 500, check\_duplicates = F)  
  
classes <- as.factor(creditcard$Class[tsne\_subset])  
tsne\_mat <- as.data.frame(tsne$Y)  
ggplot(tsne\_mat, aes(x = V1, y = V2)) + geom\_point(aes(color = classes)) + theme\_minimal() + common\_theme + ggtitle("t-SNE visualisation of transactions") + scale\_color\_manual(values = c("#E69F00", "#56B4E9"))



#PREPARING THE DATASET TO BE BALANCE FOR BETTER PREDICTION #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#Remove ‘Time’ variable

creditcard <- creditcard[,-1]  
#Change 'Class' variable to factor  
creditcard$Class <- as.factor(creditcard$Class)  
levels(creditcard$Class) <- c("Not\_Fraud", "Fraud")

#Scale numeric variables

creditcard[,-30] <- scale(creditcard[,-30])  
  
head(creditcard)

## V1 V2 V3 V4 V5 V6  
## 1 -0.6942411 -0.04407485 1.6727706 0.9733638 -0.245116153 0.34706734  
## 2 0.6084953 0.16117564 0.1097969 0.3165224 0.043483276 -0.06181986  
## 3 -0.6934992 -0.81157640 1.1694664 0.2682308 -0.364571146 1.35145121  
## 4 -0.4933240 -0.11216923 1.1825144 -0.6097256 -0.007468867 0.93614819  
## 5 -0.5913287 0.53154012 1.0214099 0.2846549 -0.295014918 0.07199846  
## 6 -0.2174742 0.58167387 0.7525841 -0.1188331 0.305008424 -0.02231344  
## V7 V8 V9 V10 V11 V12  
## 1 0.1936786 0.08263713 0.3311272 0.08338540 -0.5404061 -0.6182946  
## 2 -0.0637001 0.07125336 -0.2324938 -0.15334936 1.5800001 1.0660867  
## 3 0.6397745 0.20737237 -1.3786729 0.19069928 0.6118286 0.0661365  
## 4 0.1920703 0.31601704 -1.2625010 -0.05046786 -0.2218912 0.1783707  
## 5 0.4793014 -0.22650983 0.7443250 0.69162382 -0.8061452 0.5386257  
## 6 0.3849353 0.21795429 -0.5176177 -0.34110050 1.3140441 0.3601815  
## V13 V14 V15 V16 V17 V18  
## 1 -0.9960972 -0.3246096 1.6040110 -0.5368319 0.24486302 0.03076988  
## 2 0.4914173 -0.1499822 0.6943592 0.5294328 -0.13516973 -0.21876220  
## 3 0.7206986 -0.1731136 2.5629017 -3.2982296 1.30686559 -0.14478974  
## 4 0.5101678 -0.3003600 -0.6898362 -1.2092939 -0.80544323 2.34530040  
## 5 1.3522420 -1.1680315 0.1913231 -0.5152042 -0.27908030 -0.04556892  
## 6 -0.3597909 -0.1430569 0.5655061 0.4584589 -0.06844494 0.08190778  
## V19 V20 V21 V22 V23 V24  
## 1 0.49628116 0.32611744 -0.02492332 0.382853766 -0.17691102 0.1105067  
## 2 -0.17908573 -0.08961071 -0.30737626 -0.880075209 0.16220090 -0.5611296  
## 3 -2.77855597 0.68097378 0.33763110 1.063356404 1.45631719 -1.1380901  
## 4 -1.51420227 -0.26985475 -0.14744304 0.007266895 -0.30477601 -1.9410237  
## 5 0.98703556 0.52993786 -0.01283920 1.100009340 -0.22012301 0.2332497  
## 6 -0.04077658 0.11021522 -0.28352172 -0.771425648 -0.04227277 -0.6132723  
## V25 V26 V27 V28 Amount Class  
## 1 0.2465850 -0.3921697 0.33089104 -0.06378104 0.24496383 Not\_Fraud  
## 2 0.3206933 0.2610690 -0.02225564 0.04460744 -0.34247394 Not\_Fraud  
## 3 -0.6285356 -0.2884462 -0.13713661 -0.18102051 1.16068389 Not\_Fraud  
## 4 1.2419015 -0.4602165 0.15539593 0.18618826 0.14053401 Not\_Fraud  
## 5 -0.3952009 1.0416095 0.54361884 0.65181477 -0.07340321 Not\_Fraud  
## 6 -0.4465828 0.2196368 0.62889938 0.24563577 -0.33855582 Not\_Fraud

