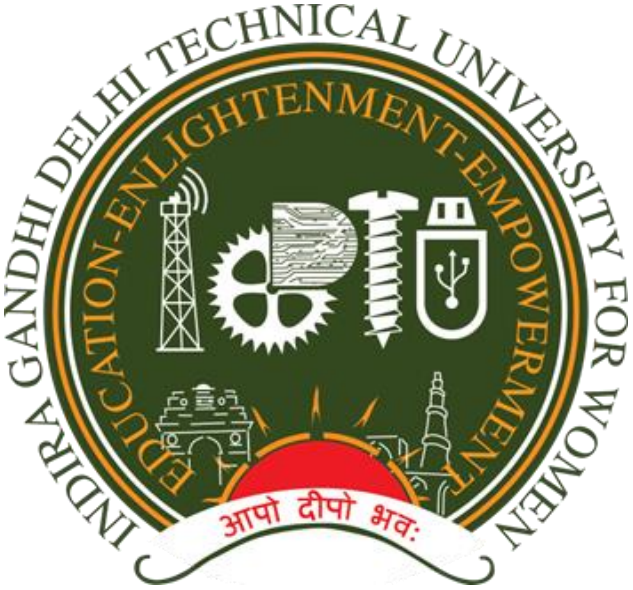
INDIRA GANDHI DELHI TECHNICAL UNIVERSITY FOR WOMEN



IT WORKSHOP (BAI 108)

SUBMITTED BY:

GUNGUN SINGH (02201192022)

HEMLATA (02301192022)

ISHA SWAMI (02401192022)

SUBMITTED TO:

MR. SANTANOO

CREDIT CARD FRAUD DETECTION

CONTENTS

- ABSTRACT
- INTRODUCTION
- PROBLEM DEFINITION
- PROBLEM OBJECTIVE
- DATASET DESCRIPTION
- RESEARCH METHODOLOGY
- IMPLEMENTATION
- AREAS OF IMPROVEMENT
- CONCLUSION
- REFERENCES

ABSTRACT

This project focuses on developing a credit card fraud detection system using machine learning techniques. The objective is to analyse a dataset of anonymized credit card transactions and identify fraudulent transactions. Various machine learning algorithms, including logistic regression, decision trees and gradient boosting, are employed and evaluated using performance metrics. Feature selection techniques and ensemble methods are explored to enhance the accuracy and robustness of the system. The results demonstrate the effectiveness of the developed system in accurately identifying credit card fraud, providing valuable insights for fraud prevention strategies.

INTRODUCTION

Credit card fraud is a significant issue faced by financial institutions and cardholders worldwide. The rise in online transactions and sophisticated fraudulent activities has made it essential to develop robust systems for detecting and preventing fraudulent transactions. Machine learning techniques have emerged as powerful tools for fraud detection, offering the potential to analyze large volumes of data and identify patterns indicative of fraudulent behaviour.

The objective of this project is to develop a credit card fraud detection system using machine learning algorithms. By leveraging a labelled dataset of anonymized credit card transactions, the project aims to build models that can accurately differentiate between legitimate and fraudulent transactions. The ultimate goal is to provide financial institutions and cardholders with an effective tool to mitigate the risks associated with credit card fraud.

In this report, we will discuss the problem definition, project objectives, research methodology, imlementation, problematic models, areas of improvement, and the conclusion of the credit card fraud detection project. We will explore various machine learning algorithms, evaluate their performance, and analyse the effectiveness of

ensemble methods in combining the outputs of multiple models. The report will highlight the importance of feature selection techniques in identifying relevant variables for fraud detection and provide insights into enhancing the overall performance of the system.

The findings and outcomes of this project will contribute to the field of credit card fraud detection by showcasing the potential of machine learning in accurately identifying fraudulent transactions. By providing a comprehensive analysis and evaluation of different models and techniques, this project aims to enhance fraud prevention strategies and assist financial institutions in safeguarding their customers' financial assets.

PROBLEM DEFINITION

Credit card fraud is a pervasive issue that can lead to significant financial losses for individuals and organisations. The objective of this project is to develop a credit card fraud detection system that can accurately identify fraudulent transactions and minimise the impact of fraud. By leveraging machine learning techniques and analysing historical credit card transaction data, the project aims to build a model that can effectively distinguish between legitimate and fraudulent transactions. The goal is to provide financial institutions with a reliable tool to detect and prevent fraudulent activities, thereby enhancing the security and trustworthiness of credit card transactions.

PROJECT OBJECTIVE

The main objective of this project is to develop a robust credit card fraud detection system that can effectively identify fraudulent transactions in real-time. The specific goals include:

- Creating a model that can accurately predict the likelihood of a transaction being fraudulent based on historical transaction data.
- Enhancing the efficiency of fraud detection to minimise false positives (legitimate transactions flagged as fraudulent) and false negatives (fraudulent transactions not detected).
- Providing insights and actionable information to financial institutions for proactive fraud prevention, allowing them to take appropriate measures to protect their customers.
- Improving the overall security and trustworthiness of credit card transactions, thereby increasing customer confidence and reducing financial losses due to fraud.

DATASET DESCRIPTION

The dataset used in this project has been extracted from Kaggle.

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 numeric input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we do not have access to the original features and more background information about the data. 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'.

TIME : Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset.

AMOUNT : The feature 'Amount' is the transaction Amount, this feature can be used for example-dependent cost-sensitive learning.

CLASS : Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

RESEARCH METHODOLOGY

1. DATA ACQUISITION

The dataset used in this project was extracted from Kaggle.The dataset had the 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.

2. DATA PREPARATION

In the data preparation phase of the project, several important steps were performed to ensure the dataset is ready for analysis and modelling. Here are the key tasks accomplished during the data preparation phase:

- Loading Libraries: The required libraries, such as ‘dplyr’, ‘stringr’, ‘caret’, ‘caTools’, ‘ggplot2’, ‘corrplot’, ’Rtsne’, ’ROSE’, ‘rpart’, ‘xgboost’ and ‘Rborist’ were loaded into the R environment. These libraries provide essential functions and tools for data manipulation, modelling, and evaluation.

- Dataset Overview: The dimensions of the "creditcard_data" dataset were examined using the dim() function, providing information about the number of rows and columns in the dataset. Additionally, the head() and tail() functions were used to display the first and last few rows of the dataset, allowing for a quick glance at the data structure and values.

- Several exploratory steps were taken to understand the dataset and its variables. The table() function was used to examine the distribution of the "Class" variable, providing a count of the number of transactions labelled as fraudulent and legitimate. Summary statistics, including mean, standard deviation, and range, were obtained using the summary(), var(), and sd() functions for the "Amount" variable. These exploratory analyses offer insights into the distribution and characteristics of the dataset.

- Train-Test Split: The dataset was split into training and testing datasets using the sample.split() function from the "caTools" library. This random split allocated 70% of the data to the training set and 30% to the testing set, ensuring separate datasets for model training and evaluation.

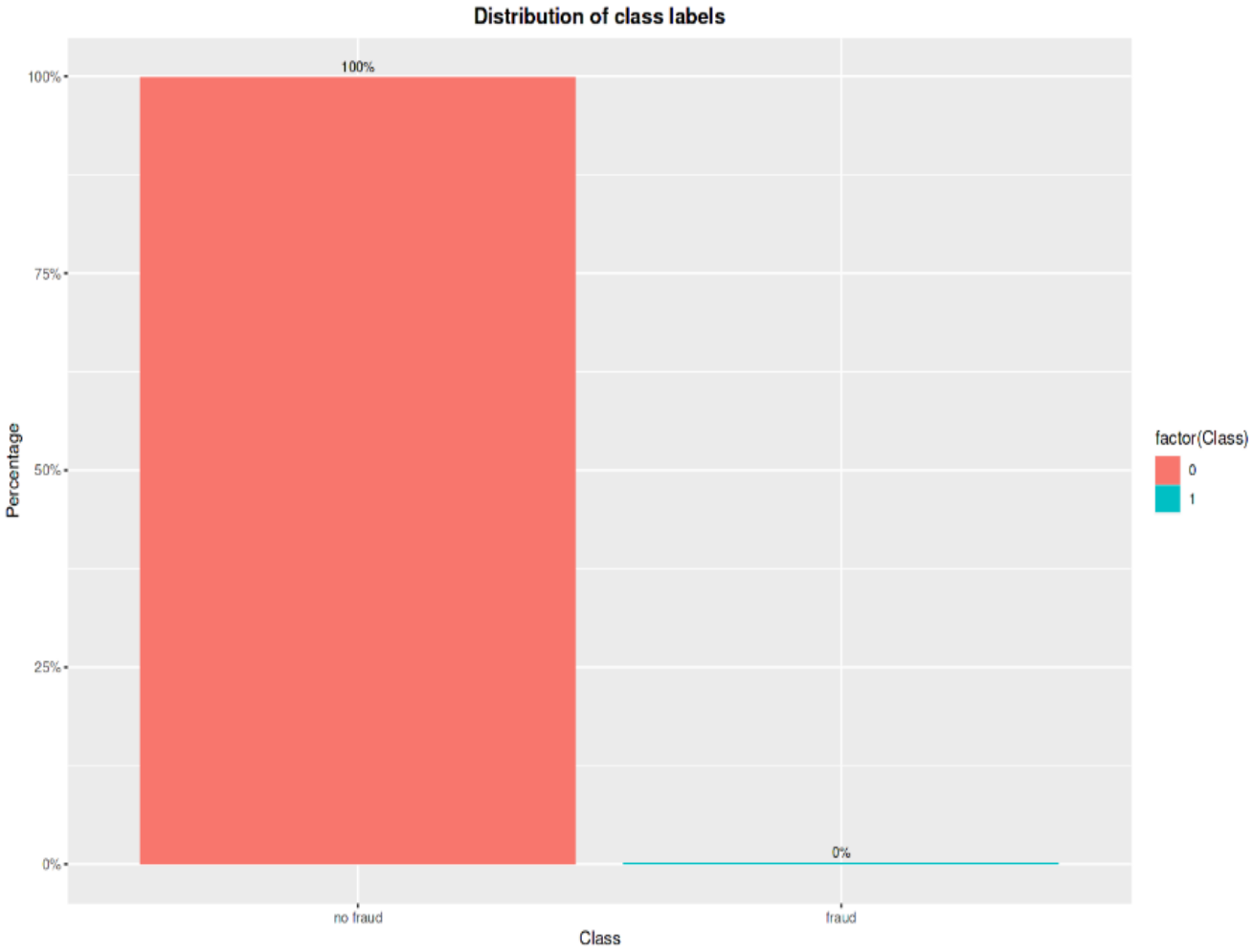
- Handling Unbalanced Dataset : In our data processing project, we encountered the challenge of unbalanced data, where the distribution of classes in our dataset was significantly skewed. To address this issue and ensure fair representation of all classes, we employed various techniques such as upsampling, downsampling, and ROSE (Random Over-Sampling Examples). Among these techniques, we chose to utilize upsampling to balance our training dataset.

