USE CASE STUDY REPORT FOR CREDIT CARD FRAUD DETECTION USING R

Group No.: 12

Team members:

1. Nikheel Navanale
2. Praudeep Namakkal Balasubramani

Executive Summary:

With the rapid growth of E-Commerce, the use of credit cards for online purchases has dramatically increased as the most popular mode of payment and it has caused an explosion in credit card fraud. Despite fraud costs increasing and cardholder confidence decreasing, financial institutions need to take action to ensure that their company and cardholders are safeguarded. These costs incorporate the hard losses that hurt the company's bottom line such as the cost incurred to replace cards, as well as case investigations, customer phone support, and damage to the company's reputation. In the event of a data breach, the customers would avoid further transactions/business with the company. This project intends to illustrate a possible solution to avoid the fraudulent transactions using Machine learning techniques and R programming language.

Background and Introduction

Credit card fraud is the main concern in the financial industry nowadays. It is determined that £20M a day were lost due to fraudulent transactions in 2016 alone, totalling approximately £770M annually. The manual analysis of fraudulent transactions is impracticable due to huge amounts of data and its difficulty. However, given adequate features, one could expect it is possible to do using Machine Learning.

Challenges involved in credit card fraud detection are:

1. Large amount of data is processed each day and the model developed must be fast enough to respond to the scam in time.
2. Imbalanced Data i.e. most of the transactions (99.8%) are not fraudulent which makes it arduous for detecting the fraudulent ones
3. Data availability, as the data is often private.
4. Misclassified data can be another influential issue, as not all fraudulent transaction is detected and reported.
5. Adaptive methods used against the model by the scammers.

How do we tackle these challenges?

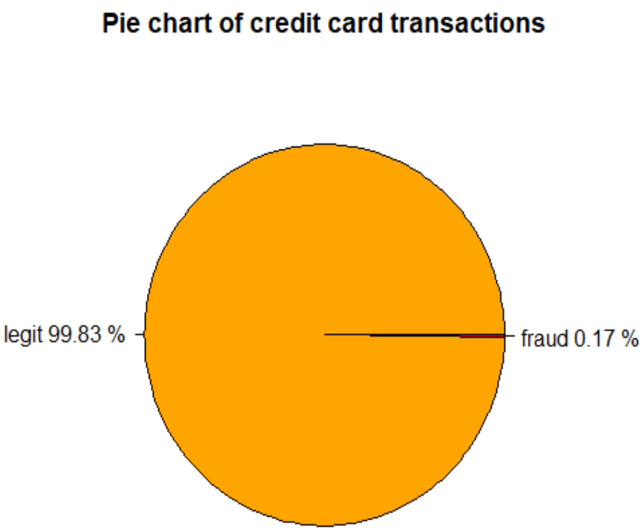
1. The model used must be a simple and more agile method to detect the exception and classify it as a fraudulent transaction instantly
2. Asymmetry can be dealt with using pre-processing methods which will be addressed in the forthcoming section
3. For protecting the privacy of the user, the dimensionality of the data can be reduced

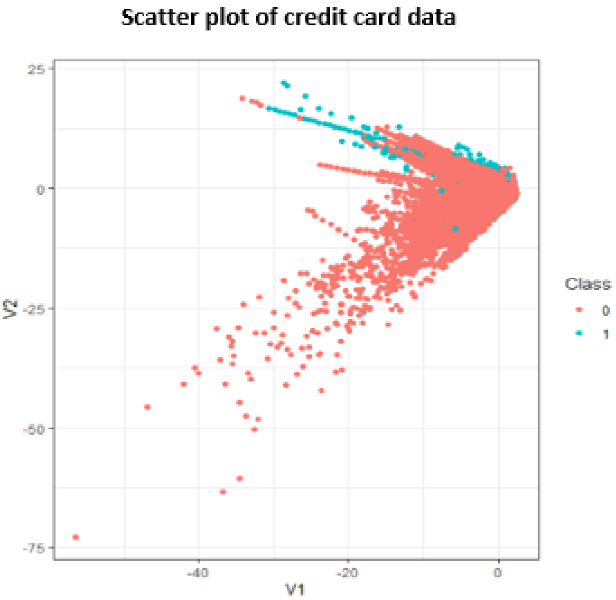
4. A more reliable source must be taken which double-checks the data, at least for training the model

These hypotheses will be explored in the project.

