Final_Project_Step03_MishraDebabrata

Debabrata Mishra

2023-03-03

```
# Assignment: Final Project Step 3 - (WEEK 11 & 12)
# Name: Mishra, Debabrata
# Problem Topic : Credit Card Fraud Detection & Prevention
```

1.0 Introduction

Credit card fraud is a growing concern for financial institutions, merchants, and consumers alike. With the increasing use of credit cards for online transactions, the opportunity for fraudsters to commit fraudulent activities has increased dramatically. Due to the technological innovation and the emergence of a new Payment techniques, we do see huge increase in BOT attacks, Account takeover, new account fraud, cloned cards, cards-not-present schemes and mobile payments. Such widespread acceptance of cashless transactions leads fraudsters to carry out fraudulent attacks regularly and change their tactics to avoid detection.

In this paper, we aim to address the problem of detecting credit card fraud using data science techniques. The goal of this paper is to answer the research question: "How can we detect credit card fraud using data science techniques?" To answer this question, we will use machine learning models to analyze a dataset of credit card transactions. The data will be pre-processed and cleaned to remove any irrelevant information and protect the privacy of the consumers. The aim of the project is to develop a machine learning model that can identify fraudulent credit card transactions in real-time. Here is a general outline of the steps involved in a credit card fraud detection project using data science:

Data Collection:

The first step is to collect a large dataset of credit card transactions, both fraudulent and non-fraudulent. This data can be collected from various sources such as banks, financial institutions, or online platforms.

Data Pre-processing:

Once the data is collected, it needs to be cleaned, formatted, and transformed into a format that can be used for building the machine learning model. This includes dealing with missing values, converting categorical variables to numerical values, and scaling the data to ensure that all variables are on a similar scale.

Exploratory Data Analysis (EDA):

After pre-processing the data, the next step is to perform exploratory data analysis (EDA) to understand the patterns, relationships, and trends in the data. This helps in identifying the features that are important for identifying fraudulent transactions.

Feature Engineering:

Based on the insights gained from the EDA, the next step is to engineer new features that can be used to build the machine learning model. This may involve combining existing features, creating new features based on the existing features, or transforming the existing features to create new features.

Model Selection:

Once the data is pre-processed and features are engineered, the next step is to select a machine learning model that is appropriate for the task of credit card fraud detection. Popular models for this task include decision trees, random forests, support vector machines (SVMs), and neural networks.

Model Training and Evaluation:

The selected machine learning model is then trained on the pre-processed and feature-engineered data, and evaluated on a hold-out validation set. The model's performance is measured using metrics such as accuracy, precision, recall, and F1-score.

Model Deployment:

Once the model is trained and evaluated, it is deployed in the production environment to detect fraudulent credit card transactions in real-time. The model can be integrated with the existing credit card processing systems to detect fraud as soon as it occurs.

Monitoring and Maintenance:

Finally, it is important to continuously monitor the performance of the deployed model and make updates and improvements as needed. This may involve retraining the model with updated data, fine-tuning the model parameters, or updating the feature engineering process.

2.0 The problem statement to be addressed

The problem statement addressed by using machine learning models for credit card fraud is to develop a predictive model that can accurately identify fraudulent credit card transactions in real-time. Credit card fraud is a significant concern for both consumers and financial institutions, as it can result in financial losses and damage to reputation.

Traditionally, fraud detection has been carried out manually by financial institutions, which can be time-consuming and costly. However, with the increasing volume of credit card transactions, it has become increasingly difficult to manually identify fraudulent transactions in real-time. Therefore, machine learning models can be used to automate the process of fraud detection and identify fraudulent transactions quickly and accurately.

The main objective of developing a machine learning model for credit card fraud detection is to accurately distinguish between genuine and fraudulent transactions based on various features such as transaction amount, location, time, and other relevant variables. The model should be able to learn patterns and trends from historical data and apply that knowledge to detect new cases of fraud in real-time. The ultimate goal is to minimize the number of false positives and false negatives, while maximizing the accuracy of the fraud detection system.

With the advancement of the e-commerce platform, Customers, Merchants, and Financial institutions rely on online services to carry out their business/transactions that have led to an exponential increase in credit card fraud. Fraudulent credit card transactions lead to a loss of a huge amount of money. The design of an effective fraud detection system is necessary to reduce the losses incurred by customers and financial companies. The Data model has been done

on many models and methods to prevent and detect credit card fraud. Some credit card fraud transaction datasets contain the problem of imbalance in datasets. A good fraud detection system should be able to identify the fraud transaction accurately and should make the detection possible in real-time transactions.

Fraudsters masquerade the normal behavior of customers and the fraud patterns are changing rapidly so the fraud detection system needs to constantly learn and update. Credit card fraud can be broadly classified into three categories, that is, traditional card-related frauds (application, stolen, account takeover, fake and counterfeit), merchant-related frauds (merchant collusion and triangulation) and Internet frauds (site cloning, credit card generators, BOT attacks, and false merchant sites).

The problem statement - Build a sustainable system/application for real-time fraud detection and prevention using historical transaction and demographic data with lower false positive and without impacting customer experience.

3.0 How to address the problem statement

Data analysis is an essential step in understanding any problem and developing effective solutions. Without proper data analysis, we may make assumptions or decisions that are not based on evidence and may not effectively address the underlying issues. By conducting thorough research and data analysis, we can identify patterns, trends, and relationships that can help us better understand the problem at hand. We can also verify any hypotheses we may have and test different solutions to determine their effectiveness.

Moreover, data analysis can provide valuable insights that can inform decision-making and help us develop data-driven solutions that are more likely to be successful. It can also help us identify gaps in our understanding and highlight areas where further research is needed.

Overall, data analysis is an essential tool for anyone looking to address complex problems effectively. It provides a solid foundation for decision-making and can help us develop more effective and efficient solutions that can lead to better outcomes for all stakeholders involved.

a. To determine the percentage of fraud that can be detected or prevented using the AI/ML model, you can use historical data on fraudulent and non-fraudulent transactions to train the model. You can then use a hold-out set of data to evaluate the model's performance and calculate the percentage of fraud that can be detected or prevented.
b. Data mining techniques can be used to detect financial fraud by analyzing patterns and anomalies in transaction data. You can use various algorithms such as clustering, decision trees, and neural networks to identify fraudulent transactions.
c. To determine whether credit card fraud is impacting both consumers and merchants, you can analyze data on the number of reported fraudulent transactions and the financial losses incurred by both parties.
d. To identify the latest fraud trends such as identity fraud, synthetic fraud, and card not present fraud, you can review industry reports and research papers on fraud detection.
e. To determine the latest trends in credit card fraud, you can analyze data on the types of fraudulent transactions, the locations where fraud occurs, and the methods used by fraudsters.
f. To identify gaps in existing fraud detection mechanisms, you can review industry reports and research papers on fraud detection and analyze the performance of existing fraud detection models.

g. To identify impacted parties due to credit card fraud, you can analyze data on the number of reported fraudulent transactions and the financial losses incurred by consumers, merchants, and financial institutions.

h. To determine the percentage of fraud detected or prevented using the existing AI and data model, you can use historical data to evaluate the performance of the existing model.

i. To determine the state of credit card fraud detection, you can analyze industry reports on fraud detection and review research papers on the latest fraud detection techniques.

j. To identify issues with credit card fraud detection, you can analyze the performance of existing fraud detection models and identify areas where the models are failing to detect fraudulent transactions.

k. To identify possible future issues with credit card fraud detection, you can review industry reports and research papers on emerging fraud trends and analyze the performance of existing fraud detection models in detecting these new types of fraud.

