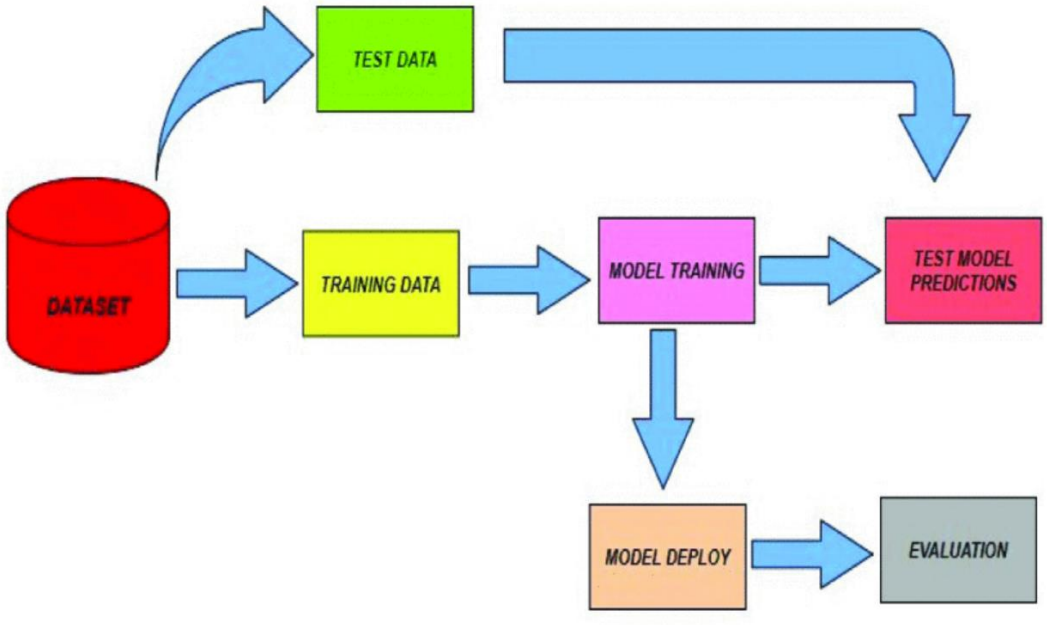
I. ABSTRACT:

Credit card fraud detection is presently the most frequently occurring problem in the present world. This is due to the rise in both online transactions and e-commerce platforms. Credit card fraud generally happens when the card was stolen for any of the unauthorized purposes or even when the fraudster uses the credit card information for his use. In the present world, we are facing a lot of credit card problems. To detect the fraudulent activities the credit card fraud detection system was introduced. This project aims to focus mainly on machine learning algorithms. The algorithms used are decision tree, random forest algorithm, artificial neural networks, Support Vector Machines(SVM) and the XGBoost algorithm. The results of these algorithms are based on Area under ROC curve. The ROC curve is plotted based on the confusion matrix. All the algorithms are compared and the algorithm that has the greatest AUC score is considered as the best algorithm that is used to detect the fraud.

II. PROPOSED WORK:

The main aim of this paper is to classify the transactions that have both the fraud and non-fraud transactions in the dataset using algorithms like that Logistic Regression, Random Forest, SVM and the XGBoost algorithms. Then these modelling techniques are compared to choose the algorithm that best detects the credit card fraud transactions. The process flow for the credit fraud detection problem includes the splitting of the data, model training, model deployment, and the evaluation criteria.



The project includes many steps from gathering dataset to deploying model and performing analysis based on results. In this model, we take the Kaggle credit card fraud dataset which contains only numerical input variables which are the result of a PCA transformation. Due to confidentiality issues, the dataset does not original features and more background information about the data. We have used transformation techniques like up-sampling, down-sampling and Random Over Sampling (ROSE) to balance the dataset. Since accuracy is not an appropriate measure of model performance for imbalanced datasets, we have used the metric AREA UNDER ROC CURVE to evaluate how different methods of over-sampling or under-sampling the response variable can lead to better model training. To prepare the model, we have to split the data into the training data and the testing data. We use the training data to prepare Logistic Regression, Random Forest, SVM and the XGBoost models. Finally, the comparison AUC for different models gives us reasonable performance of the credit card fraud transactions more accurately.

Sampling Techniques used for unbalanced dataset:

A. **Upsampling:** This method increases the size of the minority class by sampling with replacement so that the classes will have the same size.
B. **Downsampling:** In contrast to the above method, this one decreases the size of the majority class to be the same or closer to the minority class size by just taking out a random sample.
C. **ROSE**: This technique provides functions to deal with binary classification problems in the presence of imbalanced classes. Artificial balanced samples are generated according to a smoothed bootstrap approach and allow for aiding both the phases of estimation and accuracy evaluation of a binary classifier in the presence of a rare class.

Data Modelling Algorithms:

A. **Decision Tree:** Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. The decision of making strategic splits heavily affects a tree's accuracy. The decision criteria are different for classification and regression trees. The objective of applying SVMs is to find the best line in two dimensions or the best hyperplane in more than two dimensions in order to help us separate our space into classes. In Decision Trees, for predicting a class label for a record we start from the root of the tree.

   Algorithm:

   1. *Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes.*
   2. *The creation of sub-nodes increases the homogeneity of resultant sub-nodes.*
   3. *In other words, we can say that the purity of the node increases with respect to the target variable.*
   4. *The decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.*
   5. *The algorithm selection is also based on the type of target variables.*

6. *On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.*

B. Logistic regression: Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of certain classes based on some dependent variables. In short, the logistic regression model computes a sum of the input features (in most cases, there is a bias term), and calculates the logistic of the result. The output of logistic regression is always between (0, and 1), which is suitable for a binary classification task. The higher the value, the higher the probability that the current sample is classified as class=1, and vice versa.

Algorithm:

1. *The logistic regression algorithm analyzes relationships between variables.*
2. *It assigns probabilities to discrete outcomes using the Sigmoid function, which converts numerical results into an expression of probability between 0 and 1.0. Probability is either 0 or 1, depending on whether the event happens or not.*
3. *For binary predictions, you can divide the population into two groups with a cut-off of 0.5.*
4. *Everything above 0.5 is considered to belong to group A, and everything below is considered to belong to group B.*
5. *A hyperplane is used as a decision line to separate two categories (as far as possible) after data points have been assigned to a class using the Sigmoid function. The class of future data points can then be predicted using the decision boundary.*

C. Random Forest: The Random Forest algorithm is one of the widely used supervised learning algorithms. This can be used for both regression and classification purposes. But this algorithm is mainly used for classification problems. Generally, a forest is made up of trees and similarly, the Random Forest algorithm creates the decision trees on the sample data and gets the prediction from each of the sample data. Then Random Forest algorithm is an ensemble method. This algorithm is better than the single decision trees because it reduces the over-fitting by averaging the result.

Algorithm:

1. *Every decision tree has high variance, but when we combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data and hence the output doesn't depend on one decision tree but multiple decision trees.*
2. *In the case of a classification problem, the final output is taken by using the majority voting classifier.*
3. *In the case of a regression problem, the final output is the mean of all the outputs. This part is Aggregation.*
4. *The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.*
5. *Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.*

D. **SVM:** In machine learning, Support vector machines (SVM) are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis. It is mostly used in classification problems. In this algorithm, each data item is plotted as a point in n-dimensional space (where n is a number of features), with the value of each feature being the value of a particular coordinate. Then, classification is performed by finding the hyper-plane that best differentiates the two classes.