Data Exploration and Visualization

Data visualization and exploration is perhaps the fastest and most useful way to summarize and learn more about our data. The dataset that we use here is obtained from European cardholders 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. Inside this dataset, there are 31 columns out of which 28 are named as v1-v28 to protect sensitive data. The other columns represent Time, Amount and Class. Time shows the time gap between the first transaction and the following one. The amount is the amount of money transacted. Class 0 represents a valid transaction and 1 represents a fraudulent one.





```
> summary(credit_card)
      Time              V1                  V2                  V3                 V4          
 Min.   :     0   Min.   :-56.40751   Min.   :-72.71573   Min.   :-48.3256   Min.   :-5.68317  
 1st Qu.: 54202   1st Qu.: -0.92037   1st Qu.: -0.59855   1st Qu.: -0.8904   1st Qu.:-0.84864  
 Median : 84692   Median :  0.01811   Median :  0.06549   Median :  0.1799   Median :-0.01985  
 Mean   : 94814   Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.0000   Mean   : 0.00000  
 3rd Qu.:139321   3rd Qu.:  1.31564   3rd Qu.:  0.80372   3rd Qu.:  1.0272   3rd Qu.: 0.74334  
 Max.   :172792   Max.   :  2.45493   Max.   : 22.05773   Max.   :  9.3826   Max.   :16.87534  
       V5                   V6                  V7                  V8                  V9          
 Min.   :-113.74331   Min.   :-26.1605   Min.   :-43.5572   Min.   :-73.21672   Min.   :-13.43407  
 1st Qu.:  -0.69160   1st Qu.: -0.7683   1st Qu.: -0.5541   1st Qu.: -0.20863   1st Qu.: -0.64310  
 Median :  -0.05434   Median : -0.2742   Median :  0.0401   Median :  0.02236   Median : -0.05143  
 Mean   :   0.00000   Mean   :  0.0000   Mean   :  0.0000   Mean   :  0.00000   Mean   :  0.00000  
 3rd Qu.:   0.61193   3rd Qu.:  0.3986   3rd Qu.:  0.5704   3rd Qu.:  0.32735   3rd Qu.:  0.59714  
 Max.   :  34.80167   Max.   : 73.3016   Max.   :120.5895   Max.   : 20.00721   Max.   : 15.59500  
      V10                 V11                V12                V13                V14          
 Min.   :-24.58826   Min.   :-4.79747   Min.   :-18.6837   Min.   :-5.79188   Min.   :-19.2143  
 1st Qu.: -0.53543   1st Qu.:-0.76249   1st Qu.: -0.4056   1st Qu.:-0.64854   1st Qu.: -0.4256  
 Median : -0.09292   Median :-0.03276   Median :  0.1400   Median :-0.01357   Median :  0.0506  
 Mean   :  0.00000   Mean   : 0.00000   Mean   :  0.0000   Mean   : 0.00000   Mean   :  0.0000  
 3rd Qu.:  0.45392   3rd Qu.: 0.73959   3rd Qu.:  0.6182   3rd Qu.: 0.66251   3rd Qu.:  0.4931  
 Max.   : 23.74514   Max.   :12.01891   Max.   :  7.8484   Max.   : 7.12688   Max.   : 10.5268  
      V15                V16                 V17                 V18                 V19           
 Min.   :-4.49894   Min.   :-14.12985   Min.   :-25.16280   Min.   :-9.498746   Min.   :-7.213527  
 1st Qu.:-0.58288   1st Qu.: -0.46804   1st Qu.: -0.48375   1st Qu.:-0.498850   1st Qu.:-0.456299  
 Median : 0.04807   Median :  0.06641   Median : -0.06568   Median :-0.003636   Median : 0.003735  
 Mean   : 0.00000   Mean   :  0.00000   Mean   :  0.00000   Mean   : 0.000000   Mean   : 0.000000  
 3rd Qu.: 0.64882   3rd Qu.:  0.52330   3rd Qu.:  0.39968   3rd Qu.: 0.500807   3rd Qu.: 0.458949  
 Max.   : 8.87774   Max.   : 17.31511   Max.   :  9.25353   Max.   : 5.041069   Max.   : 5.591971  
      V20                 V21                 V22                  V23                 V24          
 Min.   :-54.49772   Min.   :-34.83038   Min.   :-10.933144   Min.   :-44.80774   Min.   :-2.83663  
 1st Qu.: -0.21172   1st Qu.: -0.22839   1st Qu.: -0.542350   1st Qu.: -0.16185   1st Qu.:-0.35459  
 Median : -0.06248   Median : -0.02945   Median :  0.006782   Median : -0.01119   Median : 0.04098  
 Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.000000   Mean   :  0.00000   Mean   : 0.00000  
 3rd Qu.:  0.13304   3rd Qu.:  0.18638   3rd Qu.:  0.528554   3rd Qu.:  0.14764   3rd Qu.: 0.43953  
 Max.   : 39.42090   Max.   : 27.20284   Max.   : 10.503090   Max.   : 22.52841   Max.   : 4.58455  
      V25                 V26                V27                  V28                 Amount        
 Min.   :-10.29540   Min.   :-2.60455   Min.   :-22.565679   Min.   :-15.43008   Min.   :    0.00  
 1st Qu.: -0.31715   1st Qu.:-0.32698   1st Qu.: -0.070840   1st Qu.: -0.05296   1st Qu.:    5.60  
 Median :  0.01659   Median :-0.05214   Median :  0.001342   Median :  0.01124   Median :   22.00  
 Mean   :  0.00000   Mean   : 0.00000   Mean   :  0.000000   Mean   :  0.00000   Mean   :   88.35  
 3rd Qu.:  0.35072   3rd Qu.: 0.24095   3rd Qu.:  0.091045   3rd Qu.:  0.07828   3rd Qu.:   77.17  
 Max.   :  7.51959   Max.   : 3.51735   Max.   : 31.612198   Max.   : 33.84781   Max.   :25691.16  
 Class    
 0:284315 
 1:   492 
```

Data Preparation and Pre-processing

As discussed above, due to confidentiality, the dataset contains only numerical input variables which are the result of a PCA transformation. 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.

With the help of scaling, the data is structured according to a specified range. Therefore, there are no extreme values in our dataset that might interfere with the functioning of our model. As the dataset is unbalanced, we use Random over-sampling (ROS), Random under-sampling (RUS), both ROS+RUS and SMOTE (Synthetic minority over sampling technique).

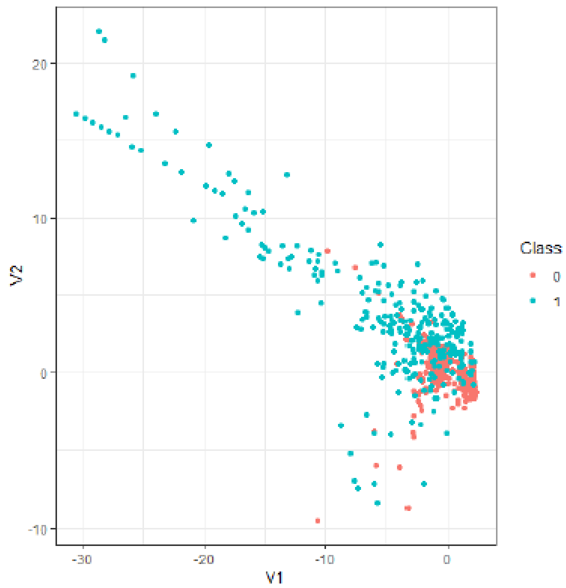
Random under-sampling:

This approach is better suited to use when there is a large data set and minimizing the number of training samples helps to reduce run time and storage problems. The random under-sampling method randomly selects observations from the class of majority which is discarded before the collection of data is balanced. Informative under-sampling mimics a pre-specified selection rule to remove the observations from the majority class.