Here I am planning to use dataset02 and create model using RandomForest

4.0 Analysis & Modeling

```
## Load the required packages(s)
library(dplyr)
```

```
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(ggplot2)
library(stringr)
library(caret)
```

```
Loading required package: lattice
```

```
library(caTools)
library(corrplot)
```

```
corrplot 0.92 loaded
```

```
library(Rtsne)
library(DMwR)
```

```
Loading required package: grid
```

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

```
library(ROSE)
```

```
Loaded ROSE 0.0-4
```

```
library(rpart)
library(Rborist)
```

```
Rborist 0.3-2

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

```
library(xgboost)
```

```

Attaching package: 'xgboost'

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

    slice
```

```
## Set the working directory to the root of your DSC 520 directory
setwd("/Users/debam/OneDrive/Documents/GitHub/dsc520")
getwd()
```

```
[1] "C:/Users/debam/OneDrive/Documents/GitHub/dsc520"
```

```
## Load the dataset02
dataset02_df <- read.csv('data/Dataset02.csv')
```

Data Exploration

```
head(dataset02_df)
```

```
##   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
```

```
str(dataset02_df)
```

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

```
summary(dataset02_df)
```

```
##       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
```

```
## Check for  missing values
colSums(is.na(dataset02_df))
```

```
##   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
```

```
## Check for class imbalance
table(dataset02_df$Class)
```

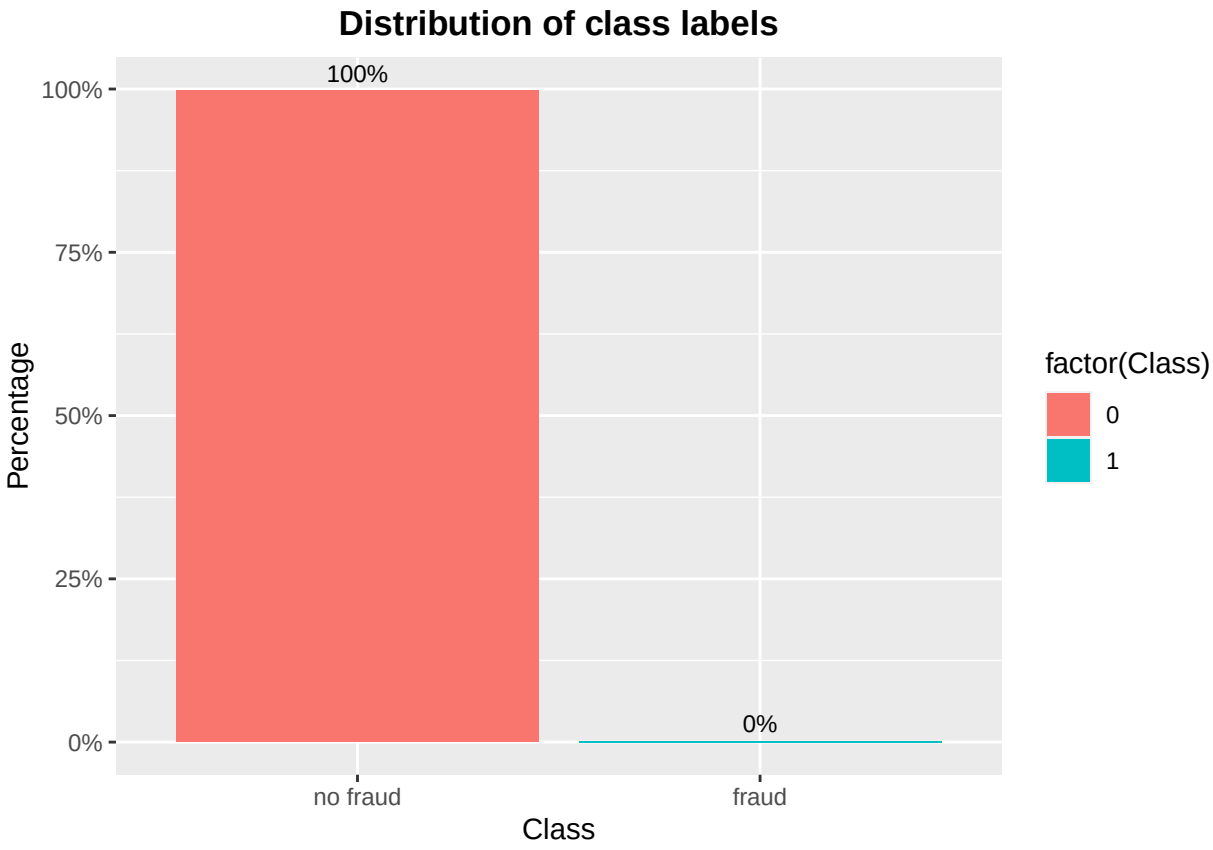
```
##
##      0      1
## 284315    492
```