  Unbalanced data refers to a situation where one class dominates the dataset while other classes are underrepresented. This can lead to biased predictions and hinder the performance of machine learning models. Upsampling, also known as oversampling, is a technique that involves increasing the number of instances in the minority class or classes to match the majority class. By augmenting the minority class data, upsampling aims to rectify the class imbalance and provide a more representative training dataset for machine learning algorithms.

  In our project, the minority class was significantly underrepresented compared to the majority class. To address this issue, we applied upsampling, which involved randomly replicating instances from the minority class. We continued this process until the number of instances in the minority class matched that of the majority class. Importantly, we ensured that the replication process preserved the distribution and characteristics of the original data.

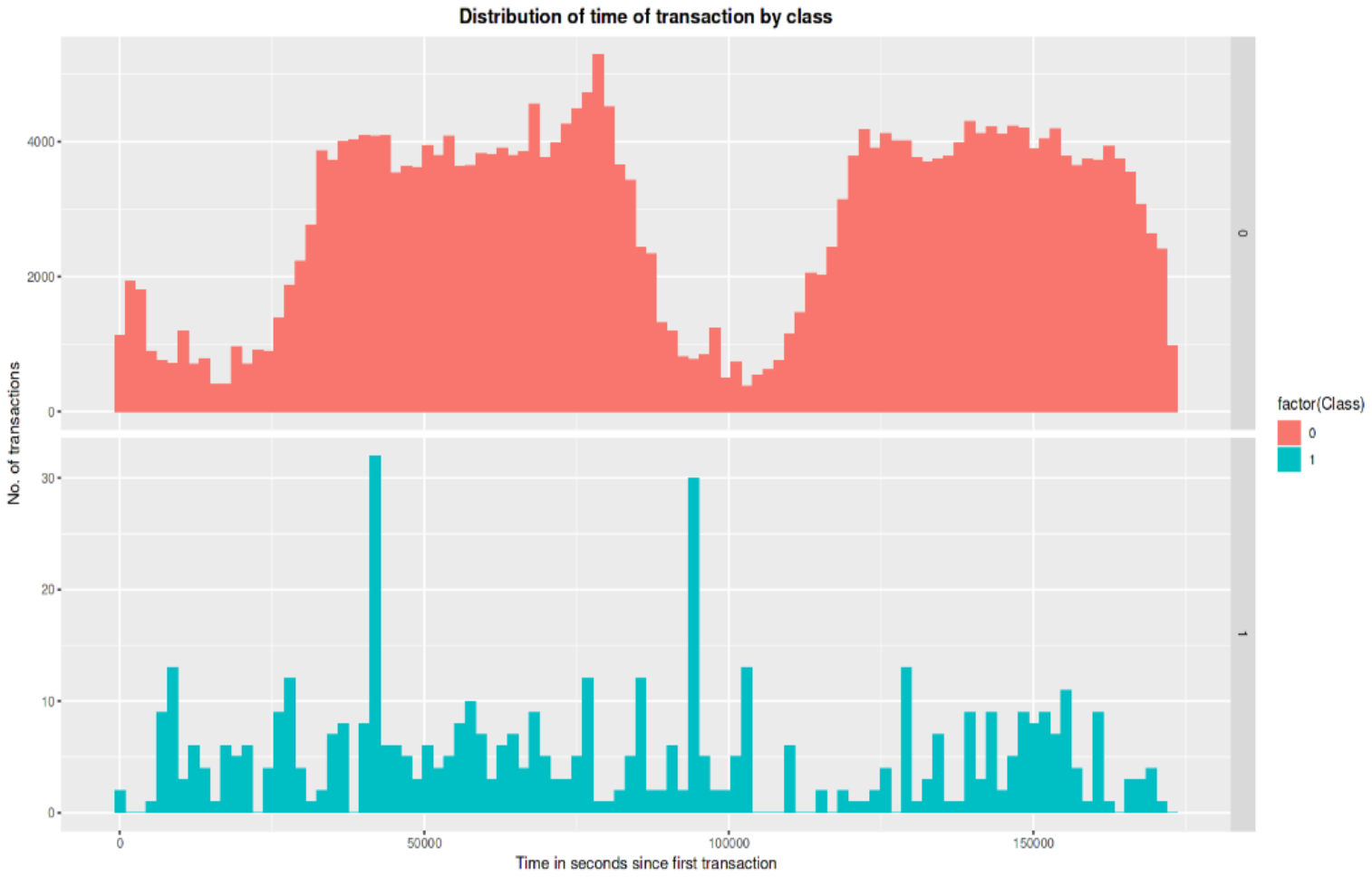
3. **DATA EXPLORATION AND DATA VISUALISATION**

1. DISTRIBUTION OF CLASS LABELS



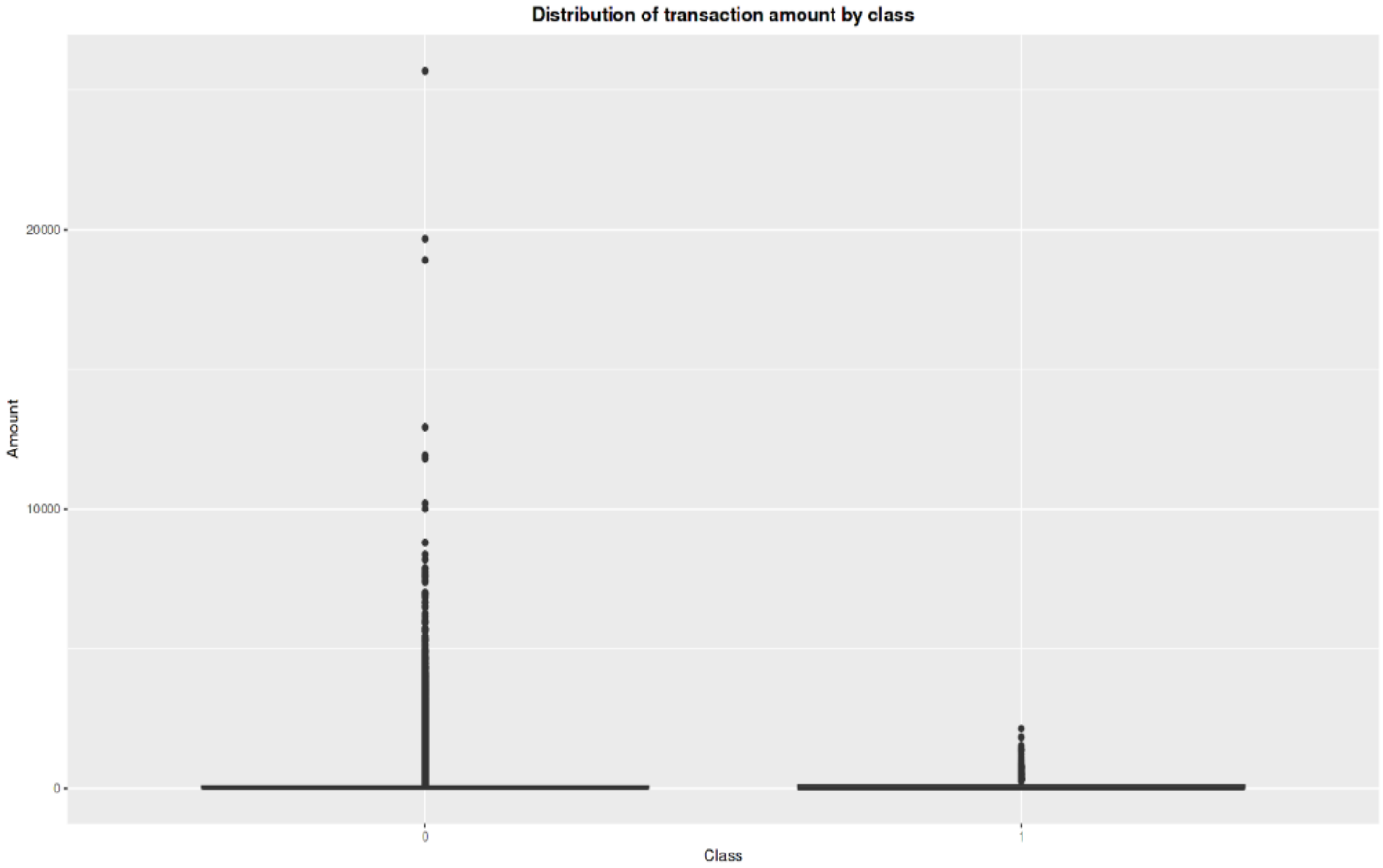
Through the above visualisation, we can conclude that the dataset is very imbalanced with 99.8% of cases being non-fraudulent transactions. A simple measure like accuracy is not appropriate here as even a classifier which labels all transactions as non-fraudulent will have over 99% accuracy.

