Credit Card Fraud Detection in R

Langat Erick

2023-06-18

Credit Card Fraud Detection in R

Introduction

Credit card fraud is a major concern for financial institutions and consumers alike. Fraudulent transactions not only result in financial loses for banks, but can damage their reputation and lose the trust of customers. The increase in online transactions over the past decade has made real-time detection and prevention of financial fraud even more important.

A dataset has been provided that contains credit card transactions over a period of 2 days from September 2013. The dataset is highly unbalanced - containing 492 fraudulent transactions out of 284,807 total. To maintain anonymity and confidentiality, most of the original features have been transformed into principle components. Only the features time, class and amount have been preserved in their original form.

This notebook represents an attempt to analyse the data within the dataset and recommend a model that could be used for a practical credit card fraud detection system.

Project Outline

1. Importing libraries and dataset
2. Data Exploration
3. Data Manipulation
4. Data Modelling
    - 4.1 Logistic Regression
    - 4.2 Naive Bayes Classifier
    - 4.3 Decision Tree
5. Results and Limitations
6. Conclusion

```
library(tidyverse)
# library(Hmisc)
library(caret)
library(ROSE)
library(ggplot2)
```

```r
#library(gridExtra)
library(e1071)
library(corrplot)
library(broom)
library(partykit)
```

```r
# Load the dataset

data <- read.csv('C:/Users/JIT/Desktop/creditcard.csv',
                 stringsAsFactors = T)
head(data)
```

Importing libraries and dataset

```
##   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
```

```r
str(data)
```

```
## '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 ...
```

```r
# Convert Class column into Factors

data$Class <- as.factor(data$Class)
levels(data$Class) <- c("Legit", "Fraud")
```

```r
summary(data)
```

Data Exploration

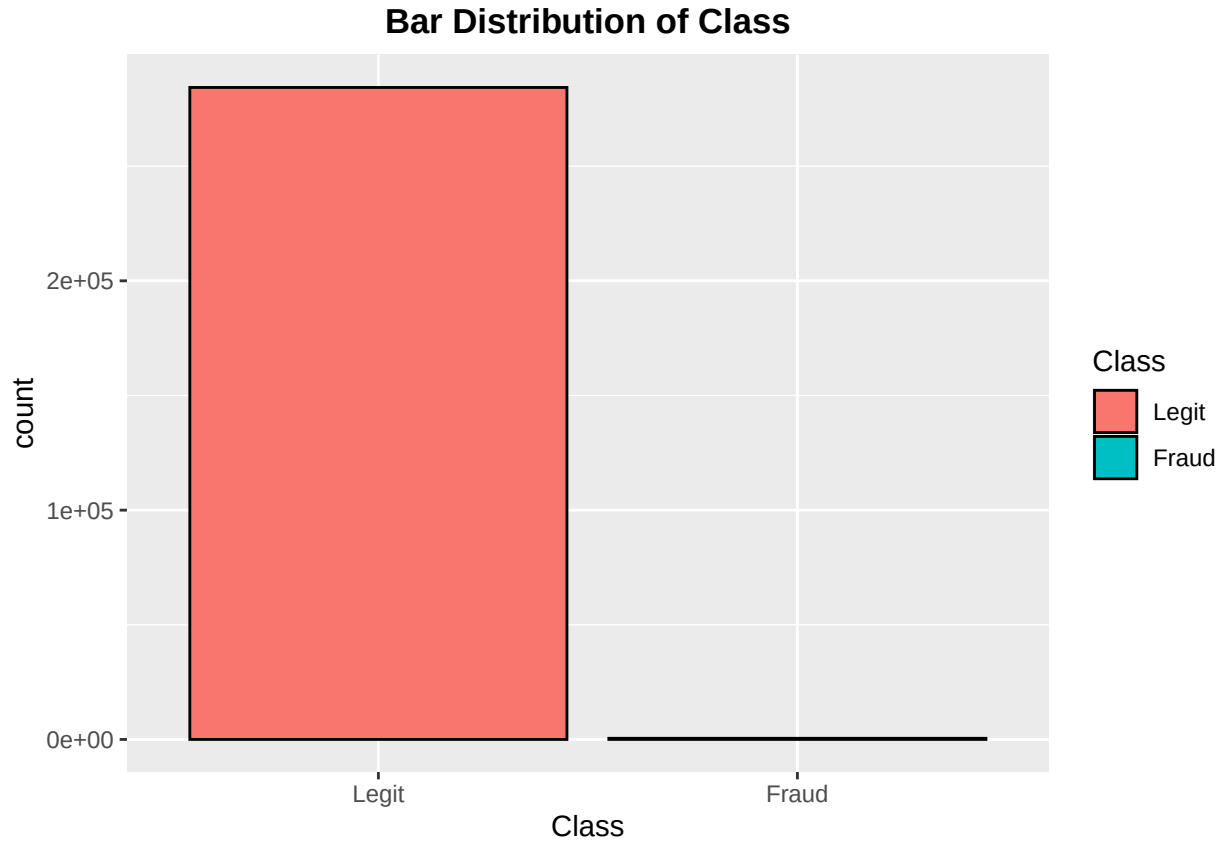
```
##       Time              V1                    V2                    V3
##  Min.   :     0   Min.   :-56.40751   Min.   :-72.71573   Min.   :-48.3256
##  1st Qu.: 54202   1st Qu.: -0.92037   1st Qu.: -0.59855   1st Qu.: -0.8904
##  Median : 84692   Median :  0.01811   Median :  0.06549   Median :  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   Legit:284315
##  1st Qu.: -0.05296   1st Qu.:    5.60   Fraud:   492
```

```
##  Median :  0.01124   Median :   22.00
##  Mean   :  0.00000   Mean   :   88.35
##  3rd Qu.:  0.07828   3rd Qu.:   77.17
##  Max.   : 33.84781   Max.   :25691.16
```