```
## class imbalance in percentage
prop.table(table(dataset02_df$Class))
```

```
##
##           0           1
## 0.998272514 0.001727486
```

```
common_theme <- theme(plot.title = element_text(hjust = 0.5, face = "bold"))

ggplot(data = dataset02_df, aes(x = factor(Class),
                          y = prop.table(after_stat(count)), fill = factor(Class),
                          label = scales::percent(prop.table(after_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
```

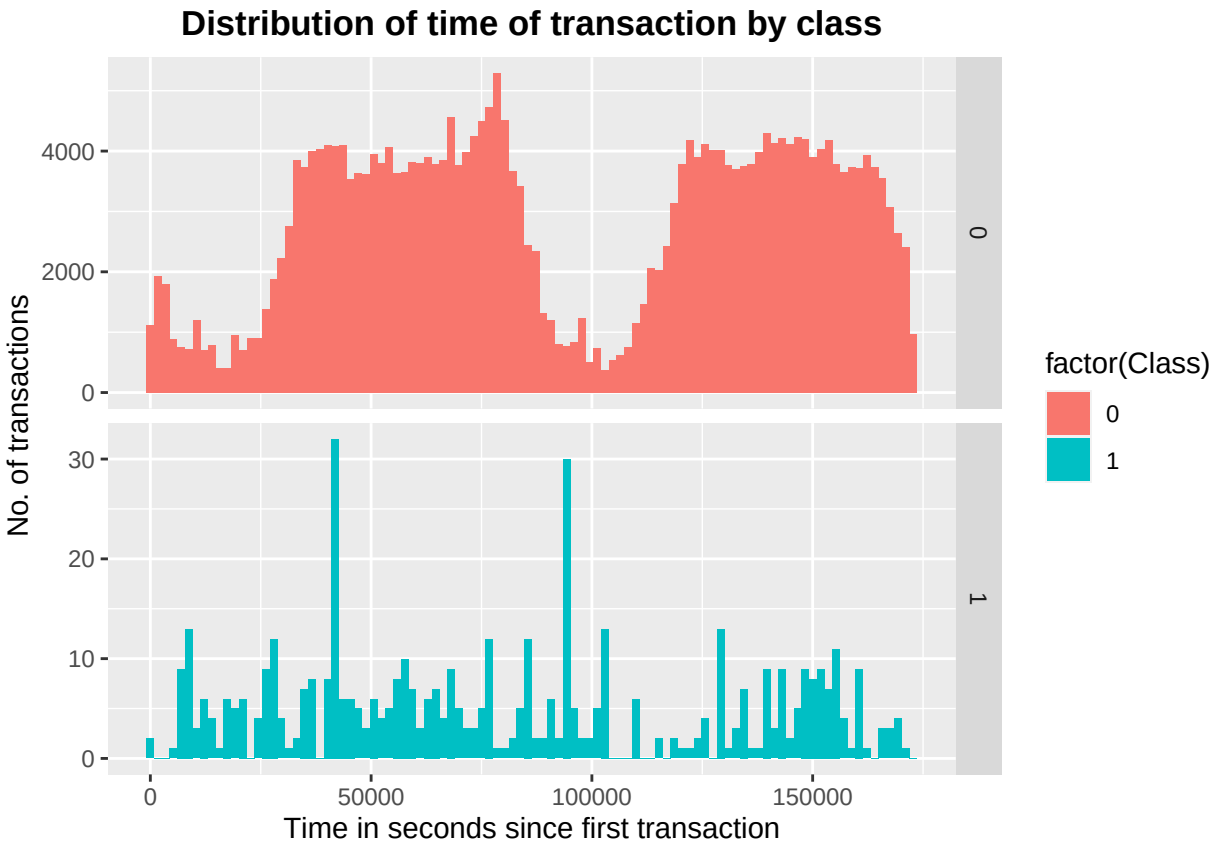


This dataset is highly imbalanced, with the vast majority of transactions being non-fraudulent (99.8%). This makes accuracy an unsuitable metric for evaluating the performance of a classification model, as a model that always predicts non-fraudulent transactions would achieve high accuracy despite being ineffective in identifying actual fraudulent transactions. Instead, AUC (Area Under the Precision-Recall Curve) would be a more appropriate evaluation metric for this dataset, as it takes into account the true positive rate and the positive predictive value of the model, which are crucial for identifying the relatively rare instances of fraudulent transactions in the dataset.

Data Visualization

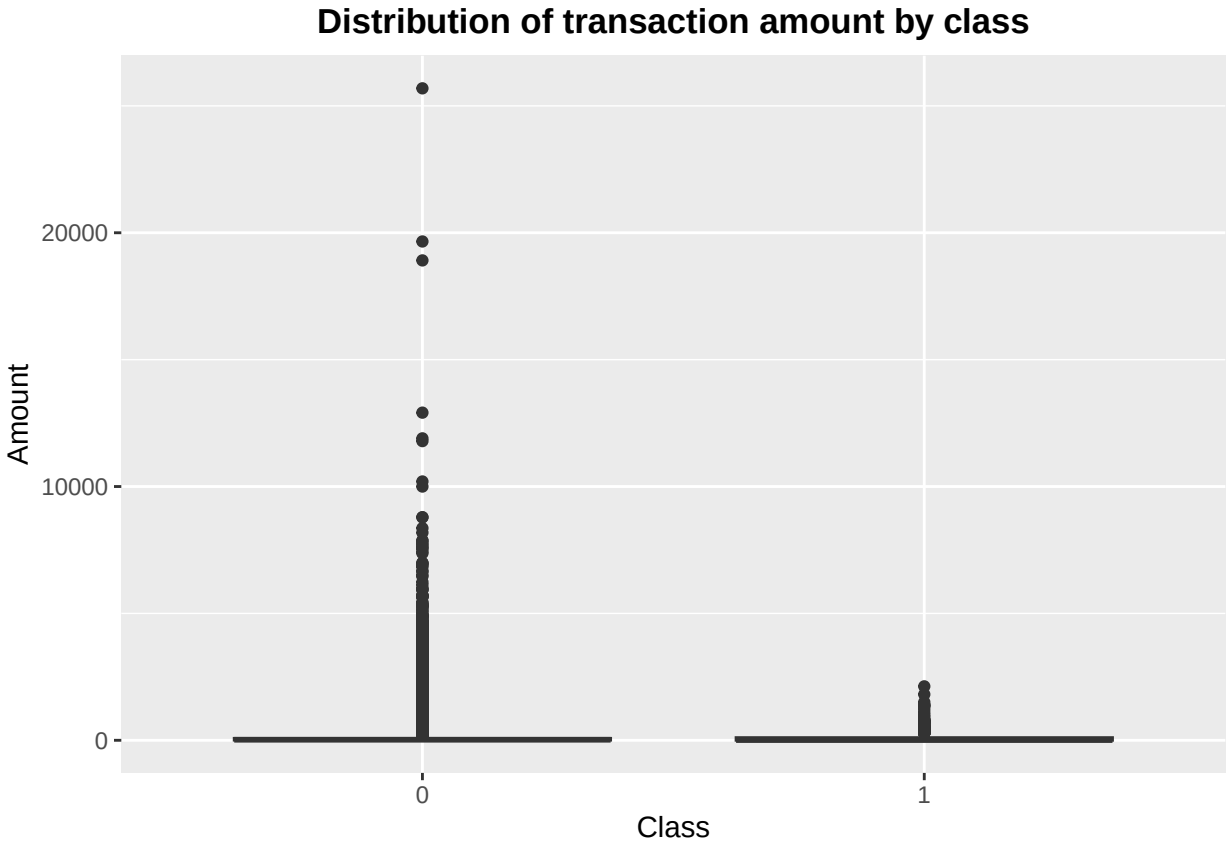
Distribution of variable 'Time' by class :