2.Distribution of variable 'Time' by class



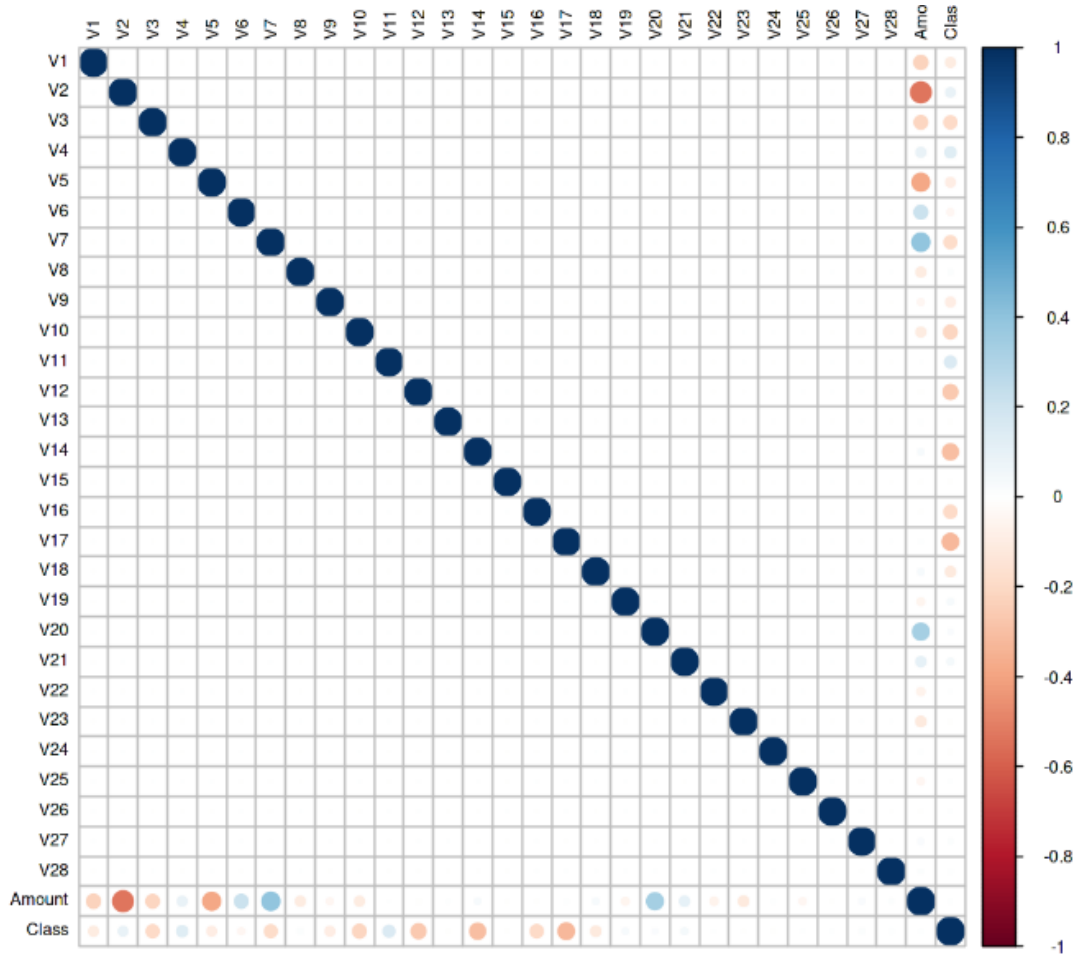
Through the above visualisation, we can conclude that the ‘Time’ feature looks similar across both types of transactions. The fraudulent transactions are more uniformly distributed, while normal transactions have a cyclical distribution.

3.Distribution of transaction amount by class



There is clearly a lot more variability in the transaction values for non-fraudulent transactions.

4.Correlation of anonymised variables and 'Amount'



Through the above visualisation, we can observe that most of the data features are not correlated. This is because before publishing, most of the features were presented to a Principal Component Analysis (PCA) algorithm. The features V1 to V28 are most probably the Principal Components resulted after propagating the real features through PCA.

Through basic Data Exploration and Data visualisation, we conclude that the dataset is very imbalanced with 99.8% of cases being non-fraudulent transactions. A simple measure like accuracy is not appropriate here as even a classifier which labels all transactions as non-fraudulent will have over 99% accuracy.

Thus, we have employed upsampling to balance our training dataset.

Thus, An appropriate measure of model performance here would be AUC(Area Under The Curve).

4. FITTING DIFFERENT ALGORITHMS ON DATA

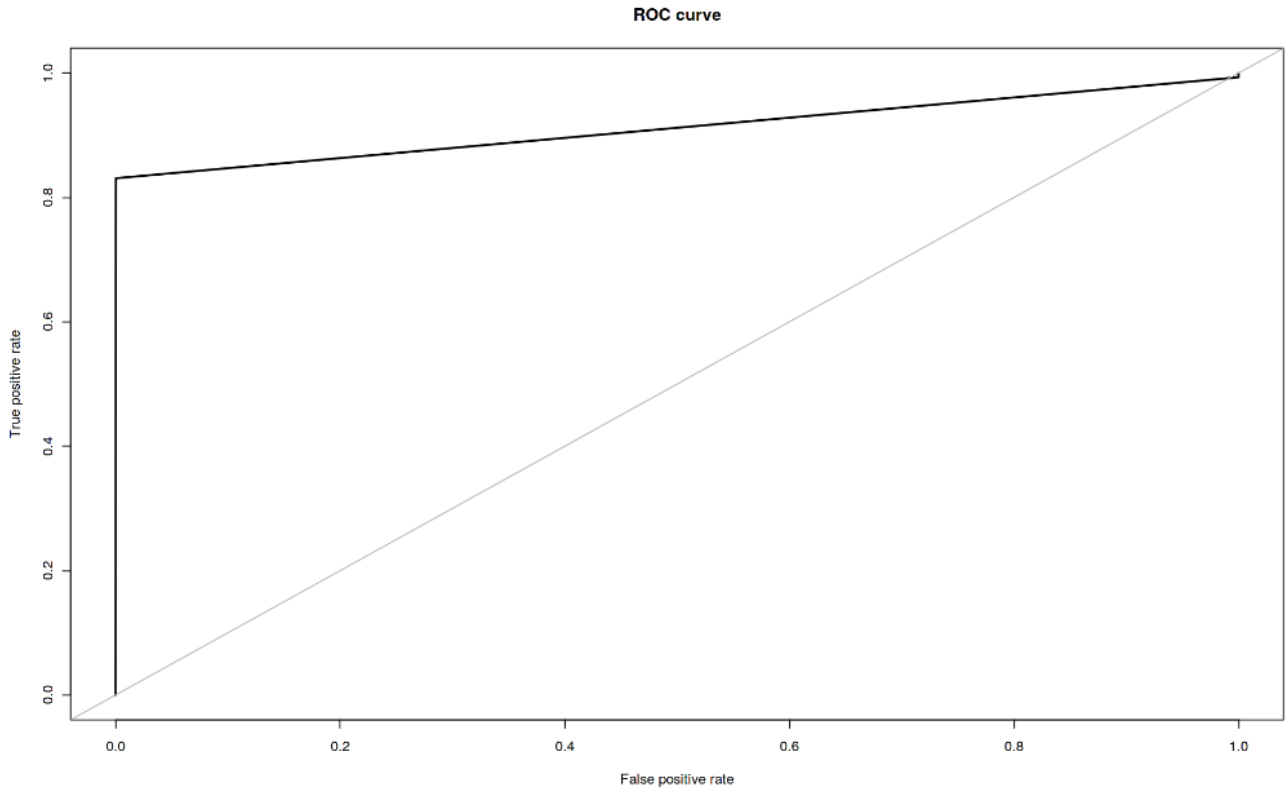
To begin, we conducted an evaluation of the decision tree model's performance using upsampled data. The decision tree model was trained on the balanced dataset, and we assessed its classification capability using the ROC AUC score. This score provides a measure of the model's ability to distinguish between the positive and negative classes, offering insights into its predictive performance.

Moving forward, we proceeded to fit a logistic regression model on the upsampled data. Similarly, we evaluated the performance of the logistic regression model using the ROC AUC score. This metric helped us assess how accurately the logistic regression model classified instances, providing valuable information about its classification accuracy.