Algorithm:

1. *Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.*
2. *The objective of applying SVMs is to find the best line in two dimensions or the best hyperplane in more than two dimensions in order to help us separate our space into classes.*
3. *The hyperplane (line) is found through the maximum margin, i.e., the maximum distance between data points of both classes.*

E. **XGB:** Extreme Gradient Boosting (xgboost) is similar to gradient boosting framework but more efficient. It has both linear model solver and tree learning algorithms. So, what makes it fast is its capacity to do parallel computation on a single machine. This makes xgboost at least 10 times faster than existing gradient boosting implementations. It supports various objective functions, including regression, classification and ranking. Since it is very high in predictive power but relatively slow with implementation, “xgboost” becomes an ideal fit for many competitions. It also has additional features for doing cross validation and finding important variables. There are many parameters which needs to be controlled to optimize the model.

Algorithm:

1. *In this algorithm, decision trees are created in sequential form.*
2. *Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results.*
3. *The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree.*
4. *These individual classifiers/predictors then ensemble to give a strong and more precise model.*
5. *It can work on regression, classification, ranking, and user-defined prediction problems.*

F. Neural Networks: Neural Network (or Artificial Neural Network) has the ability to learn by examples. ANN is an information processing model inspired by the biological neuron system. It is composed of a large number of highly interconnected processing elements known as the neuron to solve problems. It follows the non-linear path and process information in parallel throughout the nodes. A neural network is a complex adaptive system. Adaptive means it has the ability to change its internal structure by

adjusting weights of inputs. The neural network was designed to solve problems which are easy for humans and difficult for machines such as identifying pictures of cats and dogs, identifying numbered pictures. These problems are often referred to as pattern recognition. Its application ranges from optical character recognition to object detection.

Algorithm:

1. *Input layer receives input values.*
2. *A hidden layer(s) that fits between the input and output levels and contains a group of neurons (which could be a single layer or several layers).*
3. *Output layer that usually contains a neuron.*
4. *Its output is in the range of 0 to 1, or more than 0 and less than 1. However, several outputs are possible.*

III. DATA SUMMARY/PREPARE DATA/EVALUATE ALGORITHMS

- The dataset 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 0.172% of all transactions.
- It contains only numerical input variables which are the result of a PCA transformation.
- However, for modelling we have considered 10,000 random samples from the dataset.
- 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-sensitive learning.
- Feature 'Class' is the response variable, and it takes value 1 in case of fraud and 0 otherwise.
- Since accuracy is not an appropriate measure of model performance for imbalanced datasets, we have used the metric AREA UNDER ROC CURVE to evaluate how different methods of over-sampling or under-sampling the response variable can lead to better model training.
- To prepare the model, we have split the data into the training data and the testing data. We use the training data to prepare Logistic Regression, Random Forest, SVM and the XGBoost models.

IV. CODE

```
################################################################################
# Detecting Credit Card Fraud

# 1. Prepare Problem

# a) Load libraries
install.packages("smotefamily")
install.packages("e1071")
install.packages("caTools")
install.packages("class")
install.packages("rlang")
install.packages("caret")
install.packages("e1071")
install.packages("class")
install.packages("neuralnet")

library(neuralnet)
library(rlang)
library(e1071)
library(caTools)
library(class)
library(dplyr) # for data manipulation
library(stringr) # for data manipulation
library(caret) # for sampling
library(caTools) # for train/test split
library(ggplot2) # for data visualization
library(corrplot) # for correlations
library(Rtsne) # for tsne plotting
library(ROSE)# for ROSE sampling
library(rpart)# for decision tree model
library(Rborist)# for random forest model
library(xgboost) # for xgboost model

# b) Load dataset
df<- read.csv("creditcard.csv")
df<- df[sample(nrow(df), 10000), ]

# 2. Summarize Data

# a) Descriptive statistics

# Data Exploration
dim(df)
head(df,6)
tail(df,6)
```

```
table(df$Class)
summary(df$Amount)
names(df)
var(df$Amount)
sd(df$Amount)


# b) Data visualizations

# Distribution of class labels
common_theme <- theme(plot.title = element_text(hjust = 0.5, face = "bold"))

ggplot(data = df, 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


# Distribution of variable 'Time' by class
df %>%
  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


# Distribution of transaction amount by class
ggplot(df, aes(x = factor(Class), y = Amount)) + geom_boxplot() +
  labs(x = 'Class', y = 'Amount') +
  ggtitle("Distribution of transaction amount by class") + common_theme

# Feature Selection
correlations <- cor(df[,-1],method="pearson")
corrplot(correlations, number.cex = .9, method = "circle", type = "full", tl.cex=0.8,tl.col =
"black")

# 3. Prepare Data
```

```
# a) Data Cleaning

# checking missing values
colSums(is.na(df))

# Remove 'Time' variable
df <- df[,-1]

# Change 'Class' variable to factor
df$Class <- as.factor(df$Class)
levels(df$Class) <- c("Not_Fraud", "Fraud")

head(df)
# Scale numeric variables
df[,-30] <- scale(df[,-30])
head(df)


# 4. Evaluate Algorithms
# a) Split-out validation dataset

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

# b) Test options and evaluation metric

# class ratio initially
table(train$Class)

# downsampling
set.seed(9560)
down_train <- downSample(x = train[, -ncol(train)],
                         y = train$Class)
table(down_train$Class)

# Build down-sampled model
set.seed(5627)
down_fit <- rpart(Class ~ ., data = down_train)

# AUC on down-sampled data
pred_down <- predict(down_fit, newdata = test)
roc.curve(test$Class, pred_down[,2], plotit = FALSE)

# upsampling
set.seed(9560)
```

```
up_train <- upSample(x = train[, -ncol(train)],
                     y = train$Class)
table(up_train$Class)

# Build up-sampled model
set.seed(5627)
up_fit <- rpart(Class ~ ., data = up_train)

# AUC on up-sampled data
pred_up <- predict(up_fit, newdata = test)
roc.curve(test$Class, pred_up[,2], plotit = FALSE)

# rose
set.seed(9560)
rose_train <- ROSE(Class ~ ., data  = train)$data
table(rose_train$Class)

# Build rose model
set.seed(5627)
rose_fit <- rpart(Class ~ ., data = rose_train)

# AUC on ROSE fit data
pred_rose <- predict(rose_fit, newdata = test)
roc.curve(test$Class, pred_rose[,2], plotit = FALSE)

#6. Finalize Model
# a) Predictions on validation dataset
# b) Create standalone model on entire training dataset
# c) Save model for later use

# Decision Tree
set.seed(5627)
orig_fit <- rpart(Class ~ ., data = train)
pred_orig <- predict(orig_fit, newdata = test, method = "class")
roc.curve(test$Class, pred_orig[,2], plotit = TRUE)

# Logistic Regression
x = up_train[, -30]
y = up_train[,30]
model <- glm(Class ~.,family=binomial(link='logit'),data=up_train)
summary(model)
pred_glm = predict(model, newdata = test)
cm = table(test$Class, pred_glm)
roc.curve(test$Class, pred_glm, plotit = TRUE)

# Random Forest
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)

# SVM
labels <- up_train$Class # Convert class labels from factor to numeric
y <- recode(labels, 'Not_Fraud' = 0, "Fraud" = 1)
classifier = svm(formula = Class ~ .,
                 data = up_train,
                 type = 'C-classification',
                 kernel = 'linear')
classifier
y_pred = predict(classifier, newdata = test)
cm = table(test$Class, y_pred)
cm
roc.curve(test$Class, y_pred, plotit = TRUE)

# XGB
set.seed(42)
xgb <- xgboost(data = data.matrix(up_train[,-30]),
               label = y,
               eta = 0.1,
               gamma = 0.1,
               max_depth = 10,
               nrounds = 300,
               objective = "binary:logistic",
               colsample_bytree = 0.6,
               verbose = 0,
               nthread = 7,
)
xgb_pred <- predict(xgb, data.matrix(test[,-30]))
roc.curve(test$Class, xgb_pred, plotit = TRUE)

names <- dimnames(data.matrix(up_train[,-30]))[[2]]
importance_matrix <- xgb.importance(names, model = xgb) # Compute feature importance
matrix
xgb.plot.importance(importance_matrix[1:10,]) # Nice graph

# Neural Network
ANN_model =neuralnet(Class~.,up_train,linear.output=FALSE)
plot(ANN_model)
predANN=compute(ANN_model,test)
resultANN=predANN$net.result
resultANN=ifelse(resultANN>0.5,1,0)

#############################################################################
```