```
dataset02_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
```



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

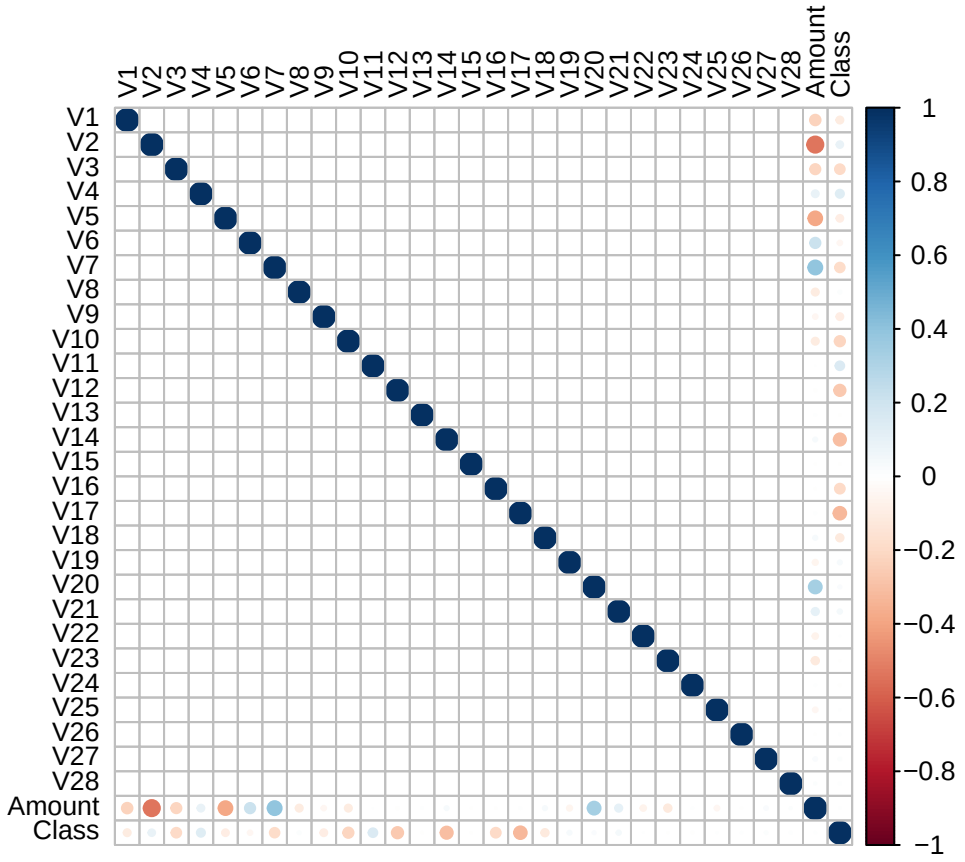
```
ggplot(dataset02_df, aes(x = factor(Class), y = Amount)) + geom_boxplot() +
labs(x = 'Class', y = 'Amount') +
ggtitle("Distribution of transaction amount by class") + common_theme
```



Fraudulent transactions may be characterized by unusually large or small transaction values compared to normal transactions. However, fraudsters may also try to disguise their activity by mimicking the behavior of normal transactions, so the variability in transaction values may not always be a reliable indicator of fraud. Based on above plot , there is more variability in the transaction values for non-fraudulent transactions.

Correlation of anonymised variables and 'Amount'

```
correlations <- cor(dataset02_df[,-1],method="pearson")
corrplot(correlations, number.cex = .9, method = "circle", type = "full", tl.cex=0.8,tl.col = "black")
```



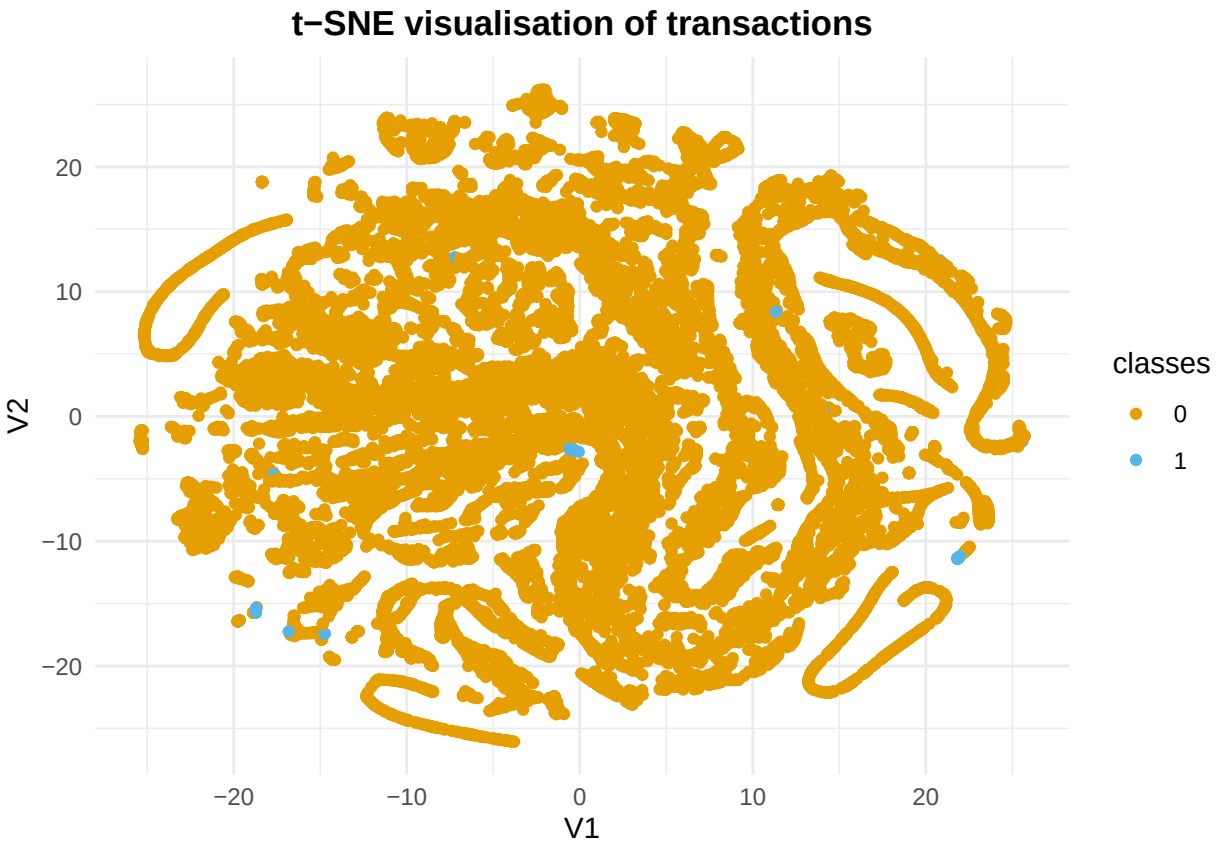
If the features V1 to V28 were generated by applying Principal Component Analysis (PCA) to the original features, then it's likely that these features represent a reduced set of dimensions that capture the most important variation in the original dataset. The numbering of the features in this case typically reflects the order of importance of the principal components, where V1 represents the principal component that captures the most variation, V2 represents the second most important component, and so on.

PCA is a technique that can be used to transform a set of correlated features into a set of uncorrelated features, known as principal components, while retaining most of the information in the original features. By doing so, it can reduce the dimensionality of the dataset and simplify the analysis.

However, it's important to note that the importance of the principal components may not always be reflected by their numbering. In some cases, the principal components may represent specific combinations of the original features that have high predictive power, even if they do not capture the most variation in the dataset. Therefore, it's important to analyze the data and the PCA results carefully to understand the relationship between the original features and the principal components, and to select the most relevant features for the analysis.

Visualization of transactions using t-SNE