#We will split the data into training and testing data. #The Training test will have 70% of the data while the #test data will contain 30% of the data

set.seed(123)  
split <- sample.split(creditcard$Class, SplitRatio = 0.7)  
train <- subset(creditcard, split == TRUE)  
test <- subset(creditcard, split == FALSE)

#USING THE BALANCING TECHNIQUES #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ #From the training set, we have a total of 199020 non fraud #transactions and 344 fraud transactions

table(train$Class)

##   
## Not\_Fraud Fraud   
## 199020 344

#Applying the undersampling technique #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#Using the Under Sampling Technique In order to balance the data set, this approach lowers the amount of observations from the majority class. When the data set is large and the number of training samples is decreased, run time and storage issues are improved. There are actually two types of undersampling techniques. They are the Random undersampling technique and the Informative undersampling technique.

In the random undersampling method, observations from the majority class are selected at random and deleted until the data set is balanced. A predetermined selection criterion is used in conjunction with informative undersampling to eliminate data from the majority class.

This method may have a flaw in that by deleting observations, the training data may lose crucial information about the majority class.

set.seed(9560)  
down\_train <- downSample(x = train[, -ncol(train)],  
 y = train$Class)  
table(down\_train$Class)

##   
## Not\_Fraud Fraud   
## 344 344

#Using the Over Sampling Technique

Minority classes can use this strategy. To balance the data, it repeats the observations from the minority class. This method, like undersampling, can be broken down into two categories: random oversampling and informative oversampling.

By oversampling the minority class at random, random oversampling balances the data. Informative oversampling creates minority class observations artificially using a predetermined criterion.

Utilizing this strategy has the benefit of not causing information loss. The drawback of this approach is that because oversampling just duplicates the original data set’s observations, it adds many observations of various types, which results in overfitting.

Using the over sampling technique, we can see that we now have a balance dataset with a total of 199020 fraud transactions and 199020 fraud transactions

set.seed(9560)  
up\_train <- upSample(x = train[, -ncol(train)],  
 y = train$Class)  
table(up\_train$Class)

##   
## Not\_Fraud Fraud   
## 199020 199020

#Synthetic Data Generation (SMOTE and ROSE) #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ In other words, it corrects imbalances by producing fake data, as opposed to duplicating and adding observations from the minority class. It also falls under the category of oversampling.

Synthetic minority oversampling technique (SMOTE) is a potent and popular methodology for creating synthetic data. By selecting points that are on the line that connects the unusual observation to one of its closest neighbors in the feature space, the SMOTE algorithm creates artifactual samples. Smoothed bootstrapping is used by ROSE (random over-sampling examples) to create artificial samples from the neighborhood of the minority class in the feature space.

Using the Smote technique, we can see that we have a total of 1376 not fraud transactions and 1032 fraud transactions

set.seed(9568)  
smote\_train <- SMOTE(Class ~ ., data = train)  
  
table(smote\_train$Class)

##   
## Not\_Fraud Fraud   
## 1376 1032

#USING THE ROSE FUNCTION #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ using the ROSE function, we can see that we have a total of 99456 not fraud transaction and 99908 fraud transactions

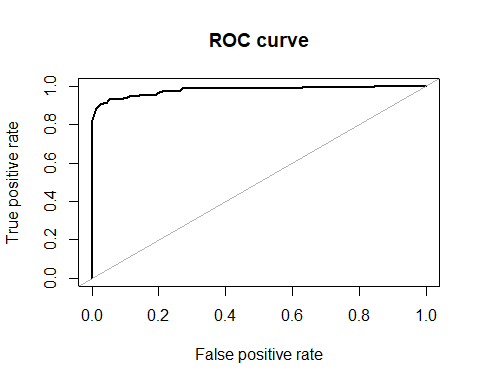
set.seed(9532)  
rose\_train <- ROSE(Class ~ ., data = train)$data   
  
table(rose\_train$Class)

##   
## Not\_Fraud Fraud   
## 99456 99908

#Random Forest Performance on the Under sample data

using random forest for the under sampling sample and then make prediction

x = down\_train[, -30]  
y = down\_train[,30]  
  
rf\_fit <- Rborist(x, y, ntree = 1000, minNode = 20, maxLeaf = 13)  
  
  
rf\_pred <- predict(rf\_fit, test[,-30], ctgCensus = "prob")  
prob <- rf\_pred$prob  
  
roc.curve(test$Class, prob[,2], plotit = TRUE)