V. SCREENSHOTS OF OUTPUT

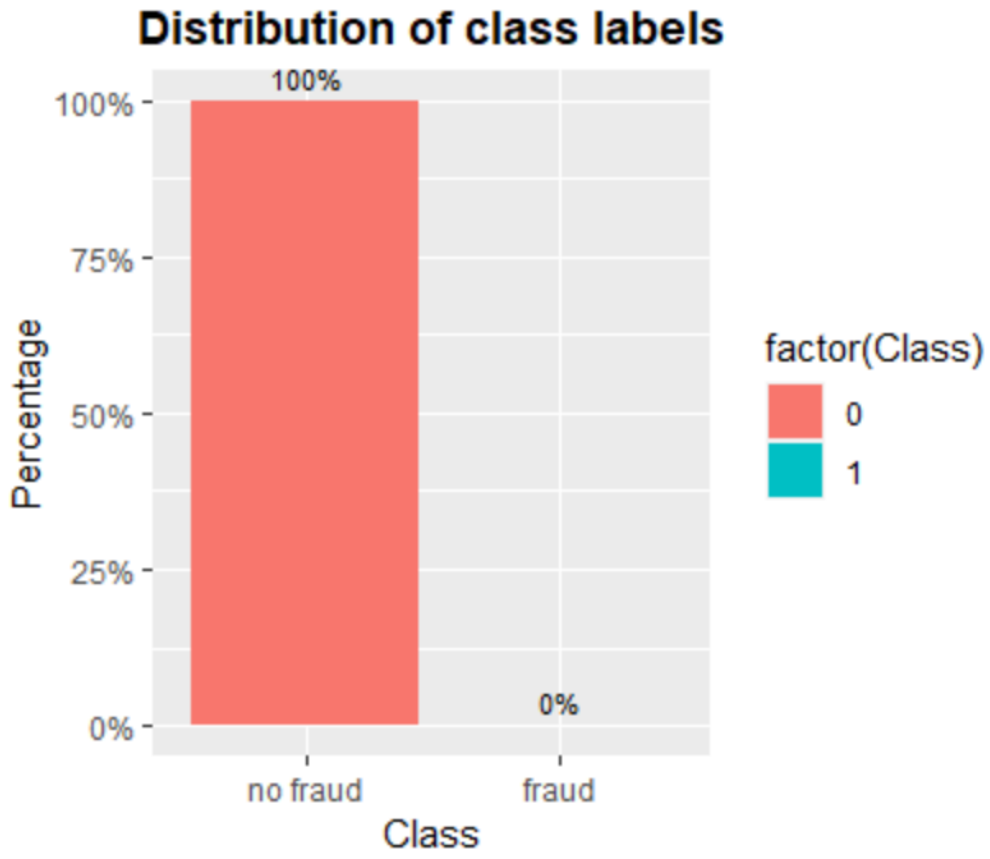
```
> library(neuralnet)
> library(rlang)
> library(e1071)
> library(caTools)
> library(class)
> library(dplyr) # for data manipulation
> library(stringr) # for data manipulation
> library(caret) # for sampling
> library(caTools) # for train/test split
> library(ggplot2) # for data visualization
> library(corrplot) # for correlations
> library(Rtsne) # for tsne plotting
> library(ROSE)# for ROSE sampling
> library(rpart)# for decision tree model
> library(Rborist)# for random forest model
> library(xgboost) # for xgboost model
> # b) Load dataset

> df<- read.csv("creditcard.csv")
> df<- df[sample(nrow(df), 10000), ]
>
```

```
> dim(df)
[1] 10000    31
> head(df,6)
        Time         V1         V2         V3         V4          V5         V6          V7          V8         V9
141744  84477  1.2542251  0.1533146  0.4135391  0.5515336 -0.52539054 -1.0167843  0.03167532 -0.20107962  0.1324351
40729   40373 -6.2076697  1.3451751 -2.8294671 -0.5862605 -2.36522608 -1.3252861 -1.06864377  2.90100795 -0.5688667
265036 161701 -1.8774760 -1.3707342  1.1534334 -0.9626916  3.03724110  0.3080279  0.36120042 -0.82618803 -0.7963844
271256 164488 -0.2257562  0.4992309  0.8066651  0.6853814  0.42488691 -0.3115593  0.41874351 -0.04778272  0.6530591
159554 112636 -0.4181440  0.7296859  0.1517136  2.5696264  2.21769750  0.1821927  0.60131038  0.16981628 -1.9444476
207018 136486  1.7142737 -0.3828553 -1.8165115  0.6017195 -0.03065004 -1.1530926  0.34643453 -0.31898749  0.9607451
              V10         V11         V12         V13         V14         V15         V16         V17          V18
141744 -0.05393577 -0.32494092  0.09503966 -0.10266794  0.3094825  1.07008844  0.27303525 -0.3795832 -0.343948744
40729  -0.26256478  0.02915084  1.24237204 -0.52022576  2.4556436 -0.27809010  1.57487864  0.3377249 -0.031950659
265036  1.41621919 -0.11983952 -0.57935005 -0.06627808 -0.6741227  0.94953146 -3.21654256  0.6080651 -0.834935519
271256 -0.38113442 -1.40404488  0.83162562  0.44408496 -0.6002898 -1.71005218 -0.47901162 -0.2210712 -0.928257195
159554  1.15557486 -0.27338698 -0.40342896 -0.40834729  0.6147487 -1.42102961  1.24028803 -1.4909484  0.728713018
207018 -0.98441149 -0.40107835  0.44728320  0.18293544 -1.6140338 -0.07838034 -0.00843839  1.2120722  0.002671239
              V19          V20         V21         V22          V23         V24         V25          V26         V27
141744 -0.169038500 -0.09785966 -0.08559376 -0.2544572  0.017874142  0.41946388  0.3511639  0.39217508 -0.04345167
40729   0.003117090 -0.31418919 -0.44936799 -1.5762536 -0.002637769  0.04423105  0.1817175  0.53130277  0.31275327
265036 -1.098715834 -1.18194225 -0.87217230 -0.5722425 -1.153049720 -0.25348575  0.1577900  0.38390161 -0.47022983
271256 -0.072565361 -0.31156307 -0.39513199 -0.9176026  0.270284196 -0.15241068 -0.8802264 -1.25730148 -0.11089346
159554 -1.374685607 -0.06474990  0.45753937  1.0145721  0.004990248  0.16856622 -0.8609766 -0.08588152  0.19155731
207018  0.004684685  0.15085864 -0.15491297 -0.5095508  0.041070850 -0.19533104 -0.1170444 -0.09877673 -0.02933647
               V28 Amount Class
141744  0.013787734   9.99     0
40729  -0.534229819  28.99     0
265036 -1.074753216  67.92     0
271256  0.125965937   8.90     0
159554  0.266144518   5.37     0
207018  0.008552109 165.98     0
```