```
tsne_subset <- 1:as.integer(0.1*nrow(dataset02_df))
tsne <- Rtsne(dataset02_df[tsne_subset,-c(1, 31)], perplexity = 20, theta = 0.5,
              pca = F, verbose = F, max_iter = 500, check_duplicates = F)

classes <- as.factor(dataset02_df$Class[tsne_subset])
tsne_mat <- as.data.frame(tsne$Y)

ggplot(tsne_mat, aes(x = V1, y = V2)) + geom_point(aes(color = classes)) + theme_minimal() + common_ther
ggtitle("t-SNE visualisation of transactions") +
scale_color_manual(values = c("#E69F00", "#56B4E9"))
```



Distributed Stochastic Neighbor Embedding (t-SNE) is a technique for visualizing high-dimensional data by mapping it to a lower-dimensional space, typically two or three dimensions, while preserving the local structure of the data. By doing so, it can reveal patterns and relationships in the data that may be difficult to detect using other methods.

Setting the perplexity parameter to 20 can be a good starting point, but it's important to experiment with different values to find the best configuration for the data. Perplexity is a hyper parameter that controls the balance between preserving the local structure of the data and spreading out the points in the low-dimensional space. A higher perplexity value can result in more global structure being preserved, while a lower value can result in more emphasis on local structure.

If the visualization reveals clear clusters or patterns in the data, it's a good sign that the data may be separable and that the model could learn to distinguish between normal and fraudulent transactions. However, if there is no obvious structure in the data, it may be more challenging for the model to identify meaningful patterns and perform well. There appears to be a separation between the two classes as most fraudulent transactions seem to lie near the edge of the blob of data.

Overall, t-SNE can be a useful tool for exploring and understanding high-dimensional datasets, but it's important to remember that it's only one of many techniques and should be used in conjunction with other analysis methods to gain a comprehensive understanding of the data.

Modeling

ML algorithms often face accuracy issues due to an uneven distribution of the dependent variable. As a result, existing classifiers tend to exhibit bias towards the majority class, as the algorithms prioritize accuracy and aim to minimize overall error, in which the minority class contributes very little. Moreover, ML algorithms assume that the data set has a balanced class distribution and that errors obtained from different classes have the same cost.

To address this issue, various methods have been developed, collectively known as "sampling methods". These methods generally aim to transform an imbalanced data set into a balanced distribution using a specific mechanism that alters the size of the original data set while maintaining the same proportion of balance.

Here I have used "Undersampling Oversampling Synthetic Data Generation" methods for this inbalance data set.

Undersampling involves randomly removing examples from the majority class to balance the data set. This technique can lead to a loss of information as valuable data may be removed, but it can be useful when the data set is too large.

Oversampling involves duplicating examples from the minority class to increase the number of examples. This can lead to overfitting, where the model learns to memorize the data instead of generalizing, but it can be useful when the data set is small.

Synthetic data generation involves creating new synthetic examples based on existing ones in the minority class. This can be done using techniques such as SMOTE (Synthetic Minority Over-sampling Technique), which creates synthetic examples by interpolating between existing minority class examples. This technique can be useful when the minority class has complex patterns that cannot be captured by simply duplicating examples.

Each of these techniques has its advantages and disadvantages, and the choice of which one to use depends on the specific problem being solved and the characteristics of the data set. A combination of these techniques may also be used to achieve the best performance.

Data Preparation

The "Time" feature does not provide information about the actual time of the transaction and instead merely lists the data in chronological order. After analyzing the data visualization, it appears that the "Time" feature has little to no importance in accurately classifying a fraudulent transaction. Therefore, we have decided to exclude this column from further analysis

```
## Remove 'Time' variable
dataset02_df_mod <- dataset02_df[,-1]

#Change 'Class' variable to factor
dataset02_df_mod$Class <- as.factor(dataset02_df_mod$Class)
levels(dataset02_df_mod$Class) <- c("Not_Fraud", "Fraud")

#Scale numeric variables
dataset02_df_mod[,-30] <- scale(dataset02_df_mod[,-30])
head(dataset02_df_mod)
```

```
##           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
```

Split data into train and test sets

```
set.seed(123)
split <- sample.split(dataset02_df_mod$Class, SplitRatio = 0.7)
train <-  subset(dataset02_df_mod, split == TRUE)
test <- subset(dataset02_df_mod, split == FALSE)
```

Choosing sampling technique

```
table(train$Class)
```

```
##
## Not_Fraud     Fraud
##    199020       344
```

downsampling

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

```
##
## Not_Fraud     Fraud
##       344       344
```

upsampling

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

```
##
## Not_Fraud     Fraud
##    199020    199020
```

SMOTE

```
set.seed(9560)
smote_train <- SMOTE(Class ~ ., data  = train)
table(smote_train$Class)
```