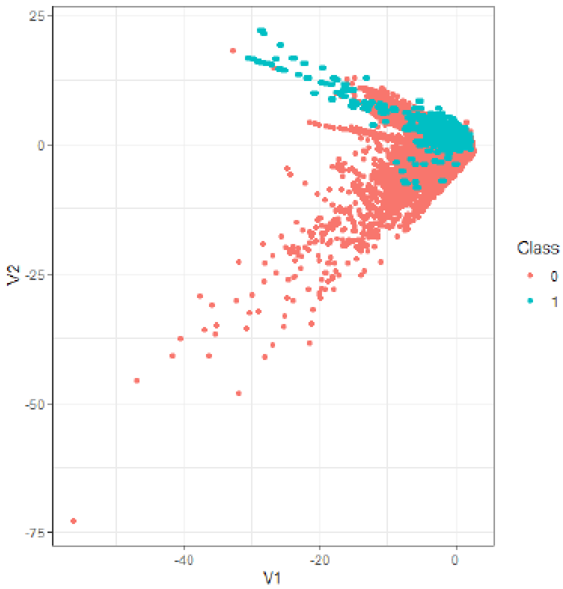
Random over-sampling:

Random oversampling balances the data by randomly oversampling the minority class. In other words, it uses a pre-specified criterion and synthetically generates minority class observations. An advantage of using this method is that there's no information loss. As oversampling merely adds repeated observations in the primary data collection, it ends up adding multiple observations of different types, leading to overfitting, which would be a downside. While these data set will have a high training accuracy, the accuracy of the unseen data will be more acute.

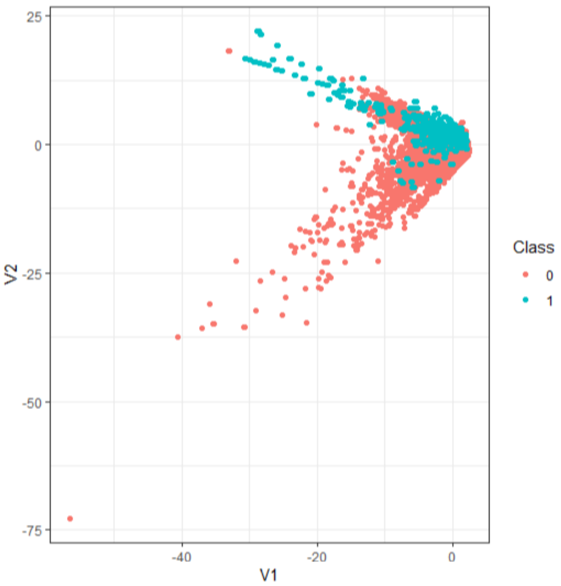
Under Sampling



Over Sampling



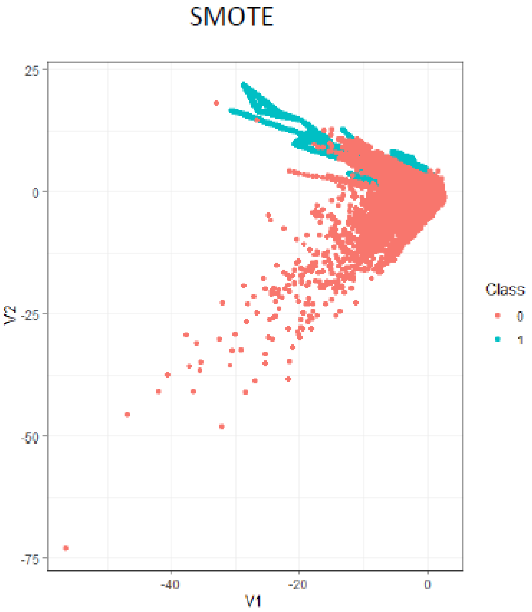
RUS and ROS



SMOTE (synthetic minority oversampling technique):

Instead of replicating and adding observations from the minority class, this method overcomes imbalances by generating artificial data. This is also a type of oversampling technique. In other words, it generates a random set of minority class observations to change the classifier learning bias towards minority class.

Below are the plots of different graphs to check for inconsistencies in the dataset and to visually comprehend it:



Confusion Matrix

A Confusion matrix is utterly a technique for summing the performance of a classification algorithm. If you have an unequal number of observations in each class or if you have more than two classes in your dataset, classification efficiency solely can be misleading. We can have a better idea of what our classification model is capturing right and what kinds of errors it is making simply by determining a confusion matrix.

Data Mining Techniques and Implementation

1. Classification and Regression Tree (CART)

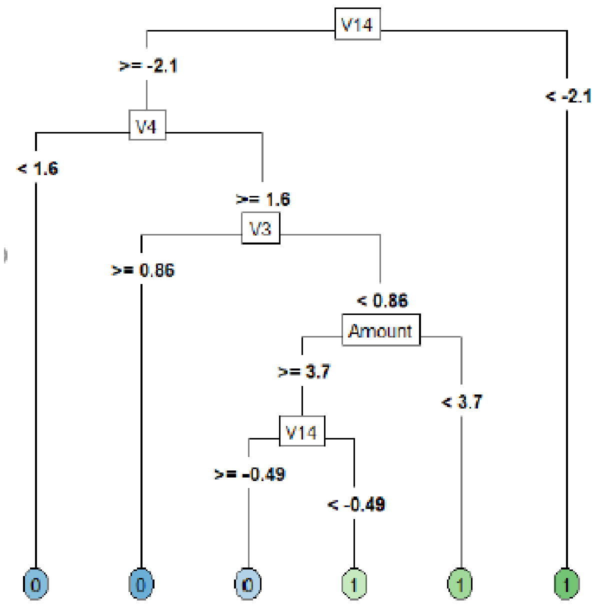
Prediction Trees are generally used to predict a response or class YY from input X1, X2, X3, …, Xn. A continuous response is called a regression tree and a categorical response is called a classification tree. The significant advantage of using decision trees is that they are intuitively very simple and straightforward to explain. They can be graphically represented, and they can manage qualitative predictors easily without the need to construct dummy variables.

Nevertheless, they aren't quite robust as decision trees usually do not have the same level of predictive accuracy as other approaches. A big change in the final estimated tree due to a slight shift in the data set. By aggregating multiple decision trees, using methods such as bagging, random forests, and boosting, the predictive performance of decision trees can be considerably improved.