The majority of the data has undergone PCA, limiting the amount of information that can be obtained from most features. However, the "Amount" and "Class" features could provide some insight of the data.

```
ggplot(data, aes(x=Class, fill = Class)) +
        geom_bar(color = "black") +
         ggtitle("Bar Distribution of Class") +
         theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```




```
counts <- table(data$Class)
result <- data.frame(table(data$Class), round(prop.table(counts), 5))
noms <- c("Class", "Value", "none", "Proportion")

names(result) <- noms

print(result[c(1,2,4)])
```

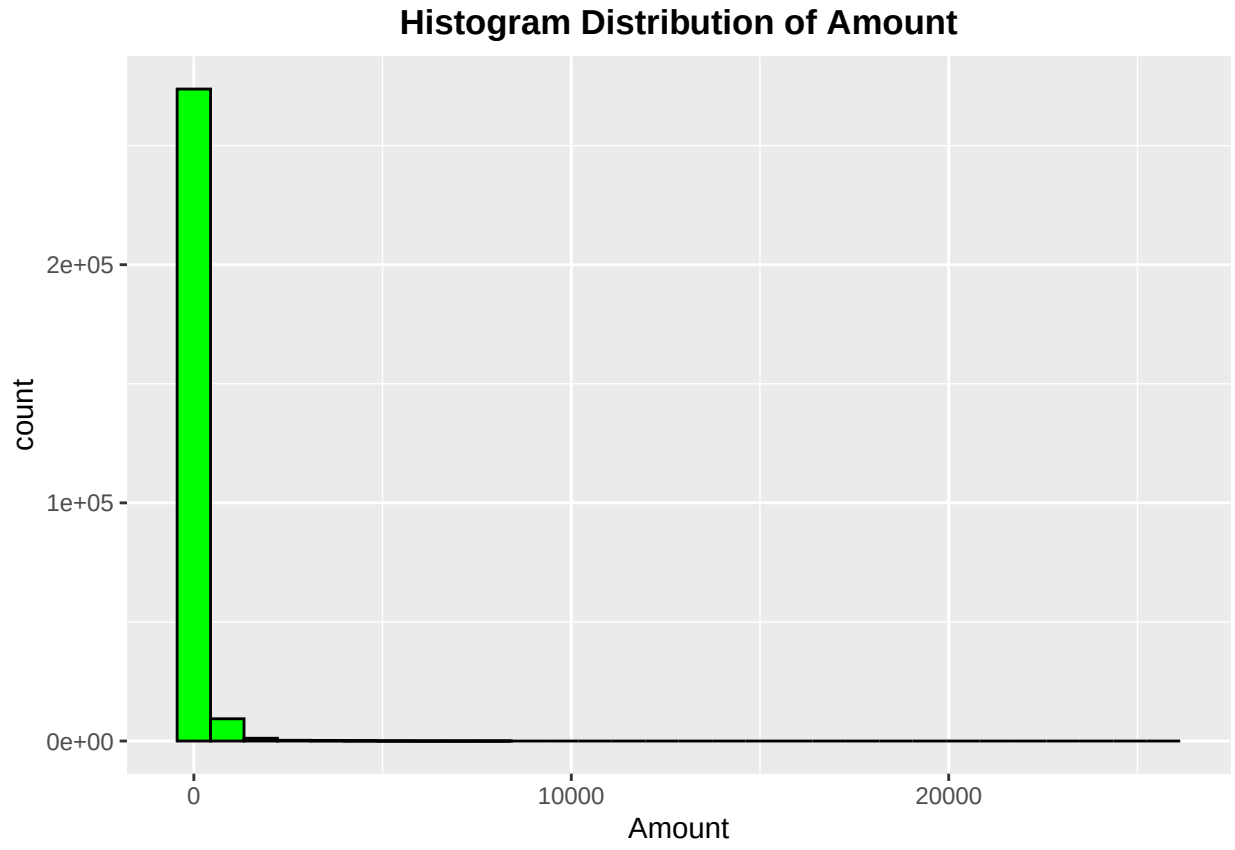
```
##   Class  Value Proportion
## 1 Legit 284315    0.99827
## 2 Fraud    492    0.00173
```

- **Legit** cases constitute 99.8% (284315) of the dataset.
- **Fraud** cases constitute 0.2% (492)

As expected from the brief, analysis of the Class feature shows that the dataset is unbalanced and the vast majority of the cases are legitimate. Accuracy will not be an appropriate measure of performance here and AUC will be used instead.