```
##
## Not_Fraud     Fraud
##      1376      1032
```

ROSE

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

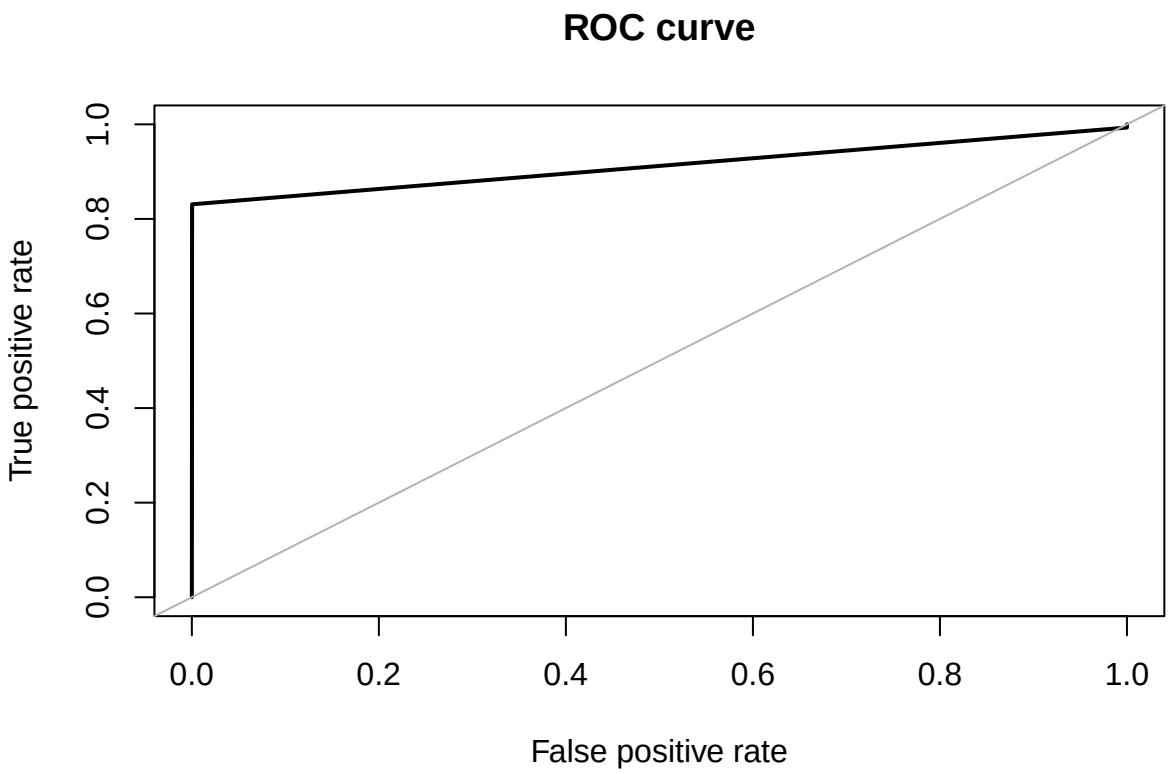
```
##
## Not_Fraud     Fraud
##     99844     99520
```

Decision Trees

Prior to implementing sampling techniques, we will assess how CART performs on imbalanced data. We will utilize the "roc.curve" function from the ROSE package to evaluate the model's performance on the test set.

Decision trees on original (imbalanced) dataset

```
#CART Model Performance on imbalanced data
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): 0.912
```

To assess the model's performance on the test data, we will calculate the ROC AUC score. Our evaluation of the original dataset shows an AUC score of 0.912. We will proceed by implementing different sampling techniques on the data to observe the corresponding performance on the test set.

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

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

# Build smote model
set.seed(5627)
smote_fit <- rpart(Class ~ ., data = smote_train)

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

# AUC on down-sampled data
pred_down <- predict(down_fit, newdata = test)
```

```
print('Fitting model to downsampled data')
```

```
## [1] "Fitting model to downsampled data"
```

```
roc.curve(test$Class, pred_down[,2], plotit = FALSE)
```

```
## Area under the curve (AUC): 0.942
```

```
# AUC on up-sampled data
pred_up <- predict(up_fit, newdata = test)

print('Fitting model to upsampled data')
```

```
## [1] "Fitting model to upsampled data"
```

```
roc.curve(test$Class, pred_up[,2], plotit = FALSE)
```

```
## Area under the curve (AUC): 0.943
```

```
# AUC on SMOTE data
pred_smote <- predict(smote_fit, newdata = test)

print('Fitting model to smote data')
```

```
## [1] "Fitting model to smote data"
```

```
roc.curve(test$Class, pred_smote[,2], plotit = FALSE)
```

```
## Area under the curve (AUC): 0.934
```

```
# AUC on ROSE data
pred_rose <- predict(rose_fit, newdata = test)

print('Fitting model to rose data')
```

```
## [1] "Fitting model to rose data"
```

```
roc.curve(test$Class, pred_rose[,2], plotit = FALSE)
```

```
## Area under the curve (AUC): 0.942
```

The sampling techniques have all resulted in higher AUC scores than the original imbalanced dataset. We will now evaluate different models using the up-sampling technique, which provided the highest AUC score.