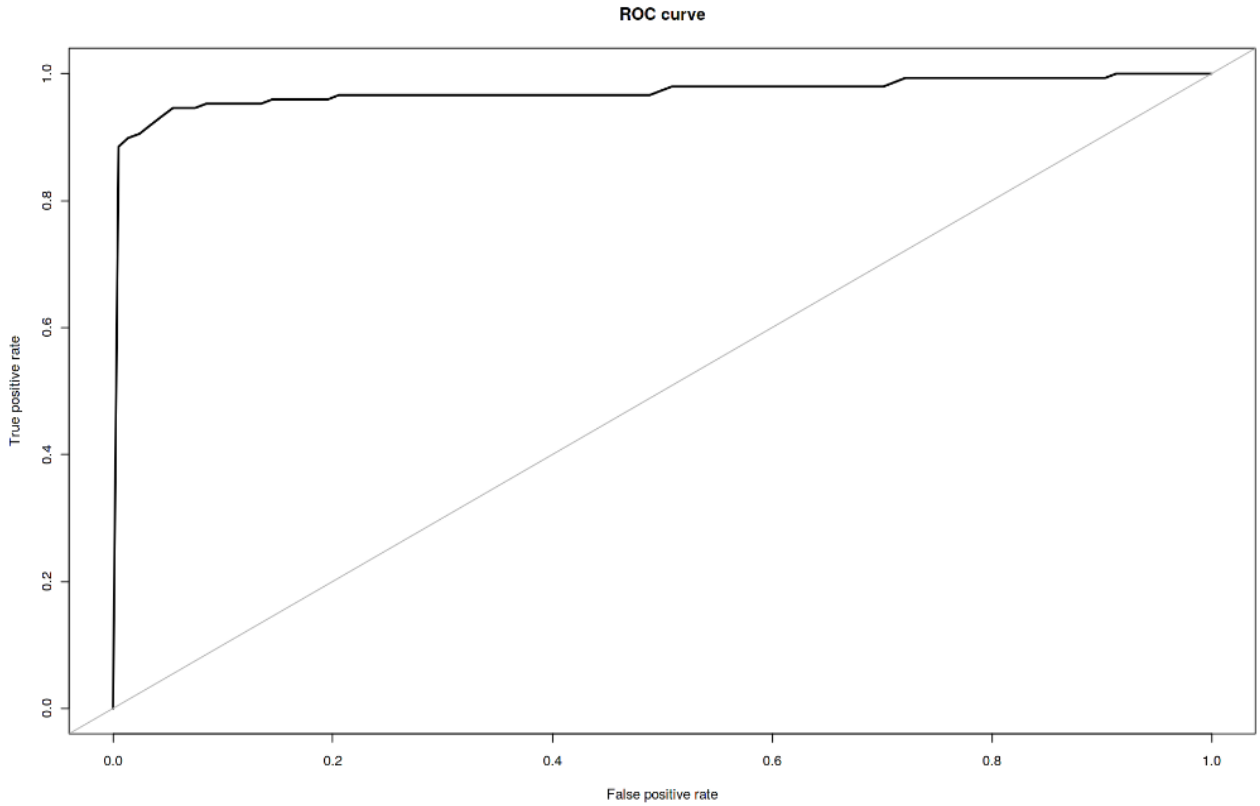
Finally, we employed the XGBoost model on the upsampled data and evaluated its performance using the ROC AUC score. The ROC AUC score allowed us to determine how effectively the XGBoost model ranked instances based on their predicted probabilities. This assessment helped us understand the model's ranking capability, which is crucial for various applications.

In summary, each model underwent evaluation based on its classification accuracy and ranking capability using the ROC AUC score. This approach provided us with valuable insights into the performance of the decision tree, logistic regression, and XGBoost models on the upsampled dataset. By utilizing this performance metric, we gained a comprehensive understanding of how well each model handled the balanced data and made informed comparisons between their respective performances.

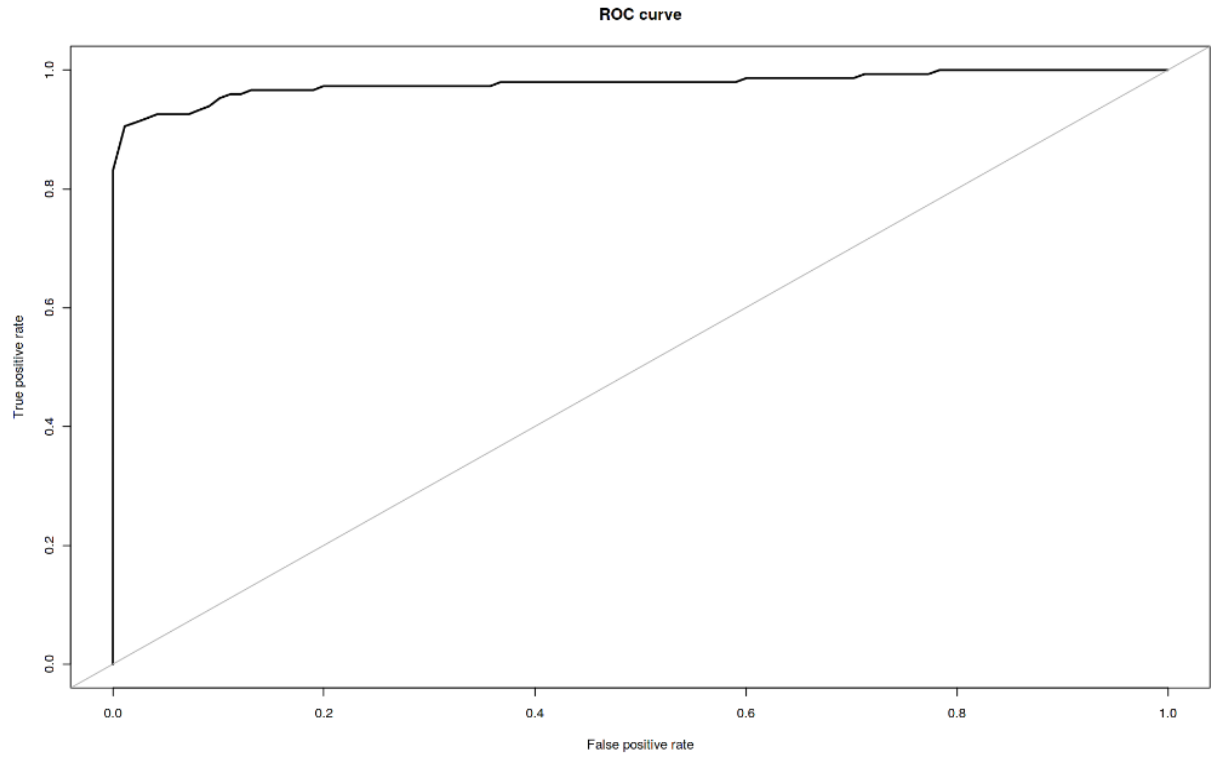
5. ROC Curve of Decision Trees



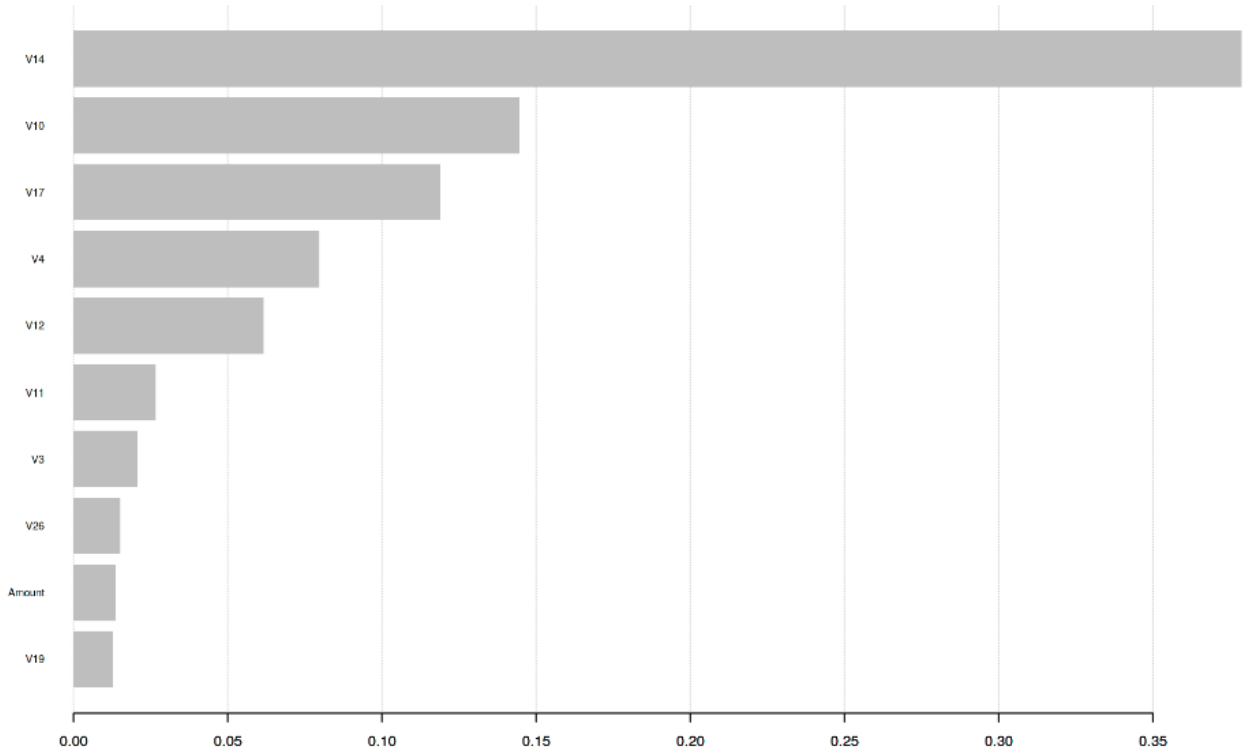
6. ROC Curve of Logistic Regression



7. ROC Curve of XGBoost



We can also take a look at the important features from the below graph:



IMPLEMENTATION

1. IMPORTING LIBRARIES

- The required libraries such as ‘dplyr’, ‘stringr’, ‘caret’, ‘caTools’, ‘ggplot2’, ‘corrplot’, ’Rtsne’, ’ROSE’, ‘rpart’, ‘xgboost’ and ‘Rborist’ were imported into the R environment.
- These libraries provide essential functions and tools for data manipulation, modelling, and evaluation.