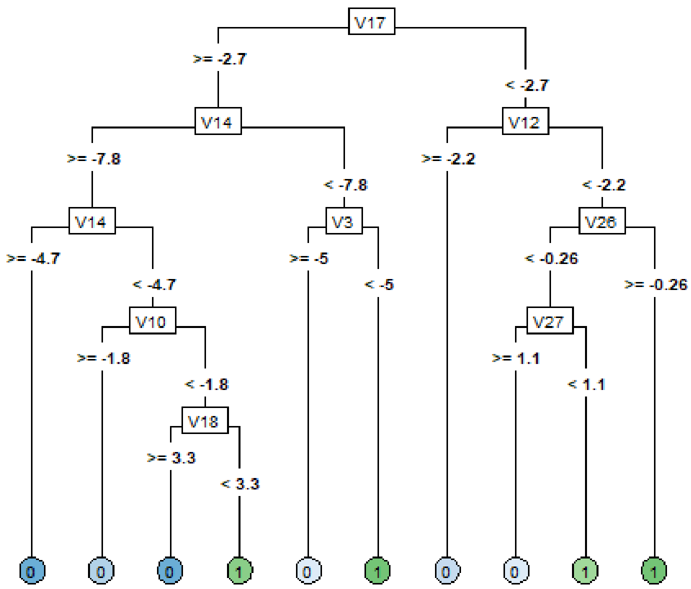
Below figure represents a decision tree and confusion matrix **using SMOTE technique**.

- 179 fraudulent transactions correctly predicted as fraudulent
- 18 fraudulent transactions were misclassified as legitimate
- 3578 were legit transactions but were classified as fraudulent
- 110148 were legitimate transactions and correctly predicted

```
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0 110148     18
         1   3578    179
```



Below figure represents a decision tree **without using SMOTE technique**.



This is the confusion matrix without SMOTE technique. From this matrix it’s clear that,

- 157 fraudulent transactions correctly predicted as fraudulent
- 40 fraudulent transactions were misclassified as legitimate
- 26 were legit transactions but were classified as fraudulent
- 113700 were legitimate transactions and correctly predicted

```
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0 113700     40
         1     26    157
```

2. Random forest

Random forest is a versatile machine learning method able to perform both regression and classification tasks. This technique provides an improvement over bagged trees through a slight tweak that decorrelates trees as it involves dimensional reduction methods, handles missing values, outer values, and other important data exploration measures, and performs a fairly good job

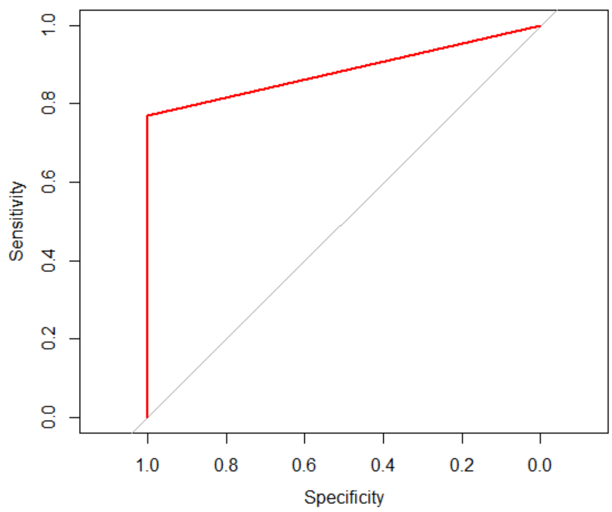
Random forest is extremely efficient at predicting missing data and preserving precision at periods where there is a large proportion of data missing. This method can also balance errors where classes are imbalanced in a dataset. Most importantly, it can handle massive datasets with the large dimensionality. However, one downside of using Random Forests, is that we can easily overfit noisy datasets (especially in the case of doing regression).

The confusion matrix for random forest model is as follows,

- 158 fraudulent transactions correctly predicted as fraudulent
- 47 fraudulent transactions were misclassified as legitimate
- 3 were legit transactions but were classified as fraudulent
- 113715 were legitimate transactions and correctly predicted

```
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0 113715     47
         1      3    158
```

ROC for Random Forest method:



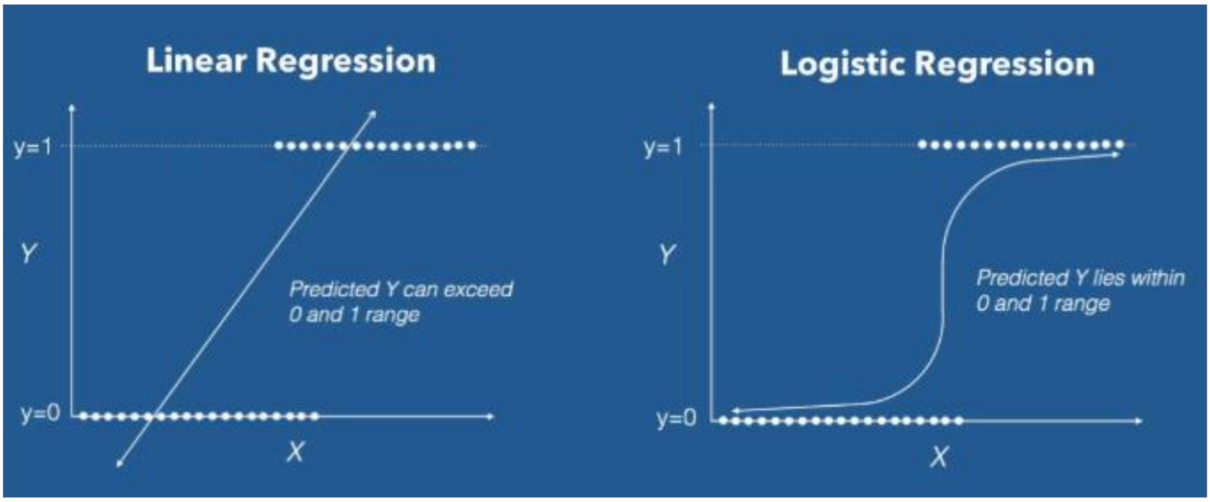
AUC = 88.54%

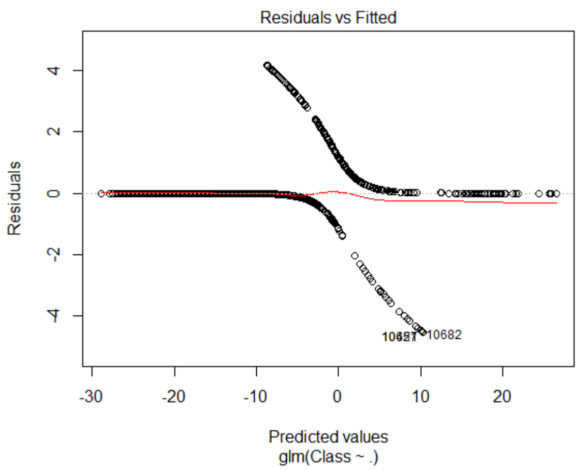
```
> print(cur)

Call:
roc.default(response = rf.testc$Class, predictor = rf.testc$Pred,     plot = TRUE, col = "red")

Data: rf.testc$Pred in 113718 controls (rf.testc$Class 0) < 205 cases (rf.testc$Class 1).
Area under the curve: 0.8854
```

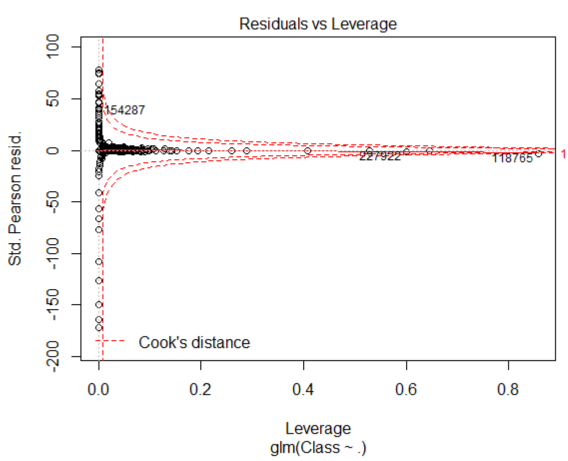
3. Logistic Regression

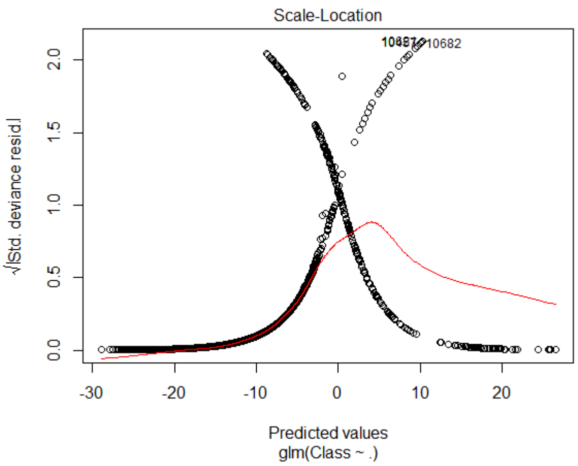
Linear regression is one of the most popularly known modelling techniques. This method allows us to use a linear relationship to predict the (average) numerical value of ‘YY’ for a given value of ‘XX’ with a straight line which is called the "regression line". Hence, the linear regression model is y=ax+by=ax+b. The model assumes that the response variable ‘YY is quantitative. Yet, in many situations, the response variable is qualitative or, in other words, categorical. For example, taking on values fraudulent or not fraudulent transactions is qualitative.

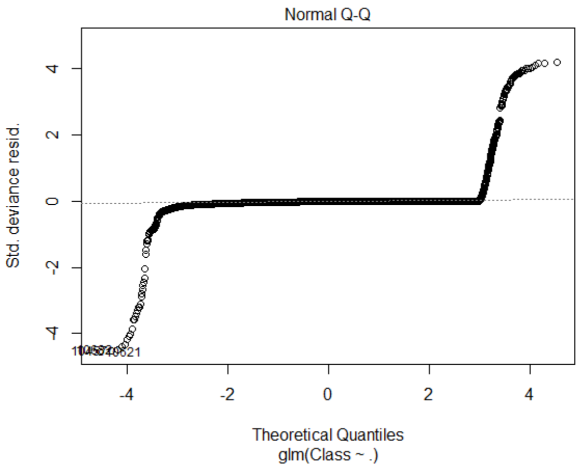
Unfortunately, Linear regression is not capable of predicting probability, but logistic regression can predict a probability score that reflects the probability of the occurrence at the event based on one or more predictor variables (*X*). This allows one to say that the existence of a predictor raises (or decreases), by a specific probability of a given outcome by a specific percentage.



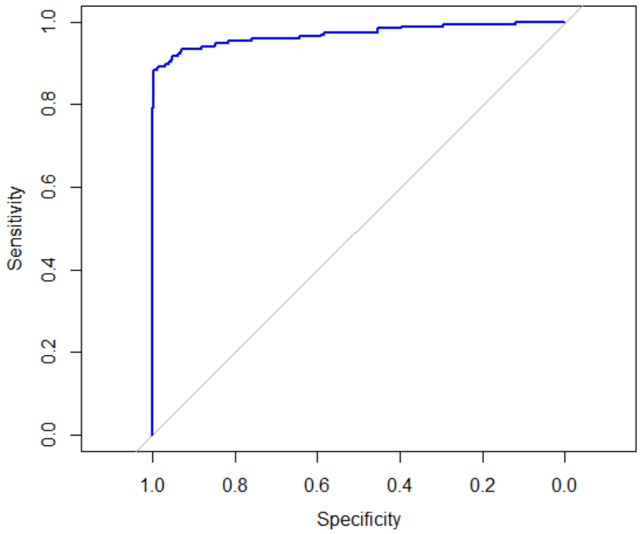








ROC for a Logistic regression model:



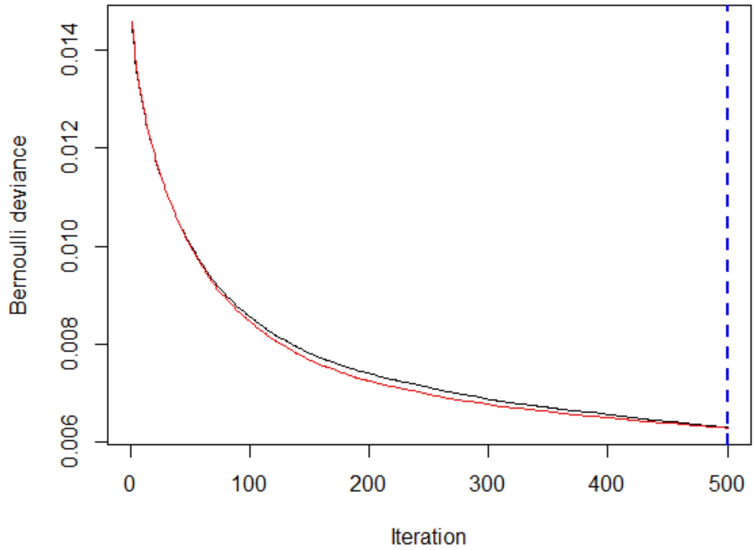
AUC = 97.07%

```
> print(auc)

Call:
roc.default(response = test_data$Class, predictor = lr.predict,     plot = TRUE, col = "blue")

Data: lr.predict in 113726 controls (test_data$Class 0) < 197 cases (test_data$Class 1).
Area under the curve: 0.9707
```

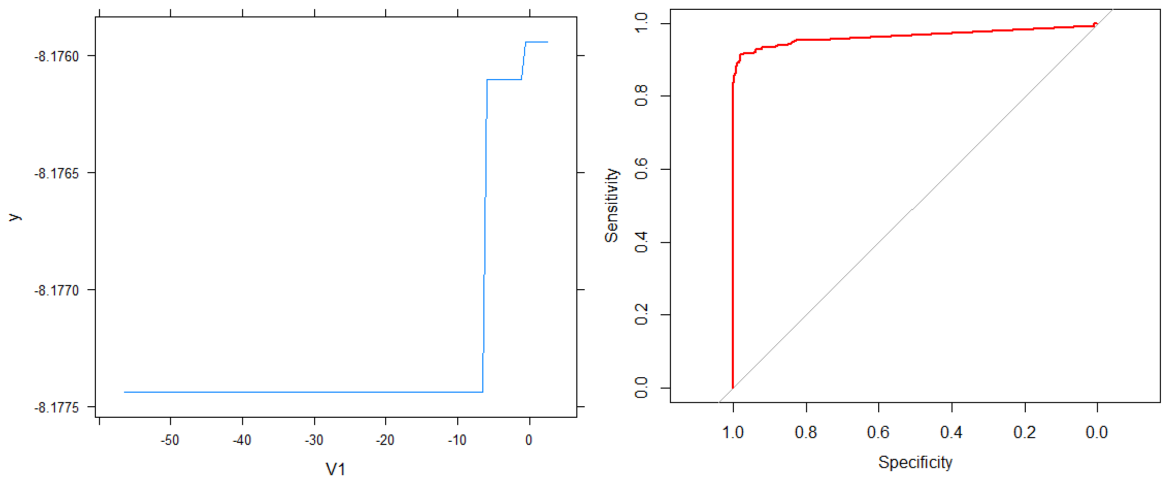
4. Gradient Boosting Method (GBM)

Gradient Boosting is a common algorithm for machine learning used to perform classification and regression tasks. This model consists of different underlying models such as weak decision trees. These decision trees combine to form a powerful gradient boosting model. In this method, each new tree is a fit on a modified version of the original data set.

Below graph determines best iteration based on test data



GBM model plot, ROC for GBM model:



AUC = 96.58%

```
> print(gbm_auc)

Call:
roc.default(response = test_data$Class, predictor = gbm_test,     plot = TRUE, col = "red")

Data: gbm_test in 113726 controls (test_data$Class 0) < 197 cases (test_data$Class 1).
Area under the curve: 0.9658
```

5. Artificial Neural Network (ANN)

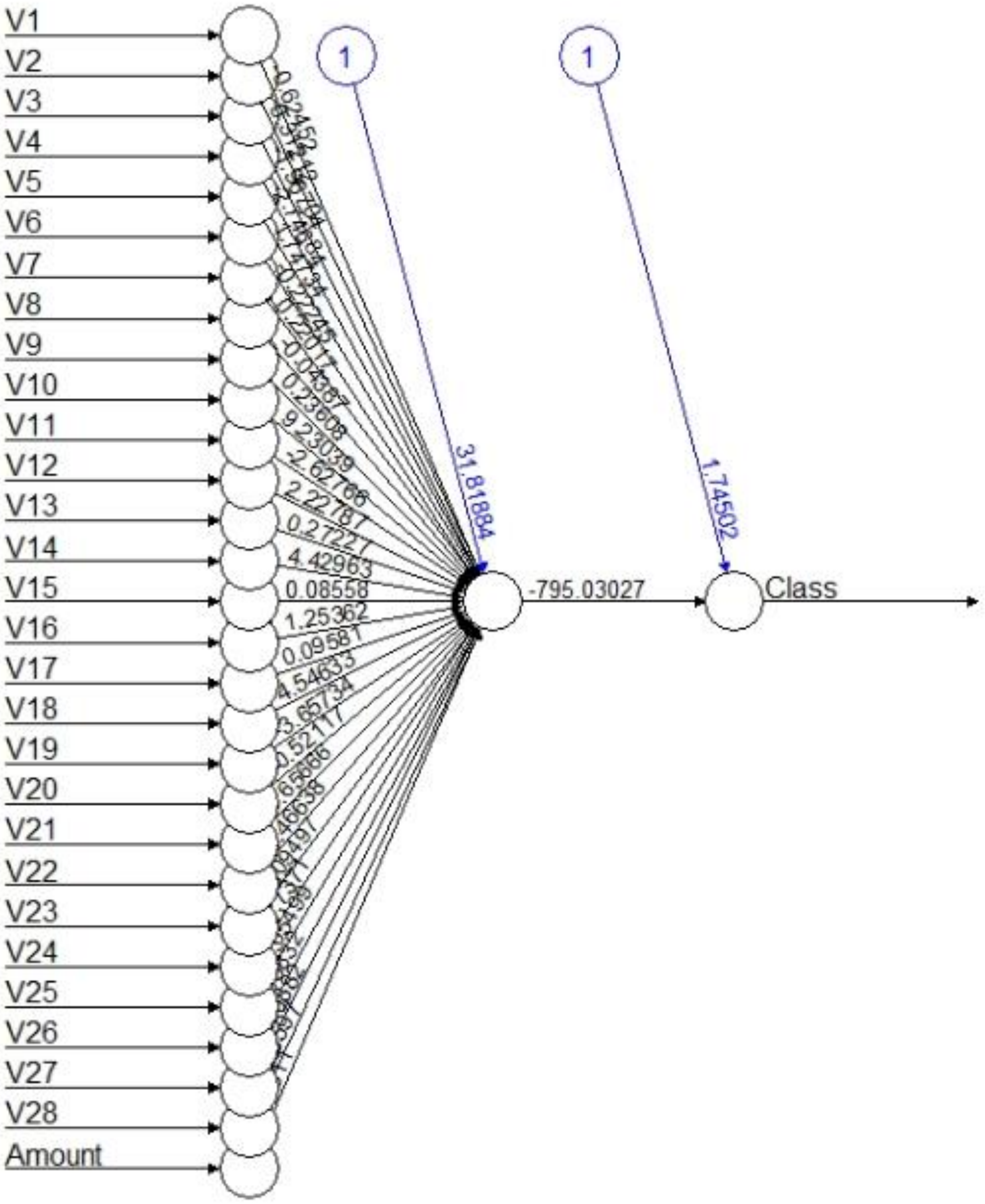
We are importing the neural net kit which helps us to implement our ANNs. Using historical data, the ANN models can learn the patterns, and can identify the input data. We then plotted it using the plot () function. In the case of ANN, we have a range of numbers that is between 1 and 0. We set a threshold as ‘X’ and values above ‘X’ will correspond to 1 and the rest will be 0.