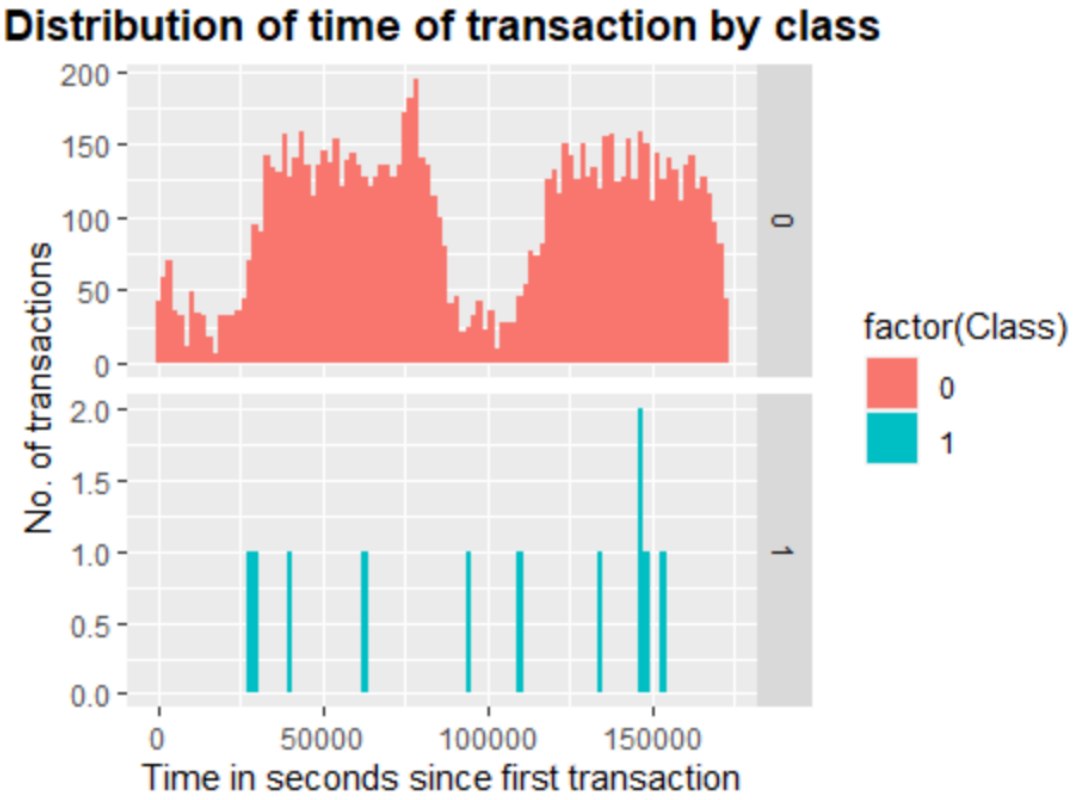
```
> tail(df,6)
        Time         V1         V2          V3         V4         V5         V6         V7          V8         V9
220570 142225 -0.9758753  0.4805713  0.26870461 -1.2199411  1.5047658  2.3045657  0.2552748 -0.09295291  2.0356523
60703   49456 -0.7223032  1.0581520  1.45238531 -0.1814653  0.4179179 -0.5035613  1.0939902 -0.51958466  0.7490385
252052 155630 -1.3901141 -0.1573343  1.78106958 -1.0324482  0.7272517 -0.4249165  0.4494269  0.10712295  0.2281043
252144 155669  1.8957470 -1.1932025 -0.03411366 -0.5356105 -1.7217103 -1.2145501 -0.8656020 -0.26500450 -0.1259599
85210   60652  0.9057432 -0.1593768  1.37877204  2.7477665 -0.8026969  0.6200164 -0.5873856  0.34016165  0.2613867
170271 120101  1.8675519  1.0527573 -1.75011290  4.0099371  1.0507453 -0.3324869  0.3154386  0.03072161 -1.1951106
              v10         v11         v12         v13         v14         v15         v16         v17         v18
220570  1.3601311 -0.54462765 -0.06362406  0.1411474 -1.3391420  1.18207231 -1.0783863 -0.08779398 -1.7234614
60703   0.8601307  0.05476572 -1.12849710 -1.6344479 -0.8760835  1.46224984 -0.1294261  0.04364798 -0.5083465
252052 -1.4142464 -1.00083315  0.21592221  0.5315095 -0.2092195  0.49651474  0.3454345 -0.94591101  0.5084341
252144  0.6791389  0.30671445  0.42179311  1.3718025 -0.6347396  0.43663802  1.0160896  0.29080072 -1.4620532
85210   0.4391994 -0.91686262 -0.32391829 -1.0561824 -0.1481842  0.07912968  0.3907612 -0.07095833 -0.4978708
170271  0.4994919  0.92910074 -0.53095803 -1.0751810 -2.1483332 -1.18550156  1.9955756  1.09971919  1.0982404
              v19         v20         v21         v22          v23         v24         v25         v26          v27
220570 -1.8235696  0.14361817  0.16027856  1.3533292  0.006431357 -1.26671071 -0.9407350  0.48671304 -0.728944087
60703  -0.4773159  0.39473365 -0.46579313 -0.6724277 -0.032799566 -0.02885727 -0.2701240  0.02072487 -0.146558228
252052 -0.8637157  0.28579685  0.33011815  0.6144503 -0.173112977  0.73168209  0.9160694 -0.56070739  0.011079697
252144 -0.1684339  0.22465141  0.53662258  1.4355825  0.148242003  0.94804853 -0.3313910 -0.13258615  0.009484914
85210  -1.2228423 -0.08028727 -0.01604977 -0.1162299  0.007687181  0.03770113  0.1541201 -0.01941589  0.035145040
170271 -1.3294042 -0.23818703 -0.39967307 -1.2116989  0.264815308 -0.67871459 -0.2755445 -0.33899309 -0.014763878
               v28 Amount Class
220570 -0.340427055   5.08     0
60703  -0.461940747   8.93     0
252052  0.090626077  95.00     0
252144 -0.015951550 112.40     0
85210   0.045751358  91.25     0
170271  0.003970285  13.01     0
> table(df$class)

   0    1
9978   22
```

```
> summary(df$Amount)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
   0.000    5.367   21.485   90.473   79.000 8790.260
> names(df)
 [1] "Time"   "V1"     "V2"     "V3"     "V4"     "V5"     "V6"     "V7"     "V8"     "V9"     "V10"    "V11"
[13] "V12"    "V13"    "V14"    "V15"    "V16"    "V17"    "V18"    "V19"    "V20"    "V21"    "V22"    "V23"
[25] "V24"    "V25"    "V26"    "V27"    "V28"    "Amount" "Class"
> var(df$Amount)
[1] 65362.89
> sd(df$Amount)
[1] 255.6617
```