```
p <- ggplot(data, aes(x=Amount)) + geom_histogram(fill = "green",
                                  color = "black", bins = 30) +
       ggtitle("Histogram Distribution of Amount") +
        theme(plot.title = element_text(hjust = 0.5, face = "bold"))

p
```




```
# Function to find range of outliers in Amount

find_outlier_range <- function(x){
    outliers <- boxplot.stats(x)$out
    return(range(outliers))
}

find_outlier_range(data$Amount)
```

```
## [1]   184.52 25691.16
```

The histogram above shows that the vast majority of the transactions have low values. However, there are a non-neglible number of outliers with values ranging from 184.52 up to 25691.16 dollars. There is clearly a positive skew in the feature that will need to be considered during feature selection and modelling.

Data Manipulation

```
# Check for missing values

colSums(is.na(data))
```

```
##   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
```

```
sum(is.na(data))
```

```
## [1] 0
```

There are no missing values in the dataset.

```
# Check for duplicate rows

sum(duplicated(data))
```

```
## [1] 1081
```

```
# Remove any duplicate rows

data <- distinct(data)
```

There are 1081 duplicate rows in the dataset which can be removed.

Also the Time feature can be removed as each value is unique and provides no predictive power.

```
data$Time <- NULL
```

```
# Standardise Amount feature

data$Amount <- scale(data$Amount)

summary(data$Amount)
```

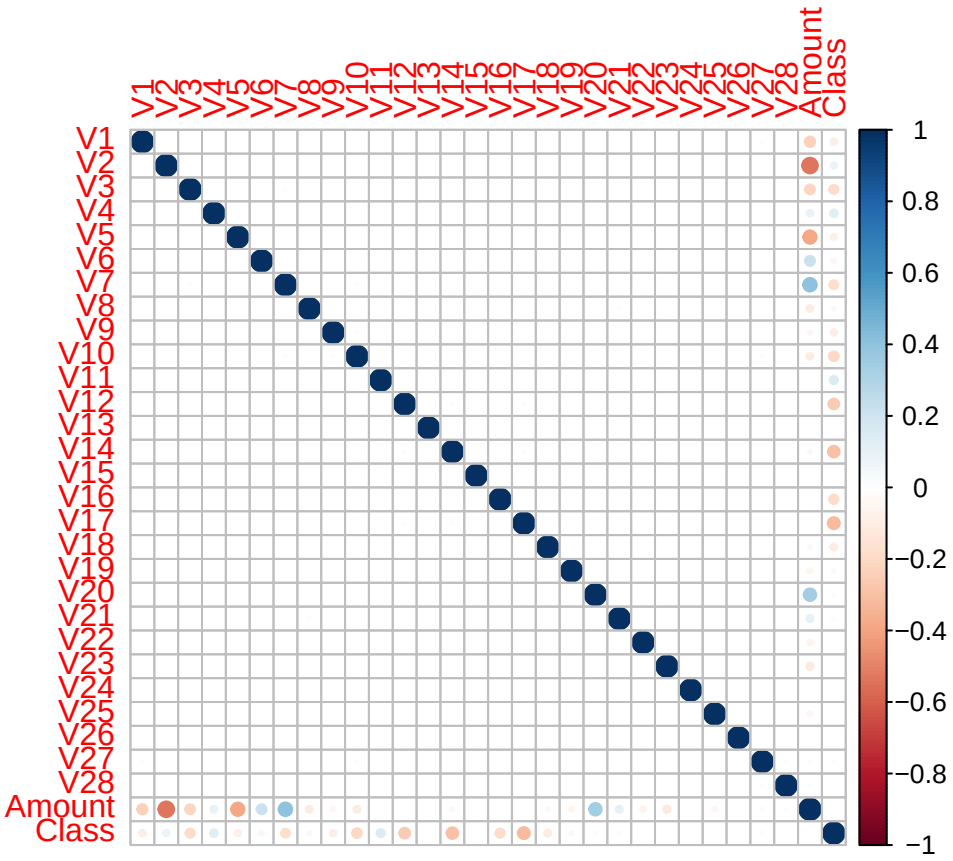
```
##        V1
##  Min.   : -0.35333
##  1st Qu.: -0.33096
##  Median : -0.26547
##  Mean   :  0.00000
##  3rd Qu.: -0.04378
##  Max.   :102.24738
```