There are two major types of neural artificial networks:

- Feedforward Artificial Neural Networks
- Feedback Artificial Neural Networks

Feedforward neural network is a non-recursive network. In Feedforward, signals travel towards the output layer in one direction only.

Feedback neural networks contain cycles. Signals travels in bi direction by introducing loops in the network. The feedback cycles can cause the network's behaviour change over time based on its input. Feedback neural network also known as recurrent neural networks



Performance Evaluation

With respect to CART method used above,

- CART method with SMOTE technique is predicting fraudulent transactions more accurately compared to CART method without SMOTE
- But the drawback of this method is it misclassifies a greater number of legit transactions as fraudulent

With respect to AUC,

- Logistic regression (97.07%) performs the best followed by Gradient boosting (96.58%) and Random forest (88.54%)

Advantage and disadvantage of ANN:

- Neural networks are more versatile and can be used for problems of regression and classification.
- Neural Networks requires more data than other Machine Learning algorithms. NNs can be used only with numerical inputs and non-missing value datasets.
- Neural networks with several inputs, such as images, are ideal for the nonlinear dataset. Neural networks can operate with inputs and layers of any number. Neural networks have the power of numbers that can conduct jobs in parallel.

Discussion and Recommendation

- Logistic regression has the best performance. Random forest is not preferred as it overfits the training data and GBM is preferred as it improves upon both models.
- ANN is not preferred as alternative algorithms such as Decision Tree, Regression which are available and are simple, fast, easy to train. They also provide better performance.

Summary

To implement this model a variety of ML algorithms and plotted the respective performance curves for the models. We learned how to analyse and visualize data, in order to distinguish fraudulent transactions from other data forms.

Future Enhancement

While we couldn’t reach our goal of 100% accuracy in fraud detection, we did end up building a model that can get very close to that goal with enough time and data. More room for improvement can be found in the project and dataset. As discussed earlier, the precision of the algorithms increases when the size of the dataset is increased. Hence, more data will certainly make the model more accurate in recognizing frauds and diminish the number of false positives. However, this requires official support from the banks themselves.

Citations:

Definitions for various methods used in this project are taken from official documents that are available on the web and textbooks.

https://www.financialfraudaction.org.uk/fraudfacts16/assets/fraud_the_facts
https://www.datacamp.com/community/tutorials/neural-network-models-r
https://data-flair.training/blogs/data-science-machine-learning-project-credit-card-fraud-detection/
https://rpubs.com/slazien/fraud_detection
http://www.di.fc.ul.pt/~jpn/r/tree/tree.html
https://www.datacamp.com/community/tutorials/decision-trees-R
https://www.statmethods.net/advstats/cart.html