2.DATA LOADING AND PREPROCESSING

- The credit card fraud dataset was loaded or imported to the R environment.
- Through basic Data Exploration, we concluded that the dataset is very imbalanced with 99.8% of cases being non-fraudulent transactions. A simple measure like accuracy is not appropriate here as even a classifier which labels all transactions as non-fraudulent will have over 99% accuracy.

3.DATA PREPARATION

- Dataset Overview: The dimensions of the "creditcard_data" dataset were examined using the dim() function, providing information about the number of rows and columns in the dataset. Additionally, the head() and tail() functions were used to display the first and last few rows of the dataset, allowing for a quick glance at the data structure and values.
- Variable Exploration: Several exploratory steps were taken to understand the dataset and its variables. The table() function was used to examine the distribution of the "Class" variable, providing a count of the number of transactions labelled as fraudulent and legitimate. Summary statistics, including mean, standard deviation, and range, were obtained using the summary(), var(), and sd() functions for the "Amount" variable. These exploratory analyses offer insights into the distribution and characteristics of the dataset.
- Train-Test Split: The dataset was split into training and testing datasets using the sample.split() function from the "caTools" library. This random split allocated 70% of the data to the training set and 30% to the testing set, ensuring separate datasets for model training and evaluation.

Since, the dataset was highly imbalanced So, we used certain methods used here to treat the imbalanced dataset. Those methods are :

- ❖ UNDERSAMPLING
- ❖ OVERSAMPLING
- ❖ ROSE (Random over-sampling)

3.DATA VISUALISATION AND DATA EXPLORATION

- Through basic Data Exploration AND Data Visualisation, we concluded that the dataset is very imbalanced with 99.8% of cases being non-fraudulent transactions. A simple measure like accuracy is not appropriate here as even a classifier which labels all transactions as non-fraudulent will have over 99% accuracy.
- Thus, we have employed upsampling to balance our training dataset.
- An appropriate measure of model performance here would be AUC(Area Under The Curve).

5.FITTING DIFFERENT ALGORITHMS ON DATA

- Firstly, we applied a decision tree model to upsampled data and assessed its performance using the ROC AUC score, which measures how well the model distinguishes between positive and negative classes.

- Next, we fitted a logistic regression model on the upsampled data and evaluated its performance using the ROC AUC score. This metric helps determine how effectively the logistic regression model classifies instances.

- Lastly, we utilised the XGBoost model on the upsampled data and assessed its performance using the ROC AUC score. This score provides insights into how well the XGBoost model ranks instances in terms of their predicted probabilities.

In summary, each model was evaluated based on its ability to classify instances correctly and rank them effectively, using the ROC AUC score as the performance metric.

AREAS OF IMPROVEMENT

The field of credit card fraud detection plays a crucial role in safeguarding financial transactions and protecting individuals from fraudulent activities. In this report, we aim to present an overview of our project on credit card fraud detection using R, highlighting our achievements and identifying key areas for improvement. While our current model has yielded promising results, it is essential to recognize that there are opportunities for enhancement to ensure more accurate and reliable fraud detection.

Current Performance Evaluation:
Our credit card fraud detection model, implemented using the R programming language, has shown commendable performance in identifying fraudulent transactions. The model's accuracy, precision, and recall metrics have demonstrated notable effectiveness in distinguishing between genuine and fraudulent transactions. These achievements lay a solid foundation for our work but call for further exploration to achieve even higher levels of accuracy.

Identifying Areas for Improvement:
Despite our model's success, there remain specific areas that warrant attention and improvement. By focusing on these areas, we can elevate our credit card fraud detection system to new levels of efficacy and efficiency.

- Exploring ensemble methods, such as stacking or model averaging, to combine the predictions of multiple models and leverage the strengths of each model.
- Incorporating more advanced anomaly detection algorithms or techniques specific to fraud detection, such as clustering algorithms or outlier detection methods, to identify anomalous patterns or behaviors associated with fraud.
- Incorporating external data sources, such as IP geolocation data, user behavior patterns, or additional transaction details, to augment the feature set and improve model performance. These external data sources can provide supplementary information that may enhance the model's ability to detect fraudulent transactions accurately.
- Optimizing the hyperparameters of our model is another area where improvements can be made. Techniques such as grid search, random search,

or Bayesian optimization can help us fine-tune the model's parameters and optimize its performance. By systematically exploring the parameter space, we can identify the optimal configuration that maximizes the model's accuracy and minimizes any overfitting concerns.

- To address the dynamic nature of credit card fraud, we can focus on enhancing the real-time capabilities of our model. By implementing streaming algorithms or developing efficient batch processing techniques, we can expedite the fraud detection process, allowing for timely interventions and minimizing potential losses. This improvement would significantly contribute to the effectiveness and responsiveness of our system.

CONCLUSION

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 are few compared to the instances of normal transactions. We have seen through different visualisations 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 oversampling or undersampling the response variable can lead to better model training. We concluded that the oversampling technique works best on the dataset and achieved significant improvement in model performance over the imbalanced data. The best score of 0.977 was achieved using an XGBOOST model though both decision tree and logistic regression models performed well too. It is likely that by further tuning the XGBOOST model parameters we can achieve even better performance. However, this project has demonstrated the importance of sampling effectively, modelling and predicting data with an imbalanced dataset.

In conclusion, this project focuses on developing a credit card fraud detection system using machine learning techniques. By leveraging historical transaction data and employing various analysis models, the project aims to accurately identify fraudulent transactions and enhance the security of credit card transactions.

REFRENCES

- https://data-flair.training/blogs/data-science-machine-learning-project-credit-card-fraud-detection/
- https://www.kaggle.com/code/atharvaingle/credit-card-fraud-detection-with-r-sampling
- https://www.r-project.org/other-docs.html

CODE

Introduction

Billions of dollars of loss are caused every year due to fraudulent credit card transactions. The design of efficient fraud detection algorithms is key to reducing these losses, and more algorithms rely on advanced machine learning techniques to assist fraud investigators. The design of fraud detection algorithms is however particularly challenging due to non-stationary distribution of the data, highly imbalanced classes distributions and continuous streams of transactions. At the same time public data are scarcely available for confidentiality issues, leaving unanswered many questions about which is the best strategy to handle this issue.

The dataset here contains transactions made by credit cards in September 2013 by european cardholders. This dataset has been taken from Kaggle. 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. Unfortunately, due to confidentiality issues, we do not have access to the original features and more background information about the data. 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-senstive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

The objective of the project is to train a machine learning algorithm on the dataset to successfully predict fraudulent transactions.

Given the class imbalance ratio, we will be using measuring the accuracy using the Area Under the Precision-Recall Curve (AUC). Confusion matrix accuracy is not meaningful for unbalanced classification.

We will also use different sampling techniques on the train dataset in order to address the issue of imbalanced classes while training our models.

Type *Markdown* and LaTeX: α^2

Importing libraries

In [23]:
```
install.packages("xgboost")
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
```

In [ ]:
```
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(xgboost) # for xgboost model
```

In [2]:
```
# function to set plot height and width
fig <- function(width, heigth){
     options(repr.plot.width = width, repr.plot.height = heigth)
}
```

In [3]:
```
# loading the data
df = read.csv('/kaggle/input/credit-card-fraud-detection-using-ml/Credit card
```

Basic Data Exploration

In [4]:
```
head(df)
```

A data.frame: 6 × 31

	Time	V1	V2	V3	V4	V5	V6	V7
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	0	-1.3598071	-0.07278117	2.5363467	1.3781552	-0.33832077	0.46238778	0.23959855
2	0	1.1918571	0.26615071	0.1664801	0.4481541	0.06001765	-0.08236081	-0.07880298
3	1	-1.3583541	-1.34016307	1.7732093	0.3797796	-0.50319813	1.80049938	0.79146096
4	1	-0.9662717	-0.18522601	1.7929933	-0.8632913	-0.01030888	1.24720317	0.23760894
5	2	-1.1582331	0.87773676	1.5487178	0.4030339	-0.40719338	0.09592146	0.59294075
6	2	-0.4259659	0.96052304	1.1411093	-0.1682521	0.42098688	-0.02972755	0.47620095

In [5]:
```
str(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 ...
```

In [6]:
```
summary(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.:139320   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.   :-43.55
72
 1st Qu.:-0.84864   1st Qu.:  -0.69160   1st Qu.: -0.7683   1st Qu.: -0.55
41
 Median :-0.01985   Median :  -0.05434   Median : -0.2742   Median :  0.04
01
 Mean   : 0.00000   Mean   :   0.00000   Mean   :  0.0000   Mean   :  0.00
00
 3rd Qu.: 0.74334   3rd Qu.:   0.61193   3rd Qu.:  0.3986   3rd Qu.:  0.57
04
 Max.   :16.87534   Max.   :  34.80167   Max.   : 73.3016   Max.   :120.58
```

In [7]:
```
# checking missing values
colSums(is.na(df))
```

```
Time
0
V1
0
V2
0
V3
0
V4
0
V5
0
V6
0
V7
0
V8
```

None of the variables have missing values

In [8]:
```
# checking class imbalance
table(df$Class)
```

```
     0      1
284315    492
```

In [9]:
```
# class imbalance in percentage
prop.table(table(df$Class))
```

```
          0           1
0.998272514 0.001727486
```

In [10]:
```
fig(12, 8)
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
```