```r
data2 <- data

data2$Class <- as.numeric(data2$Class)

corr <- cor(data2[], method = "pearson")

corrplot(corr)
```



As the data has previously gone through PCA, there is little to no correlation between most of the features. Correlation between the target variable and the PCA features varys with no particular features standing out.

Data Modelling The dataset will be split for training and testing, with 80% of the data used for training and 20% used for testing. The seed will also be set to reproduce results.

```r
set.seed(123)

indices <- createDataPartition(data$Class, p=0.8, list = F)
trainData <- data[indices,]
testData <- data[-indices,]
```

We are going to be building using the following models:

- Logistic Regression

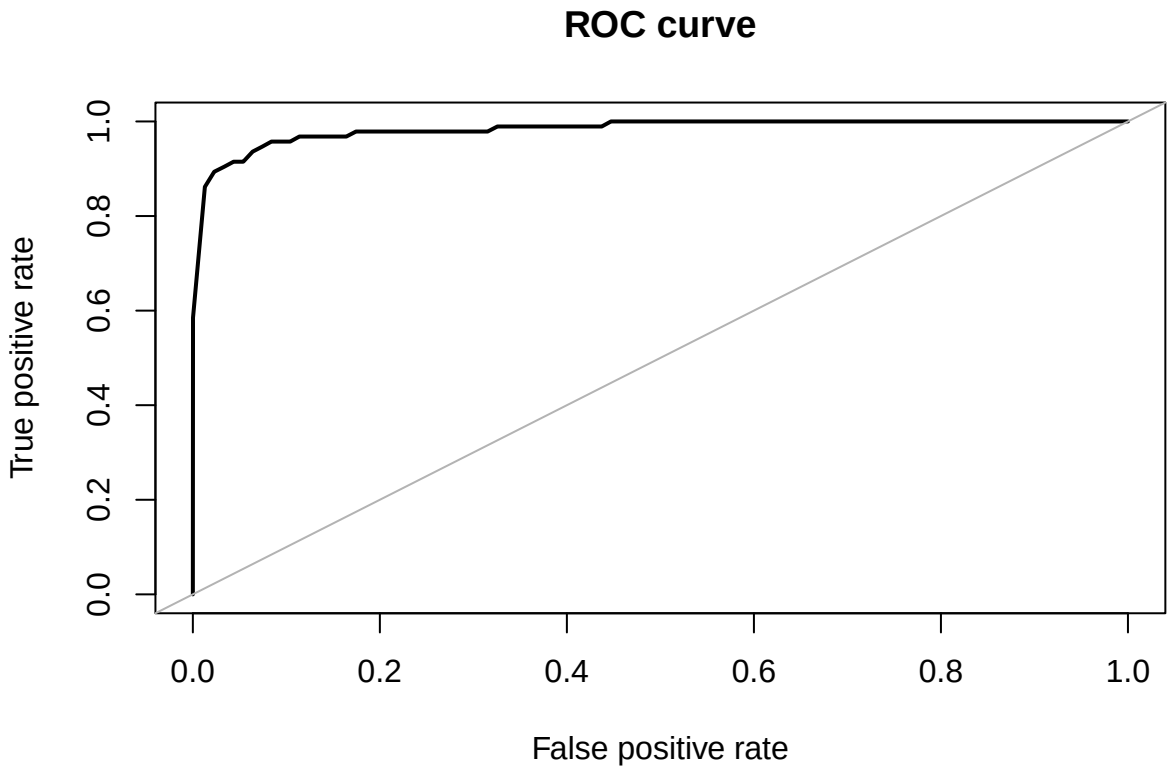
- Naive Bayes Classifier
- Decision Tree

Logistic Regression