Appendix – R Code

```
library(data.table)
library(ggplot2)
library(plyr)
library(dplyr)
library(corrplot)
library(pROC)
library(glmnet)
library(caret)
library(Rtsne)
library(xgboost)
library(caret)
library(e1071)
library(caTools)
library(ROSE)
library(smotefamily)
library(rpart)
library(rpart.plot)
library(caTools)
library(ranger)
library(caret)
library(data.table)
library(randomForest)
library(neuralnet)
library(gbm, quietly=TRUE)


#importing datatset
credit_card <- read.csv(file.choose())
str(credit_card)

q<-credit_card

#convert class to a factor variable
credit_card$Class <- factor(credit_card$Class, levels = c(0,1))

summary(credit_card)

#count the missing values
sum(is.na(credit_card))

#Distribution and % of fraud and legit transactions
table(credit_card$Class)
prop.table(table(credit_card$Class))

#Pie chart of credit card transactions
labels <- c("legit", "fraud")
labels <- paste(labels, round(100*prop.table(table(credit_card$Class)),2))
labels <- paste(labels, "%")
```

```
pie(table(credit_card$Class), labels, col = c("orange", "red"), main = "Pie chart of credit card transactions")

#no model prediction
predictions <- rep.int(0,nrow(credit_card))
predictions <- factor(predictions, levels = c(0,1))


confusionMatrix(data = predictions, reference = credit_card$Class)


ggplot(data = credit_card, aes(x = V1, y = V2, col = Class)) +
  geom_point() +
  theme_bw()


#Creating training and test sets
set.seed(123)
data_sample = sample.split(credit_card$Class, SplitRatio = 0.60)
train_data = subset(credit_card, data_sample==TRUE)
test_data = subset(credit_card, data_sample==FALSE)
dim(train_data)
dim(test_data)

#ROS
table(train_data$Class)
n_legit <- 170589
new_frac_legit <- 0.50
new_n_total <- n_legit/new_frac_legit


oversampling_result <- ovun.sample(Class ~ ., data = train_data,
                                   method = "over", N = new_n_total, seed = 2019)
oversampled_credit <- oversampling_result$data
table(oversampled_credit$Class)
ggplot(data = oversampled_credit, aes(x = V1, y = V2, col = Class)) +
  geom_point(position = position_jitter(width = 0.1)) +
  theme_bw()

#RUS
table(train_data$Class)
n_fraud <- 295
new_frac_fraud <- 0.50
new_n_total <- n_fraud/new_frac_fraud
undersampling_result <- ovun.sample(Class ~ ., data = train_data,
                                    method = "under",
                                    N = new_n_total,
                                    seed = 2019)
undersampling_credit <- undersampling_result$data
ggplot(data = undersampling_credit, aes(x = V1, y = V2, col = Class)) +
```

```
  geom_point() +
  theme_bw()

#ROS and RUS
n_new <- nrow(train_data)
fraction_fraud_new <- 0.50
sampling_result <- ovun.sample(Class ~ ., data = train_data,
                               method = "both",
                               N = n_new,
                               p = fraction_fraud_new,
                               seed = 2019)
sampled_credit <- sampling_result$data
table(sampled_credit$Class)
prop.table(table(sampled_credit$Class))
ggplot(data = sampled_credit, aes(x = V1, y = V2, col = Class)) +
  geom_point(position = position_jitter(width = 0.1)) +
  theme_bw()

#SMOTE to balance the data
table(train_data$Class)
n0 <- 170589
n1 <- 295
r0 <- 0.6

#calculate value for dup_size
ntimes <- ((1-r0)/r0)*(n0/n1)-1
smote_output <- SMOTE(X = train_data[,-c(1,31)],
                      target = train_data$Class,
                      K = 5,
                      dup_size = ntimes)
credit_smote <- smote_output$data
colnames(credit_smote)[30] <- "Class"

#Class distribution for oversampled dataset using SMOTE
prop.table(table(credit_smote$Class))
ggplot(data = credit_smote, aes(x = V1, y = V2, col = Class)) +
  geom_point() +
  theme_bw()


CART_model <- rpart(Class ~ ., credit_smote)
rpart.plot(CART_model, extra = 0, type = 5, tweak = 1.2)

#Predict fraud classes
predicted_val <- predict(CART_model, test_data, type = 'class')

#Build confusion matrix
confusionMatrix(predicted_val, test_data$Class)

#Decision tree without SMOTE
```

```
CART_model <- rpart(Class ~ ., train_data[,-1])
rpart.plot(CART_model, extra = 0, type = 5, tweak = 1.2)
predicted_val <- predict(CART_model, test_data[,-1], type = 'class')
confusionMatrix(predicted_val, test_data$Class)


#RANDOM FOREST
rf <- q
rf$Class <- as.factor(rf$Class)

rows <- nrow(rf)
cols <- ncol(rf)

set.seed(40)
rf <- rf[sample(rows),]
ntr <- as.integer(round(0.6*rows))

rf.train <- rf[1:ntr, 1:cols]
rf.test <- rf[(ntr+1):rows, -cols]
rf.testc <- rf[(ntr+1):rows, cols]

rf.testc <- as.data.frame(rf.testc)
colnames(rf.testc)[1] <- c("Class")

samp <- as.integer(0.5 * ntr)

model <- randomForest(Class~.,data = rf.train, importance = TRUE, ntree = 35, samplesize = samp,
                maxnodes = 45)

rf.pred <- predict(model, rf.test)
rf.testc$Pred <- rf.pred

confusionMatrix(rf.testc$Pred, rf.testc$Class)

rf.testc$Class <- ordered(rf.testc$Class, levels = c("0", "1"))
rf.testc$Pred <- ordered(rf.testc$Pred, levels = c("0", "1"))
auc(rf.testc$Class, rf.testc$Pred)

cur = roc(rf.testc$Class, rf.testc$Pred, plot = TRUE, col = "red")
print(cur)


#LOGISTIC REGRESSION
cc <- q

cc$Amount=scale(cc$Amount)
ND=cc[,-c(1)]

set.seed(100)
data_sample = sample.split(ND$Class,SplitRatio=0.60)
```

```
tr_data = subset(ND,data_sample==TRUE)
te_data = subset(ND,data_sample==FALSE)

Log_Mod=glm(Class~.,tr_data,family=binomial())
summary(Log_Mod)
plot(Log_Mod)

lr.p <- predict(Log_Mod,te_data, probability = TRUE)
auc = roc(te_data$Class, lr.p, plot = TRUE, col = "blue")
print(auc)


#NEURAL NET
ANN_mod = neuralnet (Class~.,tr_data,linear.output=FALSE)
plot(ANN_mod)
p.ANN=compute(ANN_mod,te_data)
res.ANN=p.ANN$net.result
res.ANN=ifelse(res.ANN>0.5,1,0)


#GRADIENT BOOSTING
mod_gbm <- gbm(Class ~ .
               , distribution = "bernoulli"
               , data = rbind(tr_data, te_data)
               , n.trees = 500
               , interaction.depth = 3
               , n.minobsinnode = 100
               , shrinkage = 0.01
               , bag.fraction = 0.5
               , train.fraction = nrow(tr_data) / (nrow(tr_data) + nrow(te_data))
  )

i = gbm.perf(mod_gbm, method = "test")

mod.inf = relative.influence(mod_gbm, n.trees = i, sort. = TRUE)

plot(mod_gbm)

gbm.test = predict(mod_gbm, newdata = te_data, n.trees = i)
gbm.auc = roc(te_data$Class, gbm.test, plot = TRUE, col = "red")
print(gbm.auc)
```

--END OF REPORT--