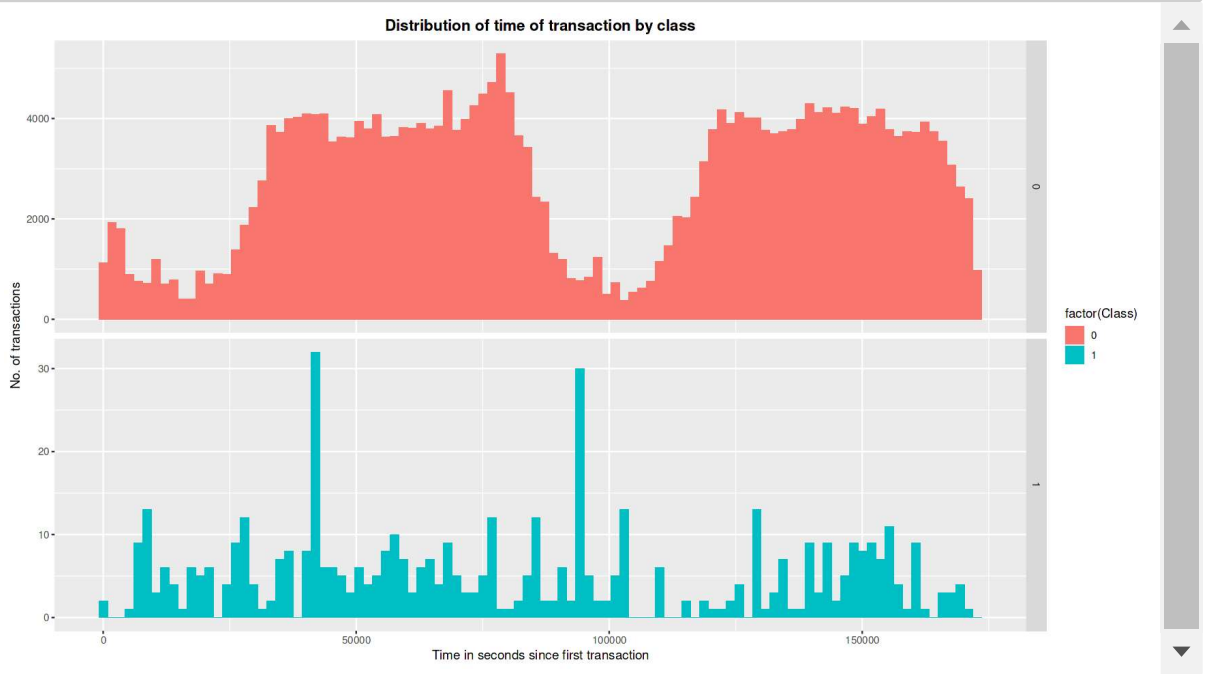
```
Warning message:
“`stat(count)` was deprecated in ggplot2 3.4.0.
ℹ Please use `after_stat(count)` instead.”
```

Clearly the dataset is very imbalanced with 99.8% of cases being non-fraudulent transactions. A simple measure like accuracy is not appropriate here as even a classifier which labels all transactions as non-fraudulent will have over 99% accuracy. An appropriate measure of model performance here would be AUC (Area Under the Precision-Recall Curve)

Data Visualization

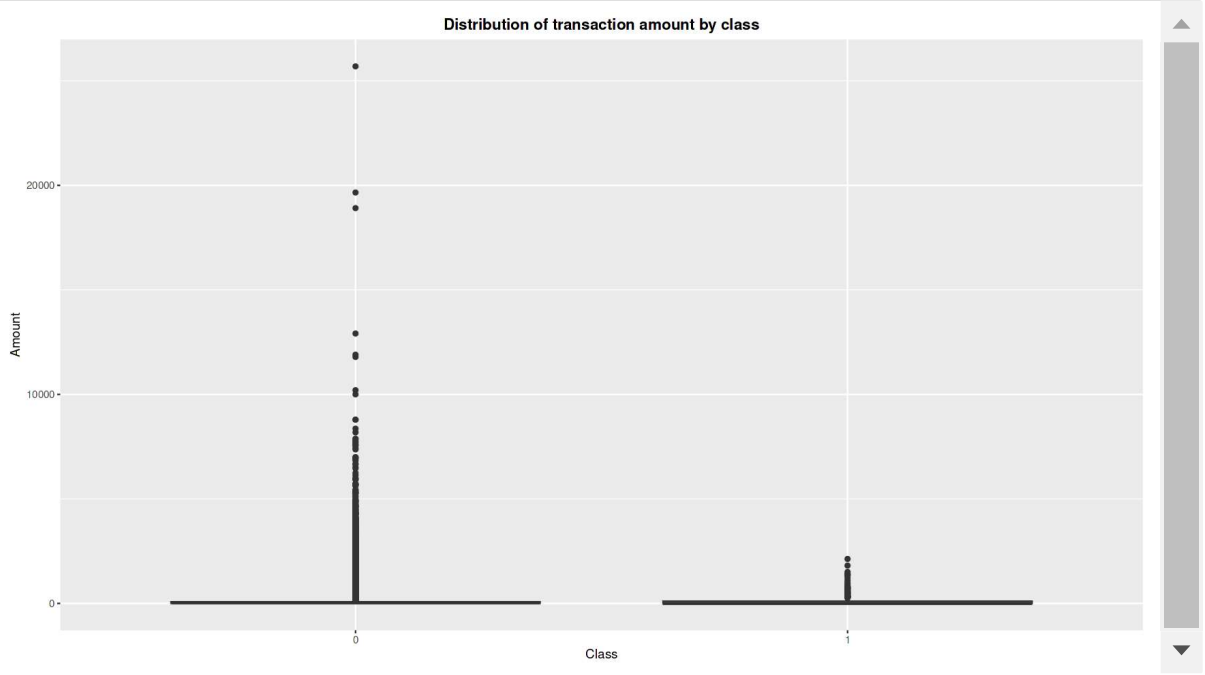
Distribution of variable 'Time' by class

In [11]:
```
fig(14, 8)
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

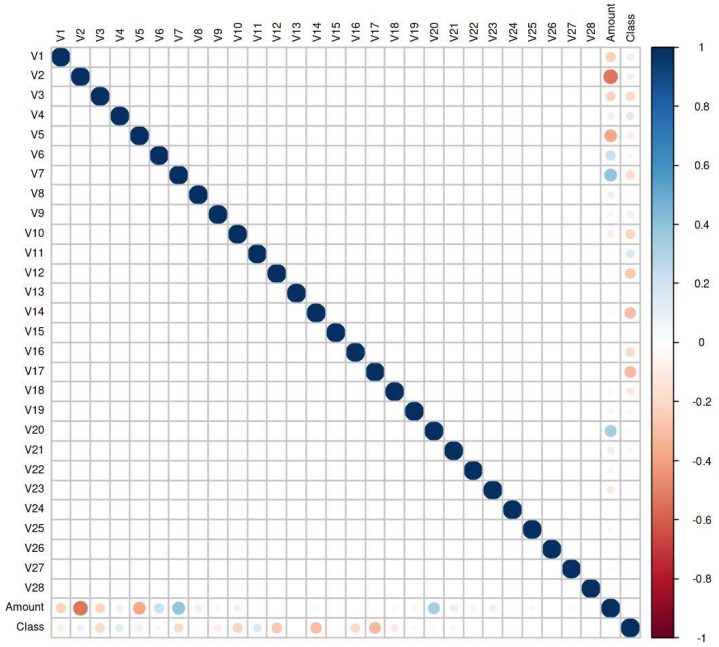
In [12]:
```
fig(14, 8)
ggplot(df, aes(x = factor(Class), y = Amount)) + geom_boxplot() +
labs(x = 'Class', y = 'Amount') +
ggtitle("Distribution of transaction amount by class") + common_theme
```



There is clearly a lot more variability in the transaction values for non-fraudulent transactions.

Correlation of anonymised variables and 'Amount'

In [13]:
```
fig(14, 8)
correlations <- cor(df[,-1],method="pearson")
corrplot(correlations, number.cex = .9, method = "circle", type = "full", tl.c
```



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

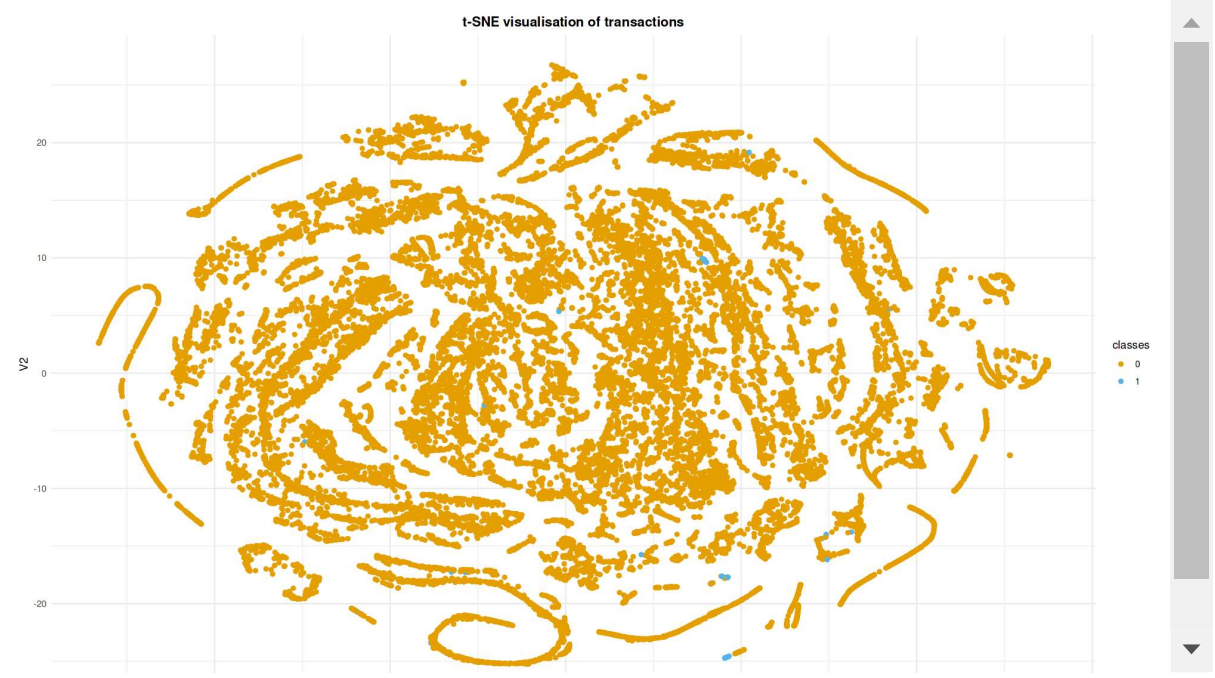
Visualization of transactions using t-SNE

To try to understand the data better, we will try visualizing the data using t-Distributed Stochastic Neighbour Embedding, a technique to reduce dimensionality.

To train the model, perplexity was set to 20.

The visualisation should give us a hint as to whether there exist any “discoverable” patterns in the data which the model could learn. If there is no obvious structure in the data, it is more likely that the model will perform poorly.

In [14]:
```
fig(16, 10)