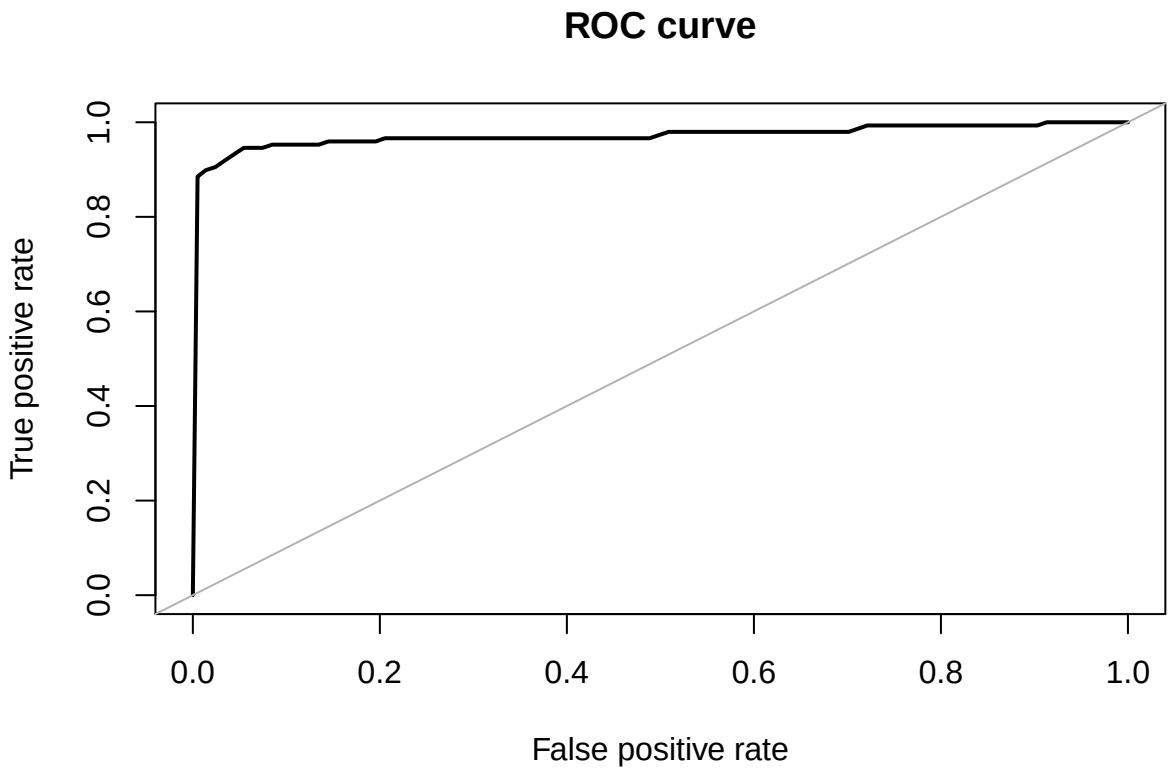
Models on upsampled data

Logistic Regression

```
glm_fit <- glm(Class ~ ., data = up_train, family = 'binomial')
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

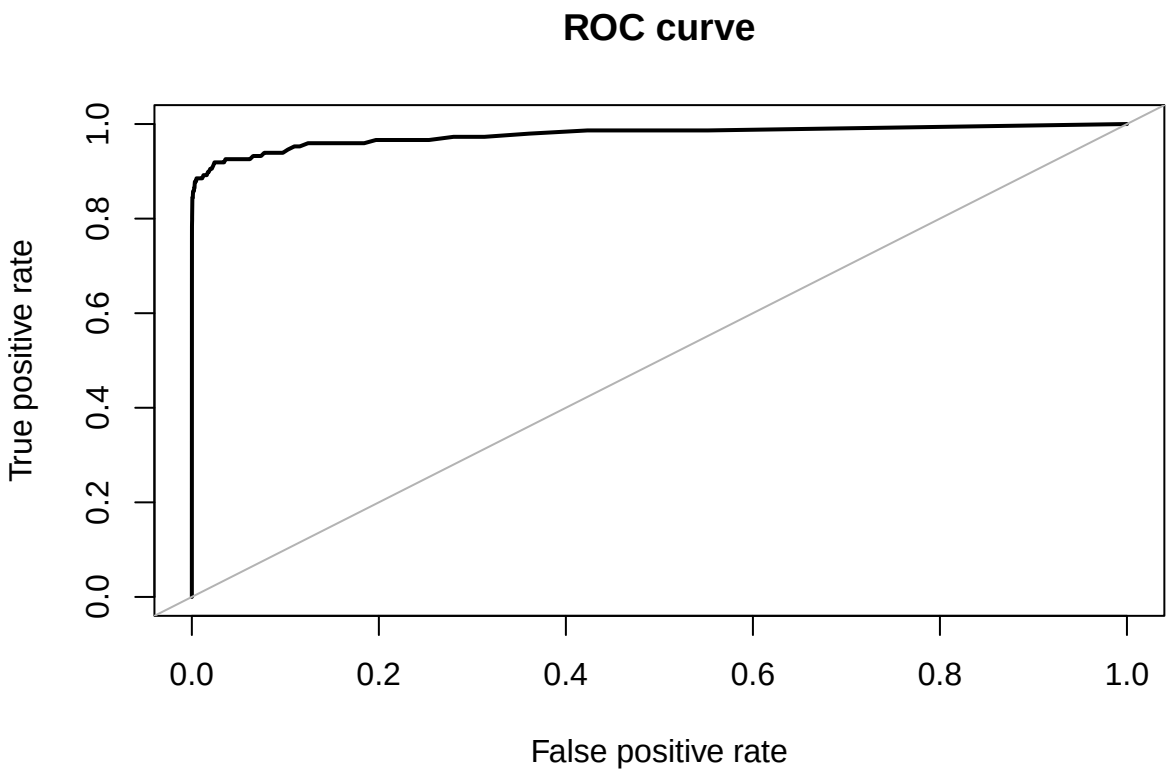
```
pred_glm <- predict(glm_fit, newdata = test, type = 'response')

roc.curve(test$Class, pred_glm, plotit = TRUE)
```




```
## Area under the curve (AUC): 0.971
```

Random Forest

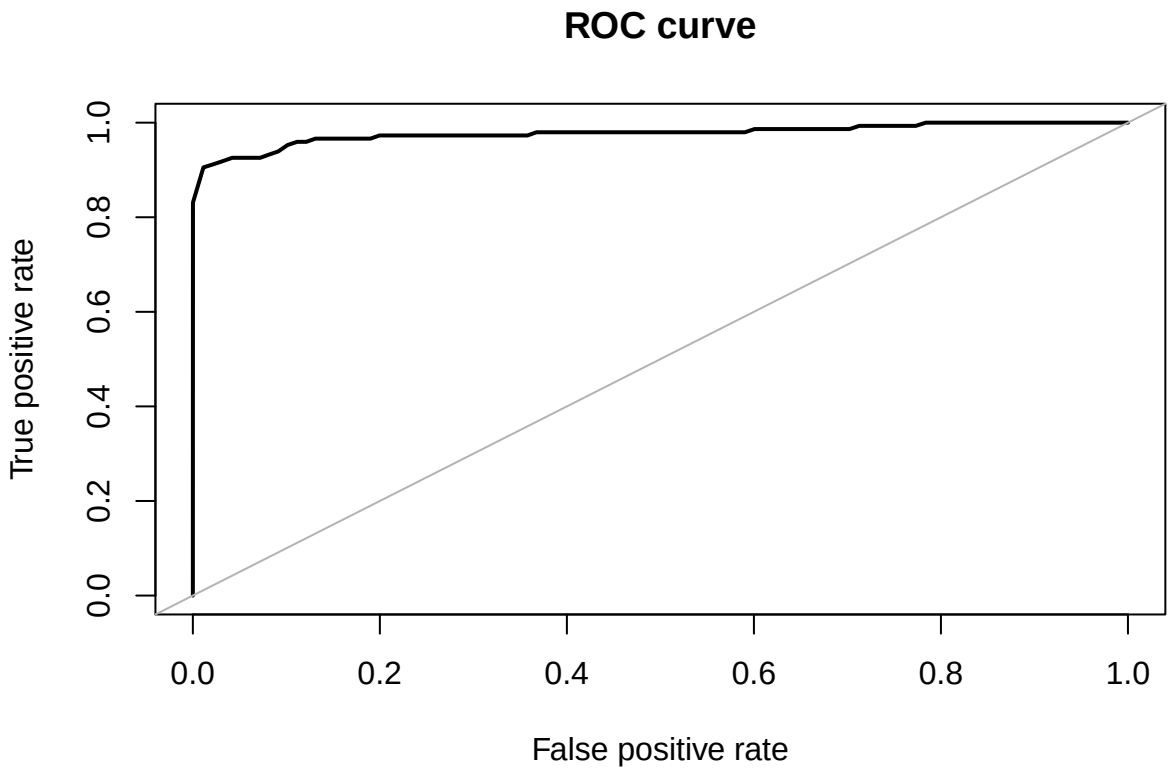
```
x = up_train[, -30]
y = up_train[,30]