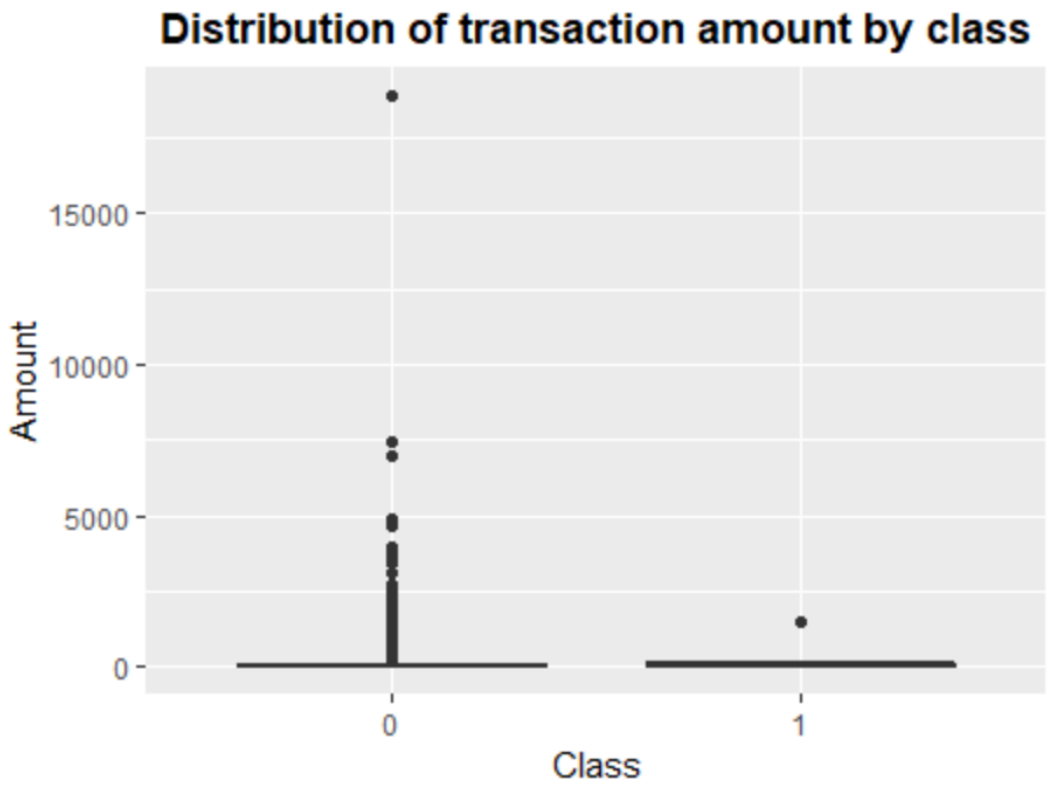
```
> # b) Data visualizations
>
> # Distribution of class labels
> common_theme <- theme(plot.title = element_text(hjust = 0.5, face = "bold"))
>
> ggplot(data = df, aes(x = factor(Class),
+                       y = prop.table(stat(count)), fill = factor(Class),
+                       label = scales::percent(prop.table(stat(count))))) +
+   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
```



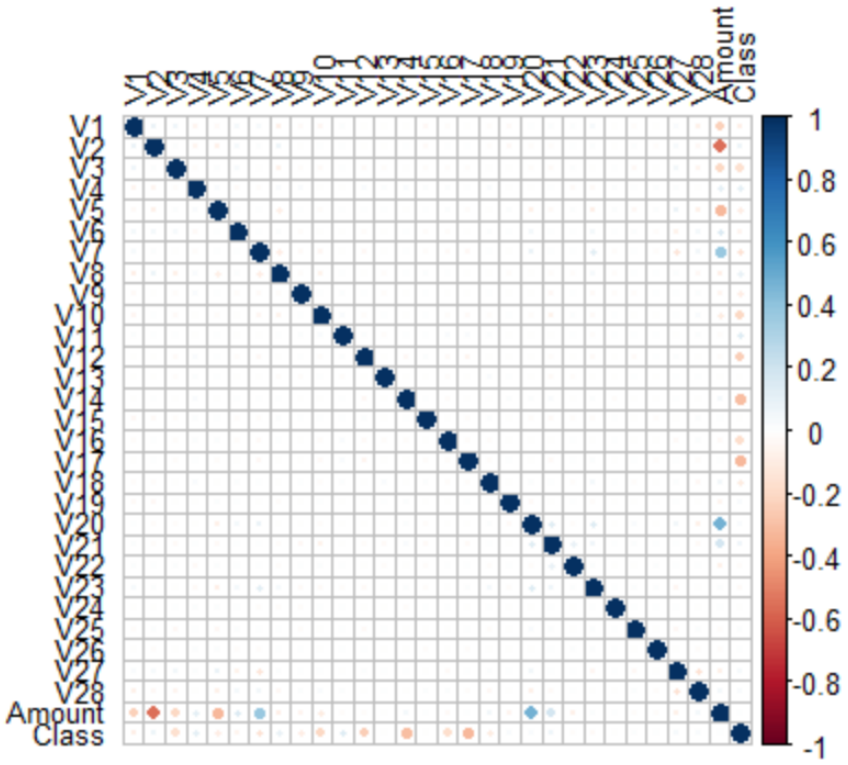
```
> # Distribution of variable 'Time' by class
> df %>%
+   ggplot(aes(x = Time, fill = factor(Class))) + geom_histogram(bins = 100)+
+   labs(x = 'Time in seconds since first transaction', y = 'No. of transactions') +
+   ggtitle('Distribution of time of transaction by class') +
+   facet_grid(Class ~ ., scales = 'free_y') + common_theme
```



```
> # Distribution of transaction amount by class
> ggplot(df, aes(x = factor(Class), y = Amount)) + geom_boxplot() +
+   labs(x = 'Class', y = 'Amount') +
+   ggtitle("Distribution of transaction amount by class") + common_theme
```



```
> # Feature Selection
> correlations <- cor(df[,-1],method="pearson")
> corrplot(correlations, number.cex = .9, method = "circle", type = "full", tl.cex=0.8,tl.col = "black")
```




```
> # 3. Prepare Data
>
> # a) Data Cleaning
>
> # checking missing values
> colSums(is.na(df))
  Time     V1     V2     V3     V4     V5     V6     V7     V8     V9
     0      0      0      0      0      0      0      0      0      0
   V10    V11    V12    V13    V14    V15    V16    V17    V18    V19
     0      0      0      0      0      0      0      0      0      0
   V20    V21    V22    V23    V24    V25    V26    V27    V28 Amount
     0      0      0      0      0      0      0      0      0      0
 Class
     0
>
> # Remove 'Time' variable
> df <- df[,-1]
```

```
> # Change 'Class' variable to factor
> df$Class <- as.factor(df$Class)
> levels(df$Class) <- c("Not_Fraud", "Fraud")
>
> head(df)
                V1          V2         V3         V4          V5
210899 -0.74439031  0.80085649 -0.7577166 -0.6003097  1.05337393
13593   1.17723017  0.28630092  0.5809844  1.1873753 -0.03078551
128840  1.06931512 -0.78224479  1.3040650  0.6229991 -1.36130896
139825 -0.91094258 -0.04687148  1.5274494 -0.1708919 -0.75349471
160134 -0.07698506  1.03047442 -0.1345232 -0.4265377  0.75463498
216092 -0.46301520  0.32577193 -0.1394111 -0.9059659  0.95995595
                V6         V7          V8         V9         V10
210899 -0.11373980  0.5391535  0.39677984 -0.1511842 -0.4182998
```

```
> # Scale numeric variables
> df[,-30] <- scale(df[,-30])
> head(df)
                V1          V2         V3         V4          V5
210899 -0.35520284  0.48234226 -0.4995781 -0.4296951  0.76524071
13593   0.61005207  0.17213892  0.3524244  0.8058966 -0.01095193
128840  0.55584494 -0.47204118  0.8126217  0.4158175 -0.96352631
139825 -0.43886422 -0.02871634  0.9547924 -0.1328950 -0.52836804
160134 -0.01995652  0.62076901 -0.1029531 -0.3095893  0.55136168
216092 -0.21386447  0.19593429 -0.1060639 -0.6409550  0.69835908
                V6         V7          V8         V9         V10
210899 -0.09392956  0.4311022  0.35414282 -0.1415164 -0.3659627
```