# Use 10% of data to compute t-SNE
tsne_subset <- 1:as.integer(0.1*nrow(df))
tsne <- Rtsne(df[tsne_subset,-c(1, 31)], perplexity = 20, theta = 0.5, pca = F

classes <- as.factor(df$Class[tsne_subset])
tsne_mat <- as.data.frame(tsne$Y)
ggplot(tsne_mat, aes(x = V1, y = V2)) + geom_point(aes(color = classes)) + the
```



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.

Modeling Approach

Standard machine learning algorithms struggle with accuracy on imbalanced data for the following reasons:

ML algorithms struggle with accuracy because of the unequal distribution in dependent variable.This causes the performance of existing classifiers to get biased towards majority class. The algorithms are accuracy driven i.e. they aim to minimize the overall error to which the minority class contributes very little. ML algorithms assume that the data set has balanced class distributions. They also assume that errors obtained from different classes have same cost.

The methods to deal with this problem are widely known as **'Sampling Methods'**. Generally, these methods aim to modify an imbalanced data into balanced distribution using some mechanism. The modification occurs by altering the size of original data set and provide the same proportion of balance.

These methods have acquired higher importance after many researches have proved that balanced data results in improved overall classification performance compared to an imbalanced data set. Hence, it's important to learn them.

Below are the methods used here to treat the imbalanced dataset:

Undersampling Oversampling Synthetic Data Generation

Undersampling

This method reduces the number of observations from majority class to make the data set balanced. This method is best to use when the data set is huge and reducing the number of training samples helps to improve run time and storage troubles.

Undersampling methods are of 2 types: Random and Informative.

Random undersampling method randomly chooses observations from majority class which are eliminated until the data set gets balanced. Informative undersampling follows a pre-specified selection criterion to remove the observations from majority class.

A possible problem with this method is that removing observations may cause the training data to lose important information pertaining to majority class.

Oversampling

This method works with minority class. It replicates the observations from minority class to balance the data. It is also known as upsampling. Similar to undersampling, this method also can be divided into two types: Random Oversampling and Informative Oversampling.

Random oversampling balances the data by randomly oversampling the minority class. Informative oversampling uses a pre-specified criterion and synthetically generates minority class observations.

An advantage of using this method is that it leads to no information loss. The disadvantage of using this method is that, since oversampling simply adds replicated observations in original data set, it ends up adding multiple observations of several types, thus leading to overfitting.

Synthetic Data Generation (ROSE)

In simple words, instead of replicating and adding the observations from the minority class, it overcome imbalances by generates artificial data. It is also a type of oversampling technique.

ROSE (random over-sampling examples) uses smoothed bootstrapping to draw artificial samples from the feature space neighbourhood around the minority class.

It is important to note that sampling techniques should only be applied to the training set and not the testing set.

Our modeling approach will involve training a single classifier on the train set with class imbalance suitably altered using each of the techniques above. Depending on which technique yields the best roc-auc score on a holdout test set, we will build subsequent models using that chosen technique.

Data Preparation

'Time' feature does not indicate the actual time of the transaction and is more of listing the data in chronological order. Based on the data visualization above we assume that 'Time' feature has little or no significance in correctly classifying a fraud transaction and hence eliminate this column from further analysis.

In [15]:
```
#Remove 'Time' variable
df <- df[,-1]
```

In [16]:
```
#Change 'Class' variable to factor
df$Class <- as.factor(df$Class)
levels(df$Class) <- c("Not_Fraud", "Fraud")

#Scale numeric variables

df[,-30] <- scale(df[,-30])

head(df)
```

A data.frame: 6 × 30

	V1	V2	V3	V4	V5	V6	V7	
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<
1	-0.6942411	-0.04407485	1.6727706	0.9733638	-0.245116153	0.34706734	0.1936786	0.0826
2	0.6084953	0.16117564	0.1097969	0.3165224	0.043483276	-0.06181986	-0.0637001	0.0712
3	-0.6934992	-0.81157640	1.1694664	0.2682308	-0.364571146	1.35145121	0.6397745	0.2073
4	-0.4933240	-0.11216923	1.1825144	-0.6097256	-0.007468868	0.93614819	0.1920703	0.3160
5	-0.5913287	0.53154012	1.0214099	0.2846549	-0.295014918	0.07199846	0.4793014	-0.2265
6	-0.2174742	0.58167387	0.7525841	-0.1188331	0.305008424	-0.02231344	0.3849353	0.2179

Split data into train and test sets

In [17]:
```
set.seed(123)
split <- sample.split(df$Class, SplitRatio = 0.7)
train <-  subset(df, split == TRUE)
test <- subset(df, split == FALSE)
```

Choosing sampling technique

In [18]:
```
# class ratio initially
table(train$Class)
```

```
Not_Fraud     Fraud
   199020       344
```

In [19]:
```
# downsampling
set.seed(9560)
down_train <- downSample(x = train[, -ncol(train)],
                         y = train$Class)
table(down_train$Class)
```

```
Not_Fraud     Fraud
      344       344
```

In [20]:
```
# upsampling
set.seed(9560)
up_train <- upSample(x = train[, -ncol(train)],
                         y = train$Class)
table(up_train$Class)
```

```
Not_Fraud     Fraud
   199020    199020
```

In [27]:
```
# rose
set.seed(9560)
rose_train <- ROSE(Class ~ ., data  = train)$data

table(rose_train$Class)
```

```
Not_Fraud     Fraud
    99844     99520
```

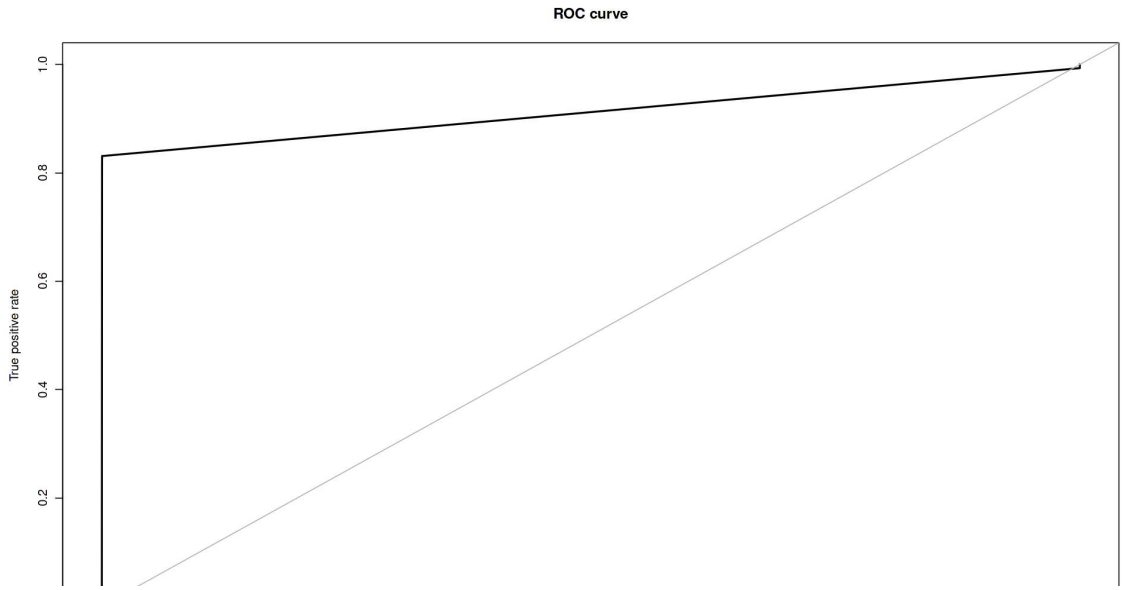
Decision Trees (CART)

Before we start using sampling let us first look at how CART performs with imbalanced data. We use the function roc.curve available in the ROSE package to gauge model performance on the test set

Decision trees on original (imbalanced) dataset

In [28]:
```
#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
```



We evaluate the model performance on test data by finding the roc auc score

We see that the auc score on the original dataset is 0.912 . We will now apply various sampling techniques to the data and see the performance on the test set.

Decision Tree on various sampling techniques

In [30]:
```
set.seed(5627)
# Build down-sampled model