```
model_lr <- glm(Class ~ ., data = trainData, family = "binomial")
```

```
model_lr_prediction <- predict(model_lr, newdata = testData, type = 'response')

roc.curve(testData$Class, model_lr_prediction, plotit = TRUE)
```

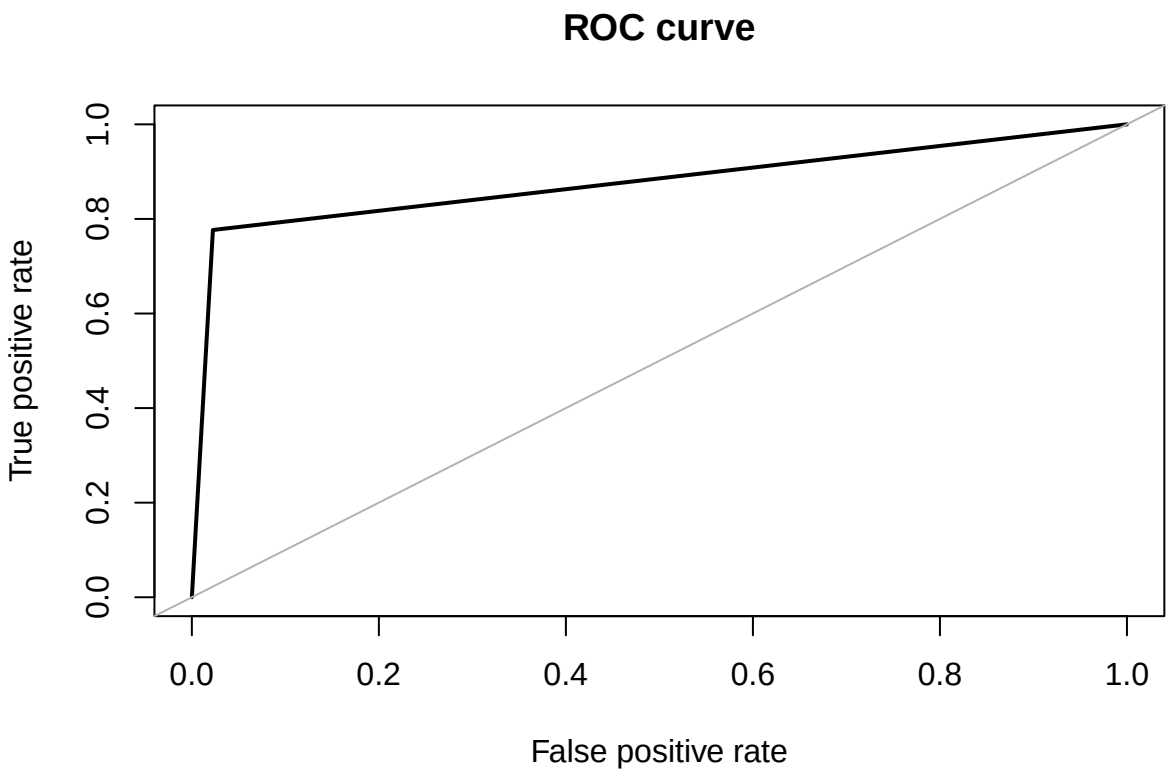



```
## Area under the curve (AUC): 0.983
```

Naive Bayes Classifier

```
NBmodCCF <- naiveBayes(Class ~ ., data = trainData, laplace = 1)
model_nb_prediction <- predict(NBmodCCF,
                               newdata = testData,
                               type = "class")