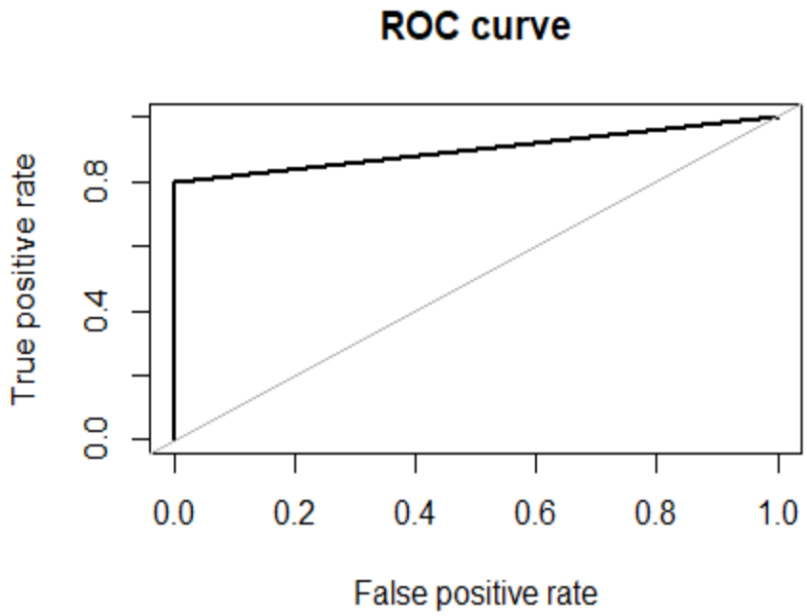
```
> set.seed(123)
> split <- sample.split(df$Class, SplitRatio = 0.7)
> train <-  subset(df, split == TRUE)
> test <- subset(df, split == FALSE)
```

```
> table(train$Class)

Not_Fraud     Fraud
     6987        13
```

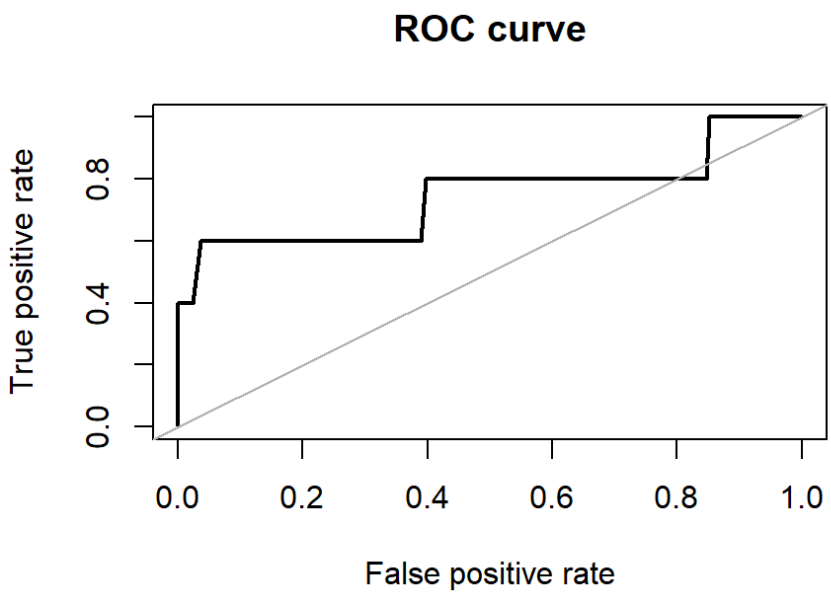
```
> set.seed(9560)
> down_train <- downSample(x = train[, -ncol(train)],
+                          y = train$Class)
> table(down_train$Class)

Not_Fraud     Fraud
       13        13
```

```
> set.seed(5627)
> down_fit <- rpart(Class ~ ., data = down_train)

> pred_down <- predict(down_fit, newdata = test)
> roc.curve(test$Class, pred_down[,2], plotit = FALSE)
Area under the curve (AUC): 0.771

> set.seed(9560)
> up_train <- upSample(x = train[, -ncol(train)],
+                      y = train$Class)
> table(up_train$Class)

Not_Fraud     Fraud
     6987      6987

> set.seed(5627)
> up_fit <- rpart(Class ~ ., data = up_train)

> pred_up <- predict(up_fit, newdata = test)
> roc.curve(test$Class, pred_up[,2], plotit = FALSE)
Area under the curve (AUC): 0.899

> set.seed(9560)
> rose_train <- ROSE(Class ~ ., data  = train)$data
>
> table(rose_train$Class)

Not_Fraud     Fraud
     3431      3569

> set.seed(5627)
> rose_fit <- rpart(Class ~ ., data = rose_train)
> pred_rose <- predict(rose_fit, newdata = test)
> roc.curve(test$Class, pred_rose[,2], plotit = FALSE)
Area under the curve (AUC): 0.890
```