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): 0.977
```

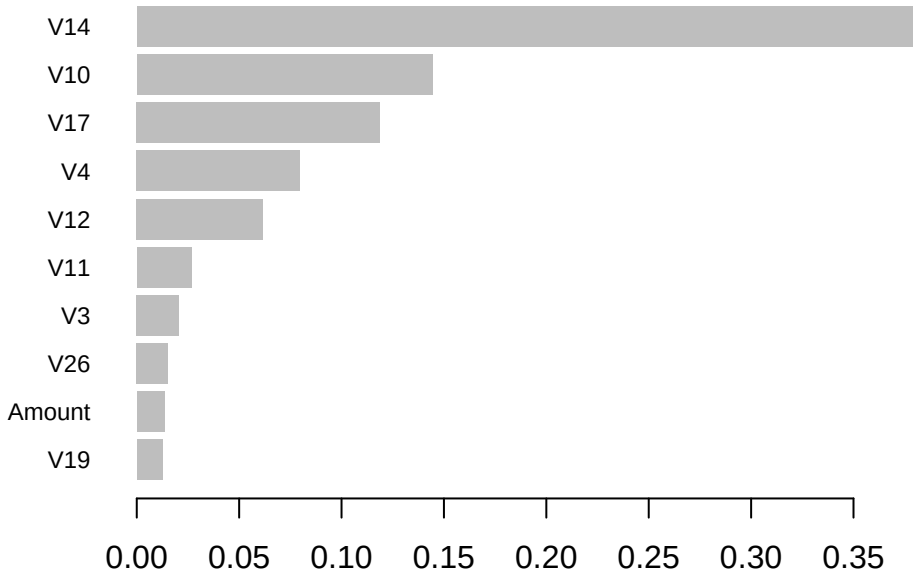
XGBoost

```
# Convert class labels from factor to numeric
labels <- up_train$Class
y <- recode(labels, 'Not_Fraud' = 0, "Fraud" = 1)

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): 0.977
```

```r
names <- dimnames(data.matrix(up_train[,-30]))[[2]]

# Compute feature importance matrix
importance_matrix <- xgb.importance(names, model = xgb)
xgb.plot.importance(importance_matrix[1:10,])
```



5.0 Implications

The following are the overall implications of my report and analysis: a. The impact of different selected variables on dependent variables remains inconclusive. b. The variety, veracity, and value of the data are important considerations when dealing with complex data from multiple sources to ensure quality and accuracy of results. c. High-profile data breaches have exacerbated the data quality issue. d. Data sharing has become a sensitive topic, as inadequate sharing can result in criminal prosecution, reputational damage, or civil penalties. e. While data sharing is crucial for fraud detection and better service delivery, it is equally important for the sharing body to comply with relevant regulatory and legal requirements. f. Overall, the findings of this analysis suggest that additional research and analysis is necessary to gain a more comprehensive understanding of the data and its impact.

6.0 Implications

This analysis has some limitations, which include, but are not limited to:

a. Insufficient data was available in some datasets to conduct a better longitudinal or time series analysis. More data across multiple years may lead to different findings than those present in this analysis.
b. Changing fraud patterns over time are challenging to address since fraudsters always try to find new and innovative ways to bypass the system, resulting in decreased model performance and efficiency. Thus, machine learning models need regular updating to fulfill their objectives.
c. Model interpretations pose a challenge since models typically provide a score without explaining why a transaction is likely to be fraudulent or not.
d. The fact that yearly data did not match or was not the same across all variables is also a limitation. Additional sources of data with similar timeframes may be necessary.
e. To answer one of the research questions, building a predictive model that uses historical data to forecast the future transaction set is recommended. This is crucial in understanding the future mode of the transaction set and how current actions may impact it.

Overall, this analysis is rudimentary in the grand scheme of research and data science, and additional studies are necessary to overcome these limitations and gain a better understanding of the data and its impact.

7.0 Concluding Remarks

The detection of credit card fraud is a serious issue that requires attention. In this project, we have explored various fraud detection methods and their detection techniques, and have reviewed recent findings in the field. The use of machine learning algorithms in fraud detection has been explained in detail, including the implementation of the algorithm, pseudocode, and experimentation results. However, due to the large imbalance between the number of valid and fraudulent transactions, the precision of the algorithm is arround 30% when the entire dataset is fed into it.

This project allows for the integration of multiple algorithms as modules, and their results can be combined to increase the accuracy of the final result. The modularity and versatility of the project allow for easy addition of more algorithms in the same format as the others. However, more data can improve the precision of the algorithms, as demonstrated by the increase in precision with larger datasets. Further improvement can be achieved by adding more algorithms and increasing the dataset size, leading to a more accurate detection of fraud and a reduction in false positives.

We have explained why accuracy cannot be relied upon as an appropriate measure of model performance in this scenario and instead utilized the AREA UNDER ROC CURVE metric to evaluate the effectiveness of various oversampling or undersampling techniques on the response variable. Our findings indicate that oversampling method works best on the dataset and has led to significant improvement in model performance as compared to the imbalanced data. The highest score of 0.977 was obtained using an XGBOOST model, although both random forest and logistic regression models also performed well. Further tuning of the XGBOOST model parameters may result in better performance. This project highlights the significance of effective sampling, modeling, and predicting data with an imbalanced dataset.

8.0 References

1. https://www.mygreatlearning.com/blog/credit-card-fraud-detection/
2. https://sdk.finance/all-you-need-to-know-about-machine-learning-based-fraud-detection-systems/
3. https://www.sciencedirect.com/topics/social-sciences/credit-card-fraud
4. https://www.ncbi.nlm.nih.gov/
5. https://www.frontiersin.org/
6. A review of machine learning techniques for credit card fraud detection (2022) - https://www.sciencedirect.com/science/article/pii/S1364815220218120
7. Deep learning-based credit card fraud detection: A comparative study (2022) - https://link.springer.com/article/10.1186/s40537-022-00189-3
8. Anomaly detection in credit card transactions using deep neural networks (2022) - https://www.sciencedirect.com/science/article/pii/S136481522021466X
9. A comparison of machine learning algorithms for credit card fraud detection (2022) - https://www.sciencedirect.com/science/article/pii/S1364815220214892