roc.curve(testData$Class, model_nb_prediction, plotit = TRUE)
```



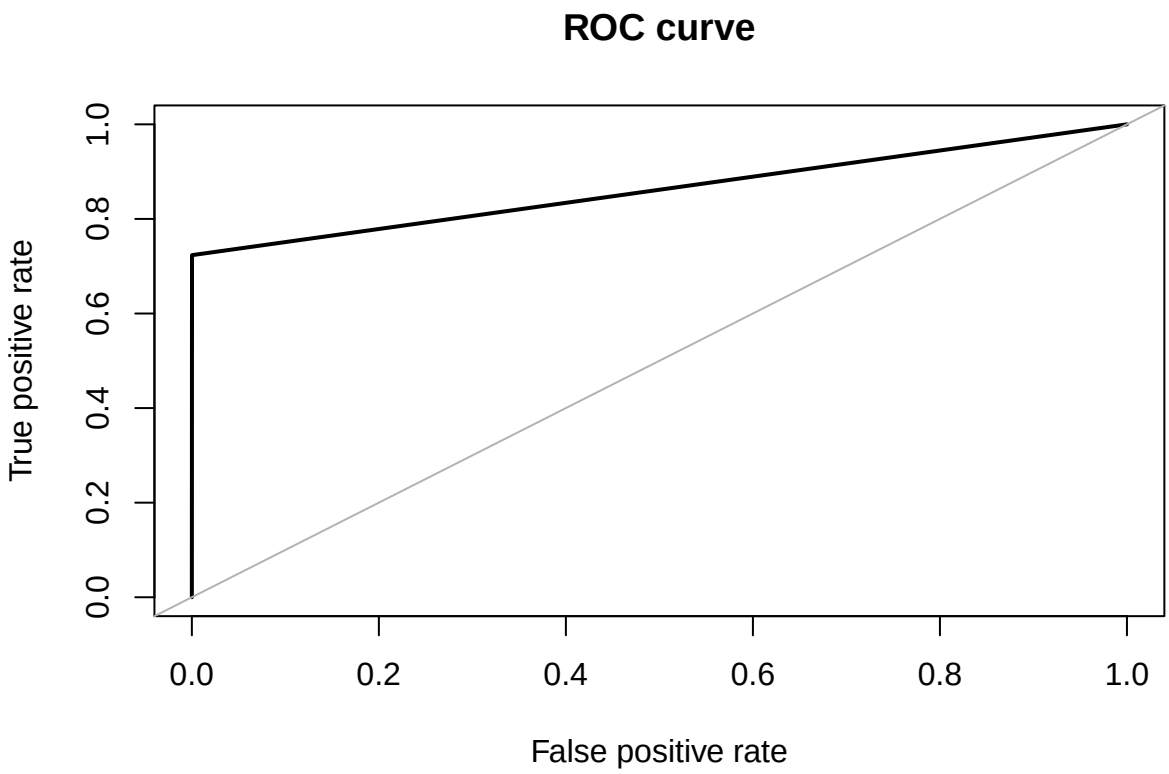

```
## Area under the curve (AUC): 0.877
```

Decision Tree

```
set.seed(123)

DTmodCCF <- ctree(Class ~ .,
                  data=trainData)
model_ctree_prediction <- predict(DTmodCCF, newdata = testData, type = "response")

roc.curve(testData$Class, model_ctree_prediction, plotit = TRUE)
```




```
## Area under the curve (AUC): 0.862
```

Results and Limitations The AUC performance results for the 3 models are as follows:

- Logistic Regression: 0.983
- Naive Bayes Classifier: 0.877
- Decision Tree: 0.856

Although the naive bayes classifier and decision tree performed well, logistic regression is the clear winner in this group.

There are some limitations of the project that may be addressed in the future that may change the outcome. These would include:

- Use of only 3 modelling techniques. If we add more modelling types in the future, there may be a model that performs even better than the LR model.
- Lack of different sampling techniques. This is something I will learn more about and add to future projects.

Conclusion

In conclusion, this project aimed to develop machine learning models to detect credit card fraud. Three models were evaluated: Logistic Regression, Naive Bayes Classifier, and Decision Tree. The results showed that the Logistic Regression model achieved the highest AUC score of 0.983, while the Naive Bayes Classifier and Decision Tree models achieved AUC scores of 0.877 and 0.856, respectively.

The high AUC score of the Logistic Regression model indicates that it is able to accurately distinguish between fraudulent and non-fraudulent transactions with a high degree of confidence. Both the Naive Bayes Classifier and Decision Tree models also demonstrated good performance, with AUC scores of 0.877 and 0.856.

However, it is important to note that there are some limitations to the project, including the use of only three modeling techniques and the lack of different sampling techniques. These limitations could be addressed in future work to improve the performance of the models.

Overall, the results of this project demonstrate the potential of machine learning models in detecting credit card fraud, and highlight the importance of continued research and development in this field. With the increasing prevalence of credit card fraud, it is essential to develop accurate and efficient methods for detecting and preventing fraud. This project provides a starting point for further research in this area, and offers insights into the potential of machine learning models for detecting credit card fraud.