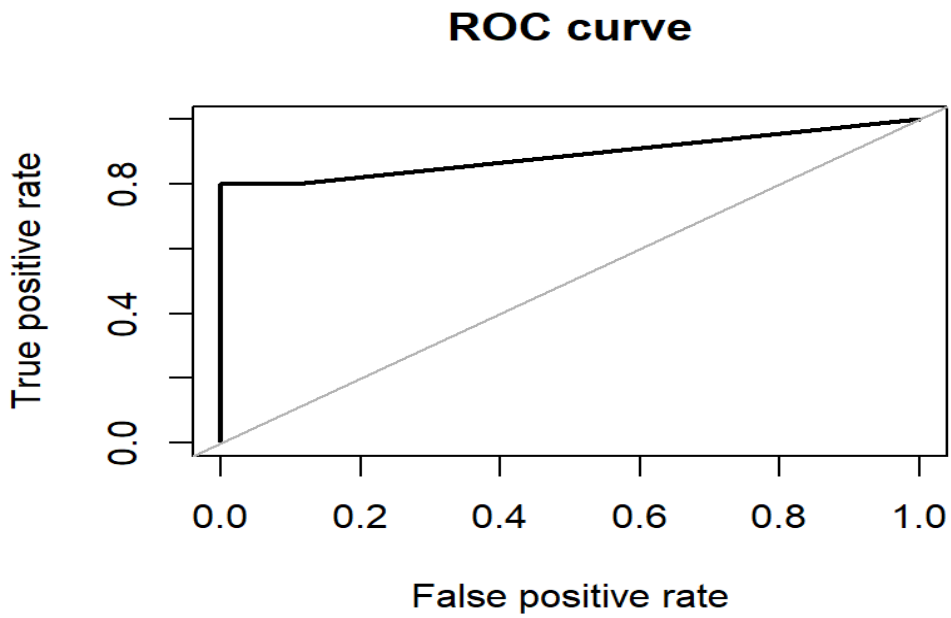
```
> set.seed(5627)
> orig_fit <- rpart(Class ~ ., data = train)
> set.seed(5627)
> orig_fit <- rpart(Class ~ ., data = train)
>
> #Evaluate model performance on test set
> pred_orig <- predict(orig_fit, newdata = test, method = "class")
> roc.curve(test$Class, pred_orig[,2], plotit = TRUE)
Area under the curve (AUC): 0.900
```



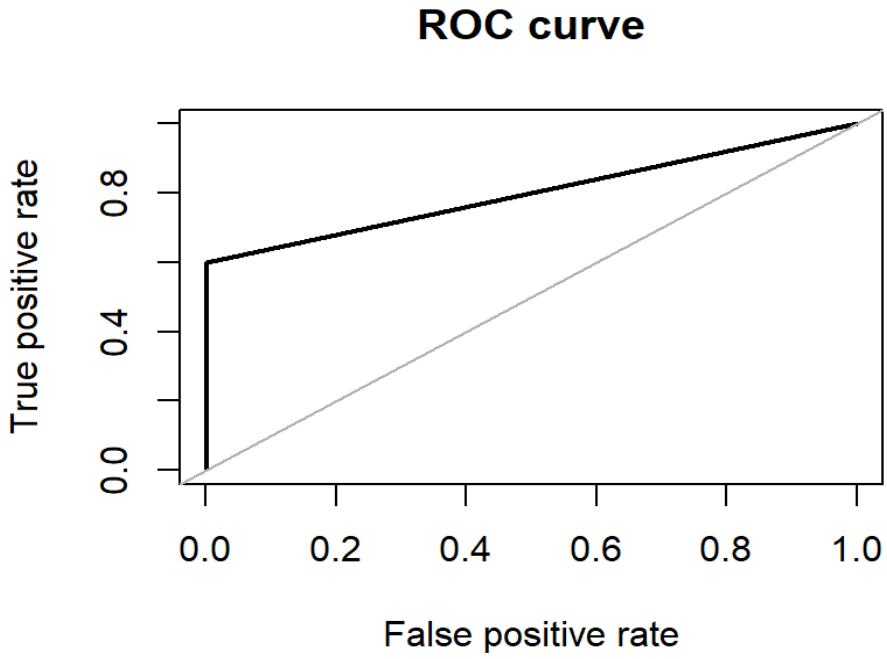
```
> x = up_train[, -30]
> y = up_train[,30]
> model <- glm(Class ~.,family=binomial(link='logit'),data=up_train)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> summary(model)

Call:
glm(formula = Class ~ ., family = binomial(link = "logit"), data = up_train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-8.4904   0.0000   0.0000   0.0054   0.0788

Number of Fisher Scoring iterations: 21

> pred_glm = predict(model, newdata = test)
> cm = table(test$Class, pred_glm)
> roc.curve(test$Class, pred_glm, plotit = TRUE)
Area under the curve (AUC): 0.745
```



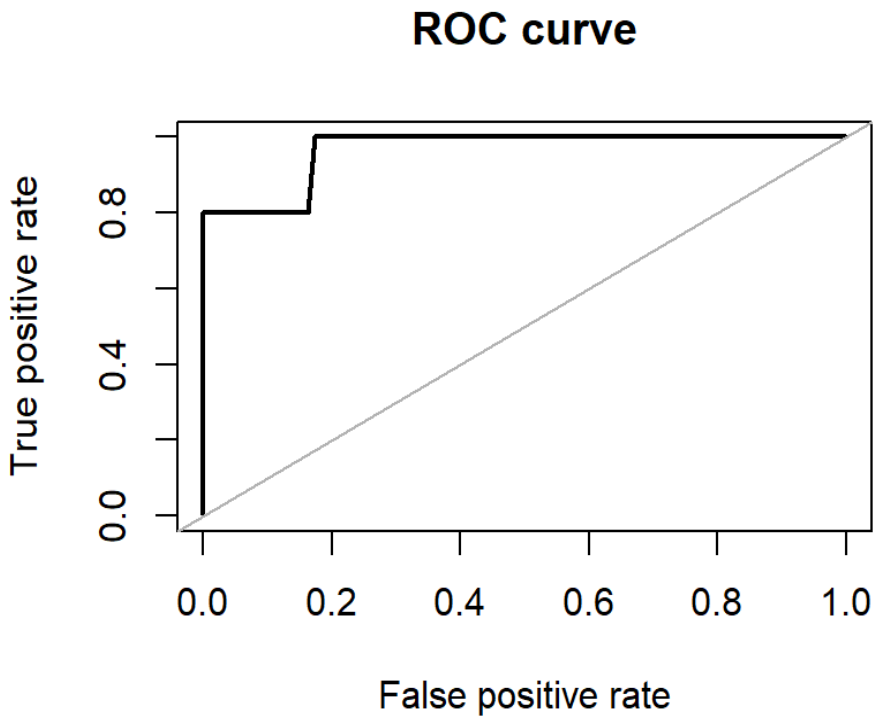
```
> labels <- up_train$Class
> y <- recode(labels, 'Not_Fraud' = 0, "Fraud" = 1)
> classifier = svm(formula = Class ~ .,
+                  data = up_train,
+                  type = 'C-classification',
+                  kernel = 'linear')
> classifier

Call:
svm(formula = Class ~ ., data = up_train, type = "C-classification",
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  54

>
> y_pred = predict(classifier, newdata = test)
>
> cm = table(test$Class, y_pred)
>
> cm
           y_pred
            Not_Fraud Fraud
  Not_Fraud      2992     3
  Fraud             2     3
>
> roc.curve(test$Class, y_pred, plotit = TRUE)
Area under the curve (AUC): 0.799
```



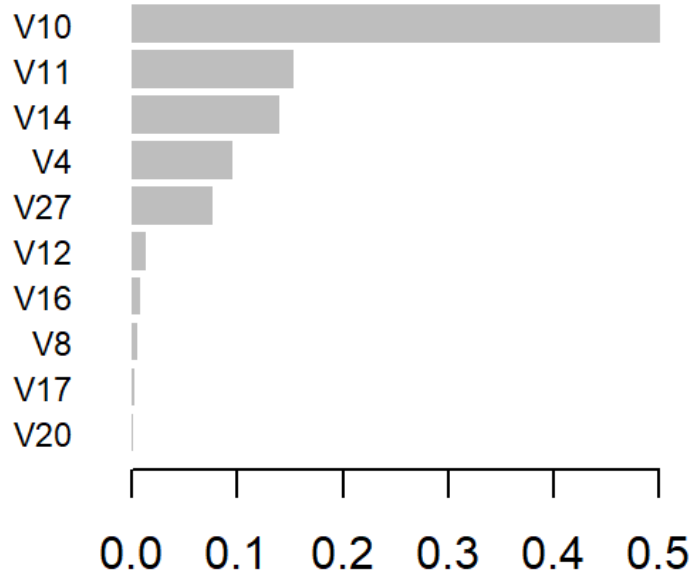
```
> labels <- up_train$Class
> y <- recode(labels, 'Not_Fraud' = 0, "Fraud" = 1)
> classifier = svm(formula = Class ~ .,
+                  data = up_train,
+                  type = 'C-classification',
+                  kernel = 'linear')
> classifier

Call:
svm(formula = Class ~ ., data = up_train, type = "C-classification",
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  54

>
> y_pred = predict(classifier, newdata = test)
>
> cm = table(test$Class, y_pred)
>
> cm
           y_pred
            Not_Fraud Fraud
  Not_Fraud      2992     3
  Fraud             2     3
>
> roc.curve(test$Class, y_pred, plotit = TRUE)
Area under the curve (AUC): 0.799
```



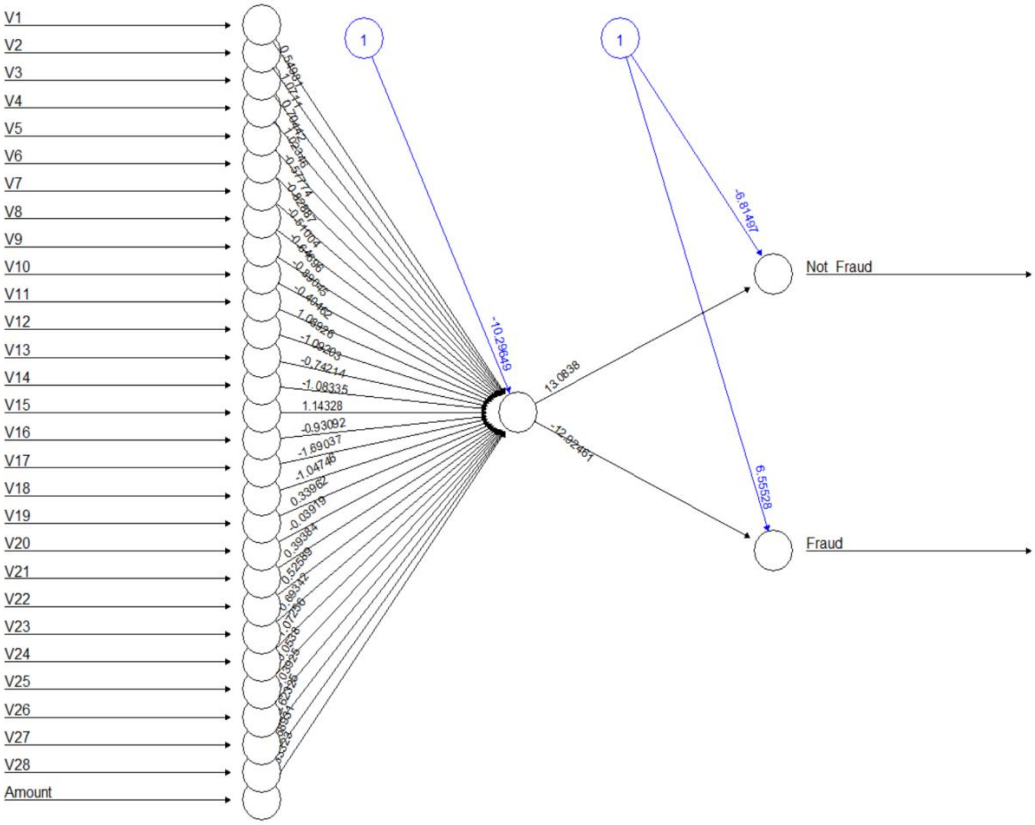
```
> set.seed(42)
> xgb <- xgboost(data = data.matrix(up_train[,-30]),
+                label = y,
+                eta = 0.1,
+                gamma = 0.1,
+                max_depth = 10,
+                nrounds = 300,
+                objective = "binary:logistic",
+                colsample_bytree = 0.6,
+                verbose = 0,
+                nthread = 7,
+ )
>
>
> xgb_pred <- predict(xgb, data.matrix(test[,-30]))
>
> roc.curve(test$Class, xgb_pred, plotit = TRUE)
Area under the curve (AUC): 0.966
```



```
> names <- dimnames(data.matrix(up_train[,-30]))[[2]]
> # Compute feature importance matrix
> importance_matrix <- xgb.importance(names, model = xgb)
> # Nice graph
> xgb.plot.importance(importance_matrix[1:10,])
```



```
> ANN_model =neuralnet(Class~.,up_train,linear.output=FALSE)
> plot(ANN_model)
> predANN=compute(ANN_model,test)
> resultANN=predANN$net.result
> resultANN=ifelse(resultANN>0.5,1,0)
```



VI. CONCLUSION

- As shown below, the dataset is highly imbalanced.

```
> table(df$Class)

Not_Fraud     Fraud
     9982        18
```

- It is imperative to make sure that the dataset is balanced before applying modelling techniques.
- Hence, we used transformation techniques like up-sampling, down-sampling and Random Over Sampling (ROSE) to balance the dataset.
- Since accuracy is not an appropriate measure of model performance for imbalanced datasets, we have used the metric AREA UNDER ROC CURVE to evaluate how different methods of over-sampling or under-sampling the response variable can lead to better model training.
- For modelling techniques, we used Decision Tree Algorithm on all the three sampled datasets.
- As observed below for each sampling technique, up-sampling has the highest AUC score, so we have used up-sampled dataset for modelling.

Sampling Techniques	AUC Score
Up-sampling	0.899
Down-sampling	0.771
ROSE	0.890

- With an AUC score of 0.966 the XGBOOST model has performed the best though both the random forest and logistic regression models have shown reasonable performance.

Modelling Techniques	AUC Score
Logistic Regression	0.745
Random Forest	0.799
SVM	0.799
XGB	0.966

In this project we have tried to show different methods of dealing with unbalanced datasets like the fraud credit card transaction dataset where the instances of fraudulent cases is few compared to the instances of normal transactions. We have argued why accuracy is not an appropriate measure of model performance here and used the metric AREA UNDER ROC CURVE to evaluate how different methods of over-sampling or under-sampling the response variable can lead to better model training. We concluded that the over-sampling technique works best on the dataset and achieved significant improvement in model performance over the imbalanced data. The best score of 0.966 was achieved using an XGBOOST model though both random forest and logistic regression models performed well too. This project has demonstrated the importance of sampling effectively, modelling and predicting data with an imbalanced dataset.