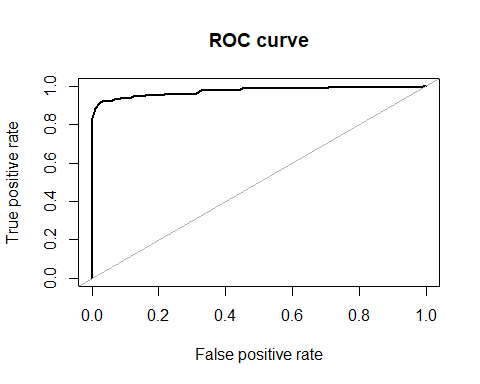


## Area under the curve (AUC): 0.978

#Random Forest Performance on the over sampling data

using random forest for the over sampling sample and then make prediction

x = up\_train[, -30]  
y = up\_train[,30]  
  
rf\_fit <- Rborist(x, y, ntree = 1000, minNode = 20, maxLeaf = 13)  
  
  
rf\_pred <- predict(rf\_fit, test[,-30], ctgCensus = "prob")  
prob <- rf\_pred$prob  
  
roc.curve(test$Class, prob[,2], plotit = TRUE)

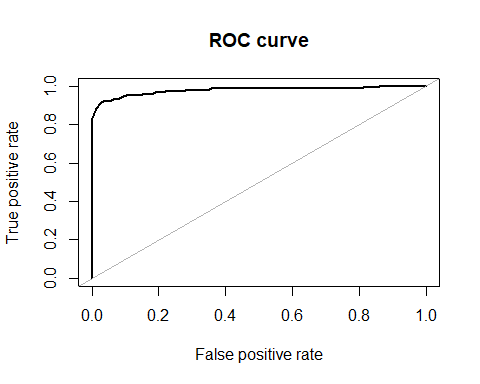


## Area under the curve (AUC): 0.973

#Random Forest Performance on the SMOTE sample data

using random forest for the SMOTE sampling sample and then make prediction

x = smote\_train[, -30]  
y = smote\_train[,30]  
  
rf\_fit <- Rborist(x, y, ntree = 1000, minNode = 20, maxLeaf = 13)  
  
  
rf\_pred <- predict(rf\_fit, test[,-30], ctgCensus = "prob")  
prob <- rf\_pred$prob  
  
roc.curve(test$Class, prob[,2], plotit = TRUE)

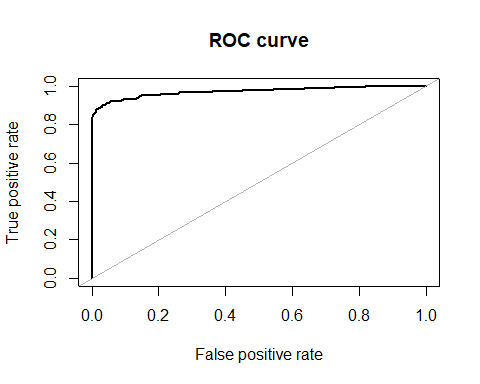


## Area under the curve (AUC): 0.977

#Random Forest Performance on the ROSE sample data

using random forest for the ROSE sampling sample and then make prediction

x = rose\_train[, -30]  
y = rose\_train[,30]  
  
rf\_fit <- Rborist(x, y, ntree = 1000, minNode = 20, maxLeaf = 13)  
  
  
rf\_pred <- predict(rf\_fit, test[,-30], ctgCensus = "prob")  
prob <- rf\_pred$prob  
  
roc.curve(test$Class, prob[,2], plotit = TRUE)



## Area under the curve (AUC): 0.973

#SUMMARY OF THE WORK DONE #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ #In this project, the main aim is to try to make our #imbalance dataset to be balance. #dealing with imbalanced datasets like the fraud credit #cases is few compared to the instances of normal transactions. #We have argued why accuracy is not a appropriate measure of #model performance here and used the metric AREA UNDER ROC #CURVE to evaluate how #undersampling, the over sampling , the SMOTE and the ROSE response variable can lead to better model #training.we applied different balancing techniques in this project. Using the Random forest to test each of the techniques, We concluded that the smote technique works with an accuracy of 0.976. #well on the dataset and achieved significant improvement in #model performance over the imabalanced data.Random forest #model performed well with SMOTE balancing techniques.