down_fit <- rpart(Class ~ ., data = down_train)


set.seed(5627)
# Build up-sampled model


up_fit <- rpart(Class ~ ., data = up_train)


set.seed(5627)
# Build smote model



set.seed(5627)
# Build rose model


rose_fit <- rpart(Class ~ ., data = rose_train)
```

In [31]:
```
# AUC on down-sampled data
pred_down <- predict(down_fit, newdata = test)

print('Fitting model to downsampled data')
roc.curve(test$Class, pred_down[,2], plotit = FALSE)
```

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

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

In [32]:
```
# AUC on up-sampled data
pred_up <- predict(up_fit, newdata = test)

print('Fitting model to upsampled data')
roc.curve(test$Class, pred_up[,2], plotit = FALSE)
```

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

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

In [33]:
```
# AUC on up-sampled data
pred_rose <- predict(rose_fit, newdata = test)

print('Fitting model to rose data')
roc.curve(test$Class, pred_rose[,2], plotit = FALSE)
```

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

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

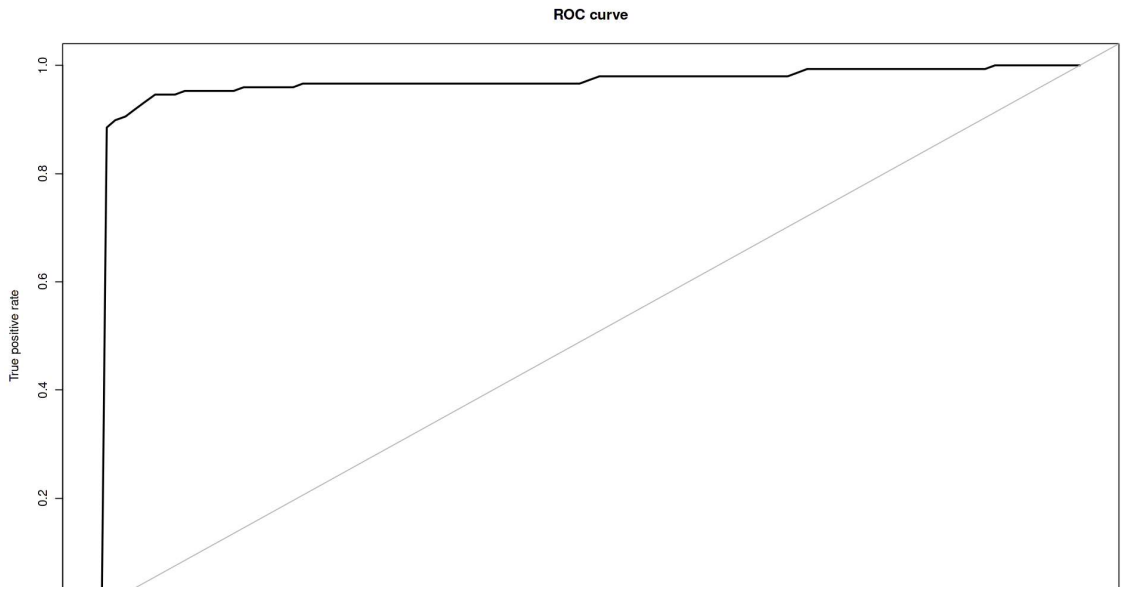
We see that all the sampling techniques have yielded better auc scores than the simple imbalanced dataset. We will test different models now using the **up sampling technique** as that has given the highest auc score

Models on upsampled data

Logistic Regression

In [34]:
```
glm_fit <- glm(Class ~ ., data = up_train, family = 'binomial')

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

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

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



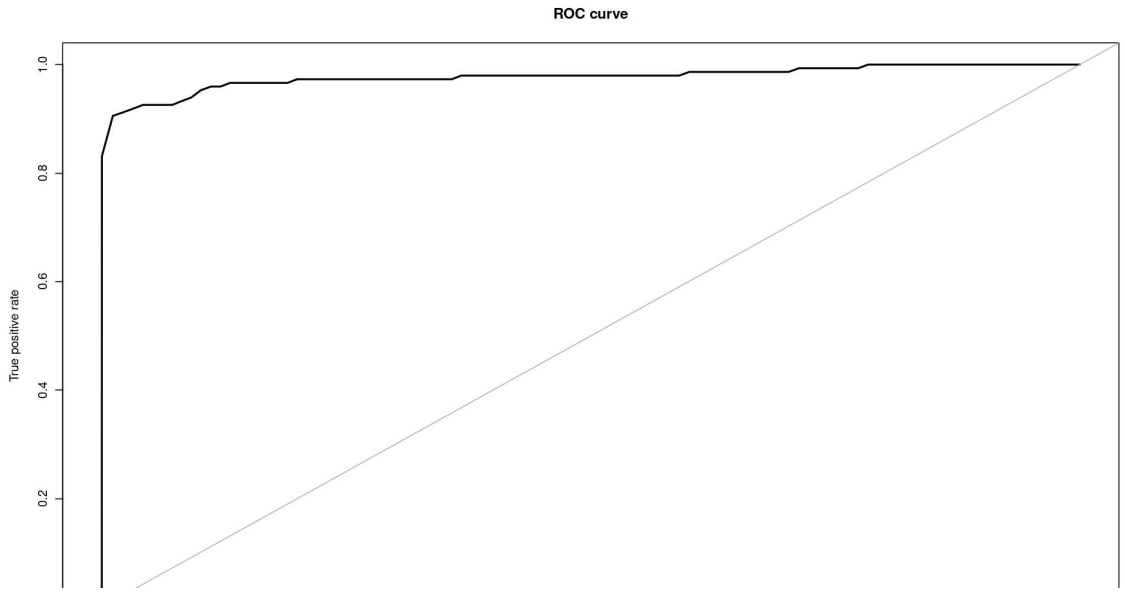
XGBoost

In [35]:
```
# Convert class labels from factor to numeric

labels <- up_train$Class

y <- recode(labels, 'Not_Fraud' = 0, "Fraud" = 1)
```

In [36]:
```
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,
)
```

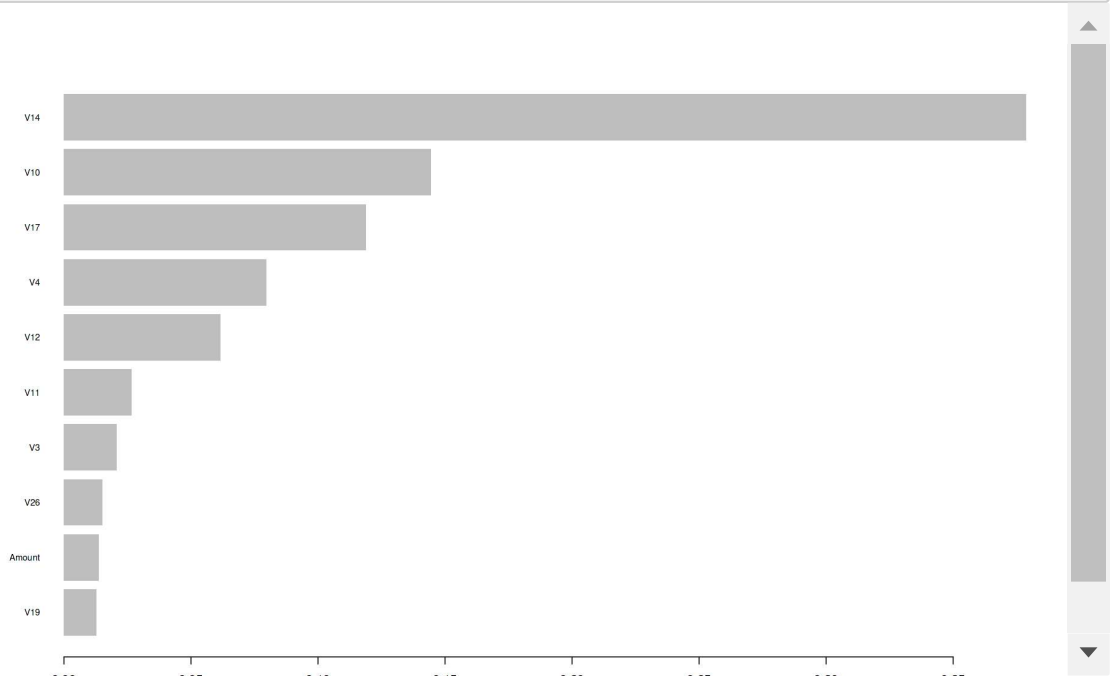
In [37]:
```
xgb_pred <- predict(xgb, data.matrix(test[,-30]))

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

```
Area under the curve (AUC): 0.977
```



We can also take a look at the important features here.

In [38]:
```
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,])
```



With an auc score of 0.977 the XGBOOST model has performed the best though both the decision trees and logistic regression models have shown reasonable performance.

Conclusion

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 a appropriate measure of model performance here and used the metric AREA UNDER ROC CURVE to evaluate how different methods of oversampling or undersampling the response variable can lead to better model training. We concluded that the oversampling technique works best on the dataset and achieved significant improvement in model performance over the imabalanced data. The best score of 0.977 was achieved using an XGBOOST model though both decision tree and logistic regression models performed well too. It is likely that by further tuning the XGBOOST model parametres we can achieve even better performance. However this project has demonstrated the importance of sampling effectively, modelling and predicting data with an imbalanced dataset.

In [ ]: