University of Fort Hare
Together in Excellence

The classification performance of ensemble decision tree classifiers: A case study of detecting fraud in credit card transactions

by

Mcdonald Chogugudza

202014068

A dissertation

submitted in fulfilment of the requirement for the degree of

Master of Science

in

Computer Science

in the

Faculty of Science and Agriculture

of the

University of Fort Hare Alice

South Africa

Supervisor: **Mr S. Ngwenya**

Co-Supervisor**: Prof K. Sibanda**

Statement of Original Authorship

I, **Mcdonald Chogugudza** do hereby confirm that all the work contained in this dissertation has not been submitted for any other qualification at this or any other University. Furthermore, I confirm that the dissertation does not contain any published information except where due reference has been indicated.

Signature: [illegible]

Date: 07/09/2022

IN LUMINE TUO VIDE BIMUS LUMEN

University of Fort Hare

Together in Excellence

Plagiarism Declaration

I, **Mcdonald Chogugudza** student number **202014068** hereby declare that I am fully aware of the University of Fort Hare's policy on plagiarism and I have taken every precaution to comply with regulations.

Signature: [illegible]………………………………...

Abstract

In this dissertation, we propose ensemble decision tree classifiers as an ideal classification technique for solving the problem of fraud in the domain of credit card transactions. Ensemble tree classifiers have been applied in many areas like speech recognition, image recognition and medical diagnostics and have shown excellent results. At the centre of fraud, credit card fraud has been a major concern. The rise in credit card fraud is largely attributed to the nature in which it can be done. A fraudster does not need to always be physically present to commit fraud making it the number one target for criminals. Card-Not-Present refers to this type of fraud where an electronic transaction can be conducted without the need for a client to be present. This can be done via telephonic calls or the web. To be able to come up with better classifiers it was important for the researcher to first investigate what causes misclassifications in fraud detection systems. A systematic literature review was done to uncover the factors that have been identified as causes of misclassifications. It was discovered that many factors lead to misclassifications and several authors have proposed techniques to handle these factors. However, there is no universal techniques for addressing factors that lead to misclassifications as different domains have different datasets which require different techniques. This study investigates how parameters involved in modelling fraud detection systems impact the classification performance of ensemble decision tree classifiers. The factors that were investigated include sample size, sampling technique, learning method and choice of split criterion and how they affect classification performance. A series of experiments were conducted to investigate how the aforementioned factors contributed to better classifiers. E-commerce data from Vesta corporation made available on Kaggle was used in the experiments. The data was split into two sets, one for training the models and the other for testing the performance of the models. Accuracy, confusion matrix, precision and recall were used as performance measures. Our results showed that a larger sample size resulted in better classifiers. This is attributed to models having more instances to learn from which covers most patterns of fraudulent transactions. The sampling technique was shown to be pivotal in classification performance as under sampling showed a great reduction in performance as it achieved a maximum accuracy of 89.6223 while oversampling produced increased performance with maximum accuracy of 99.9531. Furthermore, our results showed that the choice of split criterion impacts the performance of ensemble tree classifiers. The use of entropy as the choice of split criterion resulted in better classifiers compared to the use of the Gini index. However, the downside is that entropy requires more time to execute compared to the Gini index. Lastly, the learning method proved to impact the performance of ensemble classifiers. Models that used supervised learning had better performance compared to those that use unsupervised learning in detecting credit card fraud. The conclusions from this research are insightful when designing fraud detection systems that use ensemble decision tree classifiers as base learners.

Acknowledgements

Firstly, I would love to acknowledge our heavenly father who is the provider of all life, intelligence and wisdom. I am grateful to God who has been my core source of strength throughout this programme and for keeping me safe during this Covid pandemic.

I am grateful to my research supervisor and co-supervisor Mr S. Ngwenya and Professor K. Sibanda, respectively for their immeasurable insight, guidance, encouragement and support throughout this dissertation. I am also grateful to the H.o.D of the Department of Computer Science Mr K. Sibanda for granting me the opportunity to be a part of this Masters programme and for being patient with me always.

I would like to acknowledge the ongoing support from my parents Mr & Mrs M.E. Chogugudza, my brothers Perseverance and Munyaradzi. I would also love to thank my son Prince Shalom Chogugudza for whom I desire to be an example. Lastly, I would like to thank Neliswa Jack and my sister Chipo Purazeni for encouraging me and lifting my head when I felt overwhelmed. Their encouragement, love, prayers and support through my academic pursuit was and still is greatly appreciated.

List of Acronyms

CNP	Card-Not-Present
DTC	Decision Tree Classifier
ETC	Extra Tree Classifier
RF	Random Forest
iForest	Isolation Forest
ADB	AdaBoost
GDB	Gradient Boosting
FDS	Fraud Detection System
FPS	Fraud Prevention System
FP	False Positive
FN	False Negative
TP	True Positive
TN	True Negative
SVM	Support Vector Machine
PCA	Principal Component Analysis
SMOTE	Synthetic Minority Oversampling Technique
RUS	Random Undersampling
ROS	Random Oversampling
PIN	Personal Identification Number
ATM	Automated Teller Machine
CRISP-DM	Cross-Industry Standard Process for Data Mining
SEMMA	Sample, Explore, Modify, Model, and Assess
KDD	Knowledge Discovery in Databases
EMV	Europay, Mastercard and Visa
KDE	Kernel Density Estimator
RKDE	Robust Kernel Density Estimator
AODE	Angle Based Outlier Detector
VM	Vector Machine
LFO	Local Factor Outlier
SVDD	Support Vector Data Description
GMM	Gaussian Mixture Model

EGMM	Extended Gaussian Mixture Model
ACFE	Association of Certified Fraud Examiners
SABRIC	South African Banking and Risk Information Centre

Publications

Part of the research work presented in this dissertation has been published or has been submitted for publication or review in the following papers:

M. Chogugudza, S. Ngwenya and K. Sibanda, *On Critical factors that lead to misclassification in Fraud Detection Systems and how false positives impact businesses*.

M. Chogugudza, S. Ngwenya and K. Sibanda, *On the Effect of Sample Size and Sampling Technique on Classification performance of Ensemble Tree Classifiers in detecting Credit Card Fraud*.

M. Chogugudza, S. Ngwenya and K. Sibanda, *On The effect of Choice of Split Criterion and Learning method on Classification performance of Ensemble Tree Classifiers in detecting Credit Card Fraud*.

Table of Contents

List of Figures

List of Tables

Chapter One

1 Introduction to the research and its setting

1.1 Background

Credit card fraud continues to rise and billions of Rands are being lost yearly due to fraudsters (Budhram, 2016). Several fraud detection systems (FDS) have been developed to combat the growing numbers of fraud cases but in the process, merchants have recorded huge losses due to legitimate transactions that are being denied after being wrongly classified as fraud. Husejinovic (2020) points out that credit card payments have been on the rise allowing individuals to transact even when not physically present. As a result, credit card fraud has also increased accounting for card losses of up to $21,84 billion globally (Husejinovic, 2020). These enormous financial losses critically affect not only individuals using credit cards but also merchants and banks (Shirgave, Suresh & Awati, Chetan & More, Rashmi & Patil, Sonam, 2019). The increase in incidences of credit card fraud is largely attributed to the weakness of traditional credit card processing systems that are less secure such as the magnetic stripe card system (Budhram, 2016). Fraudsters continue to formulate highly sophisticated techniques to commit credit card fraud (Panigrahi, Suvasini & Kundu, Amlan & Sural, Shamik & Majumdar, Arun, 2009). Stolen and lost credit card fraud constituted the highest percentage of card losses on cards issued in South Africa between 2005 and 2008 (SABRIC, Credit card fraud South Africa, 2008).

However, losses resulting from stolen or lost credit cards has continued to decrease from 2005 to 2014 (Budhram, 2016). Previously, losses resulting from Card-Not-Present (CNP) Fraud cases were comparatively low. However, this type of fraud continued to upsurge at the rate of nearly 50% yearly (SABRIC, Credit card fraud South Africa, 2008). CNP fraud losses increased by almost 7% and accounted for approximately 40% of the total credit card fraud losses in 2014 (SABRIC, Card Fraud Booklet, 2014). The upsurge in CNP fraud seen over that period indicates that South African credit card fraud is rising at a ratio relatively proportional to other Europay, Mastercard and Visa (EMV) compliant countries such as the UK. The surge in CNP fraud is attributed to banks initiatives to allow e-commerce transactions (SABRIC, Card Fraud Booklet, 2014).

According to Budhram (2016) Banks in South Africa have implemented complex software to detect, prevent and reduce credit card fraud. Some of the measures taken by banks in South Africa to reduce credit card fraud include SMS notifications, authorisation measures like One Time Pin (OTP), thresholds and the introduction of chip-and-pin to replace traditional magnetic stirp cards (Budhram, 2016). Brudhram (2016) notes that these measures are rather reactive and hence a need for an intelligent-led approach to combat credit card fraud. However, criminals are always finding new ways to commit fraud because of the emergence of advanced technologies (Awati & More, 2019). Therefore, there is a constant need to review existing models to combat new methods of committing credit card fraud.

To counter challenges resulting from credit card fraud, credit card companies are constantly reviewing security features on the cards and upgrading security systems that can detect suspected fraudulent transactions (Budhram, 2016). In 2010/11, over 11 000 cases of credit card fraud were investigated by the police and only a small fraction of the cases went through court seating (South African Police Service, 2011). This shows the level of difficulty to deal with fraud crimes.

Most banks use transactional/account level systems and behavioural model systems to classify transactions as legal or fraudulent. Transactional level systems raise fraud alerts of large sums of money that have been paid to certain merchants during certain times whereas behavioural models raise alerts when there is a change in customer shopping behaviour (C. Whitrow, D. J. Hand, P. Juszczak, D. Weston, N. M. Adams, 2008). Whitrow et al. (2008) concluded that both systems have higher chances of raising false alerts as a change in behaviour does not always mean fraud is being committed. This results in many transactions being wrongly flagged as fraud. This study did not consider location or behavioural changes when detecting fraud to determine the efficacy of models.

The current literature has largely focused on addressing how credit card fraud can be detected using several techniques including Machine learning, Artificial intelligence and Neural networks (Sonal, Mehndiratta & Mr Kamal, Gupta, 2019). The limitation of the current systems is that they largely focus on preventing credit card fraud but pay little attention to the negative effect of legitimate transactions that are falsely declined, a concept known as false positives. As a result, more research needs to be done to find ways of reducing false alerts which can be frustrating for credit card users and also negatively impact merchants in terms of revenue, brand image and customer retention. This research added value to the knowledge base of anomaly detection on large e-commerce datasets to improve the efficacy of fraudulent transaction alerts from banks in South Africa. The problem statement is outlined in the next section.

1.2 Statement of the problem

An ideal fraud detection and prevention system must be able to permit legitimate transactions to be processed while denying fraudulent ones. However, this is not the reality as most fraudsters mimic legitimate transactions to go unnoticed which makes it complex to create a full proof system. Current fraud detection and prevention systems have recorded a significant number of legitimate transactions that are denied. This has a negative impact of the businesses as it may result in lost revenue and unhappy customers who may decide never to buy from the business again. This study is essential for South Africa, as it will help financial institutions reduce losses arising from false positives thereby increasing revenue. This study can also be essential for credit card users in South Africa who are often frustrated by false positives which often result in denial of legitimate transactions.

The next section will look at the research question and sub-questions.

1.3 Research question and sub-questions

Due to the problems highlighted in the problem statement, the main research question is:

How well do ensemble decision tree classifiers perform in detecting credit card fraud?

To answer the main research question, the researcher formulated 5 sub-questions.

1. *How do misclassifications impact companies that use credit cards?*
2. *How does the choice of sampling technique impact the performance of ensemble decision tree classifiers?*
3. *How does the sample size of the training dataset impact the classification performance of ensemble decision tree classifiers?*
4. *How does split criterion affect the performance of ensemble decision tree classifiers?*
5. *How supervised and unsupervised ensemble tree classifiers compare in performing classification tasks.*

1.4 Aim and objectives

1.4.1 Aim

This dissertation aims to compare the classification performance of ensemble decision trees in detecting credit card fraud to archive the lowest rate of false positives while maintaining low time complexity.

1.5 Objectives

To achieve this, the following objectives are set:

1. To analyse how misclassification impact companies that use credit cards.
2. To determine how the choice of sampling technique affects the classification performance of ensemble decision tree classifiers in detecting credit card fraud.
3. To explore how the sample size of the training data set impacts the classification performance of ensemble decision tree classifiers.
4. To determine how the choice of split criterion impact the classification performance of decision tree classifiers.
5. To determine how supervised and unsupervised ensemble tree classifiers compare in performing classification tasks.

1.6 Contribution of the study

This study is important as it will examine the different data mining tools that can be used to detect anomalies on large banking datasets to better classify transactions as legitimate or fraudulent. This will improve the precision of fraud alerts and save billions of Rands for both individuals and merchants in South Africa. The major challenge with credit card fraud detection is the amount of data that is being generated from online transactions. This phenomenon is called big data which explains the volume, veracity and velocity of data that is being generated daily. As a result, data mining techniques are used to quickly identify patterns or classify data to allow quick decision making and detect anomalies on large datasets. Another challenge is the amount of time taken to analyse these large datasets. This study will look at technologies available that can analyse large datasets efficiently. The cost of fraud might be less compared to the loss that will arise from false positives. False positives may have long term financial implications on the business. This research seeks to explore how to reduce false positives in fraud detection systems by exploring the effectiveness of ensemble decision tree classifiers in detecting credit card fraud.

The next section will look at the literature review used in this study.

1.7 Research Methodology

Omankwu (2018) defines research as "the activity of a diligent and systematic inquiry or investigation in an area, with the objective of discovering or revising facts, theories, applications to discover and disseminate new knowledge". The research made use of both theoretical and simulation methodology. In the theoretical methodology, qualitative and quantitative methods will be used.

To tackle the problem models to detect anomalies in banking datasets to identify fraud were implemented using data mining tools. The model will be an implementation of the Isolation Forest, Random Forest, Gradient Boost, Adaboost and Extra Tree classifiers in Python. Python was chosen and implemented in the Jupyter notebook. Jupyter notebooks are preferable because of the computational power they possess compared to even the state-of-the-art Personal Computers we use. The Jupyter notebook was be accessed via the web interface. The program runs on a remote computer, that is, the researcher did not need to install Python on their personal computer. Another advantage of using Jupyter notebooks is that they can be shared on the web.

This study made use of secondary data provided by Vesta Corporation, which has made available real-world e-commerce transactions on Kaggle an online competition site for Data Science. The dataset has 284,807 transactions of which 492 are fraud. The dataset is highly unbalanced with approximately 0.01% being true positives (TP). The data was split into two groups, training and test data. The test data will be used to establish a performance score of the model by predicting the missing field of whether a transaction is fraudulent or not. This next section discusses the scope of the study.

1.8 Scope of the study

Data mining algorithms have been used to model FDS. In this study, we used Projection-based methods for this purpose. The Isolation Forest, Random Forest, Gradient Boost, Extra tree, decision tree and Adaboost were applied in credit card fraud detection. There are a lot of fraudulent behaviours that credit card users are prone to, however, this study is limited to E-commerce transactions. This study does not focus on producing a new classification algorithm but rather an implementation of existing algorithms in the chosen case study to mitigate false positives and compare their performance.

1.9 Ethical Issues

The study used publicly available dataset from Kaggle. The dataset has been made available for educational purposes and it does not have specific personal information that can compromise security and privacy. The source of the dataset has been duly acknowledged.

The next section presents the research study layout.

1.10 Research Layout

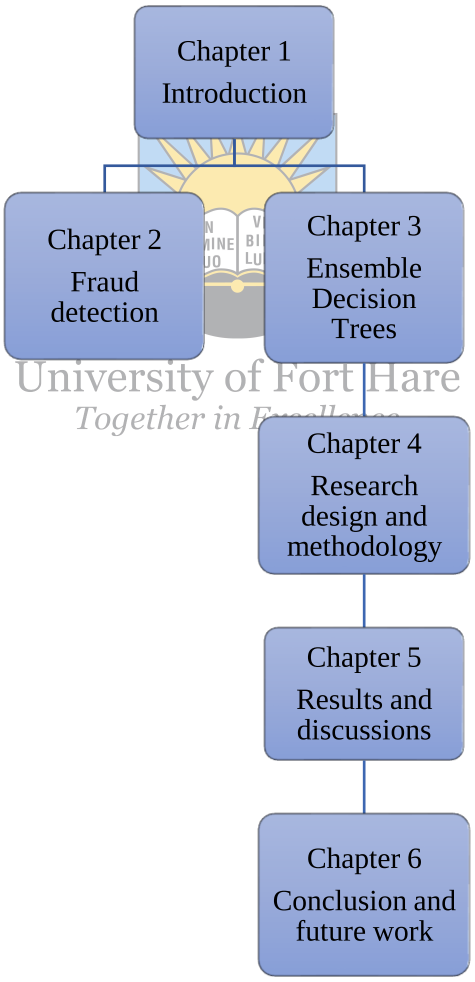


Figure 1-1: Research layout

This study has six chapters, with the first chapter being this introductory chapter.

Chapter 1: gives the introduction of the research. A background study was presented first and then a problem statement was identified. The chapter also highlighted the research questions and objectives of the study. The scope of the study is also stated and the research layout.

Chapter 2: provides an introduction to fraud and fraud detection. It also briefly discusses how machine learning has impacted the domain of fraud detection. The different types of credit card fraud were also discussed. The process of detecting fraud and the critical issues that lead to misclassifications were also discussed. Thereafter, the impact of misclassifications on businesses was highlighted.

Chapter 3: gives a detailed account of how ensemble tree classifiers perform classification tasks. The use of boosting and bagging in ensemble learning was also introduced in this chapter. The chapter also provides an understanding of the parameters involved in the modelling of ensemble tree classifiers. In-depth literature for each of the ensemble tree classifiers used in the study was provided including the underlying algorithms.

Chapter 4: provides the research methodology that was chosen for this study and why it was chosen. It also provides the research design for the experiments carried out.

Chapter 5: describes the results that were obtained from the experiments that were carried out and an evaluation of the results. The chapter also provides a discussion of the results.

Chapter 6: consolidates the research questions and the finding from the experiments to answer those questions. Each question was listed and conclusions drawn from the research were presented.

1.11 Conclusion

This chapter gave an introduction to the study. The main focus was to provide background information, identify a gap in the literature and formulate research questions and objectives to be answered by this study. The scope of the study was also stated and lastly, the structure of the study was highlighted.

The next chapter is an in-depth review of the fraud domain focusing mainly on credit card fraud.

Chapter Two

2 Fraud Detection

2.1 Introduction

This chapter introduces the concept of fraud by firstly defining fraud in section 2.1. Section 2.2 discusses the use of machine learning in detecting fraudulent transactions. section 2.3 looked at the different types of credit card fraud highlighting how they differ and how they contribute to total fraud. Section 2.4 discussed the process of detecting fraud from the time an individual initiates a transaction. Section 2.5 discussed the issues that are present when trying to detect fraud. These are critical as they often lead to misclassifications. Section 2.6 discussed how to deal with skewed data and two approaches were highlighted data level and algorithmic level approach. Section 2.7 discusses the impact of misclassifications on businesses.

2.2 Definition of Fraud

Shakya (2018) defines fraud as an "intentional crime in which a fraudster benefits himself/herself by denying a right to a victim or by obtaining financial gain". The Association of Certified Fraud Examiners (ACFE) define fraud as, "The use of one's occupation for personal enrichment through the deliberate misuse or misapplication of the employing organisation's resources or assets." ACFE classifies fraud as either internal or external. Internal fraud is when an employee misuses or commits fraud against the organisation they are working for, and external fraud is when a committed by a person who is not employed by the organisation. External fraud can also occur as a result of an individual targeting a company or also when merchants have to reverse transactions of fraudulent transactions. We can therefore clearly see that fraud involves the stealing of money by individuals in an organisation and also those who are not part of the organisation. This has a negative effect on both merchants and individuals who are defrauded. The calls for cashless societies have continued to increase and as a result, there has been an upsurge in the number of credit card users. The rise in credit card usage has not only benefited card users and merchants who can now transact from the comfort of their homes or offices but has also attracted fraudsters who seek to exploit the loopholes in the credit card system. Credit card fraud is highly lucrative to fraudsters because large amounts of money can be stolen within seconds and the fraudsters does not need to leave their home. The upsurge in credit card fraud has also led to extensive studies on how to prevent and detect fraud (Abdallah et al., 2016). These studies have led to the development of two major types of systems, Fraud Prevention Systems (FPS) and Fraud Detection Systems (FDS) as stated by Abdallah et al. (2016).

FPS act as the first in the line of defence system against computer-based fraud. Abdallah et al. (2016) state that the function of this zone is mainly to prevent or deny cyber-attacks that are used by fraudsters. This zone has computer software like firewalls that offer access control and filtering. Whereas, FDS does not prevent but used to identify and classify fraudulent transactions and raise fraud alerts. Fraudulent transactions are a very small percentage accounting for 1.12% of all transactions and hence they are classified as outliers or anomalies as they are rare and isolated occurrences (Budhram, 2016). Most FDS are trained to recognise the normal behaviour of individuals and classify anything outside the normal as fraudulent. Furthermore, Abdallah et al. (2016) argued that pattern recognition is the most popular method of detecting fraud. This is achieved by studying previous individuals spending patterns by considering factors such as time, location and amount spent to establish a pattern of their transaction behaviour, anything inconsistent with this behaviour is considered fraudulent. A combination of data mining and machine learning techniques are used to detect patterns and build FDS. These will be discussed in the next section.

2.3 Machine learning for Fraud Detection

Machine learning has brought a breakthrough in the fight against fraud. Machine learning allows computers to identify patterns without explicit programming. The greatest advantage of using machine learning is that computers can learn new patterns automatically without having to be re-programmed. There are three main techniques in machine learning and are supervised, unsupervised and semi-supervised learning (Bolton & Hand, 2001) (Jha, Montserrat, & Christopher, 2012). Firstly, supervised machine learning makes use of labelled training data sets. Labelled training data is fed into algorithms. Secondly, in unsupervised learning data sets with no labels are fed into the algorithms. Lastly, semi-supervised is a combination of both supervised and unsupervised learning. Data sets that have labels and unlabelled are used fed into the algorithm.

Most machine learning algorithms that are used to detect anomalies or fraud treat it as a classification problem, where either a transaction is classified as fraud or not a fraud. The next section will discuss the different types of credit card fraud.

2.4 Types of Credit Card Fraud

There are three main types of fraud in credit cards that is, Stolen and lost cards, counterfeit cards and card-not-present fraud. Table 2-1 gives fraud statistics as recorded in South Africa between 2018 and 2019 according to the fraud report by SABRIC. The table shows by how much, different fraud types accounted for. Figure 2-1 depicts that card-not-present fraud accounted for most of the fraud losses in South Africa between 2018 and 2019 (SABRIC, 2019).

Table 2-1: Fraud statistics according to (Chigada, 2020)

FRAUD TYPE	2018 (R)	2019 (R)	TOTAL (R)
Stolen and lost credit cards	81 497 606	78 304 617	159 802 223
Counterfeit card fraud	143 300 000	143 659 032	286 959 032
Card not present fraud (CNP)	531 900 000	528 890 000	1 060 790 000

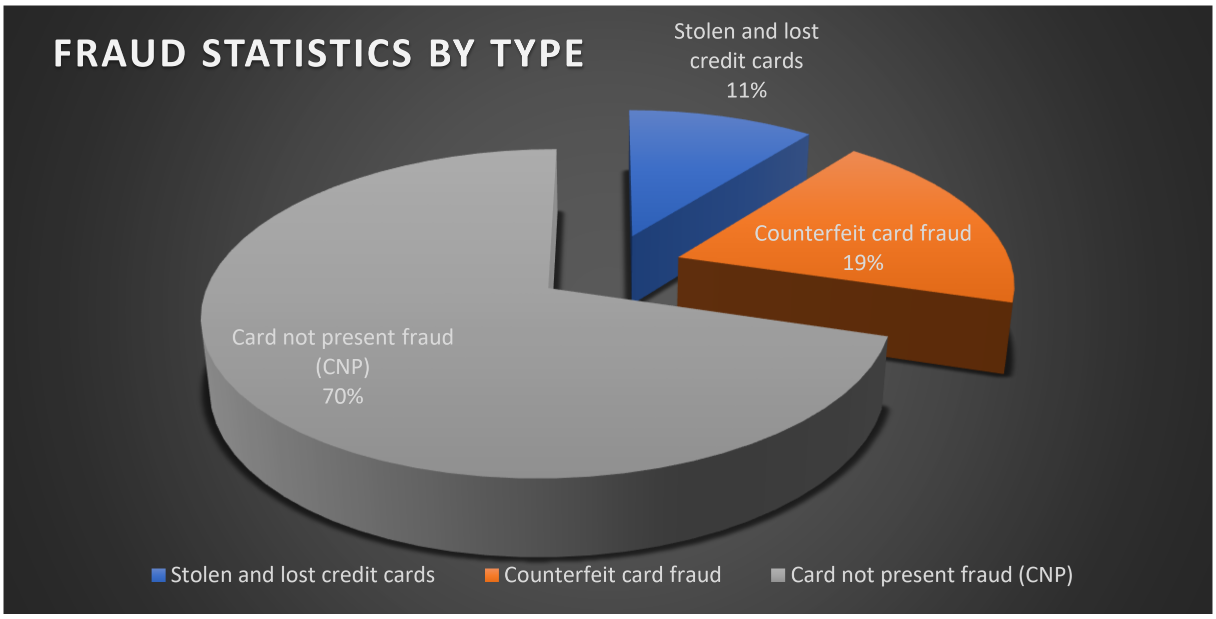


Figure 2-1: Fraud statistics according to (SABRIC, 2019)

2.4.1 **Stolen and lost credit cards**

According to (Budhram, 2016) Stolen and lost credit cards is the most popular type of fraud. This type of fraud is when an individual's credit card details are stolen as a result of a lost card or theft. These details are then used to make transactions remotely either by making a purchase on a telephone line or using the internet. The card can also be used at a point of sale to make a purchase. The fraudster would however need to know the unique PIN. Stolen or lost card fraud has become easier with the introduction of Tap-and-Go technology which does not require an individual to enter a unique PIN when they make a purchase. So once the fraudster has a credit card, they can easily make purchases.

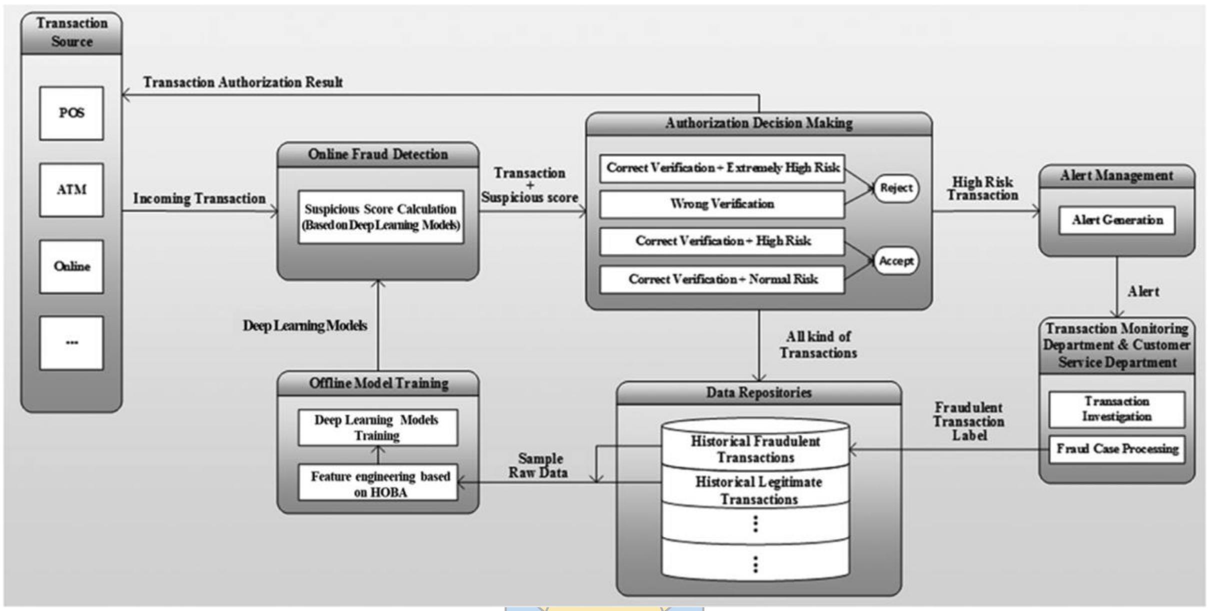
2.4.2 **Counterfeit card fraud**

A counterfeit credit card is a card that has been cloned or produced illegally from details stolen from a genuinely issued card. It is simply an imitation of a genuine card. Most fraudsters get these details by coping information contained on the magnetic strip at the back of a card using a hand-held card reader. Another way of getting this information is through the use of cloning devices which are placed at the card slot of an ATM and it will record all the details of the cards that will be used on that ATM including the PIN that will be entered for the card (Budhram, 2016).

2.4.3 **Card-not-present fraud (CNP)**

Card-Not-Present fraud is when a fraudulent transaction is performed in the absence of the card and card owner or holder. This is usually done online, telephonically or via the internet. Budhram (2016) outlines two major challenges with CNP transactions. The first is that merchants are not able to examine the card to establish the presence of all security features. The second is that the person performing the transaction will not be required to sign or enter a PIN to further check if they are the authentic cardholder (Budhram, 2016). Fraudsters capitalise on these shortfalls as they can successfully transact fraudulently without presenting physical cards or being present thereby averting fear of being arrested. CNP transactions can be performed from public facilities such as internet café, public telephones and using stollen phones which make it hard to trace the fraudsters. According to SABRIC, CNP fraud is increasing in South Africa and accounted for up to 1,06 billion Rands which is 70% of the three types of fraud discussed by the researcher between 2018 and 2019 (SABRIC,2019). This is an indication that CNP fraud is now the most lucrative type of Fraud.

2.5 Fraud detection process



Transaction Source
POS
ATM
Online
...
Transaction Authorization Result
Incoming Transaction
Online Fraud Detection
Suspicious Score Calculation (Based on Deep Learning Models)
Transaction + Suspicious score
Authorization Decision Making
Correct Verification + Extremely High Risk
Wrong Verification
Correct Verification + High Risk
Correct Verification + Normal Risk
Reject
Accept
High Risk Transaction
Alert Management
Alert Generation
Alert
Transaction Monitoring Department & Customer Service Department
Transaction Investigation
Fraud Case Processing
Deep Learning Models
Offline Model Training
Deep Learning Models Training
Feature engineering based on HOBA
Sample Raw Data
All kind of Transactions
Data Repositories
Historical Fraudulent Transactions
Historical Legitimate Transactions
Fraudulent Transaction Label

Source: Adapted from X. Zhang, Y. Han and W. Xu et al.(2019), HOBA: A novel feature engineering methodology for credit card fraud detection with a deep learning architecture, Information Sciences, https://doi.org/10.1016/j.ins.2019.05.023

Figure 2-2: Fraud detection process (X. Zhang, 2019)

The process of detecting fraud is depicted in figure 2-2. An individual must first initiate a transaction via a POS, ATM or online/telephonically. After a transaction has been initiated, they will be asked to enter a PIN or security details on the credit card which will then be verified by the computer. This will act as the first line of defence as no further action cannot be taken if the fraudster does not have the required security details. The merchant or bank will also have implemented a fraud detection and prevention system. This will assess the transaction requested to identify any anomalies and flag it as fraud or non-fraud. Some of the factors that can cause a transaction to be flagged include time, location and amount requested. The next section looked at issues in detecting fraud.

2.5.1 False alerts in fraud detection models

While there has been much research on fraud detection models, few researchers have considered the effects of false positives. This might be because more focus is on combatting fraud and building more efficient models. A false positive is a genuine transaction that is classified as fraudulent (Visa et al., 2011). Modi (2017, p. 103) states that the number of people who conduct transactions on the internet using credit cards is increasing thereby increasing the number of fraudulent transactions. Modi (2017) further stress that there are three major challenges linked with credit card fraud detection, that is the availability of live banking data, overlapping data and data imbalance. This makes it difficult to develop a holistic model that is guaranteed to reduce false positives when subjected to real-world data.

Budhram (2016) observes that newer technologies are leveraged by fraudsters to devise new ways of committing fraud. As a result, fraudsters are developing methods that closely mimic individual transaction patterns which increases the chance of incorrect classification of transactions. This creates a need to constantly review existing models to counter the increasing threat from fraudsters creating new techniques. Onukwugha (2018) proposed a model that uses multi-agents to detect fraud and seek to lessen the number of false positives instead of the two-staged model currently being used. The rate of false positives remains a concern to both individuals and merchants and this study aims to find ways to reduce false positives. This is supported by Sadgali et al. (2018) who emphasised the need for further research to minimize false alarms by deploying dynamic systems that learn the behaviours of users to better classify fraudulent transactions.

2.5.2 Anomaly detection using data mining tools

2.5.2.1 Quantile-based methods

Scholkopf et al. (1999) provides a qualitative analysis and numerical performance of the Support Vector Method (SVM) algorithm. The authors proposed an algorithm that calculates a binary function to capture areas in input space focusing on regions where the probability density function is not a zero. The proposed algorithm in the article builds on top of the Support Vector (SV) algorithm so that it also accounts for unlabelled data points. The argument from Scholkopf et al. (1999) is that the SV algorithm did not pay much attention to outliers. Fraud is an example of outlier detection and a method that purely seeks to identify outliers is more convenient. In any given dataset if a model predicts that all transactions are non-fraud it will have over 90% accuracy as the rate of fraud constitutes less than 2% of all transactions. Tax & Duin (2004) also proposed a quantile-based method called the Support Vector Data Description (SVDD). Tax & Duin state that, “the boundary of a dataset can be used to detect novel data or outliers”. Tax & Duin method obtains spherical shaped boundary around data with the same flexibility and minimised volume to reduce chances of outliers fitting in the boundary. Tax & Duin experiments using the Gaussian kernel produced comparable results to those of the SVM by Scholkopf et al. (1999). Tax & Duin, however, admit that for small error targets the SVDD performance reduces. As highlighted earlier outliers constitute a small fraction of transactions and considering that the SVDD performs less for small error targets the rate of false positives remains significantly high in the SVDD Model.

2.5.2.2 Neighbour-based methods

Kriegel et al. (2008) proposed a novel approach called angle-based outlier detection (ABOD) in data space that is multi-dimensional. ABOD is a neighbour-based model used in data mining. The experiments were done using Java version 1.5 and on Dell Precision 690 workstation. (Kriegel, Schubert, & Zimek, 2008) Kriegel et al. (2008) in the experiment used the Caltech 101 dataset to extract the top ten uncommon 2D shapes in the dataset. Kriegel et al. (2008) argue that existing approaches leverage distance in the full-dimension Euclidean data space but will eventually suffer due to the issue of dimensionality. Breunig et al. (2000) also proposed a neighbour-based outlier detection technique called the Local outlier factor (LFO) (Breunig, Kriegel, Raymond, & Sander, 2000). Breunig et al. (2000) argue that for every object there must be a level or degree of it being an outlier. The term local arises from the fact that the degree is dependent on how isolated the object is from its surrounding neighbours. Breunig et al. (2000) further argue that an outlier must be seen as a degree of isolation from surrounding neighbours in contrast to most research that describes an outlier as a binary property.

2.5.2.3 Density-based approach

Kim & Scott (2012) proposed a density-based approach to anomaly detection that is robust from a contaminated training data set. The method makes use of the traditional Kernel Density Estimator (KDE) applied in aggregation with the M-estimation which resulted in the Robust Kernel Density Estimator (RKDE). The rationale is that the sample mean is sensitive to outliers (Kim & Scott, 2012). The RKDE is implemented using the iterative re-weighted least square (IRWLS) algorithm to archive the highest effectiveness. Kim & Scott argue that despite the standard KDE being sensitive to outliers, it tends to overestimate the density in low-density regions. Glodek et al. (2013) also proposed a density-based approach to classification using an Ensemble Gaussian Mixture Model (EGMM). The approach is based on the use of the ensemble training method to estimate density by using the Gaussian Model Mixture (GMM). The EGMM was theoretically investigated and experiments were carried out to evaluate how the model performed in classification and non-Gaussian distribution (Glodek, Schels, &

Schwenker, 2013). Glodek et al. (2013) argue that the EGMM is more accurate and robust compared to contending approaches in classification.

2.5.2.4 Projection-based methods

Liu et al. (2008) points out that most existing model-based approaches or algorithms for anomaly detection create normal instances and then labels anything else that does not fit in as an anomaly. Liu et al. (2008) however proposed a different approach that focuses on identifying the outliers or anomalies primarily. They argue that the use of isolation using the iForest algorithm allows for sub-sampling to extends not achieved by existing methods at the time of writing. Liu et al. (2008) did experiments to show that outliers or anomalies have short path lengths compared to normal points indicating they are more prone to isolation under random partitioning. iForest has great efficiency when dealing with large datasets, multidimensional data and works well with data that has no anomalies Liu et al. (2008). The characteristics of the isolation forest make it suitable for fraud detection as it has been shown to work well on data that has little to no anomalies. On the other hand, Penvy (2008) argued that in supervised learning a collection of weak classifiers can give similar results and sometimes even outperform state of the art methods. However, when it comes to unsupervised learning for example anomaly detection this concept is still yet to be proved. Penvy proposed a lightweight anomaly detector suitable for online data capable of detecting causes of anomalies in datasets called the Lightweight online detector of anomalies (LODA). Penvy conducted experiments using mainly network datasets and concluded a single model to solve all kinds of classification problems existed but the LODA despite giving comparable results to other methods could also explain reasons for anomalies and account for missing data.

2.6 Issues in fraud detection systems

FDS has brought a breakthrough in the fight against credit card fraud. But like any other technology, there are also critical issues that might arise. FDS are no exception and have been faced with several critical issues which have led to the need for further research in the field to address these critical issues. One of the main critical issues is False Positives. False positives are when legitimate transactions are classified as fraudulent as an error from the FDS. Fraud detection can be thought of as a Cost-Sensitive problem as the cost of loss resulting from false positives and that of false negatives is different (Alejandro Correa Bahnsen∗, 2016). A false negative is when a fraudulent transaction is classified as a legitimate transition. This can be more costly than what the FDS is trying to prevent.

The following critical issues will be discussed: (i) concept drift; (ii) skewed data; (iii) data volume and dimensionality; (iv) real-time detection.

2.6.1 Concept drift

Concept drift is a feature or variable that exists in a feature space and changes with time and the change can not be estimated or predetermined and is therefore unforeseen (Dal Pozzolo Andrea, 2015). This happens when a model has learnt patterns of users and that of fraudsters but then by nature fraudsters are always coming up with new ways so their behaviour changes frequently. Features that can change include amount, location and time. Models can learn these behavioural changes but because they are dynamic this doesn’t happen at the same speed as the change in behaviour.

2.6.2 Skewed Data

Skewed data is another critical issue faced by FDS. According to Dal Pozzolo Andrea (2015), skewed data refers to how data is not evenly distributed. This phenomenon is also called imbalanced data. Sorournejad et al. (2016) highlight that skewed datasets negatively impact the detection of fraudulent transactions. If a model is run on skewed data the output can ignore data points that are few or considered as minority class resulting in reduced performance (Samaneh Sorournejad, 2016). Real-world credit card datasets have approximately 98% legitimate transactions and 2% as illegitimate transactions (Zareapoora & Shamsolmoalia, 2015), (Lu & Chunhua, 2011) and (Juszczak, Adams, Hand, Whitrow, & Weston, 2008).

2.6.3 Data Volume and Dimensionality

Many transactions are being performed every hour and as a result banking datasets are large. This is referred to as Big Data. Big Data explains the Velocity, Variety and Volume of data that is being generated continually. E-commerce datasets grow exponentially every hour and the data has many features which makes it a big data problem (Chan, Fan, Prodromidis, & Stolfo, 1999). Feature extraction is very complex given the volume and variety of the data.

The Principal Component Analysis (PCA) can be applied to data to reduce data dimensionality. By reducing data dimensions or features computation speed increases and feature extraction becomes easy.

2.6.4 **Real-time detection**

Credit card fraud needs to be detected in real-time. This is because fraud needs to be stopped in real-time or detected as early as possible so that appropriate action can be taken immediately. Real-time detection is important in Fraud detection and prevention as it is cost-effective to stop fraud before it occurs than to detect fraud that has already been committed. The cost of investigating and following up on the suspected fraud is an addition to the money already lost due to the fraud. Real-time detection requires high computational power and time complexity that is relatively small as credit card users want their transactions to be completed in a relatively small space of time.

2.6.5 **Availability of Training Data**

Most banks make use of centralised learning for training FDS, this is due to privacy concerns. Banks are not willing to share clients’ information as it is treated with high confidentiality. As a result, the availability of training datasets is limited to the single bank's database. Research companies and academic institutions are limited to few banking datasets made available (Bolton & Hand, 2001). This results in less effective and reliable models as more data is required to train the model more effectively (Jha, Montserrat, & Christopher, 2012), (Lu & Chunhua, 2011) and (Ngai, Hu, Wong, Chen, & Sun, 2011).

2.7 Dealing with Skewed data

Several methods have been proposed to solve the issue of skewed data. Two main approaches to deal with skewed data can be applied, the data level approach which pre-processes the data before it is fed into the classifier and the algorithmic level approach which doesn’t pre-process the data but rather modifies the model (Krawczyk, 2016).

2.7.1 **Data level Approaches**

The data level approach makes use of either of the following sampling techniques, that is, either undersampling or oversampling.

2.7.1.1 Undersampling

Undersampling is when the number of observations is reduced by removing some observations until the data becomes balanced thereby solving the data imbalance. Chawla et al. (2004) further break undersampling into Random undersampling (RUS) and Direct undersampling

(DUS). RUS means that data is removed from the majority class randomly and DUS means that predefined and known data is removed from the majority class. Figure 2-3 depicts how training data is undersampled to make it match with the test data.

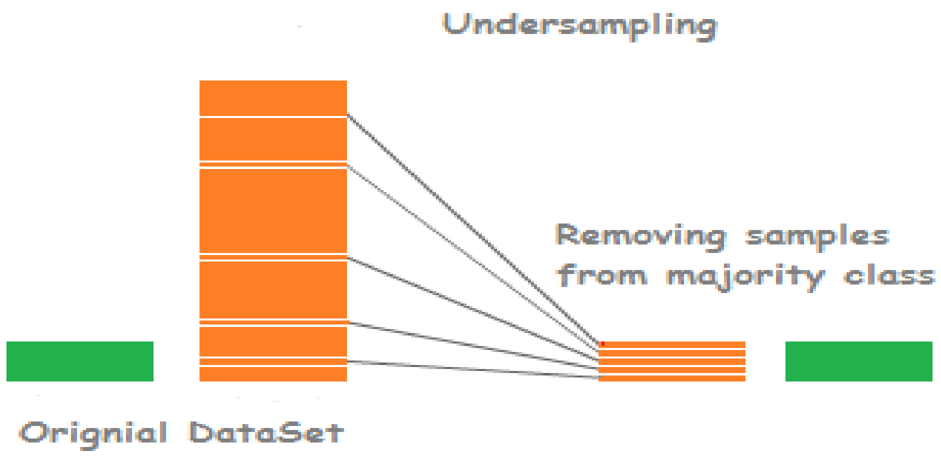


Figure 2-3: Under sampling (Roweida Mohammed, 2019)

2.7.1.2 *Oversampling*

The oversampling technique upscales the minority class by creating synthetic data points in the feature space to get more observations of that minority class (Chawla, Bowyer, Hall, & Philip, 2002). The additional data points must be within the proximity of the observed points in the minority class. The problem that may result from oversampling is overfitting the model as the points are added to the minority class without considering the majority class.

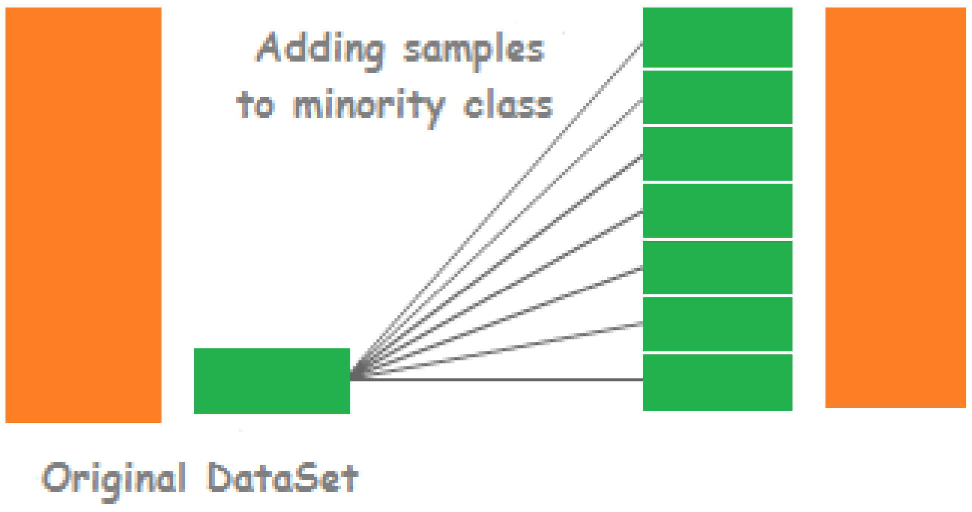


Figure 2-4: Oversampling (Roweida Mohammed, 2019)

2.7.1.3 Synthetic Minority Oversampling Technique

The widely used oversampling technique is the Synthetic Minority Oversampling Technique (SMOTE) (Aisha Abdallah, 2016; Chawla, Bowyer, Hall, & Philip, 2002). The SMOTE formulates new synthetic instances of the minority class by inserting features between close neighbours. This is a preferred method compared to random oversampling as it reduces the problem of overfitting the training data (Shakya, 2018).

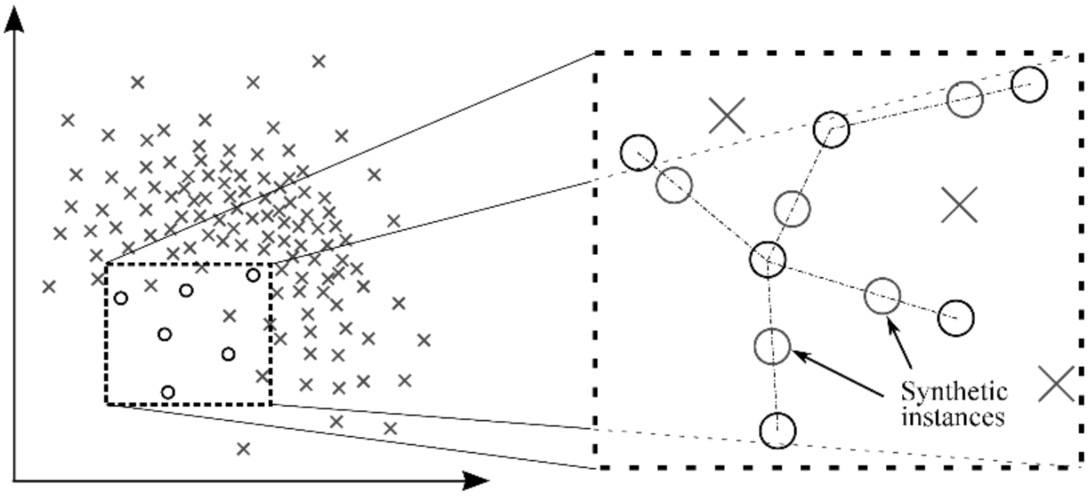


Figure 2-5: Generation of new minority classes (Shakya, 2018)

2.7.2 **Algorithmic level approaches**

Algorithmic level approaches act as tools that modify the model by either making the model adapt to handling minority classes or applying cost-sensitive learning. Data level approaches are favoured in contrast to algorithmic level approaches to solve data imbalance as they are easy to implement and do not increase computational complexity.

2.7.2.1 Bagging

Bagging is an abbreviation for Bootstrap Aggregation. Bagging makes use of bootstrapping that formulates new training data sets using only a sample of the original training dataset (Jismy & Kesavaraj, 2019). This is achieved through the process of replacement. The new training datasets are called bootstrap training samples. Each model is trained using these bootstrap training samples and a final prediction can be achieved by averaging in regression models and voting in classification problems. Random Forests implement Bagging using decision trees. Figure 2-6 shows the bagging procedure, starting with how imbalanced data is sub-sampled into bootstraps with no replacements.

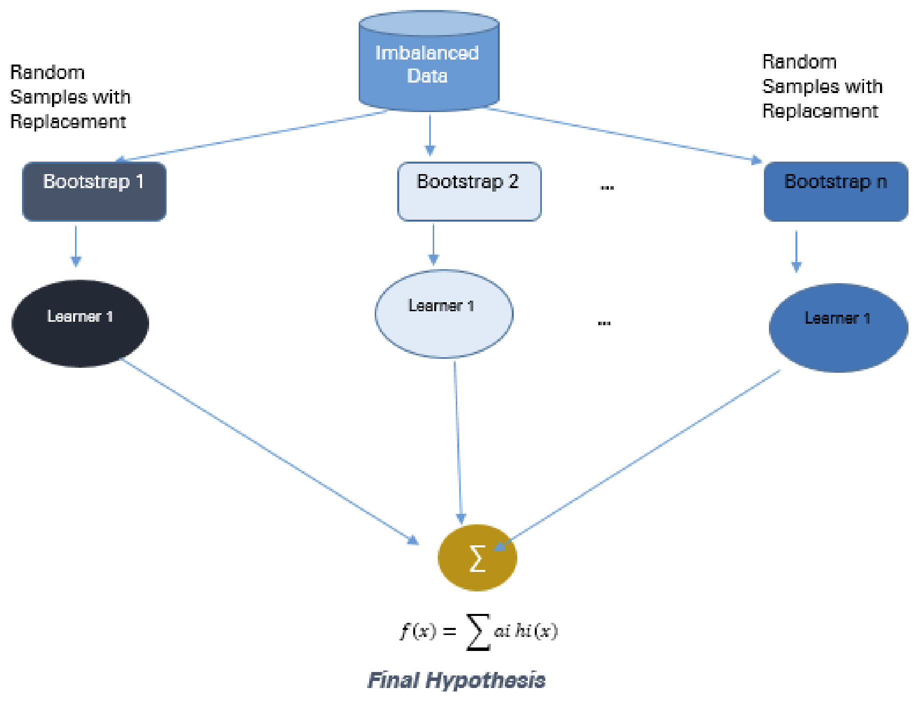


Figure 2-6: Bagging procedure (Jismy & Kesavaraj, 2019; Guest Blog, 2018)

2.7.2.2 *Boosting*

Boosting involves combining weak learners also called base learners that will work together to produce a stronger learner. Unlike in Bagging where models are trained all at the same time and then aggregated at the end, boosting trains the weak learners in a sequence such that the succeeding learner will try and correct or rectify data that was misclassified. Figure 2-7 shows how weak learners are trained with results of one learner feeding into the other so that it tries to correct the misclassifications of the previous learner.

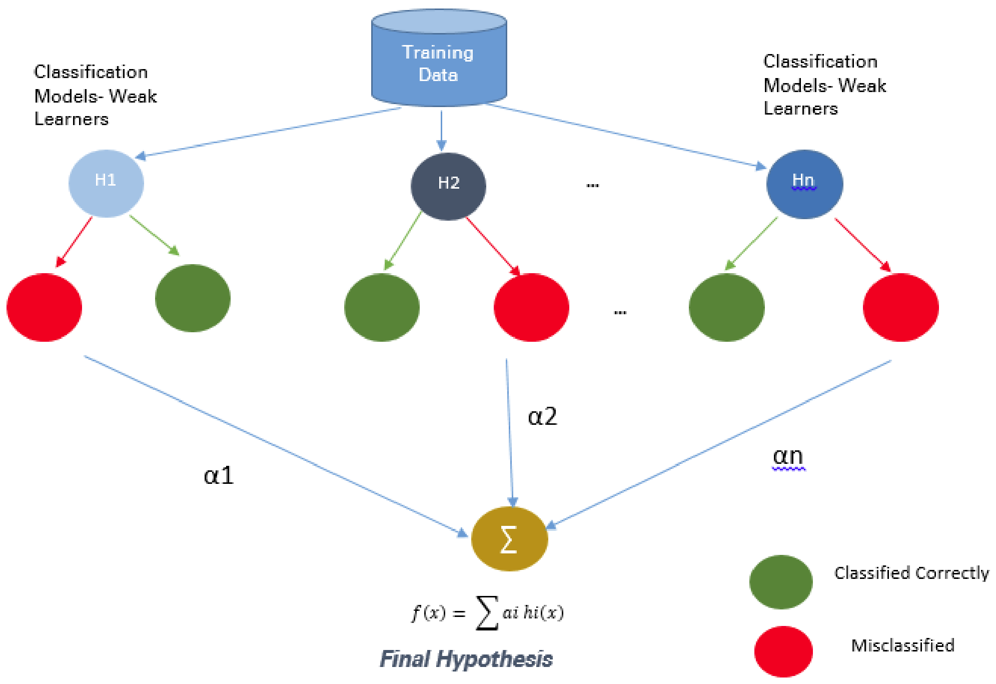


Figure 2-7: Boosting (Shakya, 2018; Guest Blog, 2018)

Examples include ADA Boost, Gradient Boost and XGBoost. Boosting takes a linear combination of predictions $p_i(X)$ by decision tree classifiers (Breiman, 2020):

$$f(x) = \sum_i a_i p_i(X) \tag{2.1}$$

Where

$$p_i(X) = \begin{cases} 1, & if\ the\ ith\ tree\ predicts\ fraud \\ -1, & otherwise \end{cases} \tag{2.2}$$

And predict

$$Y = \begin{cases} 1, & if\ f(x) \geq 0 \\ -1, & if\ f(x) < 0 \end{cases} \tag{2.3}$$

2.7.3 **Federated Learning**

Most banks make use of centralised learning for training FDS, this is due to privacy concerns. Banks are not willing to share clients' information as it is confidential. As a result, the availability of training datasets is limited to the single bank's database. The federated approach implements a decentralised approach where banks can collaborate and build more robust FDS. McMahan et al. (2016) introduced the term federated learning in 2016 (McMahan, Eider, Daniel, Seth, & Blaise, February 2016). They defined federated learning as several devices collaborating under the supervision of a central server to train a model. This collaboration does not share information but rather parameters that are required by the model only. Kairouz et. al (2019), emphasised the desire by organizations to collaborate to develop more robust and decisive models in addition to edge devices that had been highlighted by McMahan et al. (2016). McMahan grouped these two types of federated learning as cross-device and cross-silo, where cross-silo is organisations collaborating and cross-devices is between devices. This collaboration between banks can be a breakthrough as more training data from different banks will result in better Classifiers, as shown by the simulation done by Kairouz et. al (2019). Figure 2.6-6 depicts a model of federated averaging as proposed by Axelsson et al. (2020).

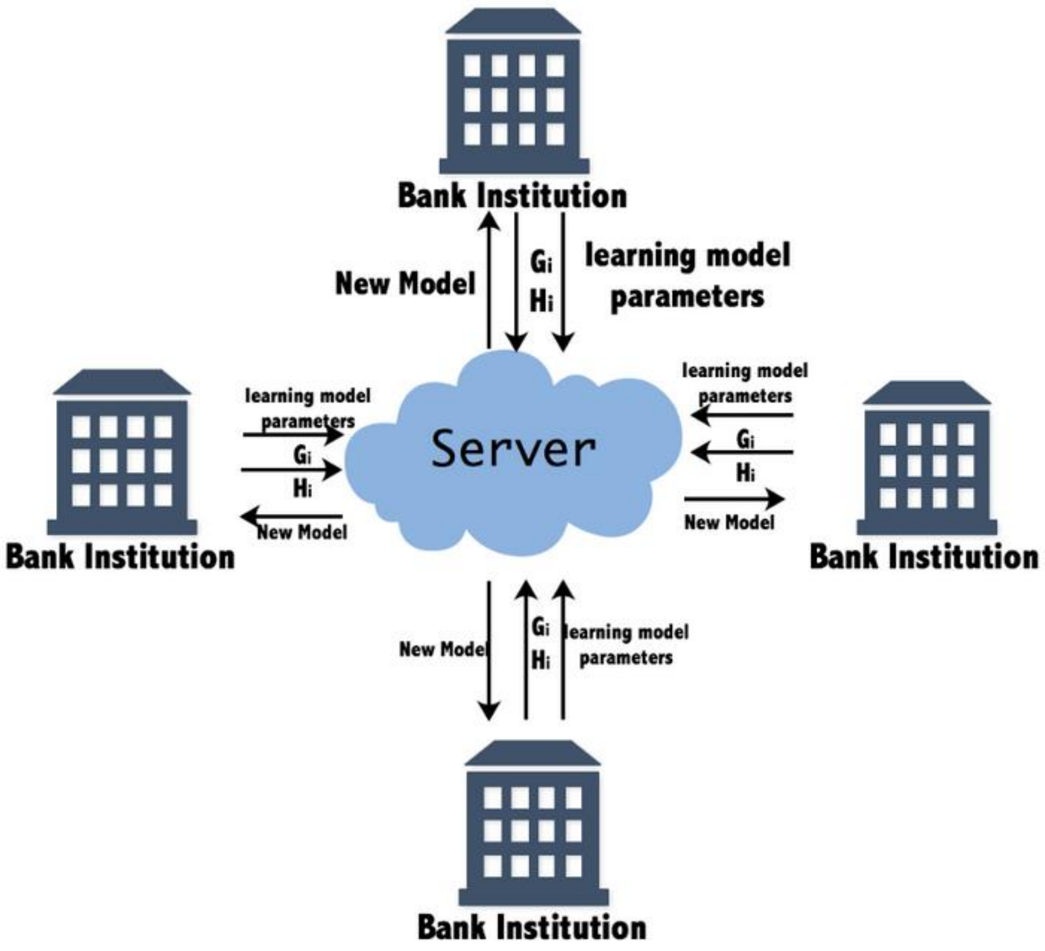


Figure 2-8: Training procedure in a Federated Learning framework (Axelsson, Jansson, & Mans, 2020)

Figure 2-8 shows how 4 banks sharing data through a central repository. The data shared will hide sensitive information and only share data that is required for training models. This will allow for more data from different banks to be available for training models. This will lead to the development of more robust models without compromising privacy and security.

2.8 Analysis of impact of false positives on businesses

According to the business insider, businesses in the United States of America were set to experience losses of up to $8.6 billion due to legitimate transactions that are falsely declined (Bakker Evan, 2016). The actual fraud that will be prevented will save businesses in the USA amounts up to $6 billion. Losses experienced from false positives are more than $2 billion compared to the actual fraud prevented by Fraud Detection and Prevention systems. As a result, false-positive undermine the effort being put in trying to combat fraud.

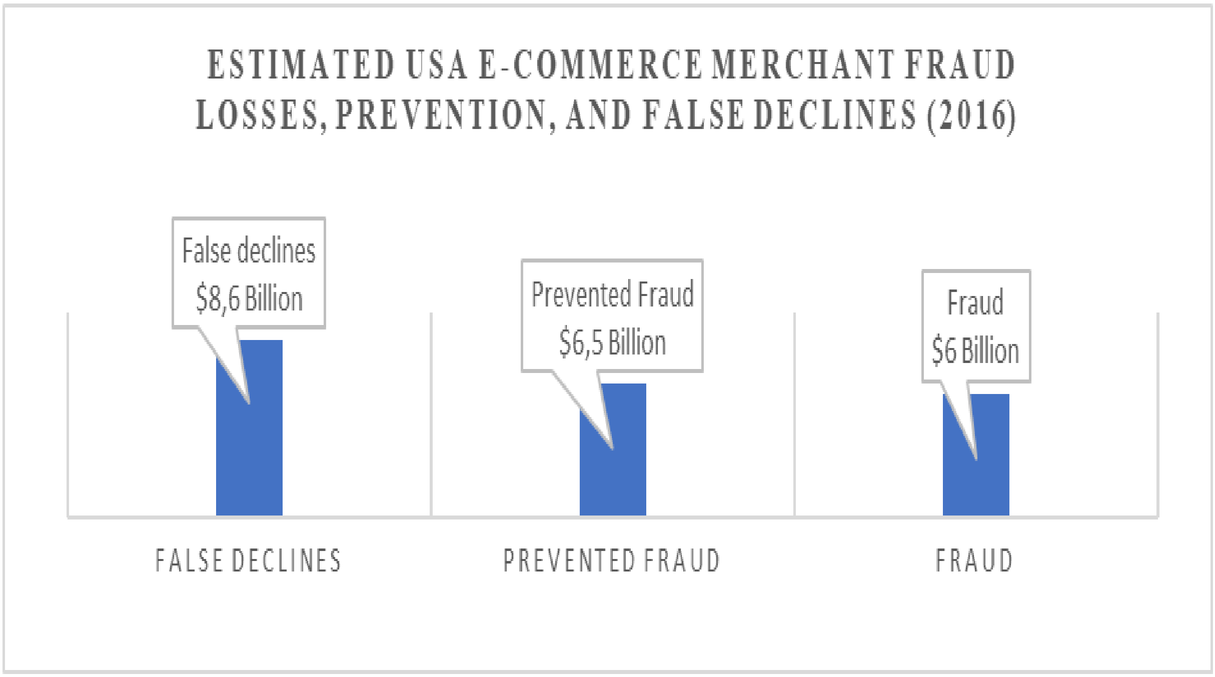


Figure 2-9: Fraud losses, prevention and false decline in the USA in 2016 (Bakker Evan, 2016)

Fraud detection and prevention systems that are designed to protect a business from losses arising from fraud might end up costing the business more than it is trying to save. The impact of false declines cannot be overlooked as it goes beyond just immediate revenue loss. False declines can affect a business in the following four critical ways: Immediate revenue loss, lost lifetime customer value, wasted acquisition cost and damage to the business brand. Figure 2-9 shows that the cost of fraud is less than that of false declines as per study done in the USA.

2.8.1 **Immediate Revenue Loss**

Around 60% of businesses that monitor false declines on their transactions have recorded a false decline rate of up to 5% (Bakker, 2016). According to a report from Business Insider (2016), a business that has an annual turnover of $50 million can lose up to $2.5 million of the annual turnover due to false declines.

2.8.2 **Lost Customer Lifetime Value**

Lifetime customer value is the overall profit a business would have projected to earn from all future purchases by a customer (Bakker Evan, 2016). A customer who has experienced purchases falsely declined as fraud may never buy from that business again. As a result, that business would have lost all the potential profit from purchases that were going to be made by that customer.

2.8.3 **Wasted Acquisition Cost**

Acquisition cost is the money that a business spends in trying to convince an individual to purchase their business (Bakker Evan, 2016). This includes all the money, time and effort spent on advertisements and targeted research to strategically meet customer needs. For example, if it costs R1000 to acquire a customer and the customer eventually makes a purchase experiences a false decline, the business will have lost the acquisition cost plus the expected revenue from the immediate sale.

2.8.4 **Brand Damage**

False declines also affect the business image. False declines can be frustrating and embarrassing to the customer. Social media and word-of-mouth are powerful tools in marketing, advertising and building brand image. A customer can write negative comments on social media platforms based on their frustration from false declines and that will negatively impact the business. Word-of-mouth refers to when people tell other people about a business. A customer who has experienced false declines from a business will likely tell other people about it, unfortunately, these could be existing customers or potential customers who might never buy from that business again.

2.9 Conclusion

The chapter introduced fraud broadly and then credit card fraud. After that, we looked at how machine learning had been used in detecting fraud   We also looked at the process of detecting and preventing fraud. The critical issues that lead to misclassification were also discussed. Lastly, we discussed how misclassifications affect businesses.

The literature discussed in this chapter answered the first research question. Critical factors that lead to misclassifications and the impact of misclassifications were explored in detail.

The next chapter discusses the concept of decision trees and how they can be used to detect credit card fraud.

Chapter Three

3 Ensemble Decision Trees

3.1 Introduction

This chapter looks at the theory behind decision trees. Section 3.2 discusses decision trees and also the two parameters that are used for selecting the best split for every node, these are Gini and entropy. Section 3.3 looked at ensemble tree classifiers and the underlying algorithms. The ensemble classifiers discussed include the random forest, Adaboost, gradient boost and Extra trees. Section 3.4 looked at the justification for algorithms that were used in this study. We concluded the chapter by giving a conclusion section which consolidated this chapter and introduced the next chapter.

3.2 Decision Trees

A decision tree is a supervised machine learning algorithm where data is split into a tree-like structure based on a chosen node or specific parameter (Jehad Ali, 2012). Decision trees are an ensemble learning technique as they make use of weak classifiers combined to produce a stronger classifier. Decision trees are made up of root nodes, edges and leaf nodes. The root node is the assumed root of all the data in a given feature space. Edges or branches are the corresponding result of a test and also act as a connection to the next node or leaf. The leaf node also referred to as terminal nodes give the outcome or prediction.

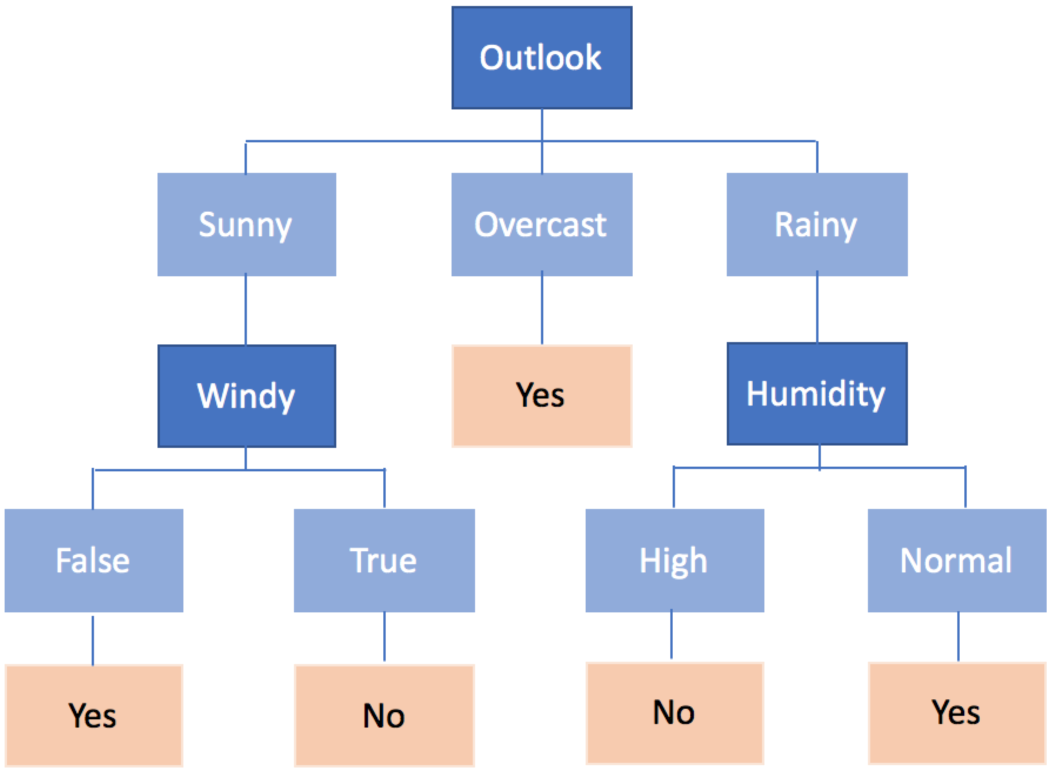


Figure 3-1: Decision tree example (Shakya, 2018; Liberman, Jan 2017)

Figure 3-1 shows a decision tree for the weather outlook of a particular day. The goal is to determine if sport can be played on that day based on the weather. There are three possible conditions, sunny, overcast and rainy. Outlook is the root node and the possible outcomes are the leaf nodes of the tree. To further split nodes there are different criteria that can be used and the next section will look at these.

3.2.1 **Entropy**

Each time a node is split entropy is used. Entropy is simply the measure of disorder in a given dataset. The purpose of entropy is to control how a decision tree is going to split the data. Entropy ranges from 0 to 1. The smaller the entropy the smaller the level of uncertainty and the higher the entropy the higher the level of uncertainty. Equation 3.1 shows how entropy is calculated.

$$H(S) = -Probability\ of\ \log_2(p+) - -probability\ of\ \log_2(p-) \qquad (3.1)$$

Where:

(p+) = % of positive class

(p-) = % of negative class

Figure 3-2 clearly shows that the maximum impurity can be attained when the division of values in a dataset are equal. Figure 3-2 depicts the relationship between entropy and probability.

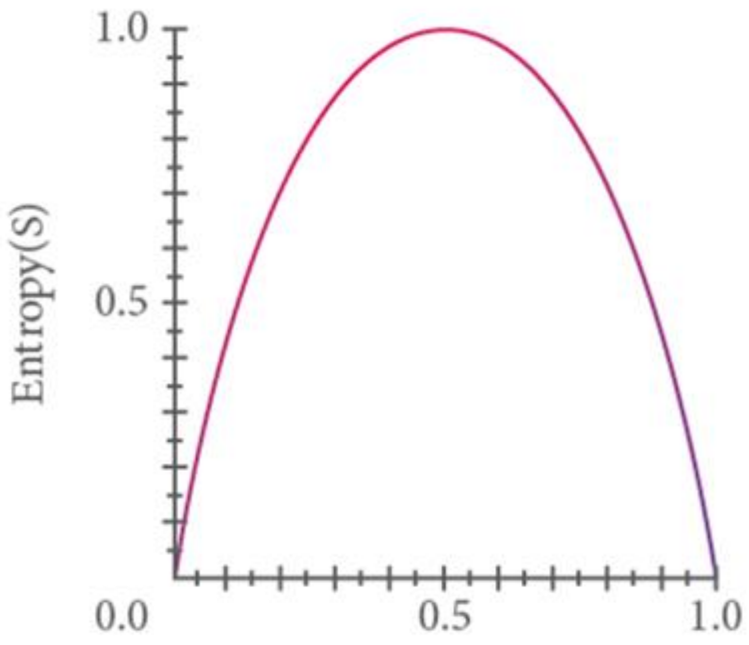


Figure 3-2: Relationship between entropy and probability (Kerecha, 2021)

3.2.2 Gini

The Gini index performs the same function as entropy but differs slightly. The Gini index measures the probability that a randomly selected feature in a dataset will be incorrectly classified. The equation 3.2 for calculating the Gini index as stated by Kerecha (2021):

$$Gini = \sum_{I=1}^{K} P(i)^2 \tag{3.2}$$

The Gini index makes use of larger partitions which makes it computationally intensive (Kerecha, 2021). The Gini index is maximum when the data is evenly balanced.

3.3 Ensemble Approach

This approach deals with highly unbalanced class distributions at the algorithm level. This is achieved by combining different classifiers to solve a classification problem and at the end, they will vote for the best solution. There are two major approaches for the ensemble approach, that is, Boosting and Bagging as discussed in sections 2.6.2.1 and 2.6.2.2. In the next section, we will look at algorithms that make use of bagging and boosting using decision trees.

3.3.1 **Random Forests**

A random forest is a group of decision trees combined to solve a problem. Each decision tree computes a prediction and the trees will vote at the end and the majority votes will be the final prediction. Each tree comprises nodes arranged in hierarchical order with information flowing from top to bottom. At each node entropy and information gain will be calculated to determine the best feature to split the data. In random forest decision trees have access to different random sub-samples of a feature set A_i which has *d* features chosen from all features A_1, A_2………A_d. Random forests use bagging and are examples of ensemble methods that combine predictions of weak classifiers.

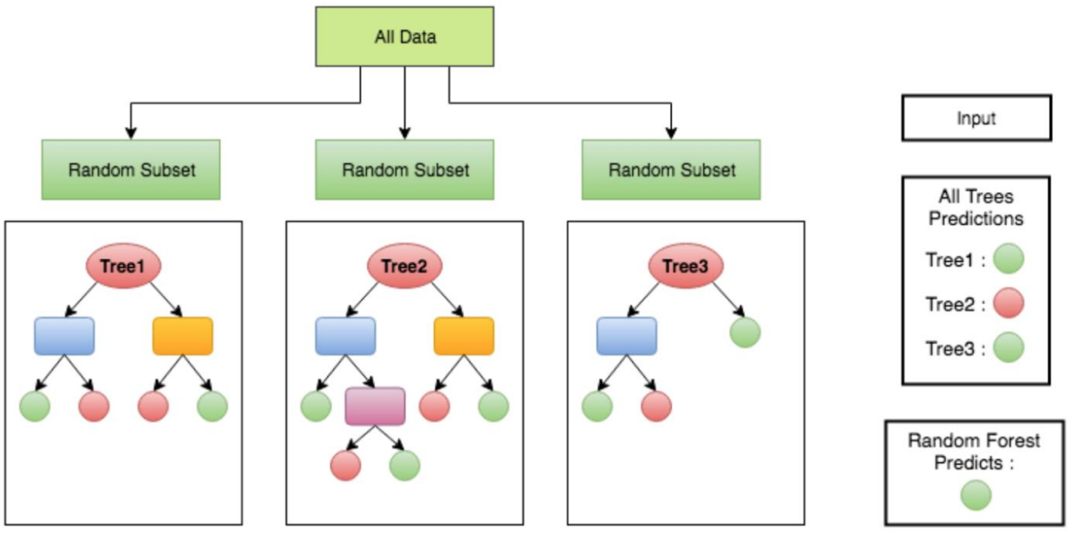


Figure 3-3: Random Forest example (Shakya, 2018; Liberman, Jan 2017)

Figure 3-3 depicts how random forests derives their outcomes. All the data present is first split into random subsets, and these random subsets were to construct random trees. Each random tree will process the data it has and come up with a prediction. After all trees have outcomes, they go through a voting process and the majority votes wins. In the case above the red dot had one vote and the green dot had two votes and the overall prediction will be green as it has the majority vote.

3.3.2 **AdaBoost**

Adaboost stands for Adaptive boosting, which is an ensemble machine learning algorithm. Boosting as described in chapter 2.6.2.2 combines weak learners in a sequence using the output of one leaner as the input of the other. Adaboost build *n* models during the training phase. The incorrectly classified outputs from the first model were given a priority and only these are sent to the next model. In this case, the model can be a decision tree.

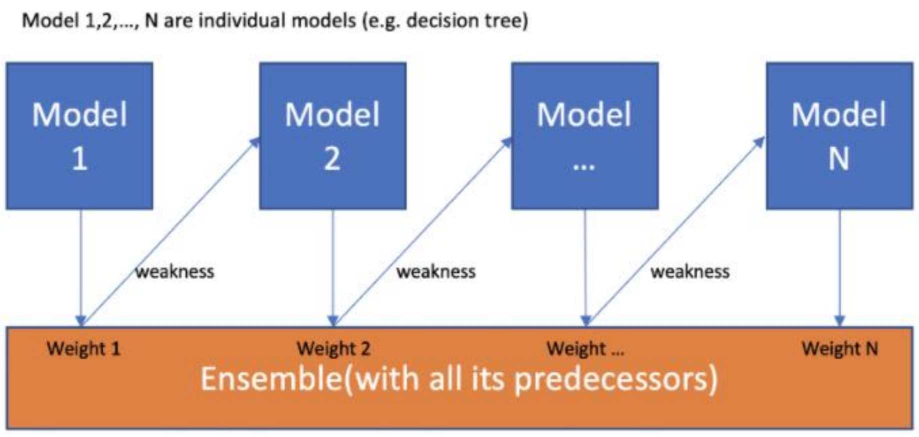

Model 1,2,..., N are individual models (e.g. decision tree)
Model 1
Model 2
Model ...
Model N
weakness
weakness
weakness
Weight 1
Weight 2
Weight ...
Weight N
Ensemble(with all its predecessors)

Figure 3-4: Adaboost training (Team Great Learning, 2020)

In figure 3-4, it can be seen that errors from the first model are recorded and passed on to the next model as input. This iteration is repeated until the ideal results are produced at the n^{th} model. Wrongly classified records are passed as weaknesses.

Unlike in Random Forests where the decision trees have no fixed depth, Adaboost has a fixed depth. Adaboost creates one node with two leaves, called a stump as shown in figure 3-5 (Team Great Learning, 2020):

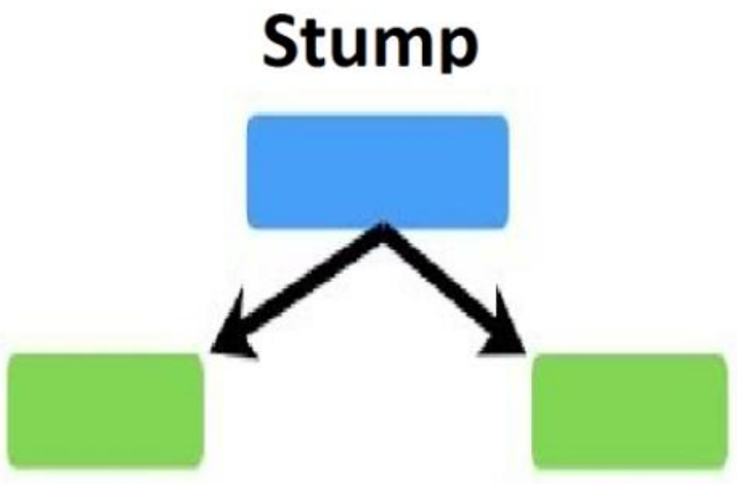


Figure 3-5: Adaboost stump (Team Great Learning, 2020)

The stump in the figure 3-5 has only two leaves. The stump is a weak learner. For example, let’s say we have a dataset with three features the Adaboost algorithm will create three stumps and from these stumps, it will make decision trees. From the three stumps, only one will be selected after calculating the entropy and Gini. The stump that has the lowest value becomes the first learner. For example, in a data set with features f1……. f3, and f1 has the lowest Gini or Entropy then f1 will be the base model as shown in figure 3-6.

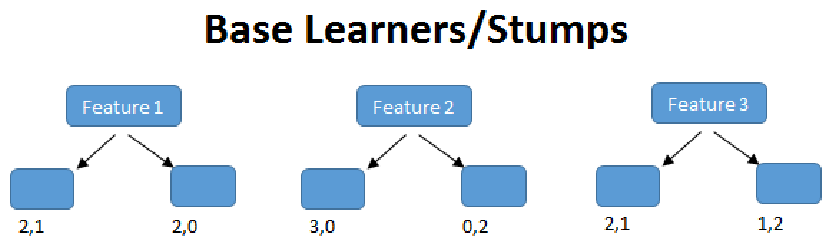


Figure 3-6: Stumps with feature 1 as a base learner (Team Great Learning, 2020)

The Adaboost algorithm will calculate total error (TE) which is the sum of the weight of all errors in data that has been misclassified. The total area will be used to calculate the performance of the stump using the formula (Team Great Learning, 2020):

$$Performance\ of\ Stump = \frac{1}{2}\ln\frac{(1-TE)}{TE} \tag{3.3}$$

After the performance has been calculated the weights of the misclassified records will be updated using the formula (Team Great Learning, 2020):

$$\text{New Sample Weight} = \text{Sample Weight} * \text{e\^{}(Performance)} \tag{3.4}$$

This is repeated until the sum of all the weights is 1.

3.3.3 Gradient Boost

According to Lu & Mazumder (2018), “Gradient Boosting Machine (GBM) is a powerful supervised learning algorithm that combines multiple weak-learners into an ensemble with excellent predictive performance”. The gradient boost also creates weak learners during training and combines these to come up with a relatively stronger learner. The weak learner will calculate a loss which is the difference between the predicted value of a feature and the actual feature value (Shakya, 2018; Chen & Guestrin, 2016; Lu & Mazumder, 2018; Friedman, 2001). A new learner will then be made based on the loss and is trained based on the errors (Shakya, 2018). GBM can handle missing data, unnormalized data and correlated data (Lu & Mazumder, 2018).

The pseudo-code for the GBM algorithm as stated by Friedman (2001) is shown below:

Algorithm 1 Gradient Boosting Machine (GBM) [10]

```
Initialization. Initialize with $f^0(x) = 0$.
For $m = 0, \dots, M-1$ do:
Perform Updates:
(1) Compute pseudo residual $r^m = -\left[\frac{\partial \ell(y_i, f^m(x_i))}{\partial f^m(x_i)}\right]_{i=1,\dots,n}$.
(2) Find the best weak-learner: $j_m = \arg\min_{j \in [K]} \min_\sigma \sum_{i=1}^n (r_i^m - \sigma b(x_i; \tau_j))^2$.
(3) Choose the step-size $\rho_m$ by line-search: $\rho_m = \arg\min_\rho \sum_{i=1}^n \ell(y_i, f^m(x_i) + \rho b(x_i; \tau_{j_m}))$.
(4) Update the model $f^{m+1}(x) = f^m(x) + \rho_m b(x; \tau_{j_m})$.
Output. $f^M(x)$.
```

Figure 3-7: Gradient Boost Machine algorithm (Friedman, 2001)

3.3.4 **Isolation Forest**

The term isolation as used by Liu et al. (2008) refers to separating a feature from the rest of the features in a feature space. The isolation forest algorithm splits datapoints in a feature space until features are isolated. Anomalies form the minority class and have characteristics that differ from normal instances making them prone to isolation (Liu, Ting, & Zhou, 2008). The separation proposed by Liu et al. (2008) implements a binary tree structure called isolation trees(iTrees). Anomalies are closer to the root node of iTrees as they have a shorter part length as compared to normal instances. Liu et al. (2008) proved that outliers or anomalies have short path lengths compared to normal points indicating they are more prone to isolation under random partitioning. iForest has great efficiency when dealing with large datasets, multidimensional data and works well with data that has no anomalies.

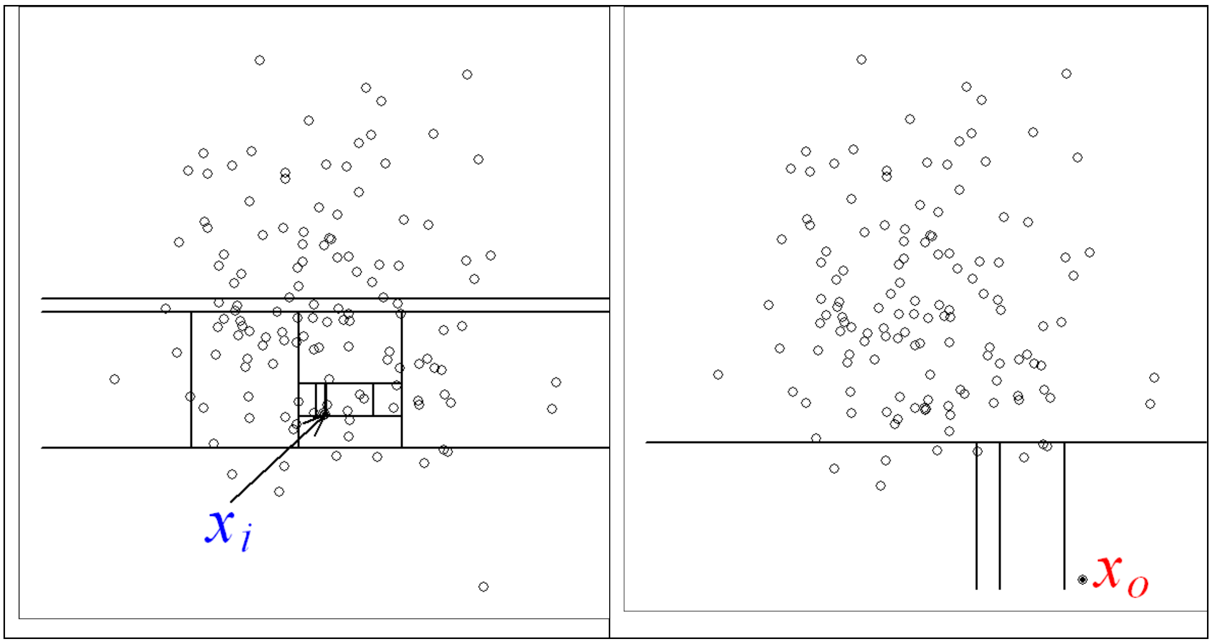


Figure 3-8: Isolating X_i and X_0 (Liu, Ting, & Zhou, 2008)

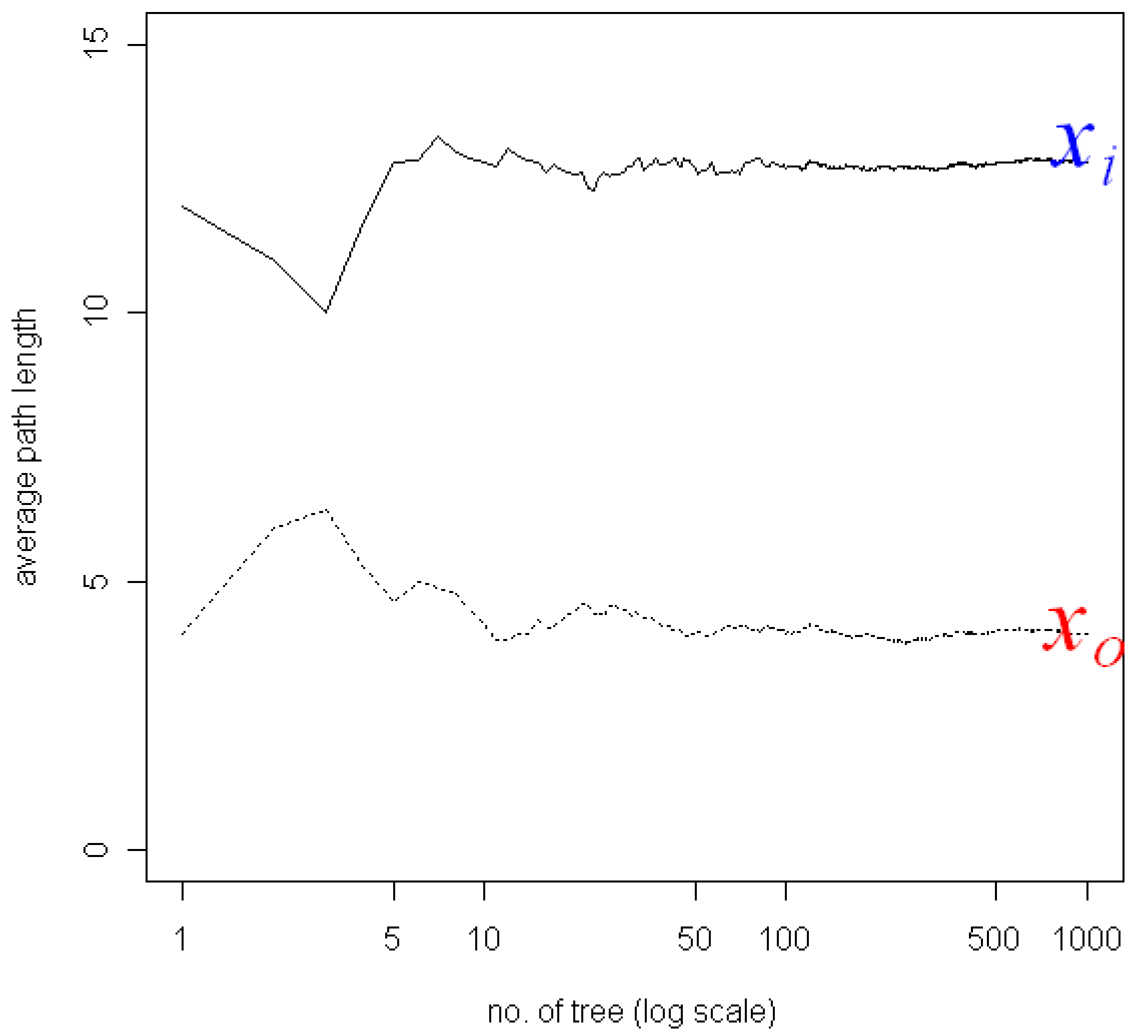


Figure 3-9: Average path length for X_i and X_0 (Liu, Ting, & Zhou, 2008)

Figures 3-8 shows that fewer random partitions are required to isolate an anomalous point in a data set compared to isolating a point in the majority class. Figure 3-9 show that anomalies generally have shorter average path lengths compared to normal instances in a data set.

3.3.4.1 Anomaly detection using Isolation Forest

Anomaly detection using the isolation forest occurs in two stages. The first stage is the training stage which builds iTrees using random sub-samples of a training dataset. The second stage is the evaluation stage. In the evaluation phase test instances pass through the iTrees to obtain an anomaly score for each instance.

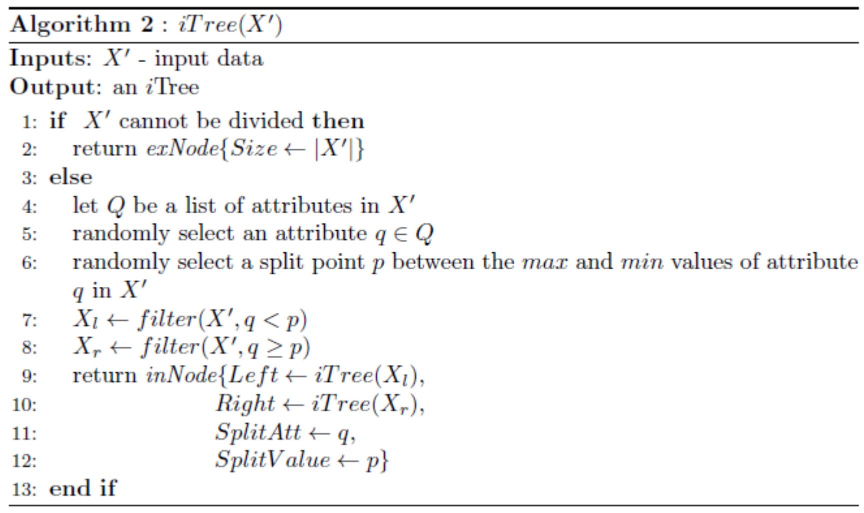
3.3.4.2 Training stage

In this stage features in a feature space X recursively partitioned in sub-sample X' until all instances are isolated, thereby creating iTrees. iTrees are constructed from sub-samples X' which is randomly selected from X without replacement. The training stage as stated by Lui et al. (2008) makes use of iForest and iTrees algorithms. Below the pseudo-code of the iForest algorithm is shown during the two training stages (Liu, Ting, & Zhou, 2008).

Algorithm 1: *i*Forest (X, t, ψ)

```
Inputs: $X$ − input data, $t$ − number of trees, $\psi$ - subsampling size
Output: a set of $t$ $i$Trees
1: Initialize Forest
2: for $i = 1$ to $t$ do
3:  $X' \leftarrow$ sample e$(X, \psi)$
4: Forest ← Forest ∪ Tre e$(X')$
5: end for
6: return Forest
```

The next stage is the building of *i*Trees (Liu, Ting, & Zhou, 2008):



```
Algorithm 2 : $iTree(X')$
Inputs: $X'$ - input data
Output: an $i$Tree
1: if $X'$ cannot be divided then
2:     return $exNode\{Size \leftarrow |X'|\}$
3: else
4:     let $Q$ be a list of attributes in $X'$
5:     randomly select an attribute $q \in Q$
6:     randomly select a split point $p$ between the $max$ and $min$ values of attribute
       $q$ in $X'$
7:     $X_l \leftarrow filter(X', q < p)$
8:     $X_r \leftarrow filter(X', q \geq p)$
9:     return $inNode\{Left \leftarrow iTree(X_l)$,
10:                $Right \leftarrow iTree(X_r)$,
11:                $SplitAtt \leftarrow q$,
12:                $SplitValue \leftarrow p\}$
13: end if
```

In Appendix J, all symbols used in the pseudo-code are explained.

3.3.5 Extremely Randomised Tree Classifier

The Extremely Randomised Tree Classifier (ETC), was first proposed by Geurts et al. (2006). They proposed an ensemble tree classifier that was fully random. The Extra tree classifier is an ensemble tree classifier that builds a forest of trees with fully randomised splits when constructing the trees (Geurts, Ernst, & Louis, 2006). When constructing the trees, the entire original sample is used and not just a bootstrapped replica of the original sample. The results produced by each tree are aggregated by voting criteria in classification problems and averaging in regression problems. The extra tree classifier uses the whole sample to reduce bias, which is a major problem in the other random tree classifiers, like the random forest. Below is the extra tree splitting algorithm as specified by Geurts et al. (2006).

Extra-Trees splitting algorithm (for numerical attributes)

```
Split a node(S)
Input: the local learning subset S corresponding to the node we want to split
Output: a split [$a < a_c$] or nothing
– If Stop split(S) is TRUE then return nothing.
– Otherwise select K attributes $\{a_1, . . . , a_K\}$ among all non constant (in S) candidate attributes;
– Draw K splits $\{s_1, . . . , s_K\}$, where $s_i$ = Pick a random split($S, a_i$), $\forall_i = 1, . . . , K$;
– Return a split $s_*$ such that Score($s_*$, S) = $\max_{i=1,\dots,K}$ Score($s_i$, S).

Pick a random split(S, a)
Inputs: a subset S and an attribute a
Output: a split
– Let $a^S_{\max}$ and $a^S_{\min}$ denote the maximal and minimal value of a in S;
– Draw a random cut-point $a_c$ uniformly in [$a^S_{\min}, a^S_{\max}$];
– Return the split [$a < a_c$].

Stop split(S)
Input: a subset S
Output: a boolean
– If $|S| < n_{\min}$, then return TRUE;
– If all attributes are constant in S, then return TRUE;
– If the output is constant in S, then return TRUE;
– Otherwise, return FALSE.
```

There are two important variables in the algorithm, that is, K and $n_{\min}$. K is the number of randomly selected features at each node and $n_{\min}$, the least number of samples required to split a node.

3.4 Justification of Algorithms

There are many algorithms that have been developed that use decision trees as their base learners. For the purposes of this study the Extra tree classifier, AdaBoost, Gradient Boost, Isolation Forest and Decision tree classifier were chosen as they had shown to be better classifiers in other domains like image recognition, speech synthesis and in networking (Husejinovic, 2020). The research made use of Jupyter notebooks and algorithms that have existing implementations were preferred over those that have not been implemented yet in Jupyter notebooks.

3.5 Conclusion

The chapter firstly gave an introduction to decision trees and how they work. Underlying concepts behind decision trees were discussed in detail. Two important factors that were discussed were the Gini index and entropy which are critical to splitting the nodes of decision trees. Thereafter, we looked at ensemble decision tree classifiers and the underlying algorithms. We started by looking at the random forest classifier which builds an ensemble forest of trees using either Gini or entropy to split the nodes. Random forests use Bagging. Then we looked at Adaptive boosting, Adaboost, which makes use of base learners to produce a more robust classifier. The gradient boosting classifier was also discussed in detail. The last classifier discussed was the Extra tree classifier.

The next chapter is the methodology section and looked at the research methodology that was adapted for this research, the research design, expected results and justification of why the methodology was chosen was ideal for this study.

Chapter Four

4 Methodology, Design and Implementation

4.1 Introduction

This chapter details the methodology framework adopted in the design of ensemble tree classifier-based fraud detection models. Sections 4.2 describes the methodology used and 4.3 outlines the research design that was followed in this research. Section 4.4 describes the dataset that was used and section 4.5 explores the data to gain insights. Section 4.6 outlines data splitting for the experiments and section 4.7 describes how the data was sampled. Section 4.8 gives details of the experimental design. Section 4.9 discussed performance measures followed by a conclusion of the chapter.

4.2 Methodology

This study employed experimental approach. We investigate the following parameters in this study: the choice of sampling technique, the sample size of the training dataset and the split criterion to establish how they influence the accuracy of credit card fraud detection. In experimental methodology we need to first determine the independent and dependent variables. The independent variables describe the cause of an outcome while the dependent variable is the outcome. In our study sample size, sampling technique and split criterion are the independent variables and the fraud class or outcome is the independent variable.

The experiments will be setup such that one independent variable will be manipulated at a time. This will enable us to see how each variable will affect outcome. Sample size was varied from 10% to 100% using intervals of 10%. Three sampling techniques were analysed: Random Oversampling, Random Undersampling and SMOTE. The split criterion will be Gini index and Entropy. Six algorithms were analysed: Decision tree classifier, Random Forest, Extra tree classifier, Adaboost, Isolation Forest and gradient boost.

The experimental methodology was chosen because it permits control of different independent variables or parameters being investigated to establish how they affect outcome. Sadgali et al. (2018) states that experiments are best used if analysing performance of different algorithms and in this study, we analyse different algorithms which makes the experimental methodology the most appropriate.

4.3 Research Design

This research was undertaken using the framework depicted in figure 4-3. We made use of the e-commerce dataset for credit card fraud detection for the construction of decision tree classifiers. The classifiers were trained using supervised machine learning and a test dataset was used to evaluate the performance of the classifiers. The classifiers are implemented for credit card fraud detection tasks.

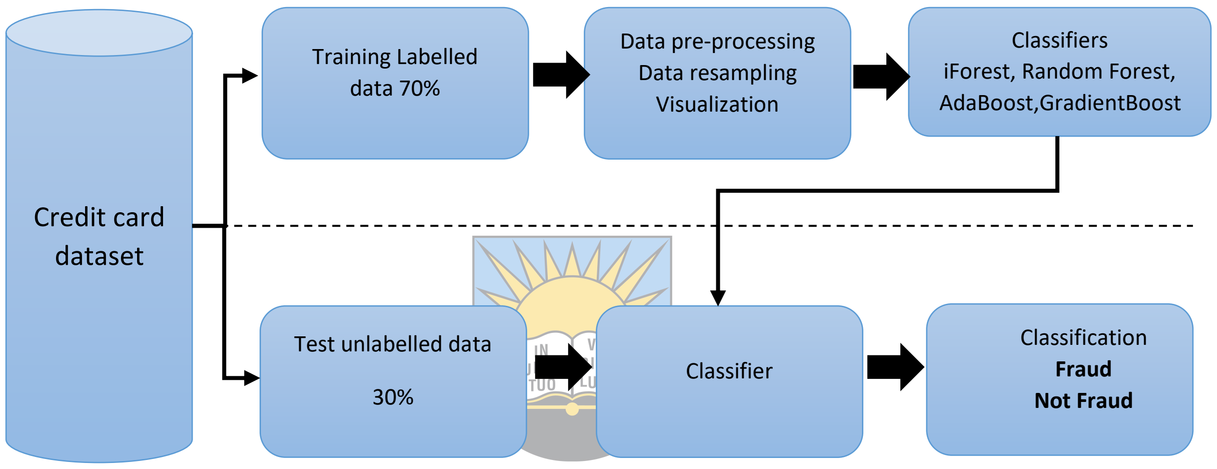


Figure 4-1: Conceptual Framework

Our research design made use of the following steps:

1. Split dataset into training and testing set
2. Data Pre-processing
3. Build decision tree classifiers using the training set
4. Evaluate classifiers using the Test set

The first step is to split the data into training data set and test data set. The data we used is secondary data and has labels in feature ‘Class’ and therefore the need to first split the data into two groups. The training dataset will be used to train the classifiers. The labels for the ‘Class’ feature will be dropped in the test dataset to allow the classifiers to predict outcomes.

Data pre-processing is critical in data mining as data is not always structured. We checked our data first for missing values using the column counts and our data did not have any missing values in all columns. We visualised our data and we could see that the data was highly imbalanced and hence we had to resample the data. How we resampled the data will be discussed in detail in Section 4.6. After data pre-processing, we used the training dataset to train our model.

4.4 Dataset

This study made use of secondary data provided by Vesta Corporation, which has made available real-world e-commerce transactions on Kaggle an online competition site for Data Science (Credit Card Fraud Detection | Kaggle, 2020). The data was collected over two days in September 2012 from European cardholders. The dataset has 284,807 transactions of which 492 are fraud making the dataset skewed.

Table 4-1: Summary of Dataset

Total dataset	**Number of Non-Fraud**	**Number of Fraud**	**Fraud label**	**Non-Fraud label**
284 807	284 315	492	1	0

Appendix K has a detailed description of the data set. Table 4-1 shows a summary of the dataset, which has a total of 284 807 transactions of which 284 315 is legitimate and only 492 are fraudulent transactions. The data is labelled with 0 and 1, where 0 is a legitimate transaction and 1 is for fraudulent transactions. Figure 4-5 shows that the dataset is highly imbalanced with the majority class being the legitimate transactions.

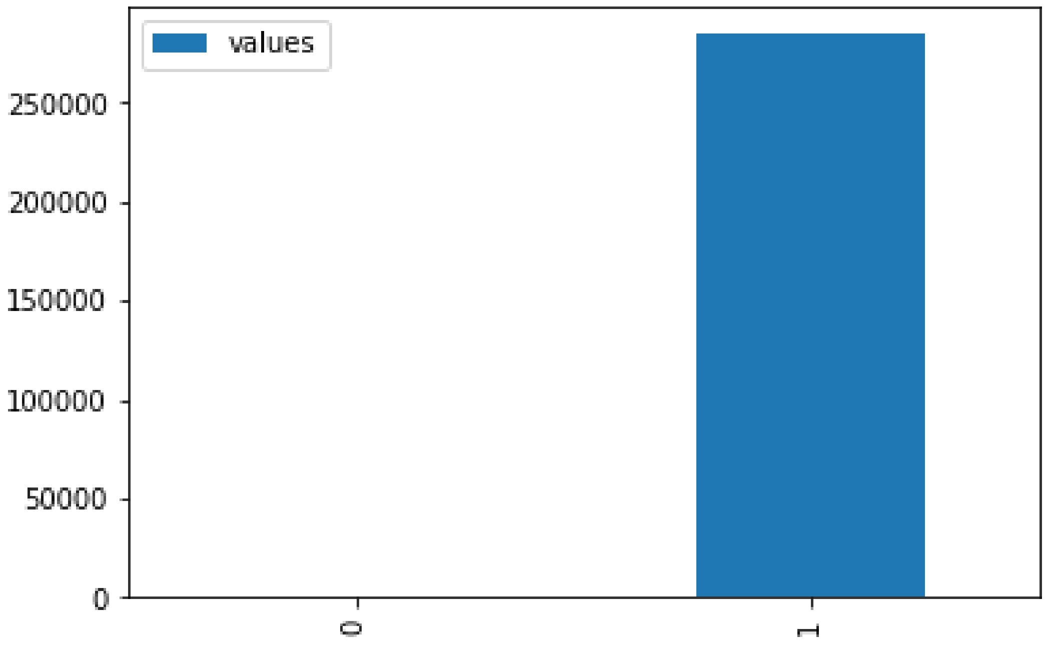
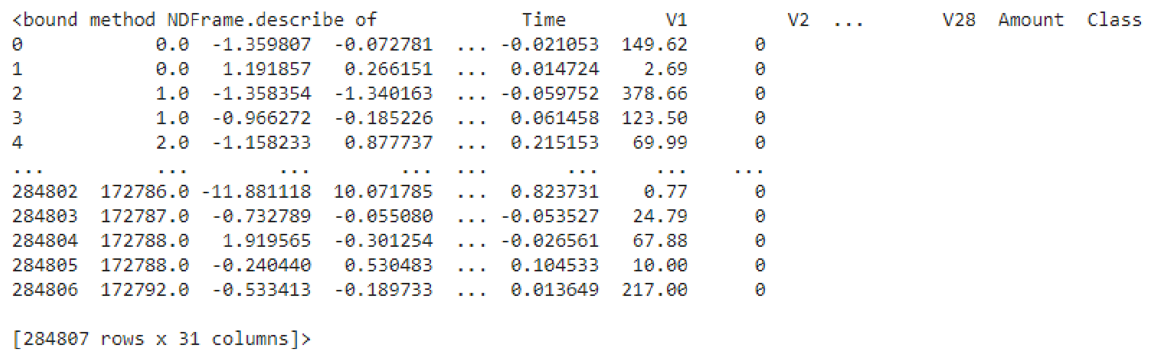


Figure 4-2 Highly imbalanced dataset

The data contains 31 features of which only three are defined, which are, amount, time and class. The amount is the money spent on the transaction, Time is the amount of time, in seconds, elapsed between any given transaction in the dataset and the first transaction in the dataset and class specify whether a transaction is fraudulent or not. The remaining features labelled as V1 – V28 are a result of Principal Component Analysis, a dimensionality reduction to protect user identity and sensitive features. There is no noise in the dataset as the data was released with labels that specify whether a transaction was fraudulent or not. Figure 4-3 shows a summary and description of the dataset.



```
<bound method NDFrame.describe of              Time         V1         V2  ...       V28  Amount  Class
0             0.0  -1.359807  -0.072781  ... -0.021053  149.62      0
1             0.0   1.191857   0.266151  ...  0.014724    2.69      0
2             1.0  -1.358354  -1.340163  ... -0.059752  378.66      0
3             1.0  -0.966272  -0.185226  ...  0.061458  123.50      0
4             2.0  -1.158233   0.877737  ...  0.215153   69.99      0
...           ...        ...        ...  ...       ...     ...    ...
284802  172786.0 -11.881118  10.071785  ...  0.823731    0.77      0
284803  172787.0  -0.732789  -0.055080  ... -0.053527   24.79      0
284804  172788.0   1.919565  -0.301254  ... -0.026561   67.88      0
284805  172788.0  -0.240440   0.530483  ...  0.104533   10.00      0
284806  172792.0  -0.533413  -0.189733  ...  0.013649  217.00      0

[284807 rows x 31 columns]>
```

Figure 4-3: Description of dataset

4.5 Data Visualisation

When dealing with big data problems and predictive analytics it is important to visualise the data. Data visualisation is when large data sets are graphically represented so that data users can have an idea of how the data is structured (Tableau, 2021). One of the important reasons to visualise data is to find out how features in a feature space may be positively or negatively related. This is called correlation. Understanding correlations between features and predictions is important during the training phase of models.

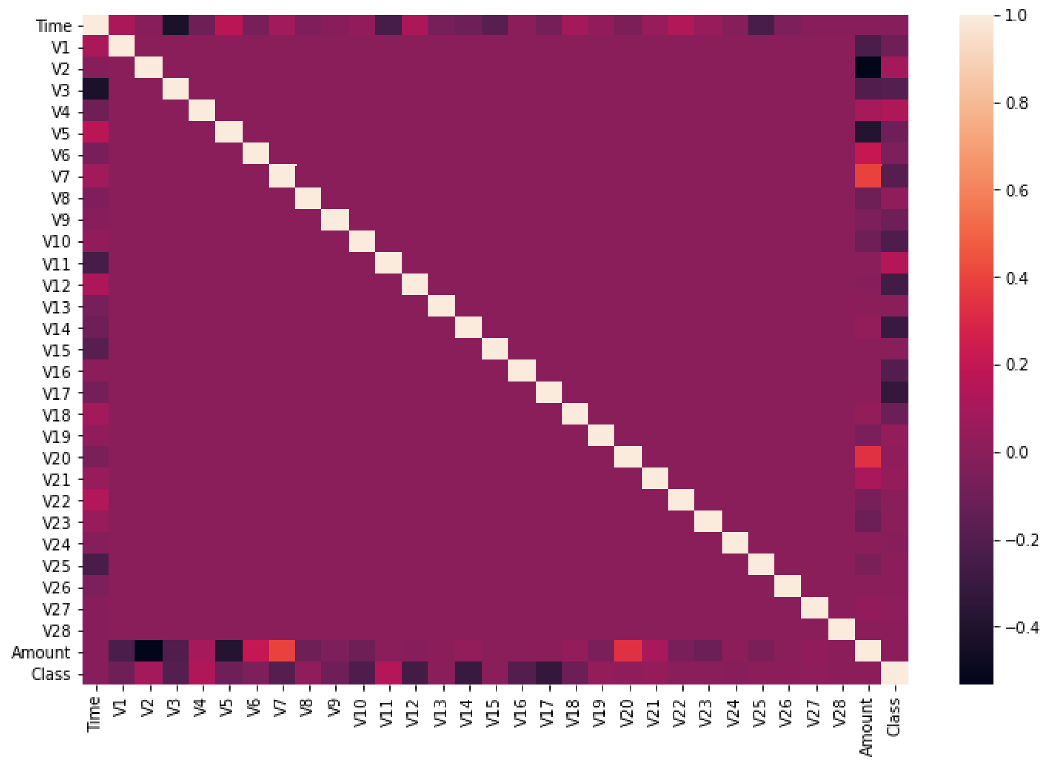


Figure 4-3: Heatmap of correlation between columns of Data Frame

Figure 4-6 shows the correlation between all the features in the feature space against each other. The correlation heatmap shows that the principal components V1 to V28 are in no way correlated. However, some of the principal components are positively correlated to the outcome variable.

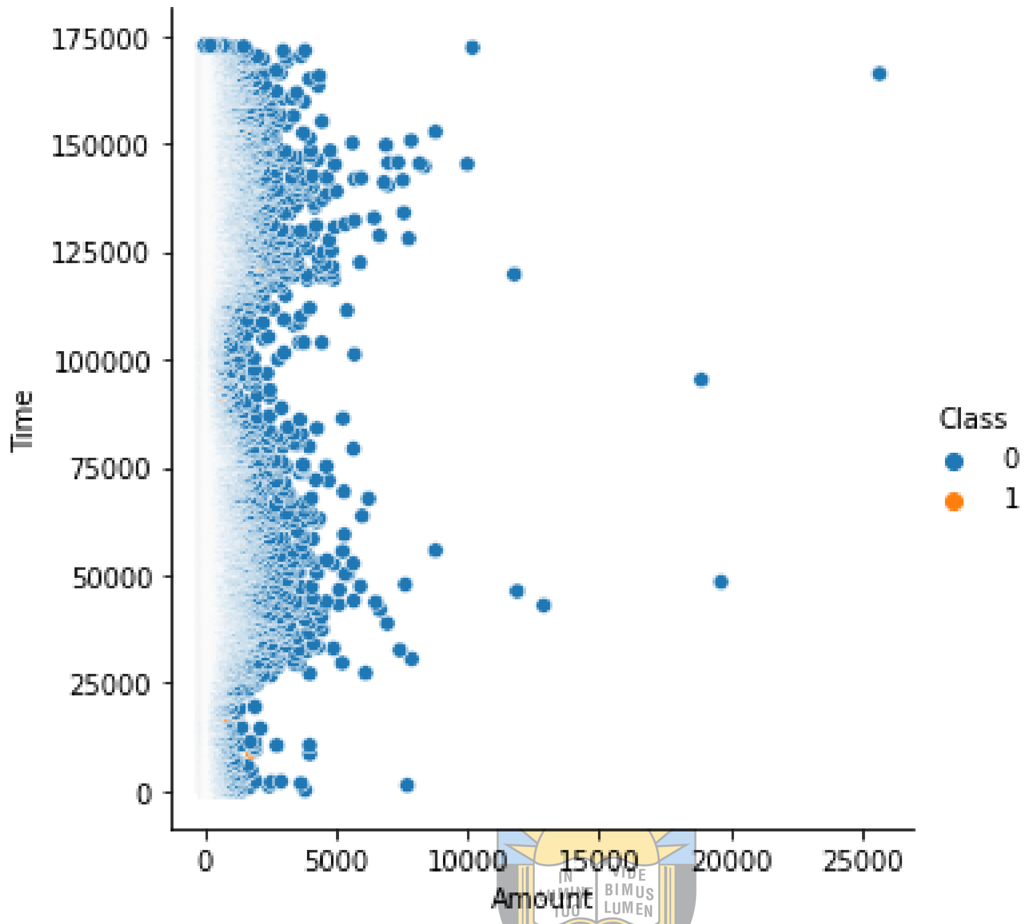


Figure 4-4: Plot of time vs amount by different classes

Figure 4-7 is a plot of all the fraud and non-fraud instances in the dataset on a scale of time and amount. We can see that all the high amount transactions were legitimate and fraudulent transactions can barely be seen as they are in the mixture of legitimate majority transactions. A boxplot of the amount by class will give us an even better picture of the different amounts on fraudulent transactions.

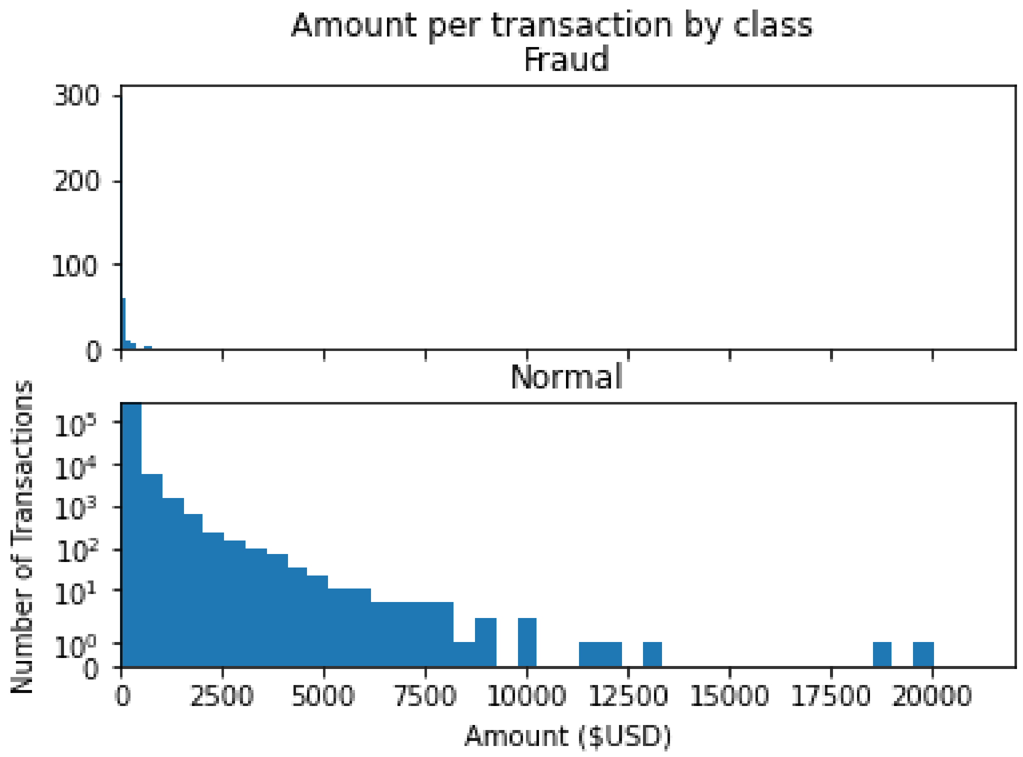


Figure 4-5: Amounts per transaction sorted by Class

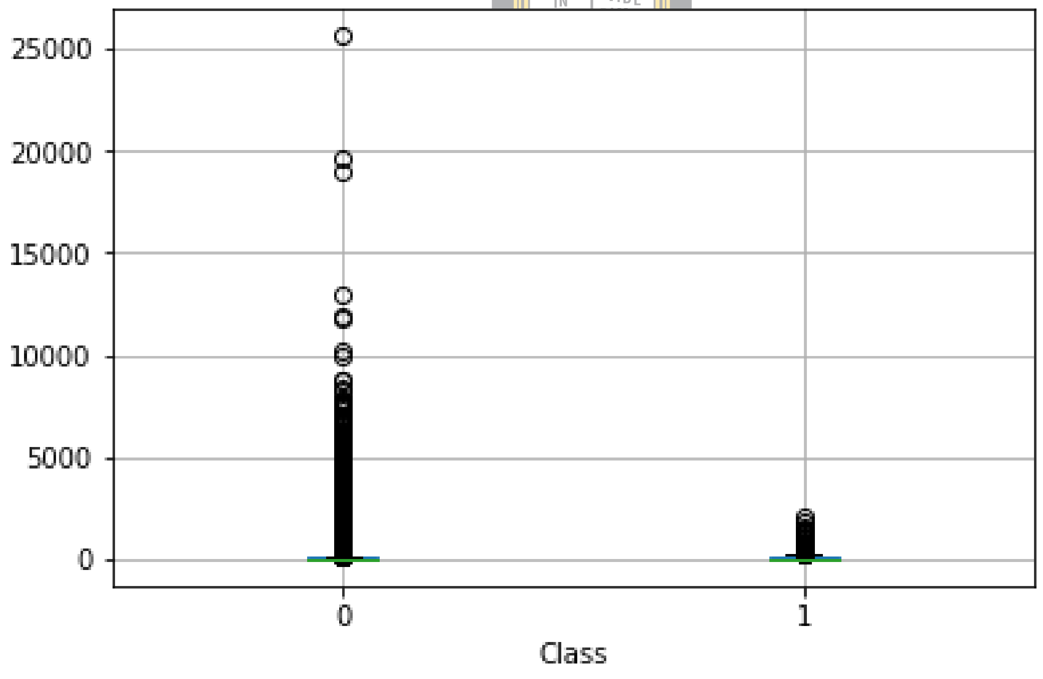


Figure 4-6: Boxplot of Amount by Class type

Figure 4-9 clearly shows that the fraudsters in the dataset made transactions with amounts less than 5000. Figure 4-8 and figure 4-9 suggests that fraudsters tried to go unnoticed by using smaller amounts. We then looked at the time series by class.

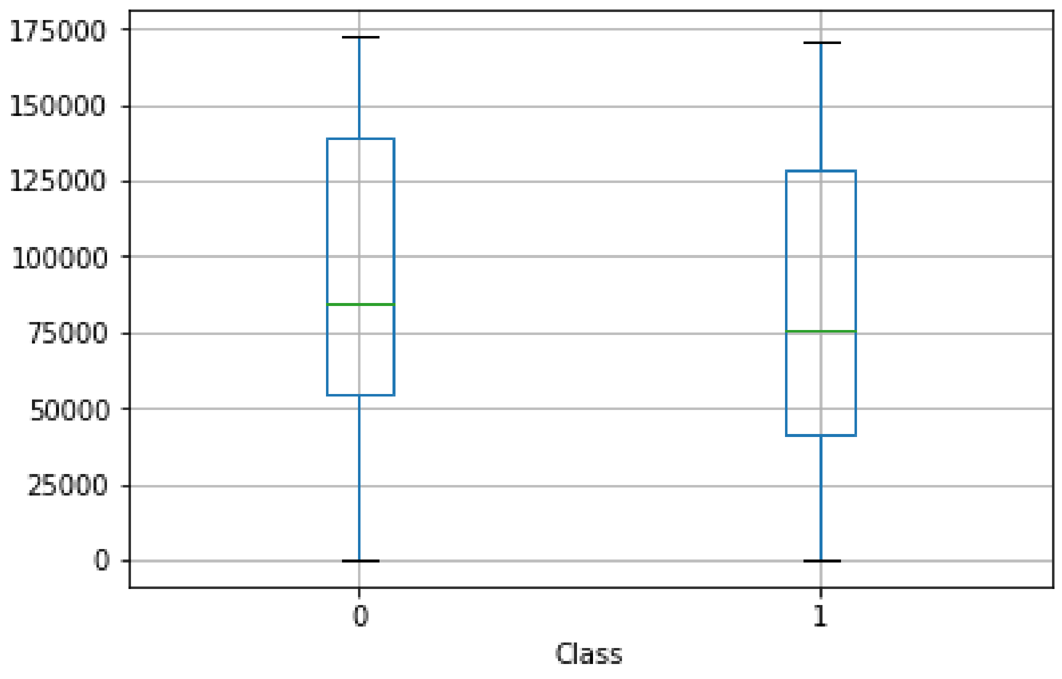


Figure 4-7: Time series classification

The feature time as highlighted earlier represents the time taken between transactions from the first recorded transaction in the dataset. The time feature is in seconds. Figure 4-10 shows that fraudulent transactions have less time between them compared to normal transactions. Next, we used histograms to visualise how the data is distributed.

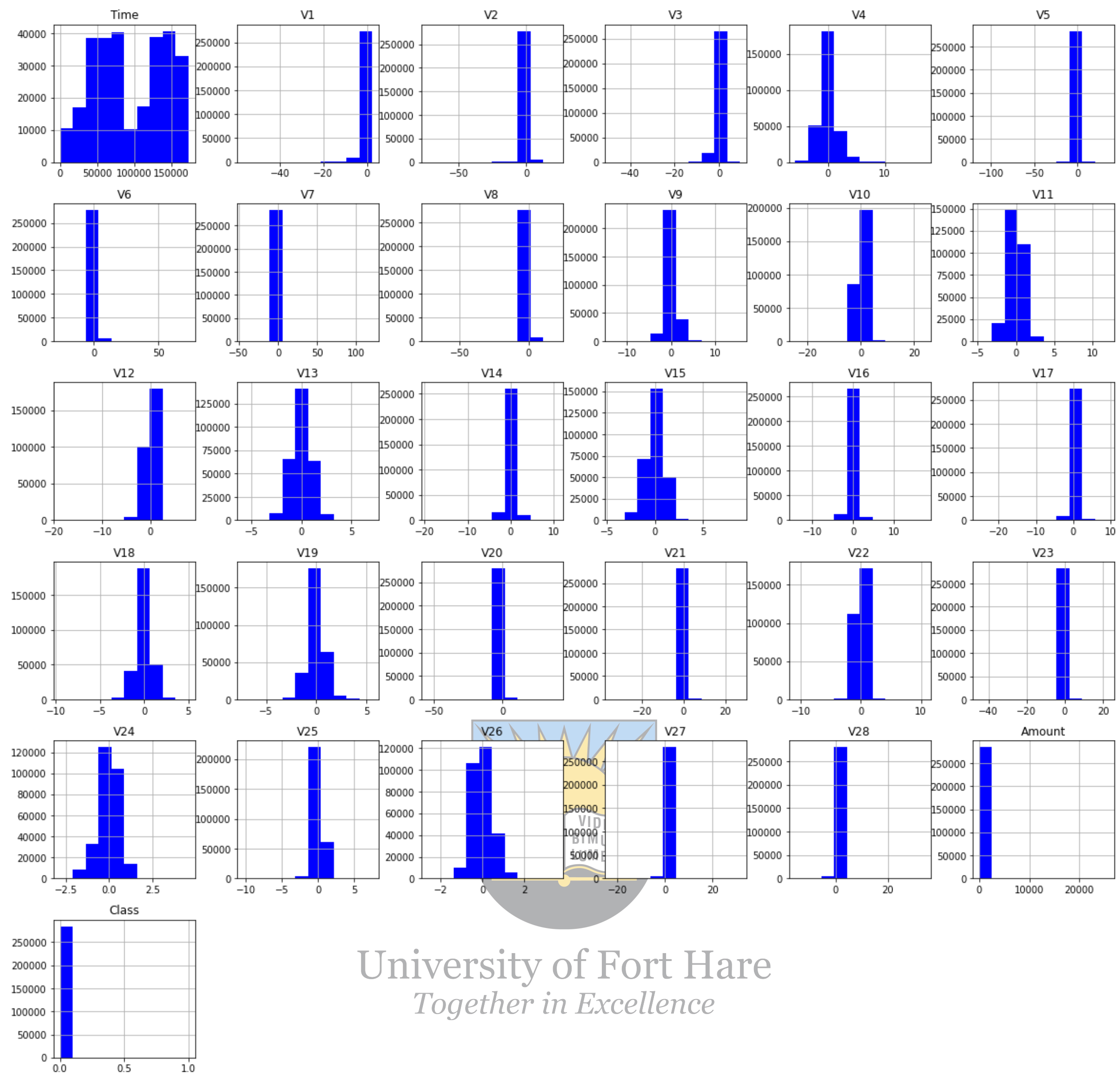


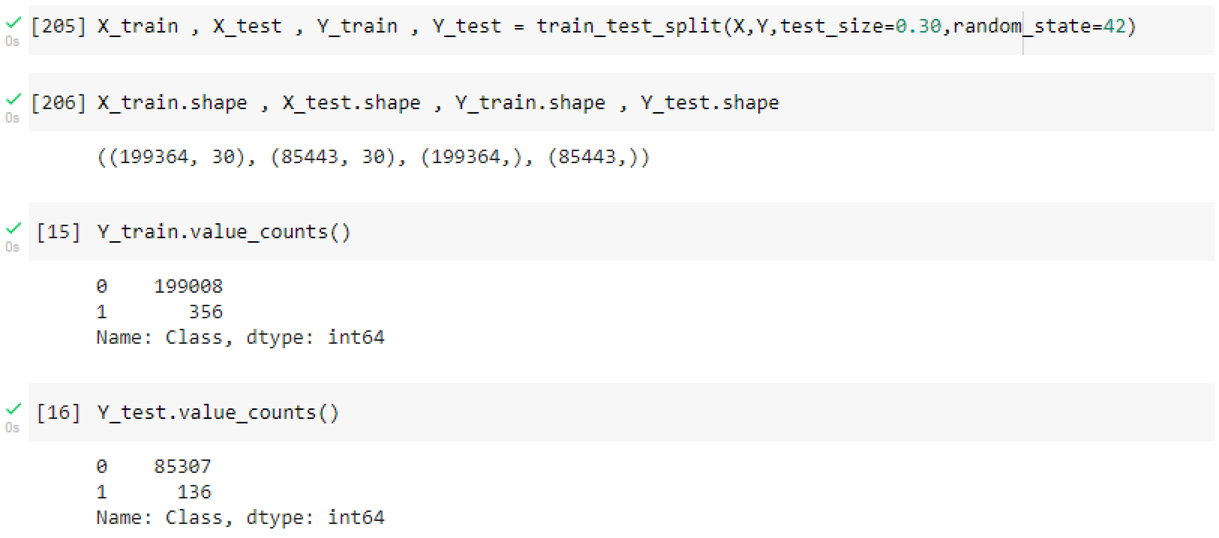
Figure 4-8: Distribution of Features in Data Set

Figure 4-11 shows the distribution of all the features in our feature space. The time distribution is bimodal. Since the data was collected over two days the different peaks are possible for the two different days. It is hard to tell which time fraudsters were most active as the time shown is the time difference from the first transition to every other transaction. However, most of the other features have a normal distribution.

4.6 Data splitting

The dataset contained labelled output variable 'Class', so we had to split the dataset into two data frames. The first is the training data and the other for testing the model. The dataset was split into 70% training data and 30% test dataset. Figure 4-12 shows how the dataset was split.

The test size specifies the percentage of the data that must be used for the test and in this case, we used 30% .



```
[205] X_train , X_test , Y_train , Y_test = train_test_split(X,Y,test_size=0.30,random_state=42)

[206] X_train.shape , X_test.shape , Y_train.shape , Y_test.shape

      ((199364, 30), (85443, 30), (199364,), (85443,))

[15]  Y_train.value_counts()

      0    199008
      1       356
      Name: Class, dtype: int64

[16]  Y_test.value_counts()

      0    85307
      1      136
      Name: Class, dtype: int64
```

Figure 4-9: Splitting data into train and test set

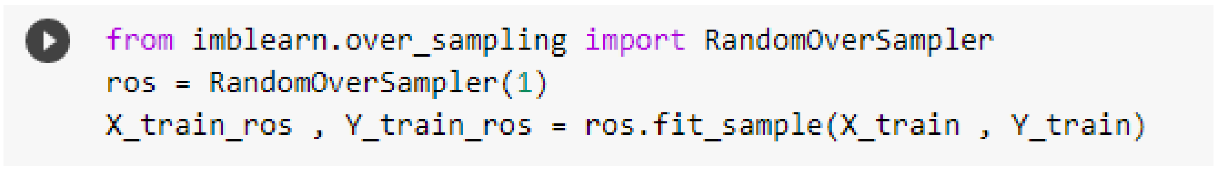
The training dataset was used for resampling and training the model. The test data was used to measure the classification performance of the trained model.

Table 4-6: Summary of training and test dataset for the experiments

	Percentage of Dataset	Number of Non-Fraud	Number of Fraud
Train	70	199 008	356
Test	30	85 307	136

4.7 Data sampling

The dataset we used is highly skewed or unbalanced. The class of non-fraudulent transactions greatly exceeds the class of fraudulent transactions. The non-fraudulent class is referred to as the majority class and the fraudulent as the minority class. As discussed earlier, if skewed data is used to train a model it will be biased towards the majority class which will result in poor classification performance. To make our training dataset balanced we used the Random Oversampling and Random Undersampling techniques so that both classes become balanced. Random oversampling was applied to the dataset and increased the instances of the minority class until it was the same as those of the majority class. Figure 4-13 shows the python code used to apply random oversampling and how the minority class which had 356 instances was oversampled to match the majority class with 199 008 instances.



```
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(1)
X_train_ros , Y_train_ros = ros.fit_sample(X_train , Y_train)
```

Figure 4-10: Code used to apply random oversampling on the training set

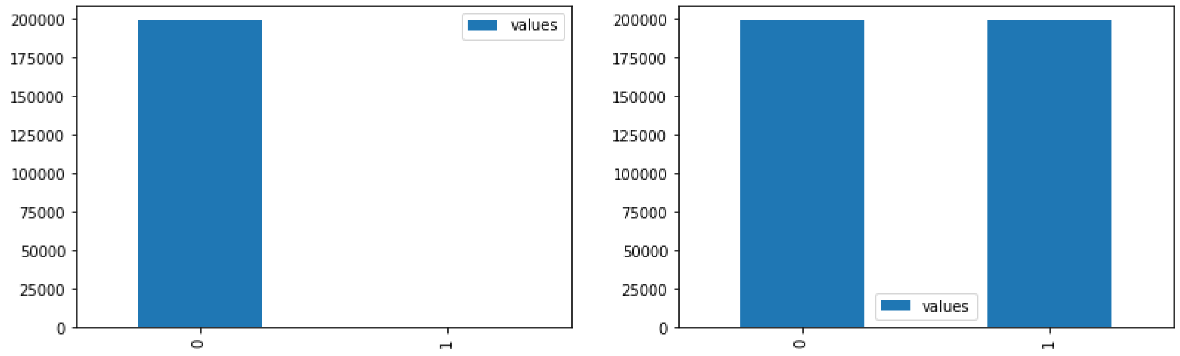


Figure 4-11: Random Oversampling

Figure 4-14 depicts the data after being Oversampled. Random undersampling was also used to reduce the number of instances in the majority class to match those in the minority class. Figure 4-15 shows how the majority class was reduced from 199 008 to 356 instances which is the same number as those in the minority class.

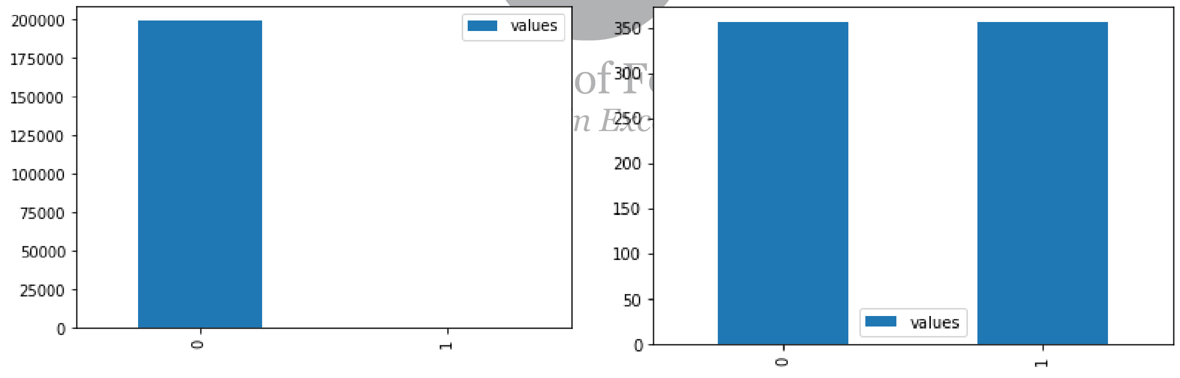
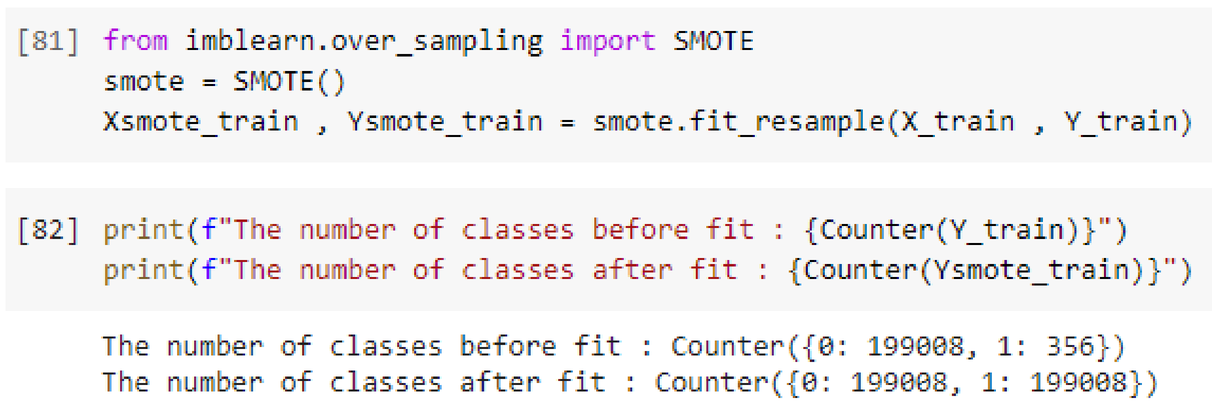


Figure 4-12: Random undersampling

Lastly, we used the Synthetic minority over-sampling technique (SMOTE). The SMOTE formulates new synthetic instances of the minority class by inserting features between close neighbours (Aisha Abdallah, 2016; Chawla, Bowyer, Hall, & Philip, 2002). Shakya (2018) preferred SMOTE method compared to random over-sampling as it reduces the problem of overfitting the training data (Shakya, 2018). Figure 4-16 shows how SMOTE was implemented in Python.

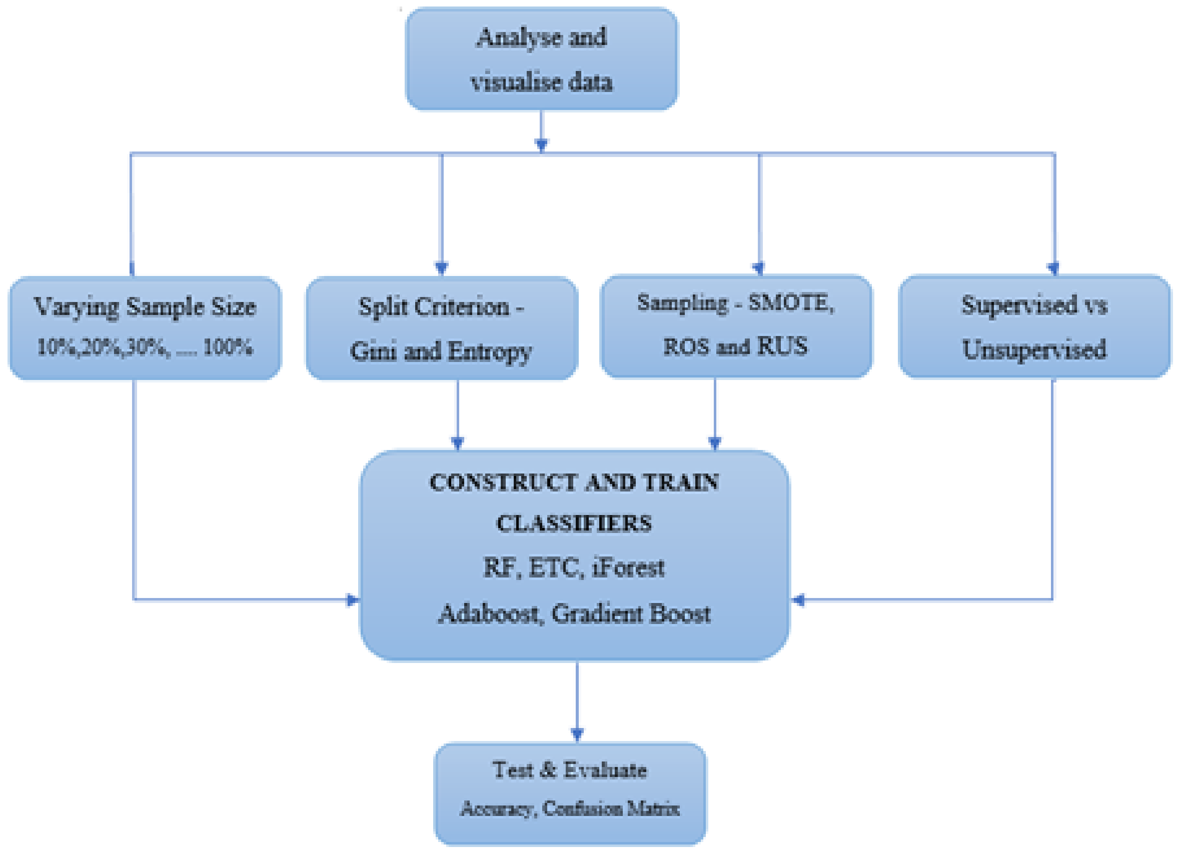


```
[81] from imblearn.over_sampling import SMOTE
     smote = SMOTE()
     Xsmote_train , Ysmote_train = smote.fit_resample(X_train , Y_train)

[82] print(f"The number of classes before fit : {Counter(Y_train)}")
     print(f"The number of classes after fit : {Counter(Ysmote_train)}")

     The number of classes before fit : Counter({0: 199008, 1: 356})
     The number of classes after fit : Counter({0: 199008, 1: 199008})
```

Figure 4-13:Applying SMOTE on training data

4.8 Experiments Setup

Figure 4-17 depicts the flow chart that we used in carrying out the experiments. In the first experiment, we investigate how different sampling techniques affect the classification performance of ensemble tree classifiers.



Analyse and visualise data
Varying Sample Size 10%,20%,30%, ... 100%
Split Criterion - Gini and Entropy
Sampling - SMOTE, ROS and RUS
Supervised vs Unsupervised
CONSTRUCT AND TRAIN CLASSIFIERS
RF, ETC, iForest
Adaboost, Gradient Boost
Test & Evaluate
Accuracy, Confusion Matrix

Figure 4-17: System flow chart that was used in the study

We used ROS, RUS and SMOTE to balance the training data as described in detail in section 4.6. The second experiment investigated the effect of sample size on classification performance. The total training data was made up of 199 364 records of which 199 008 were legitimate transactions and 356 were fraudulent as shown in table 4-6. The total training set was sampled from 10% and was incremented by a margin of 10% until 100% of the training data was used.

Table 4-2:The number of records for each sample size

Sample	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Size	19 936	39 873	59 809	79 746	99 682	119 618	139 555	159 491	179 428	199 364

Table 4-2 shows the observations for each sample, from 19 936 to 199 364 which represents the whole data set. In our third experiment, we looked at how the split criterion affected classification performance. Ensemble tree classifiers use Gini indexing as a split criterion by default, so we changed the split criterion to entropy to determine how classification accuracy was influenced. Lastly, we investigated how the choice of learning approach influence the classification performance of ensemble tree classifiers. In this experiment, we investigated how supervised learning algorithms performed compared to unsupervised ones. Random Forest, Extra Tree Classifier, Decision Tree Classifier, Adaboost, Gradient boost are supervised classifiers and were compared to the iForest which is an unsupervised classifier. The nature of the data provided had labels hence the learning approach for the iForest was semi-supervised.

4.9 Performance Measurement

A machine learning model must be evaluated to establish how correct and accurate it is. To achieve this a good performance measure is required to evaluate the model. When dealing with two-class classification problems the confusion matrix is ideal.

4.9.1 Confusion Matrix

A confusion matrix produces four outcomes, True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN). These outcomes help to establish transactions that are accurately classified and the number of those that are classified incorrectly.

		ACTUAL	
		POSITIVE	**NEGATIVE**
PREDICTION	**POSITIVE**	TRUE POSITIVE (TP)	FALSE POSITIVE (FP)
	NEGATIVE	FALSE-NEGATIVE (FN)	TRUE NEGATIVE (TN)

Figure 4-14: Confusion Matrix for Binary classification

True Positive (TP) is when the predicted value is positive and the actual value was positive. For example, a classifier predicts that a transaction is a fraud yet the transaction was fraudulent.

False Positive (FP) is when the predicted value is positive and the actual value is negative. For example, a classifier predicts that a transaction is fraudulent yet the transaction was authentic.

True Negative (TN) is when the predicted value is negative and the actual value is negative. For example, a classifier predicts that a transaction is non-fraudulent and the transaction is non-fraudulent.

False Negative (FN) is when the predicted value is negative and yet the actual is positive. For example, a classifier predicts that a transaction is non-fraudulent yet the transaction is fraudulent.

To measure the accuracy of a binary classifier the following formula can be used from the outcomes of the confusion matrix. The accuracy will be a fraction of data points that have been classified correctly. Accuracy is defined as the number of true positives and true negatives divided by the number of true positives, true negatives, false positives, and false negatives as shown in equation 4.1 (Jason, 2020):

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{4.1}$$

Macaraeg (2019) argues that accuracy is not the best measure for models that detect fraudulent transactions due to skewed data. In the case of fraudulent transactions other performance measures such as cost-sensitive learning (Andrea, Giacomo, Oliver, Cesare, & Gianluca, 2015). According to Dal Pozzolo et al. (2015) for every instance, a misclassification cost is assigned based on the effect of the misclassification on the total misclassification cost. This is because the misclassification of a non-fraudulent transaction can be more costly as compared to the misclassification of a fraudulent transaction. However, getting the information from banks on the cost of misclassification is almost impossible and as a result, two other methods can be applied.

The Area Under the Curve Receiver Operating Characteristic (AUC-ROC) and Area Under the Precision-Recall Curve (AUPRC) are other performance measures that can be used to measure the classification performance of Binary Classifiers.

4.9.2 **AUC-ROC**

The AUC-ROC is a curve that plots True Positive Rate (TPR) against False Positive Rate (FPR). TPR also called Recall is defined as the number of true positives divided by the number of true positives and false negatives as shown in equation 4.2 (Jason, 2020):

$$TPR = \frac{TP}{TP + FN} \tag{4.2}$$

TPR is a fraction that measures true fraudulent transactions that are correctly classified. FPR is defined as the number of false positives divided by the number of false positives and true negatives as shown in equation 4.3 (Jason, 2020):

$$FPR = \frac{FP}{FP + TN} \tag{4.3}$$

FPR determines non-fraudulent transactions that are not classified as fraudulent. The area under the AUC curve is called the AUC score. The AUC score is used to determine how well the model performs. If a model gives a high AUC score, then it performs better.

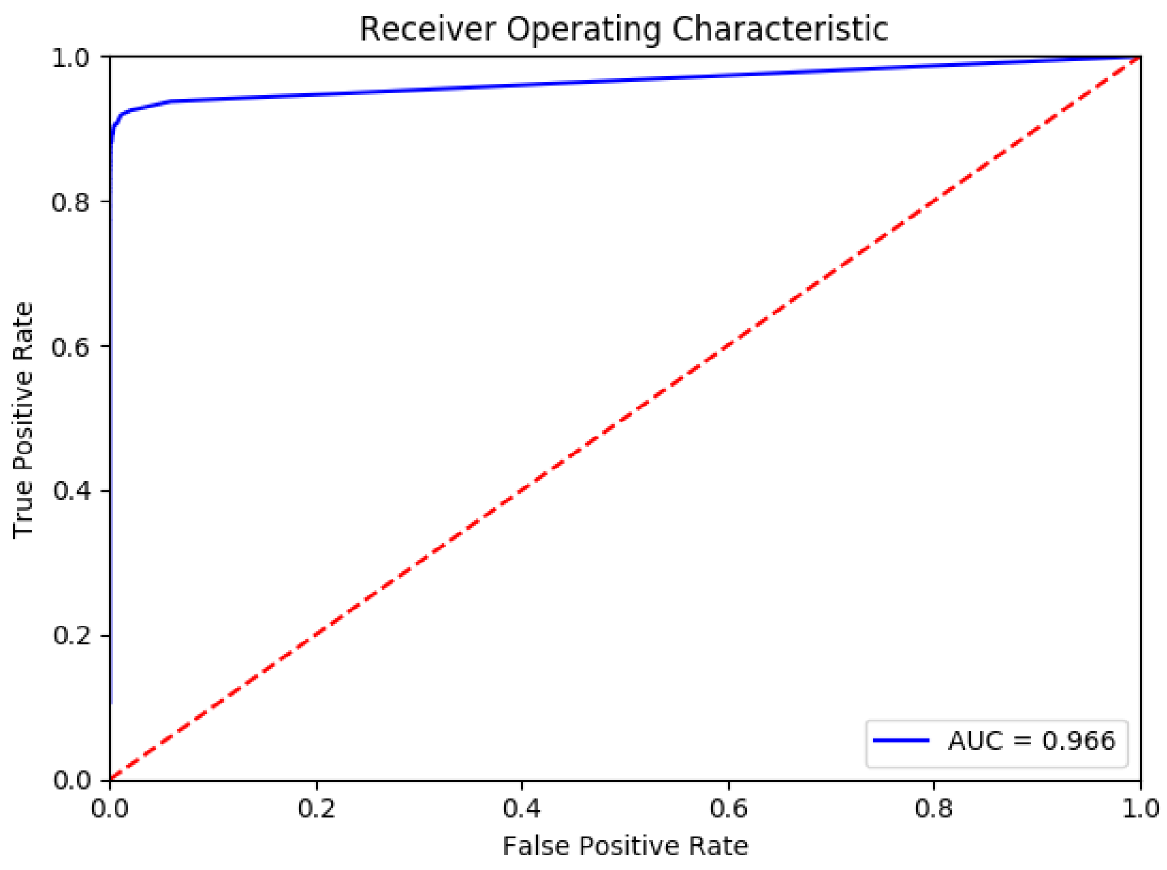


Figure 4-15: Example of ROC (Shakya, 2018)

4.9.3 **AUPRC**

The AUPRC is a plot of Precision against Recall. Recall as specified earlier is the TPR. Precision is defined as the number of true positives divided by the number of true positives and false positives as shown in equation 4.4 (Jason, 2020):

$$Precision = \frac{TP}{TP + FP} \tag{4.4}$$

The area under the Precision-Recall Curve shows the efficacy of a model. The larger the area under the curve the better the model performs. The maximum area under the curve is one, and a model with an area of one can correctly classify all fraudulent transactions s fraud. According to Brownlee (2018), the AUC-ROC performs best when used in instances where the data points in classes are balanced, else the AUPRC measure must be used as it handles imbalanced datasets better.

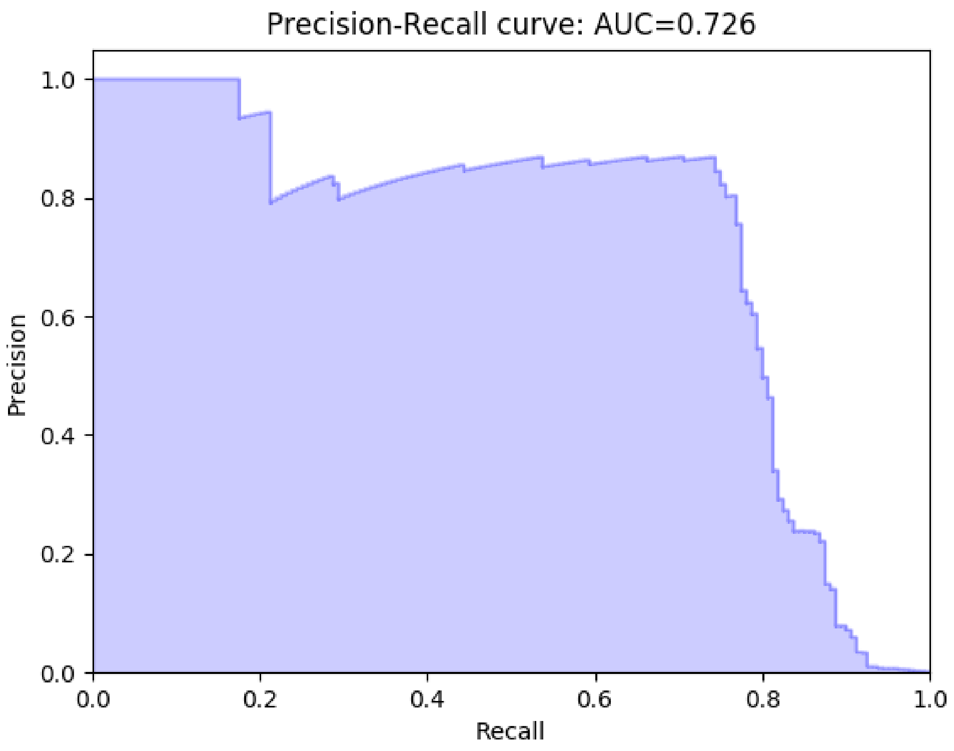


Figure 4-16: Example of Precision-Recall Curve (Shakya, 2018)

4.9.4 F1 Score

F1 Score also called F score or F-measure is the harmonic mean of the recall and precision. Its value ranges from 0 to 1, where 0 is considered worst, and 1 is considered best. It can be calculated as follows (Jason, 2020):

$$F1 = \frac{2 * (Precesion * Recall)}{Precision + Recall} \tag{4.5}$$

4.10 Conclusion

This chapter outlined the methodology that was adopted and the research design implemented in this study. A conceptual framework was also designed for the experiments we carried out. The training data used was described and data visualisation techniques were used in data pre-processing to gain insight into the data before using it. The last chapters of the chapter discussed the parameters that are used to evaluate the performance of classifiers. The confusion matrix is one of the most used performance measures for two-class classification problems. The next chapter presents the analysis of the results of this study.

Chapter Five

5 Data presentation, Analysis and Results

5.1 Introduction

This chapter presents the results from the empirical experiments discussed in chapter four and a discussion of the results. The chapter is arranged as follows, section 5.2 presents the results from the empirical experiments, section 5.3 contains analysis and discussions of the experiments conducted and lastly, section 5.4 gives an overall conclusion of the chapter. This chapter used different graphs drawn from the experiments to establish how the split criterion, sample size, sampling technique and learning technique affected the classification performance of ensemble tree classifiers.

5.2 Data presentation

5.2.1 Sampling Technique

In our first experiment, we look at how different sampling techniques affect the classification performance of classifiers when interrogated using a test data set. We used two main techniques in our experiments, random oversampling and random undersampling. Several authors have investigated the use of hybrid, SMOTE (Chawla Nitesh V, 2002) and SHRINK system to sample datasets to deal with the imbalance problem. Our study investigates which sampling technique between oversampling and undersampling leads to a better classifier and gives the lowest amount of false positives on the credit card dataset. We also implemented the SMOTE to solve the class imbalance in addition to the random oversampling technique. Firstly, the dataset was split into 70%, training set, and 30% test set. After, the decision tree classifiers were trained with the randomly oversampled dataset. In this case, the minority class was replicated to match the majority class, where the minority class was the fraud class. The results of the classifiers are shown in table 5-1. Figure 5-1 and 5-2 shows the confusion matrix of the classifiers.

Classifier	Accuracy	Precision		Recall		F1-Score	
		0	1	0	1	0	1
Decision Tree	99.9262	1	0.81	1	0.75	1	0.78
Extra Tree	99.9531	1	0.97	1	0.75	1	0.78
Random Forest	99.9531	1	0.95	1	0.77	1	0.85
AdaBoost	98.7886	1	0.11	0.99	0.84	0.99	0.19
Gradient Boost	99.4522	1	0.22	0.99	0.86	1	0.35

Table 5-1: Performance of Classifiers using Random oversampling technique

Table 5-1 shows the classification performance of ensemble tree classifiers. The performance matrix listed are accuracy, precision, recall and f1-score.

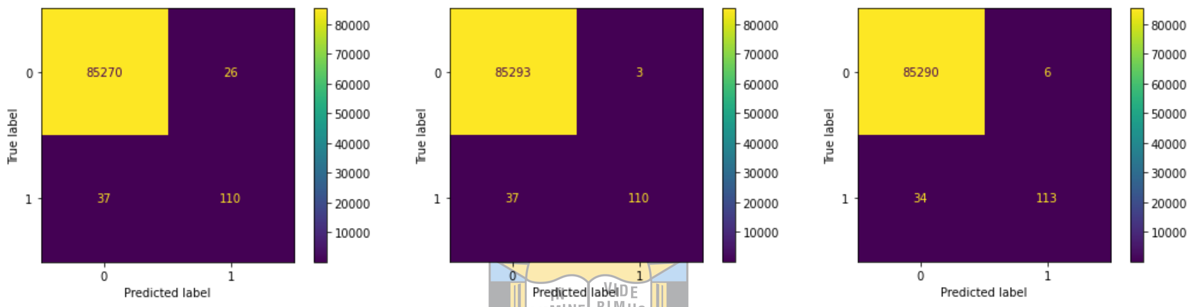


Figure 5-1: Confusion matrix of Random over-sampling applied on test data. Left: Decision Tree Classifier. Centre: Extra Tree Classifier. Right: Random Forest Classifier

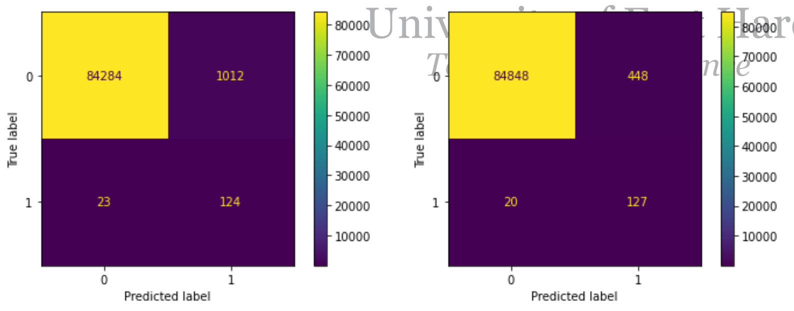


Figure 5-2: Confusion matrix of Random over-sampling applied on test data. Left: AdaBoost Classifier. Centre: Gradient Boost Classifier

Figures 5-1 and 5-2 depict the confusion matrix of when ROS was applied to the data. This is important as it will show us the actual number of false positives and false negatives.

Secondly, we trained the classifiers with a random under-sampled dataset. The randomly under-sampled dataset reduced the majority class to match the minor class. Table 5-2 shows the results. Figures 5-3 and 5-4 show the confusion matrix for the classifiers.

Classifier	Accuracy	Precision		Recall		F1-Score	
		0	1	0	1	0	1
Decision Tree	39.1489	1	0.00	0.39	0.97	0.56	0.01
Extra Tree	89.6223	1	0.02	0.90	0.93	0.95	0.03
Random Forest	68.5790	1	0.01	0.69	0.94	0.81	0.01
AdaBoost	49.8706	1	0.00	0.50	0.97	0.66	0.01
Gradient Boost	58.9375	1	0.00	0.59	0.96	0.74	0.01

Table 5-2: Performance of classifiers using the random under-sampling technique

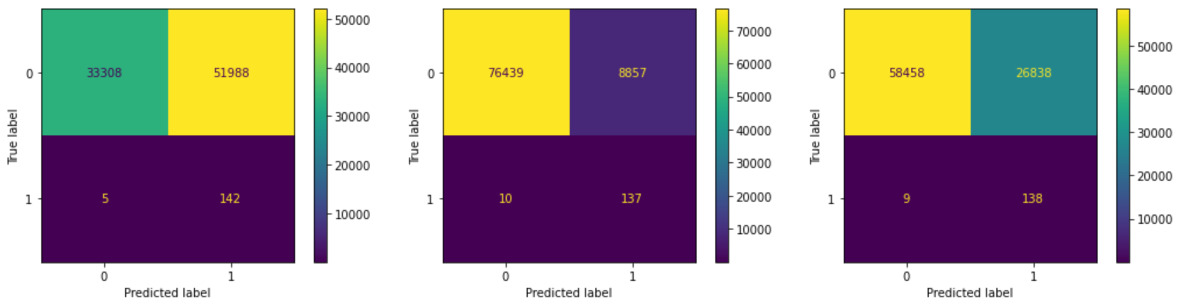


Figure 5-3: Confusion matrix of random under-sampled training data applied on test data. Left: Decision Tree Classifier. Centre: Extra Tree Classifier. Right: Random Forest Classifier

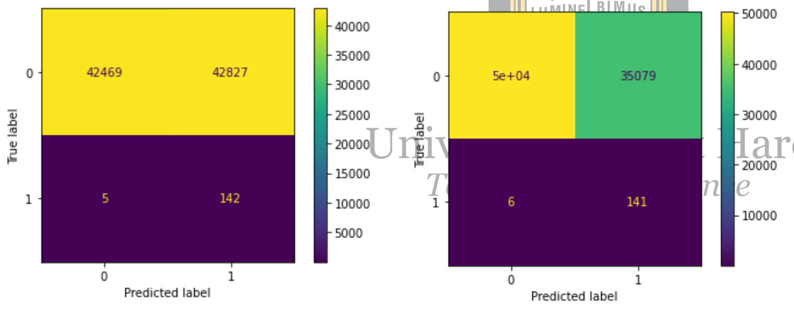


Figure 5-4: Confusion matrix of random under-sampled training data applied on test data. Left: AdaBoost Classifier. Centre: Gradient Boost Classifier.

Lastly, we trained our classifiers with a training set that was sampled using SMOTE. This technique reduces overfitting from classifiers and we will see how this will influence classification performance. Table 5-3 shows the results. Figures 5-5 to 5-6 show the confusion matrix plots for the classifiers.

Classifier	Accuracy	Precision		Recall		F1-Score	
		0	1	0	1	0	1
Decision Tree	99.7741	1	0.42	1	0.77	1	0.54
Extra Tree	99.9566	1	0.91	1	0.83	1	0.87
Random Forest	99.9520	1	0.90	1	0.82	1	0.85
AdaBoost	99.1034	1	0.14	0.99	0.84	1	0.24
Gradient Boost	99.5552	1	0.26	1	0.86	1	0.40

Table 5-3: Performance of classifiers using SMOTE

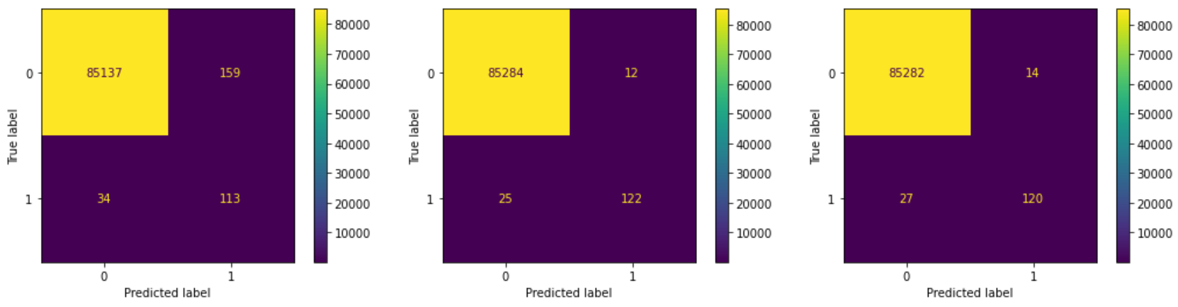


Figure 5-5: Confusion matrix of SMOTE training data applied on test data. Left: Decision Tree Classifier. Centre: Extra Tree Classifier. Right: Random Forest Classifier

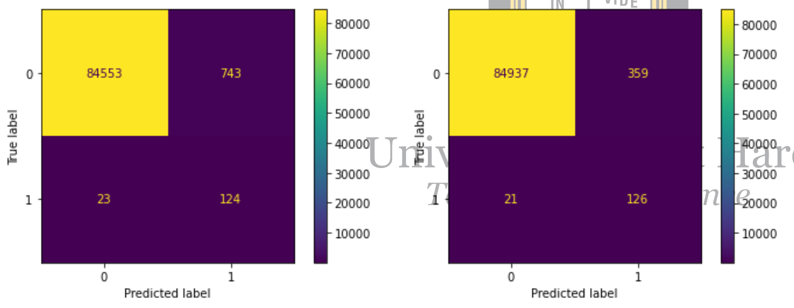


Figure 5-6: Confusion matrix of SMOTE training data applied on test data. Left: AdaBoost Classifier. Centre: Gradient Boost Classifier.

Table 5-4 and Figure 5-7 depict how accuracy is impacted by the sampling technique used in handling the data imbalance problem.

Classifier	**ROS**	**RUS**	**SMOTE**
Decision Tree	99.9262	39.1489	99.7741
Extra Tree	99.9531	89.6223	99.9566
Random Forest	99.9531	68.5790	99.9520
AdaBoost	98.7886	49.8706	99.1034
Gradient Boost	99.4522	58.9375	99.5552

Table 5-4: Comparison of different sampling techniques

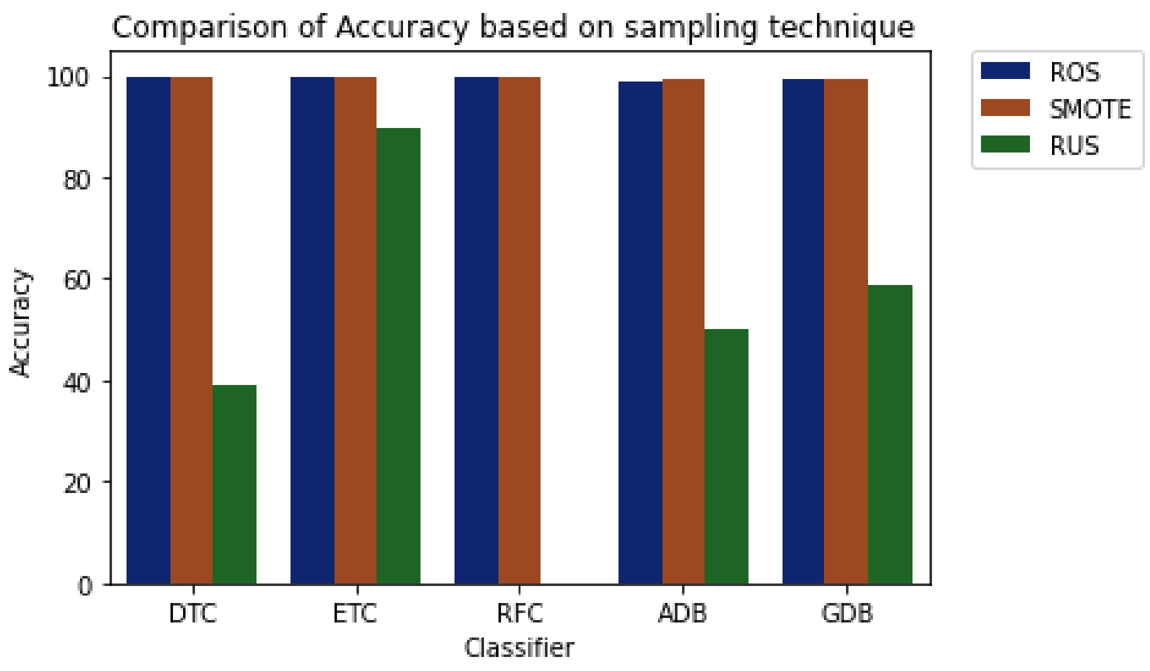


Figure 5-7: Comparison of how the accuracy of classifiers is impacted by ROS, RUS and SMOTE

5.2.2 **Sample size**

In this experiment, we investigated how sample size affected the classification performance of ensemble decision tree classifiers. The data was partitioned into subsets of 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 100% of the entire dataset.

Table 5-5: Training size and accuracy of classifiers

Sample Size	**Data Size**	**Decision Tree**	**Extra Tree**	**Random Forest**	**AdaBoost Classifier**	**Gradient Boost**
10%	19936	99.9169	99.9309	99.9180	99.9075	99.9075
20%	39873	99.9087	99.9344	99.9332	99.9286	99.9040
30%	59809	99.9180	99.9367	99.9356	99.9262	99.9204
40%	79746	99.9204	99.9438	99.9461	99.9332	99.9169
50%	99682	99.9321	99.9508	99.9473	99.9286	99.9356
60%	119618	99.9110	99.9520	99.9473	99.9274	99.9356
70%	139555	99.9169	99.9508	99.9485	99.9262	99.8993
80%	159491	99.9063	99.9508	99.9520	99.9250	99.9309
90%	179428	99.9063	99.9508	99.9554	99.9286	99.9087
100%	199364	99.9192	99.9531	99.9531	99.9098	99.9274

Table 5-5 shows the sample size, data size and the accuracy of the classifiers at varying sample sizes from 10% through to 100% of the training data. Figure 5-8 and figure 5-9 graphically depict the results.

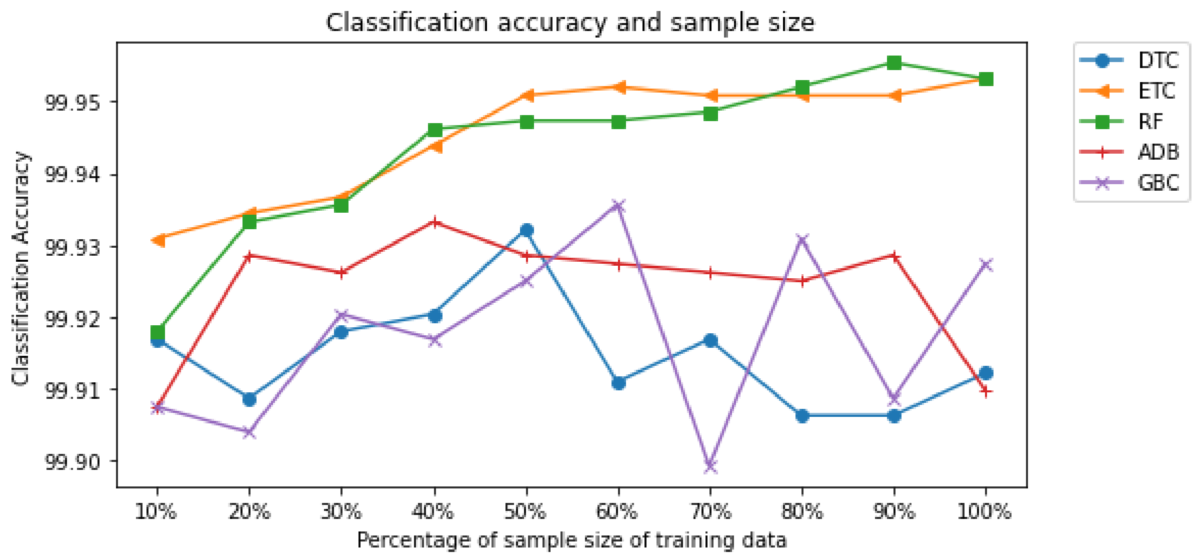


Figure 5-8: Classification accuracy for different training sizes

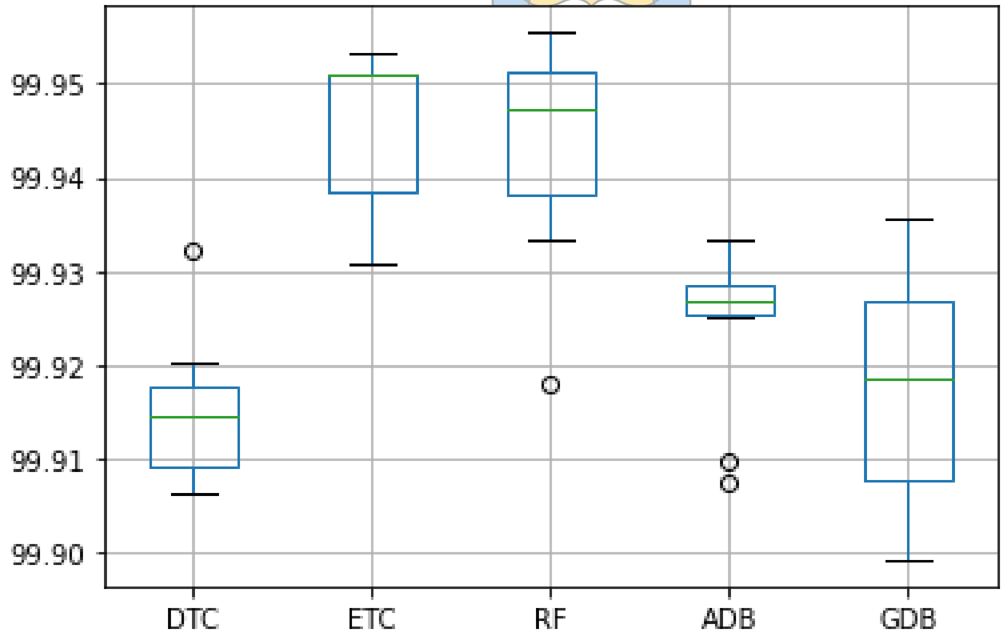


Figure 5-9: Boxplot of classifiers and accuracy

5.2.3 **Split Criterion**

Decision tree classifiers make use of either Gini or Entropy to look for the best node to split the data. Both methods evaluate how impure the data is to choose the best split that will result in a better tree classifier. In this experiment, we looked at how the two split criteria affected classification performance. Out of curiosity we also investigated how each of the split criteria affected execution times of ensemble tree classifiers. Table 5-6 shows accuracy and execution times for the classifiers when using varying split criteria. Figure 5-10 depicts how the accuracy of the classifiers was affected by split criteria. Figures 5-11 and 5-12 depict how varying split criteria affected execution time during the training and testing phases. Appendix E shows the code and results of both Gini and entropy.

Table 5-6: Accuracy, training and test time of ensemble tree classifiers based on a split criterion

Classifier	**Entropy Accuracy**	**Gini Accuracy**	**Entropy Train**	**Entropy Test**	**Gini Train**	**Gini Test**
DTC	99.9215	99.9192	11.6	0.0163	16.5	0.0138
ETC	99.9555	99.9531	16.7	1.19	18	1.27
RFC	99.9531	99.9496	148	0.731	200	0.932
ADB	99.9180	99.9157	12.5	0.0235	16.9	0.0249

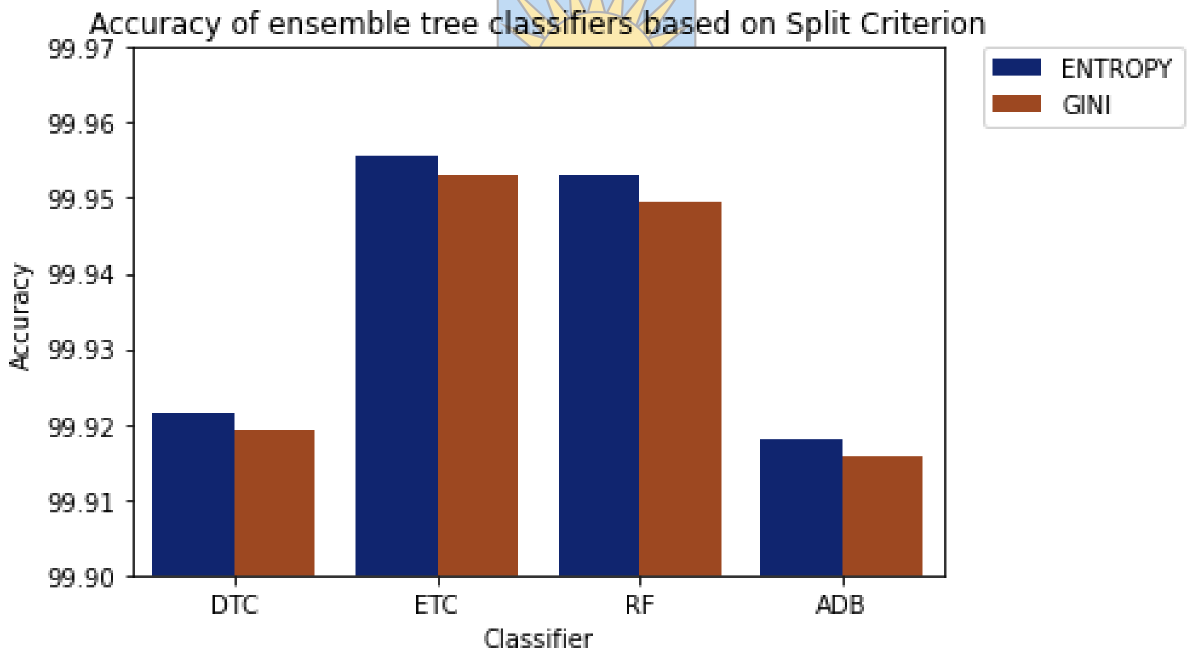


Figure 5-10: Accuracy of ensemble tree classifiers based on a split criterion

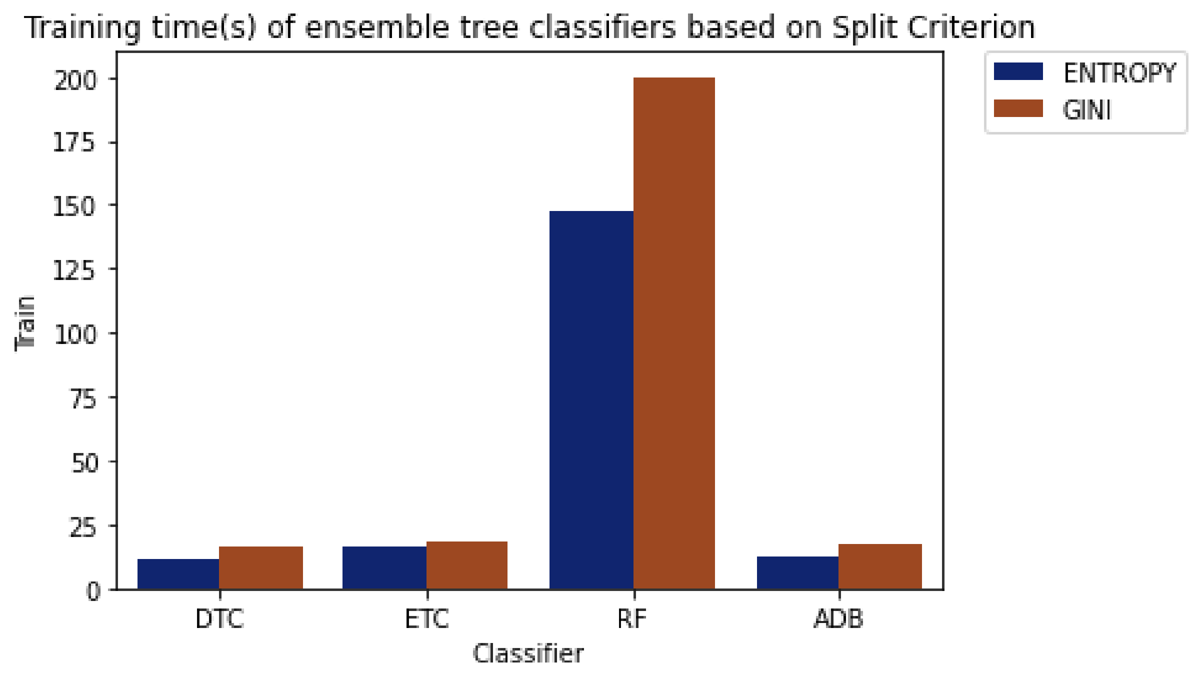


Figure 5-11: Training time of ensemble tree classifiers based on a split criterion

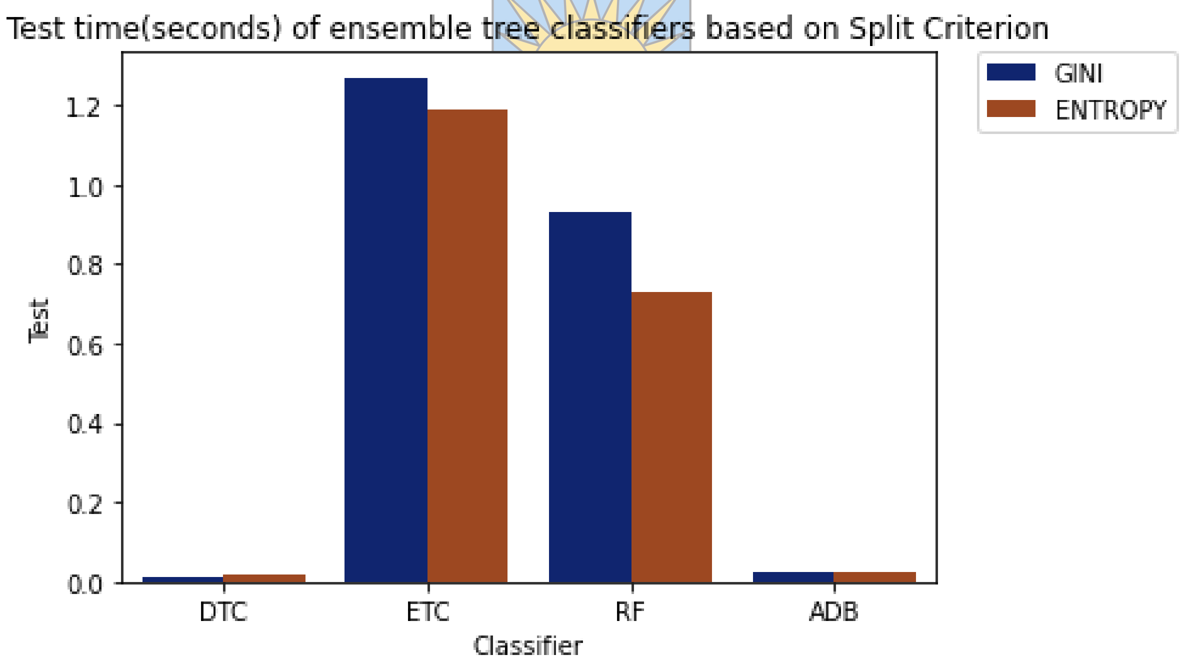


Figure 5-12: Test time of ensemble tree classifiers based on a split criterion

5.2.4 Supervised vs unsupervised ensemble tree classifiers

There are different types of learning techniques that are used by machine learning algorithms. The most popular are supervised and unsupervised learning. Ensemble tree classifiers also can use the two-learning technique in performing classification tasks. In this experiment, we explore if the choice of learning technique leads to an improved classifier when detecting credit card fraud. The Isolation Forest is an unsupervised anomaly detection algorithm proposed by Liu et al. (2008). We explored how the iForest compared to supervised algorithms in detecting fraud. Table 5-7 shows the accuracy of the algorithms when detecting credit card fraud. Figure 5-13 depicts a comparison of supervised and unsupervised ensemble tree algorithms.

Table 5-7: Comparison of classification performance of supervised and unsupervised algorithms in detecting credit card fraud

Classifier	**Learning**	**Accuracy**
Decision Tree	Supervised	99.9192
Extra Tree	Supervised	99.9531
Random Forest	Supervised	99.9531
Adaboost	Supervised	99.9098
Gradient Boost	Supervised	99.9274
Isolation Forest	Unsupervised	99.7659

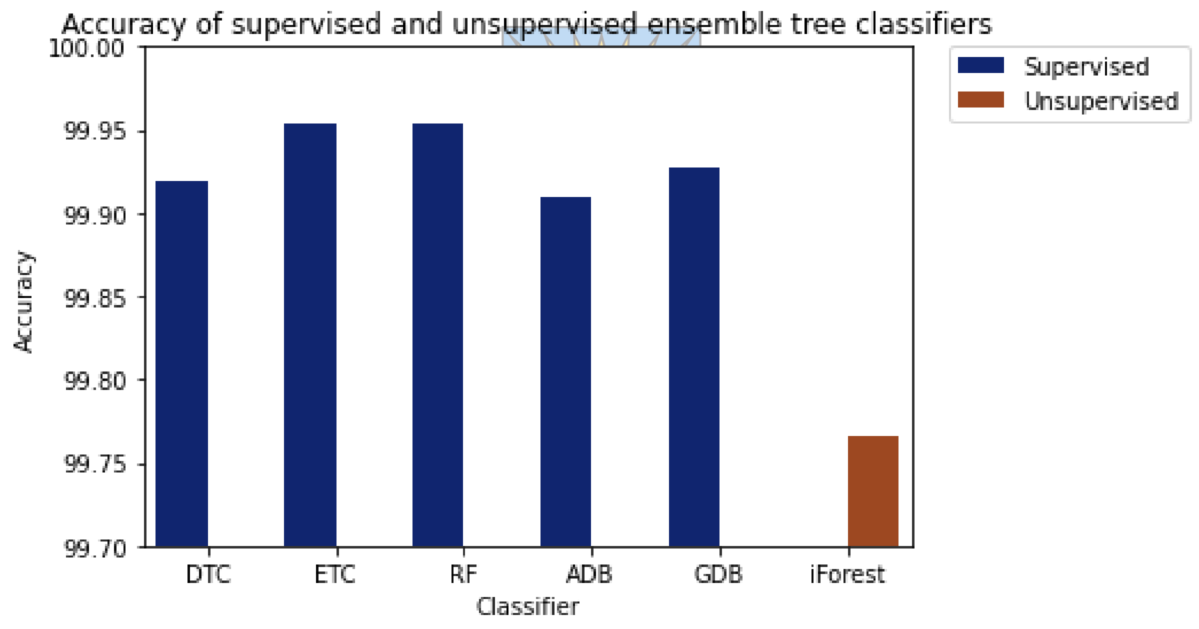


Figure 5-13: A comparison of accuracy between supervised and unsupervised ensemble tree classifiers

We also considered the AUPRC because our data is highly imbalanced. AUPRC ignores true negatives so it is useful to see how true positives or fraudulent transactions are classified.

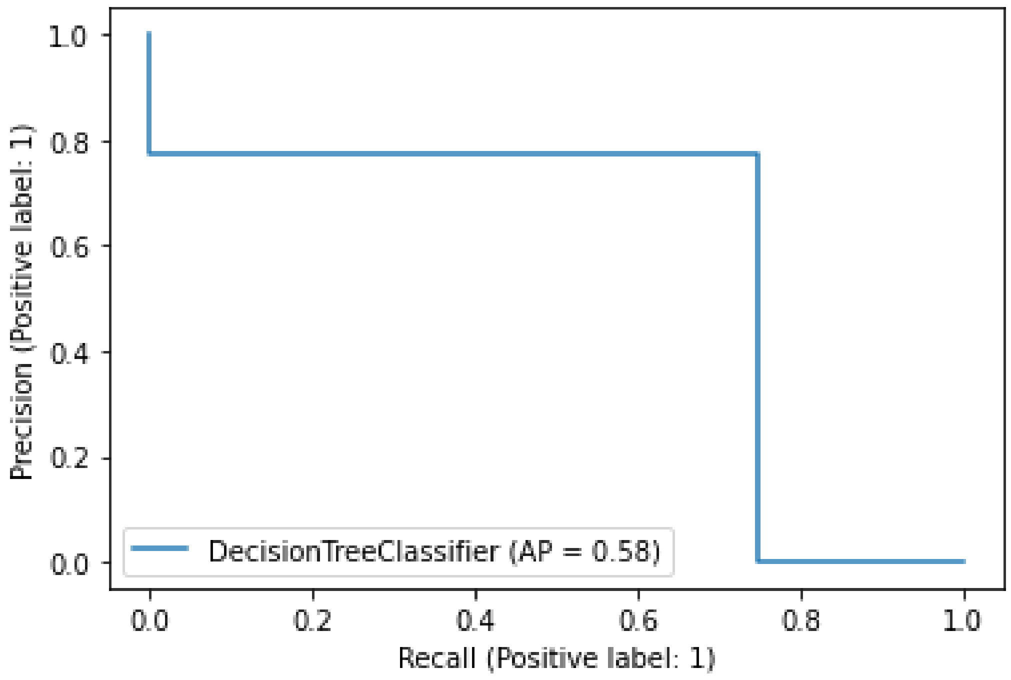


Figure 5-14: AUPRC for Decision Tree Classifier

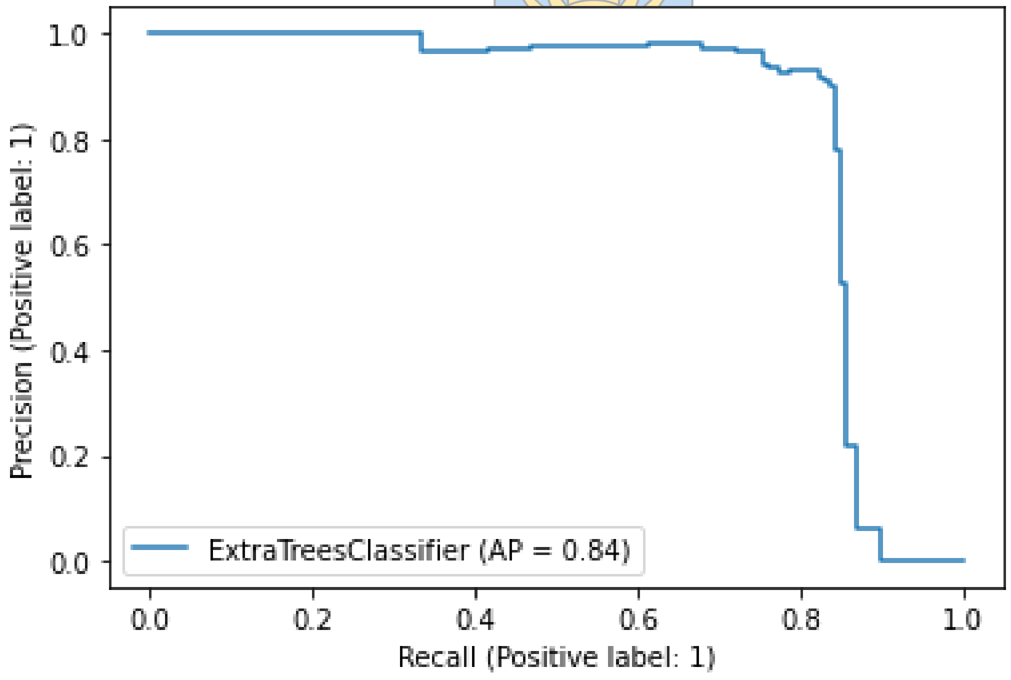


Figure 5-15: AUPRC for Extra Tree Classifier

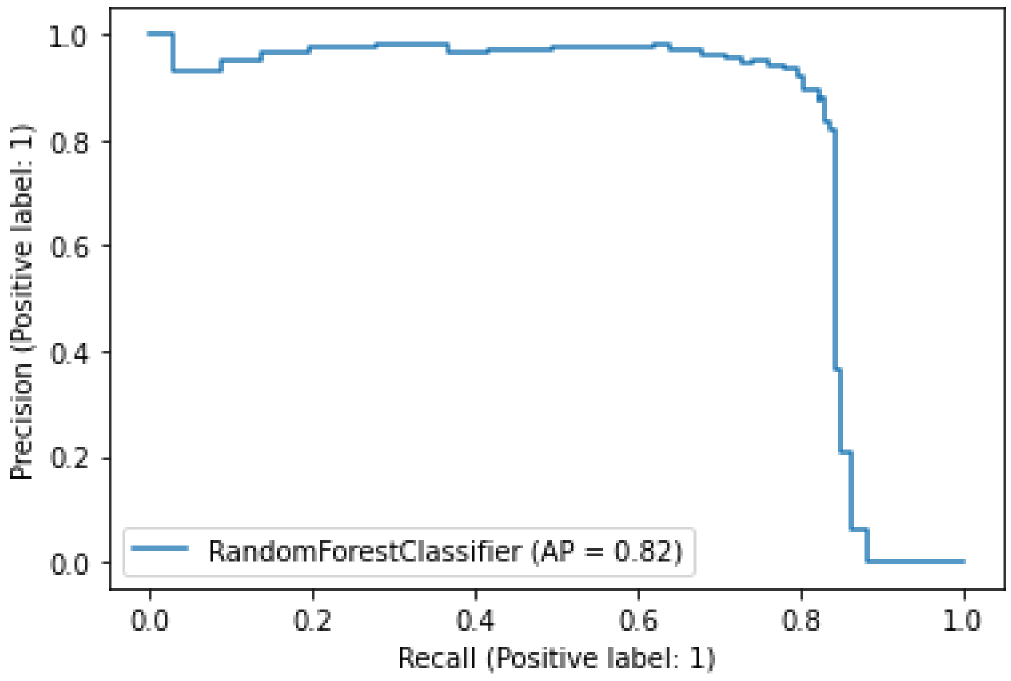


Figure 5-16: AUPRC for Random Forest Classifier

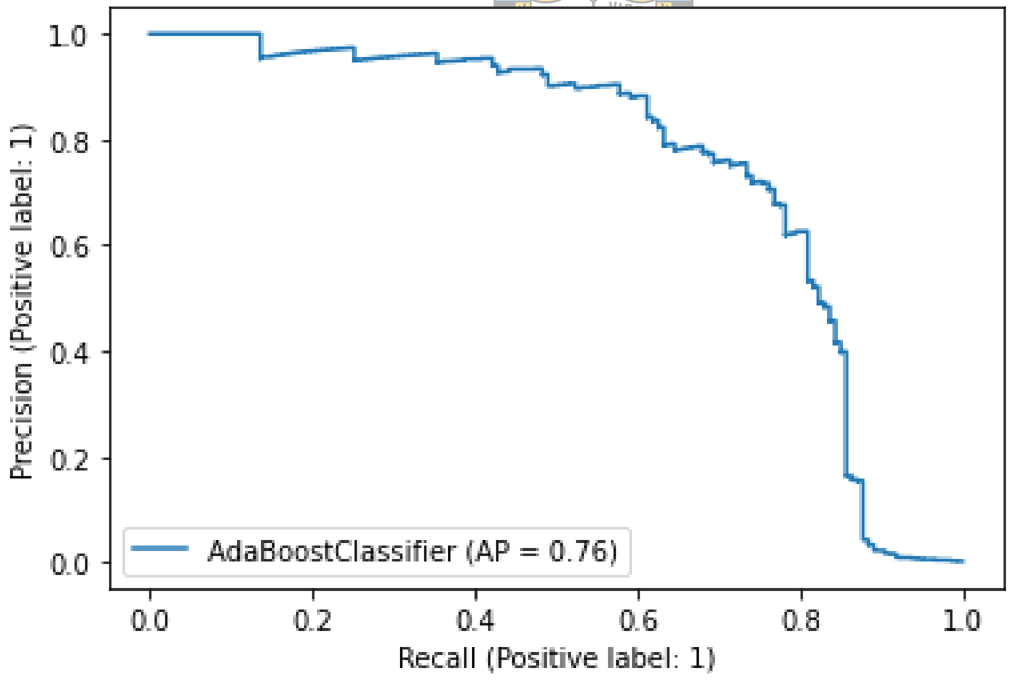


Figure 5-17: AUPRC for AdaBoost Classifier

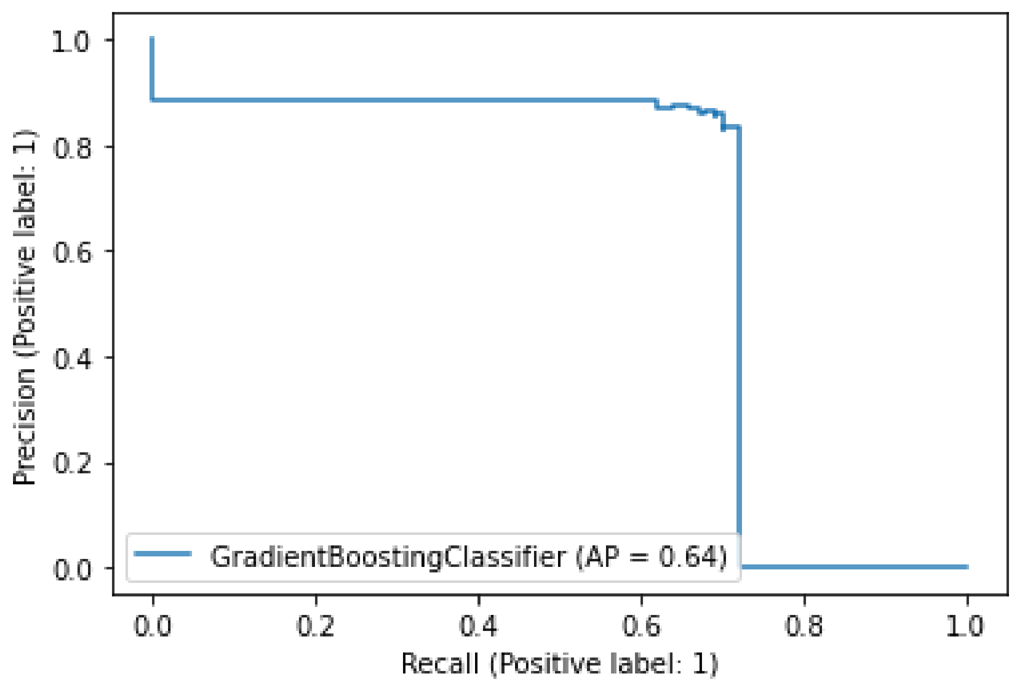


Figure 5-18: AUPRC for Gradient Boost Classifier

5.3 Discussions

This subsection is devoted to discussing the results that have been presented in section 5.2. The results presented are based on experiments that comprised of parameters that were kept constant while varying others to observe the behaviour of classifiers. For example, in one of the experiments, we investigated how the choice of split criterion influenced the performance of classifiers. To achieve this, we used spilt criterion as a variable while keeping all the other parameters like sample size and training data constant. The goal was to determine how different parameters affect the classification performance of ensemble decision tree classifiers in credit card fraud detection.

5.3.1 **Sampling Technique**

In this section, we analyse how the choice of sampling technique affects the classification performance of ensemble decision tree classifiers in the detection of credit card fraud. We start our analysis by making reference to Figure 5-7 which depicts the classification accuracy of different ensemble tree classifiers against sampling techniques. In Figure 5-7 it can be seen that oversampling produced better classification accuracy as compared to undersampling. Table 5-4 shows that for RUS, the ETC classifier achieved the highest accuracy of 89.6223% while for ROS the highest accuracy was 99.9531% with the ETC classifier. For SMOTE the highest accuracy was 99.9566% with the ETC classifier. Based on the results SMOTE technique is the favourite of the 3 techniques investigated as it produced the highest classification accuracy. SMOTE outperformed ROS as it reduces the problem of overfitting (Shakya, 2018).

Figures 5.-1 and 5-2 depict the confusion matrix of the ensemble classifiers when using ROS. When using ROS, the ETC classifier had the lowest rate of false positives, 3, and ADB had the worst rate of false positives, 1012. Figures 5-3 and 5-4 depict the confusion matrix of the ensemble classifiers when using RUS. When using RUS, the ETC classifier had the lowest false positives of 8 859 while the DTC classifier had the highest rate of false positives, 51 988. Figures 5-5 and 5-6 depict the confusion matrix of the ensemble classifiers when using SMOTE. When using SMOTE, the ETC classifier had the lowest number of false positives, 12, while the ADB classifier had the highest ratio of false positives, 359.

Based on the rate of false positives, the ROS technique would be the best to use as it produced the lowest number of false positives. Marrying the two, Accuracy and rate of False positives, the ideal sampling technique would be ROS as its accuracy is insignificantly behind that produced when using SMOTE and has the best rate of false positives. The best classifier to detect fraud and keep false positives under check is the ETC classifier trained on the data that has been sampled using the random oversampling technique.

5.3.2 Sample size

This section deals with the analysis of how varying training sample sizes affects the classification performance of ensemble tree classifiers in detecting credit card fraud. Firstly, figure 5-8 shows a general improvement in accuracy as the training sample size increases. The graph shows a clear trend for RF, ETC Classifiers as the accuracy increases steadily as the training sample size increases. ADB, GDB and DTC classifier increase proportionally to increase in sample size until approximately 50% of the sample size thereafter the graph becomes haphazard as indicated by the sharp decline from 50% to 80% of the training sample. This decline can be attributed to bias as the data becomes highly skewed as the sample size increases. The training data has fewer fraudulent transactions compared to normal instances and this variance will continue to grow as sample size increases as more and more normal points are added compared to fraudulent points. The RF classifier achieved its highest accuracy of 99.9554% at 90% of the training sample size. The ETC classifier achieved its highest accuracy of 99.9531% at 100% of the training sample. The sharp decline of the DTC, ADB and GDB classifiers from 50% of the training sample suggests that they do not perform well when exposed to very large datasets.

Figure 5-9 depicts how the range of accuracy is spread for the different ensemble tree classifiers. DTC classifier has a thin box which suggests that its accuracies are more dependable. However, there is a dot outside the maximum of the bar plot which shows the presence of an outlier, which in this case is a rare high accuracy. The ETC classifier has a relatively narrow range which shows less variance. Its median, however, is interesting as it is on the same point as the upper quartile(Q3) of the box plot. There is also a small gap between the median and the maximum accuracy which clearly shows that the ETC classifiers will give an accuracy close to the maximum accuracy with high dependence. The ETC classifier has no outliers. The RF classifier bar plot is almost similar to the ETC classifier bar plot but differs in that the RF classifier has a higher max and its median is lower than that of the ETC classifier. There is also an outlier that shows an unusually low accuracy that was recorded as the sample size varied. Adaboost classifier had a thin box which shows consistency but two outliers show a lower accuracy compared to the rest of the recorded accuracies. Gradient Boost classifier has a large range and thicker box which shows that the accuracies are not dependable as they vary so much when sample size varies.

5.3.3 Split criterion

This section deals with the analysis of how the choice of split criterion affects the classification performance of ensemble tree classifiers in detecting credit card fraud. As discussed in sections 3.1.1 and 3.1.2 two impurity measures are used in decision trees. These are Entropy and Gini index. Table 5-6 shows the accuracy, training time and test time of the ensemble tree classifiers using Gini and Entropy as split criteria. Figure 5-10 clearly shows that ensemble tree classifiers produce better classification accuracy when entropy is used as the split criterion. For all the classifiers in this study, the use of entropy resulted in higher accuracy compared to using the Gini index. The ETC classier had the highest accuracy of 99.9555% when using entropy, followed by the RF classifier which had an accuracy of 99.9531% using entropy.

Figure 5-11 shows how the choice of split criterion affected the training times of the classifiers. Training times for all the classifiers were highest when the Gini index was used as a split criterion. This clearly shows that entropy increases classification performance and has the least training time complexity. RF classifier had the highest training times for both split criteria, that is, 200 seconds using the Gini index and 150 seconds using entropy. The rest of the classifiers had training times under 25 seconds for both split criteria. Although RF uses a similar approach to ETC classifiers have a longer execution time. This may result from RF selecting the optimal splits which will require more time while ETC classifiers do not select optimal splits but split the data randomly.

Figure 5-12 depicts how the choice of split criterion affected the time required for the classifiers to compute predictions of the test data set. Classifiers with the Gini index generally had a greater execution time compared to those that used entropy. The ETC classifier had the highest execution time, slightly above 1.2 seconds, compared to the other classifiers. RF had a lower execution time compared to the ETC classifier. However, DTC and Adaboost had almost equal execution times for both split criteria. Based on split criterion and execution times, the ETC classifier gives the highest accuracy and has a relatively low training time. Despite having the highest test execution time, it remains the better classifier as it has better accuracy and far lower training time compared to its closest rival, the RF classifier.

5.3.4 **Supervised vs unsupervised ensemble tree classifiers**

This section deals with the comparison of supervised and unsupervised classifiers to determine the learning method that gives better classification performance when detecting credit card fraud using ensemble tree classifiers. Table 5-7 shows the accuracy of both supervised and unsupervised ensemble tree classifiers in detecting credit card fraud. The RF and the ETC classifier had the highest classification accuracy of 99.9531%. iForest had the least classification accuracy of 99.7659%. Figure 5-13 depicts a comparison of the accuracy of ensemble tree classifiers for supervised and unsupervised learning. Figure 5-14 to 18 depicted the AUPRC for the ensemble tree classifiers. The ETC classifier had the highest AP score of 0.84 and the DTC had the lowest of 0.58.

Lui (2008) stated that iForest can detect anomalies even if the training data has no anomalies because it has a contamination feature which allows it to estimate a certain percentage of the data to be anomalies. The iForest did not do so in this case as fraudulent transactions were too close to the normal points and as a result, required more random split to separate them. This resulted in higher path length resulting in most fraudulent transactions being classified as not fraud. The iForest had very low accuracy and gave the highest number of false-positive compared to all the supervised classifiers.

5.4 Summary

This chapter discussed the classification performance of ensemble decision tree classifiers when subjected to varying empirical settings. The experiments were described in detail in chapter 4 and a detailed analysis of the results was presented and discussed in this chapter. Different graphs and plots were used to depict the observations made and conclusions were drawn based on the graphs. For all the experiments conducted the ETC classifier gave the highest effectiveness in detecting fraud and also minimising false positives. The Random Forest also proved to be effective in detecting credit card fraud. The isolation forest did not show effectiveness compared to other ensemble decision tree classifiers. This may be as a result of parameters like contamination which makes the model have bias. The model will be biased as contamination estimates the percentage of dataset that is expected to be fraud forcing the model to aim for that percentage.

The final chapter of this dissertation is for the conclusion and future works that can be explored from this research.

Chapter Six

6 Conclusion and Future Work

6.1 Introduction

Our study was an empirical investigation to answer the following critical questions: how do misclassifications impact companies that use e-commerce? how does the choice of sampling technique impact the performance of ensemble decision tree classifiers? how does the sample size of the training dataset impact the classification performance of ensemble decision tree classifiers? how does split criterion affect the performance of ensemble decision tree classifiers and how supervised and unsupervised ensemble tree classifiers compare in performing classification tasks. We made use of e-commerce data provided by Vesta corporation to perform experiments to help answer these questions. This chapter aims to consolidate the research questions and the findings of this study. We conclude this chapter by looking at future work that can be adopted from this study.

6.2 Empirical findings vs research questions

1) How do misclassifications impact companies that use e-commerce?

In chapter 2 a systematic literature review was done to explore causes and ways of handling misclassifications and how they impact businesses. From the review, there were strong indications that the cost of false declines might be harming companies more than harm from fraud. The main causes of misclassifications include lack of real-life data for training models, skewed data, concept drift and the lack of cooperation between banks to come up with more robust FDS. To handle misclassifications approaches could be implemented at the data level and algorithm level. At the data level concepts of sampling techniques like under sampling and oversampling could be used to balance the training data. At the algorithm level bagging and boosting algorithms could be used to establish classifiers with higher efficacy.

2) How does the choice of sampling technique impact the performance of ensemble decision tree classifiers?

Empirical experiments were done to investigate how varying sampling techniques impact the classification performance of ensemble tree classifiers when detecting fraud. The most popular techniques were undersampling the majority class or oversampling the minority class to have a balanced dataset. When random undersampling has been implemented the accuracy of classifiers dropped significantly and also the number of false positives. This is largely due to the small amount of training data as the minority class has less than 1% of the entire dataset. When the random oversampling technique and SMOTE were used the accuracy was high and the number of false positives was relatively low.

3) How does the sample size of the training dataset impact the classification performance of ensemble decision tree classifiers?

Empirical experiments were done and results were discussed in chapter 5 to establish the impact of sample size on the performance of ensemble tree classifiers. The results showed a general improvement in classification performance as the sample size increased. These were the expected results as most machine learning algorithms learn better when there are large volumes of training instances. The extra tree classifier and the random forest had the best overall performance as the sample size increased. The decision tree classifier, Adaboost and Gradient boost improved accuracy score until about 50% of the sample and thereafter the accuracy fluctuated.

4) How does split criterion affect the performance of ensemble decision tree classifiers?

For decision tree classifiers splitting of nodes is very important as it ensures that each point is optimally split to produce the highest information gain. The Gini index and entropy are the two main split criteria used by decision tree classifiers. When entropy was chosen for splitting nodes the best accuracy score was achieved for all the ensemble decision tree classifiers. The Gini index was not far off but the results showed that entropy was the better of the two when dealing with credit card fraud data. We also looked at how execution time was affected by the split criterion. From the results, it was observed that entropy took the least time to execute during the training stage while the Gini index had the shortest execution time during the testing stage.

5) How supervised and unsupervised ensemble tree classifiers compare in performing classification tasks.

The empirical experiment results for comparison between supervised and unsupervised ensemble tree classifiers in performing classification tasks were discussed in section 5.1.4. The ETC, DTC, Adaboost and Gradient Boost are supervised algorithms and their performance was compared to that of the iForest which is an unsupervised algorithm. All the supervised algorithms outperformed the iForest greatly. We, therefore, concluded that supervised algorithms perform better when dealing with credit card fraud detection compared to unsupervised algorithms.

6.3 Thesis Conclusion

The main research question was to investigate how well ensemble decision tree classifiers perform the classification task of detecting fraud. The research question was broken down into sub-questions that helped answer the main question. Based on the results collected in this research we can conclude that ensemble decision tree classifiers can detect credit card fraud with high accuracy and while also keeping the false positives minimum. Ensemble tree classifiers have parameters that need to be set and if these are not set correctly their performance can drop drastically. One such parameter is contamination. By default, contamination is set to 0.1 which tells the model that 10% of the data has anomalies. This parameter will influence the model as it will try to classify 10% of the data as anomalies.

6.4 Recommendations and Future work

Since our study answered many questions there remains questions that can be further explored. The study can be extended to explore how the models would behave if exposed to real-time test datasets. This would be crucial to observe the execution time required per each transaction. The study can also be extended to observe how the models would perform if data with more features are used. The data used in this study have undergone PCA to reduce features and hide sensitive data. It, therefore, remains ambiguous how the model's accuracy would change when all features are available such as location and time as well. Another area to explore in future would be investigating the feasibility of collaborations between financial institutions to come up with more robust FDS, while not compromising on privacy.

References

Aisha Abdallah, M. A. (2016). Fraud detection system: A survey. *Journal of Network and Computer Applications*, 90–113. doi:https://doi.org/10.1016/j.jnca.2016.04.007

Alejandro Correa Bahnsen∗, D. A. (2016). Feature engineering strategies for credit card fraud detection. *Expert Systems With Applications*, 134–142.

Awati, C. J., & More, R. (2019). A Review On Credit Card Fraud Detection Using Machine Learning. *International Journal of Scientific & Technology Research*, 1217 - 1219.

Axelsson, Jansson, M., & Mans. (2020). Federated Learning Used to Detect Credit Card Fraud. 1.

Azevedo, A. (2008). Kdd, semma and crisp-dm: a parallel overview. *Iadis European Conference Data Mining*.

Bakker Evan. (2016, July 29). *Business Insider*. Retrieved from Business Insider: https://www.businessinsider.com/us-bank-and-visa-are-working-to-reduce-false-declines-2016-10?IR=T

Bakker, E. (2016, July 29). Retrieved from Business Insider: https://www.businessinsider.com/the-false-declines-report-the-86-billion-problem-undermining-e-commerce-merchants-fraud-prevention-strategies-2016-7?IR=T

Bolton, R. J., & Hand, D. J. (2001). Unsupervised profiling methods for fraud detection. *Credit scoring and credit control VII*, 235-255.

Breiman, L. (2020, August 14). *berkeley.edu*. Retrieved from http://www.stat.berkeley.edu/~breiman/RandomForests: http://www.stat.berkeley.edu/~breiman/RandomForests

Breunig, M., Kriegel, H.-P., Raymond, & Sander, J. (2000). LOF: Identifying Density-Based Local Outliers. *Proc. ACM SIGMOD 2000 Int. Conf. On Management of Data, Dalles, TX, 2000*.

Budhram, T. (2016). Lost, stolen or skimmed : Overcoming credit card fraud in South Africa. *Institute for Security Studies*, 31-36.

C. Whitrow, D. J. Hand, P. Juszczak, D. Weston, N. M. Adams. (2008). Transaction aggregation as a strategy for credit card fraud detection. *Springer Science+Business Media, LLC*, 31-54.

Chan, P. K., Fan, W., Prodromidis, A. L., & Stolfo, S. J. (1999). Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems and Their Applications 14, no. 6*, 67-74.

Chawla Nitesh V, B. K. (2002). SMOTE: Synthetic Minority Over-sampling Technique). *J. Artif. Intell. Res. (JAIR)*, 321-357. Retrieved from Doi: 10.1613/jair.953

Chawla, N., Bowyer, K., Hall, L., & Philip, K. (2002). SMOTE: Synthetic Minority Over-sampling Technique). *J. Artif. Intell. Res. (JAIR)*, 321-357. Retrieved from Doi: 10.1613/jair.953

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *KDD*.

Chigada, J. (2020). A qualitative analysis of the feasibility of deploying biometric authentication systems to augment security protocols of bank card transactions. *South African Journal of Information Management*. Retrieved from https://doi.org/10.4102/

Credit Card Fraud Detection | Kaggle. (2020, 10 10). Retrieved from kaggle.com: https://www.kaggle.com/mlg-ulb/creditcardfraud

Creswell, J. (2009). Research Design: Qualitative, Quantitative and Mixed Method (3rd Ed.). *Los Angeles: SAGE Publications*.

Dal Pozzolo Andrea, B. G. (2015). Credit card fraud detection and concept-drift adaptation with delayed su-pervised information. *International Joint Conference on Neural Networks (IJCNN)*.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). Knowledge Discovery and Data Mining:Towards a Unifying Framework. *AAAI*.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Geurts, P., Ernst, D., & Louis, W. (2006). Extremely randomized trees. *Mach Learn (2006)*.

Glodek, M., Schels, M., & Schwenker, F. (2013). Ensemble Gaussian mixture models for probability density estimation. *Computational Statistics*, 127-138.

Guest Blog. (2018). *How to handle imbalanced classification problems in machine learning?*

Husejinovic, A. (2020). Credit card fraud detection using naive Bayesian and C4.5 decision tree classifiers. *Periodicals of Engineering and Natural Sciences*, 1 - 5.

Imane, Sadgali & Faouzia, Benabbou & Nawal, Sael. (2018). Detection of credit card fraud: State of art. *International Journal of Computer Network and Information Security*, 77-82.

Jason, B. (2020, 08 20). *How to Use ROC Curves and Precision-Recall Curves for Classification in Python*. Retrieved from https://machinelearningmastery.com/ roc-curves-and-precision-recall-curves-for-classification-in-\ python

Jehad Ali, R. K. (2012). Random Forests and Decision Trees. *IJCSI International Journal of Computer Science Issues*, 272-278.

Jha, S., Montserrat, G., & Christopher, J. (2012). Employing transaction aggregation strategy to detect credit card fraud. *Expert systems with applications, 39(16)*, 12650-12657.

Jismy, J., & Kesavaraj, D. (2019). Ensemble Learning Technique to Improve Classification Accuracy for Credit Data. *Journal of Emerging Technologies and Innovative Research (JETIR)*, 352-356.

Juszczak, P., Adams, N. M., Hand, D. J., Whitrow, C., & Weston, D. J. (2008). Off-the-peg and bespoke classifiers for fraud detection. *Computational Statistics & Data Analysis 52, no. 9* , 4521-4532.

K. Randhawa, C. K. Loo, M. Seera, C. P. Lim and A. K. Nandi. (2018). Credit Card Fraud Detection Using AdaBoost and Majority Voting," in. *IEEE Access, vol. 6*, 14277 - 14284.

Kerecha, N. T. (2021, 11 10). *Entropy and Gini Index In Decision Trees*. Retrieved from Geek Culture: https://medium.com/geekculture/entropy-and-gini-index-in-decision-trees-cb99ba5d7dcc

Kim, J., & Scott, C. D. (2012). Robust Kernel Density Estimation. *Journal of Machine Learning Research 13 (2012) 2529-2565*.

Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. *Prog Artif Intell (2016) 5:221–232*, 221–232.

Kriegel, H.-P., Schubert, M., & Zimek, A. (2008). Angle-Based Outlier Detection in High-dimensional Data.

Liberman, N. (Jan 2017). *Decision trees and random forests towards data science.*

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. *DOI: 10.1109/ICDM.2008.17 · Source: IEEE Xplore*.

Lu, H., & Mazumder, R. (2018). Randomized Gradient Boosting Machine.

Lu, Q., & Chunhua, J. (2011). Research on credit card fraud detection model based on class weighted support vector machine. *Journal of Convergence Information Technology, 6(1)*.

McMahan, B., Eider, M., Daniel, R., Seth, H., & Blaise, A. A. (February 2016). Communication-E cient Learning of Deep Networks from Decentralized Data. (Doi: 1602.05629.).

Modi, K. (2017). Review on fraud detection methods in credit card transactions. *International conference on intelligent computing and control (I2C2)*, 102-106.

Ngai, E., Hu, Y., Wong, Y., Chen, Y., & Sun, X. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision support systems, 50(3)*, 559-569.

Omankwu, O. C., Nwagu, C. K., & Inyiama, H. (2018). Computer Science Research Methodologies. *International Journal of Computer Science and Information Security (IJCSIS),Vol. 16, No. 4, April 2018*, 142-144.

Onukwugha, C. (2018). Credit Card Fraud Detection System In Nigeria Banks Using Adaptive Data Mining And Intelligent Agents: A Review. *International journal of scientific & technology research volume 7, issue 7, ,* 176-183.

Panigrahi, Suvasini & Kundu, Amlan & Sural, Shamik & Majumdar, Arun. (2009). Credit card fraud detection. *A fusion approach using Dempster–Shafer theory and Bayesian learning*, 354-363.

Peter, K., Brendan, M., & Brendan, A. (December 2019). Advances and Open Problems in Federated Learning. *Doi: 1912.04977.*

Pevný, T. (2016). Loda: Lightweight on-line detector of anomalies. *Department of Computers and Engineering*.

Roweida Mohammed, J. R. (2019). Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results. 2.

SABRIC. (2008). *Credit card fraud South Africa.*

SABRIC. (2014). *Card Fraud Booklet.*

SABRIC. (2019). *SABRIC annual crime stats.* South Africa: SABRIC.

Samaneh Sorournejad, Z. Z. (2016). A Survey of Credit Card Fraud Detection Techniques: Data and Technique Oriented Perspective. 1-26.

Scholkopf, B., Williamson, R., Smola, A., Shawe-Taylort, J., & Platt, J. (1999). Support Vector Method for Novelty Detection. *Microsoft Research Ltd., 1 Guildhall Street, Cambridge, UK*, 583-588.

Shakya, R. (2018). Application of Machine Learning Techniques in Credit Card Fraud Detection. *UNLV Theses, Dissertations, Professional Papers, and Capstones*, 23.

Shirgave, Suresh & Awati, Chetan & More, Rashmi & Patil, Sonam. (2019). A Review On Credit Card Fraud Detection Using Machine Learning. *International Journal of Scientific & Technology Research*, 12-17.

Sonal, Mehndiratta & Mr Kamal, Gupta. (2019). Credit Card Fraud Detection Techniques: A Review. *International Journal of Computer Science and Mobile Computing*, 43-49.

South African Police Service. (2011). *Annual Report.*

Tableau. (2021, 9 9). *What Is Data Visualization? Definition, Examples, And Learning Resources*. Retrieved from tableau.com: https://www.tableau.com/learn/articles/data-visualization

Tax, D. M., & Duin, R. P. (2004). Support Vector Data Description. *Pattern Recognition Group, Faculty of Applied Sciences, Delft University of Technology, Lorentzweg 1,2628 CJ Delft, The Netherlands*, 45-66.

Team Great Learning. (2020, May 28). Retrieved from The Ultimate Guide to AdaBoost Algorithm | What is AdaBoost Algorithm?: https://lh6.googleusercontent.com/4knEPzYYjQOkHDOtpaZHSF0Pb7X8D52G6ij9g

31TkLL7vPq9fUb-RhmiW0u2LisDPZF1tD1JEDyrCggtsuF-324YHPXAwMEbqfzJSDqYqaDlzuGF_lfkQSiA04WYVid8aA7g0-nM

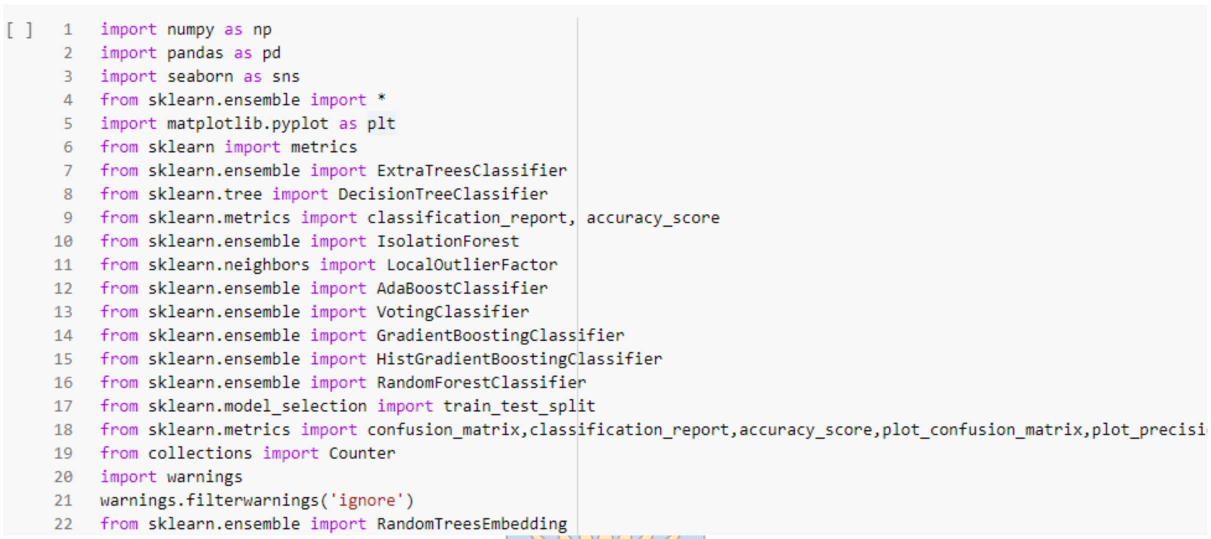
Visa, S., Ramsay, B., Ralescu, A., & Knaap, E. v. (2011). Confusion Matrix-based Feature Selection. *Conference Paper*.

X. Zhang, Y. H. (2019). HOBA: A Novel Feature Engineering Methodology for Credit Card Fraud Detection with a Deep Learning Architecture. *Information Sciences*, 10.1016/j.ins.2019.05.023. Retrieved from https://doi.org/10.1016/j.ins.2019.05.023

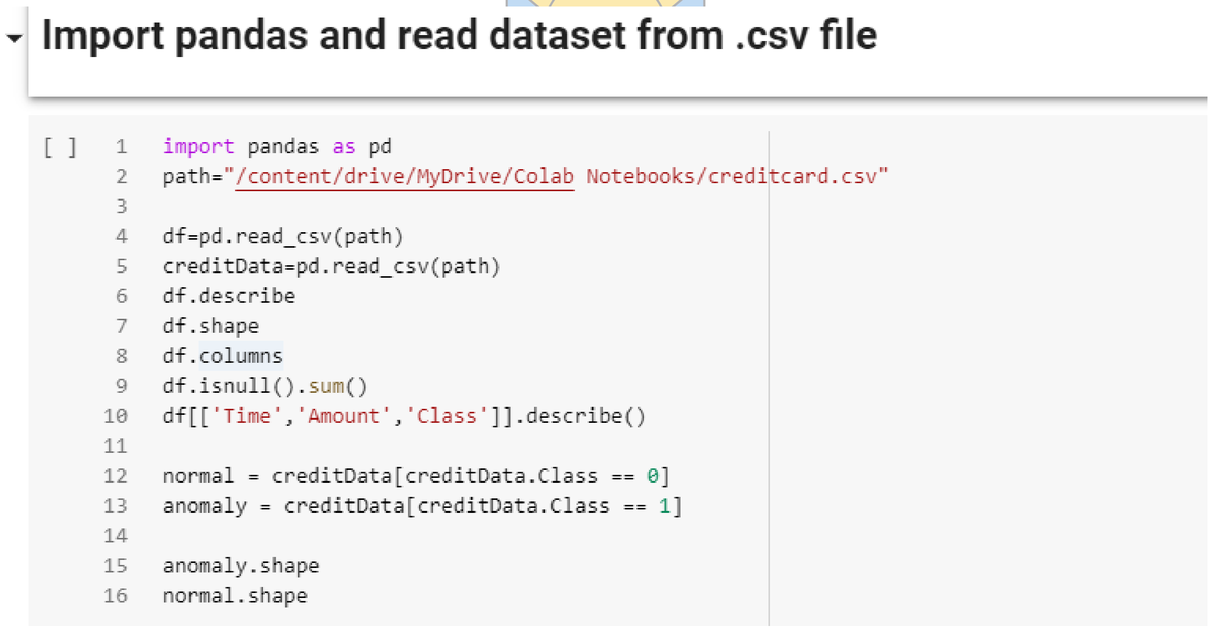
Zareapoora, M., & Shamsolmoalia, P. (2015). Application of Credit Card Fraud Detection: Based on Bagging Ensemble Classifier. *International Conference on Intelligent Computing, Communication & Convergence*.

Appendix A

Importing libraries and credit card fraud data file into Notebook.



```
[ ]
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.ensemble import *
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import HistGradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score,plot_confusion_matrix,plot_precisi
from collections import Counter
import warnings
warnings.filterwarnings('ignore')
from sklearn.ensemble import RandomTreesEmbedding
```

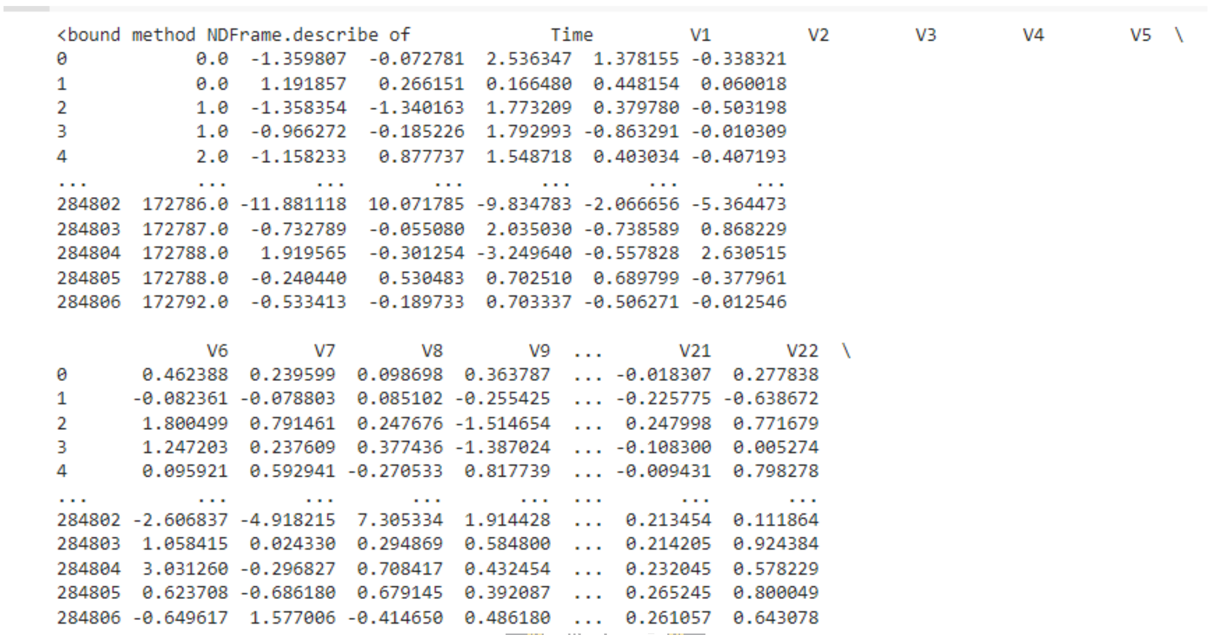


Import pandas and read dataset from .csv file

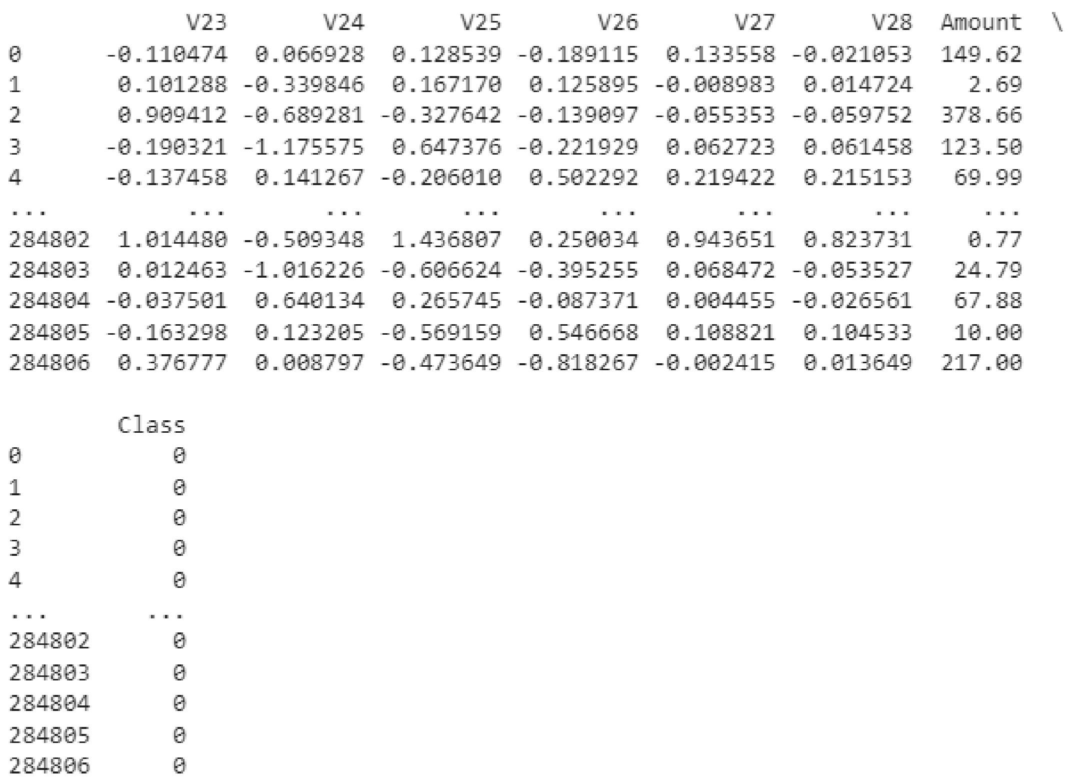
```
[ ]
import pandas as pd
path="/content/drive/MyDrive/Colab Notebooks/creditcard.csv"

df=pd.read_csv(path)
creditData=pd.read_csv(path)
df.describe
df.shape
df.columns
df.isnull().sum()
df[['Time','Amount','Class']].describe()

normal = creditData[creditData.Class == 0]
anomaly = creditData[creditData.Class == 1]

anomaly.shape
normal.shape
```

Appendix B

Description of the data set



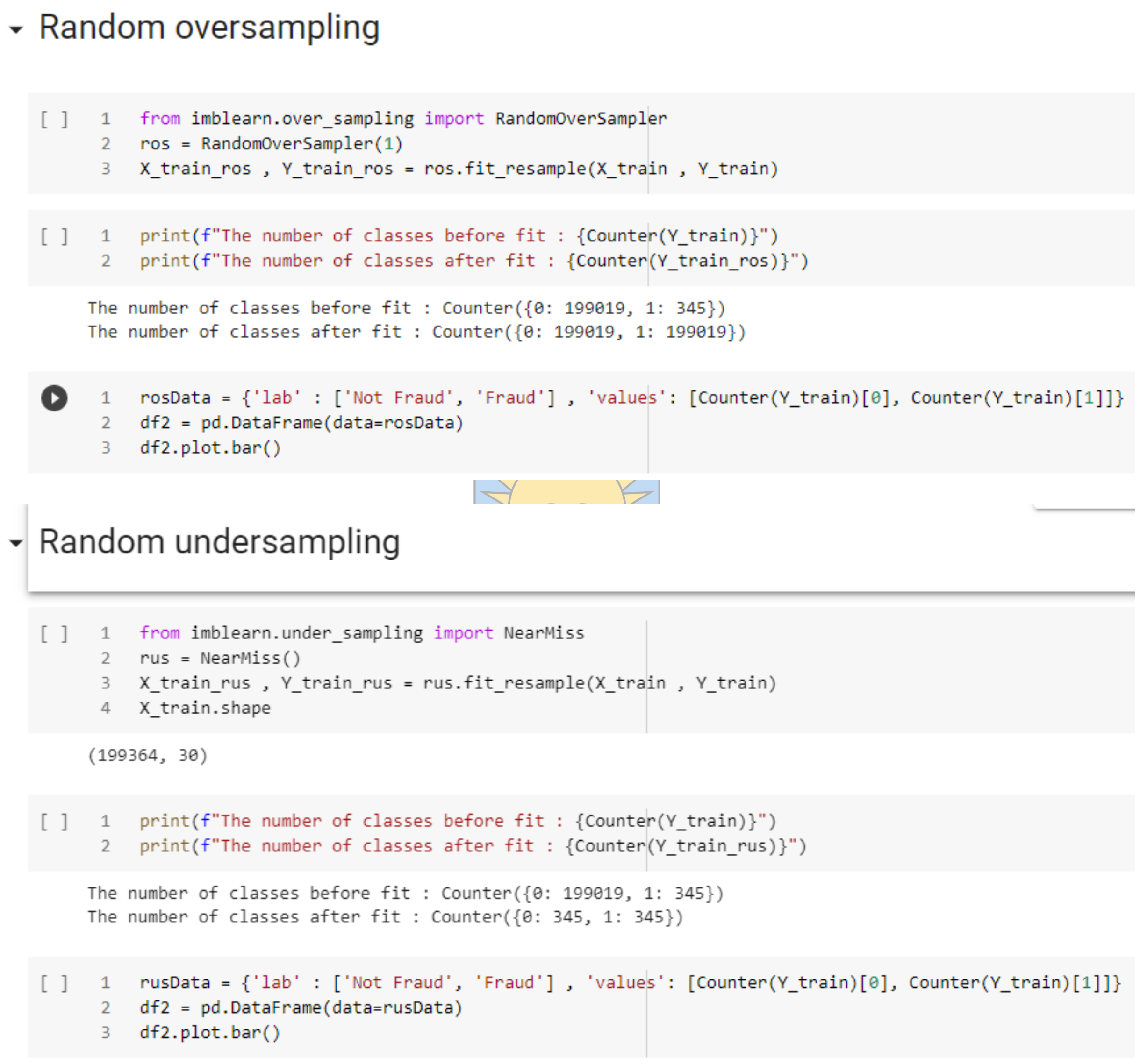
```
<bound method NDFrame.describe of            Time         V1         V2        V3        V4        V5  \
0            0.0  -1.359807  -0.072781  2.536347  1.378155 -0.338321
1            0.0   1.191857   0.266151  0.166480  0.448154  0.060018
2            1.0  -1.358354  -1.340163  1.773209  0.379780 -0.503198
3            1.0  -0.966272  -0.185226  1.792993 -0.863291 -0.010309
4            2.0  -1.158233   0.877737  1.548718  0.403034 -0.407193
...          ...        ...        ...       ...       ...       ...
284802  172786.0 -11.881118  10.071785 -9.834783 -2.066656 -5.364473
284803  172787.0  -0.732789  -0.055080  2.035030 -0.738589  0.868229
284804  172788.0   1.919565  -0.301254 -3.249640 -0.557828  2.630515
284805  172788.0  -0.240440   0.530483  0.702510  0.689799 -0.377961
284806  172792.0  -0.533413  -0.189733  0.703337 -0.506271 -0.012546

              V6        V7        V8        V9  ...       V21       V22  \
0       0.462388  0.239599  0.098698  0.363787  ... -0.018307  0.277838
1      -0.082361 -0.078803  0.085102 -0.255425  ... -0.225775 -0.638672
2       1.800499  0.791461  0.247676 -1.514654  ...  0.247998  0.771679
3       1.247203  0.237609  0.377436 -1.387024  ... -0.108300  0.005274
4       0.095921  0.592941 -0.270533  0.817739  ... -0.009431  0.798278
...          ...       ...       ...       ...  ...       ...       ...
284802 -2.606837 -4.918215  7.305334  1.914428  ...  0.213454  0.111864
284803  1.058415  0.024330  0.294869  0.584800  ...  0.214205  0.924384
284804  3.031260 -0.296827  0.708417  0.432454  ...  0.232045  0.578229
284805  0.623708 -0.686180  0.679145  0.392087  ...  0.265245  0.800049
284806 -0.649617  1.577006 -0.414650  0.486180  ...  0.261057  0.643078
```



```
             V23       V24       V25       V26       V27       V28  Amount  \
0      -0.110474  0.066928  0.128539 -0.189115  0.133558 -0.021053  149.62
1       0.101288 -0.339846  0.167170  0.125895 -0.008983  0.014724    2.69
2       0.909412 -0.689281 -0.327642 -0.139097 -0.055353 -0.059752  378.66
3      -0.190321 -1.175575  0.647376 -0.221929  0.062723  0.061458  123.50
4      -0.137458  0.141267 -0.206010  0.502292  0.219422  0.215153   69.99
...          ...       ...       ...       ...       ...       ...     ...
284802  1.014480 -0.509348  1.436807  0.250034  0.943651  0.823731    0.77
284803  0.012463 -1.016226 -0.606624 -0.395255  0.068472 -0.053527   24.79
284804 -0.037501  0.640134  0.265745 -0.087371  0.004455 -0.026561   67.88
284805 -0.163298  0.123205 -0.569159  0.546668  0.108821  0.104533   10.00
284806  0.376777  0.008797 -0.473649 -0.818267 -0.002415  0.013649  217.00

        Class
0           0
1           0
2           0
3           0
4           0
...       ...
284802      0
284803      0
284804      0
284805      0
284806      0
```

Appendix C

Code for ROS, RUS and SMOTE



Random oversampling

```
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(1)
X_train_ros , Y_train_ros = ros.fit_resample(X_train , Y_train)
```

```
print(f"The number of classes before fit : {Counter(Y_train)}")
print(f"The number of classes after fit : {Counter(Y_train_ros)}")
```

```
The number of classes before fit : Counter({0: 199019, 1: 345})
The number of classes after fit : Counter({0: 199019, 1: 199019})
```

```
rosData = {'lab' : ['Not Fraud', 'Fraud'] , 'values': [Counter(Y_train)[0], Counter(Y_train)[1]]}
df2 = pd.DataFrame(data=rosData)
df2.plot.bar()
```

Random undersampling

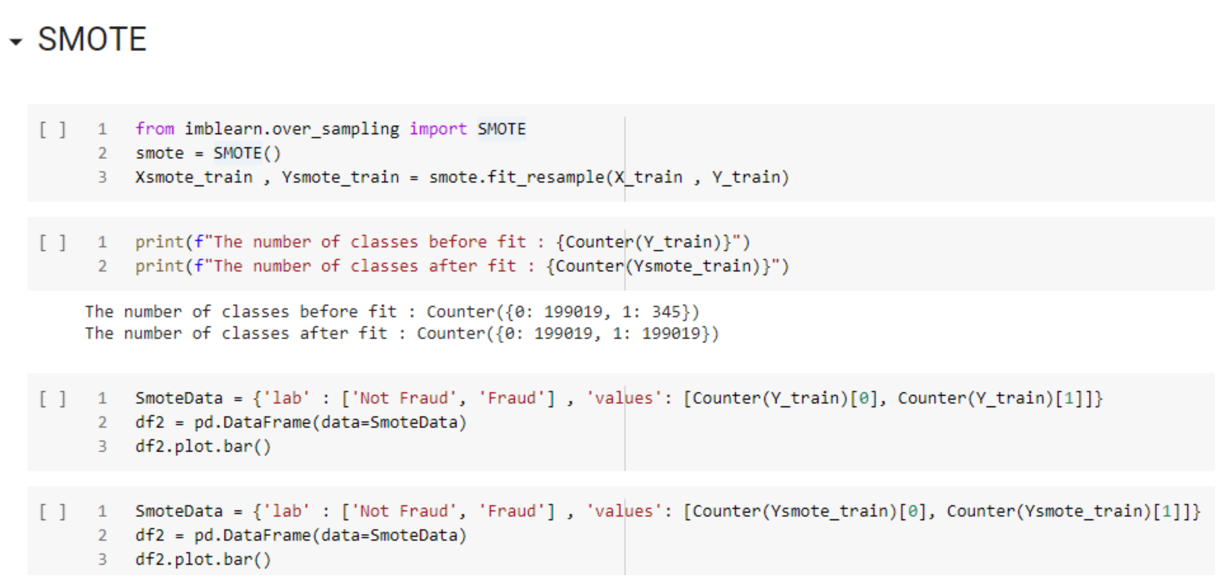
```
from imblearn.under_sampling import NearMiss
rus = NearMiss()
X_train_rus , Y_train_rus = rus.fit_resample(X_train , Y_train)
X_train.shape
```

```
(199364, 30)
```

```
print(f"The number of classes before fit : {Counter(Y_train)}")
print(f"The number of classes after fit : {Counter(Y_train_rus)}")
```

```
The number of classes before fit : Counter({0: 199019, 1: 345})
The number of classes after fit : Counter({0: 345, 1: 345})
```

```
rusData = {'lab' : ['Not Fraud', 'Fraud'] , 'values': [Counter(Y_train)[0], Counter(Y_train)[1]]}
df2 = pd.DataFrame(data=rusData)
df2.plot.bar()
```



SMOTE

```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
Xsmote_train , Ysmote_train = smote.fit_resample(X_train , Y_train)
```

```
print(f"The number of classes before fit : {Counter(Y_train)}")
print(f"The number of classes after fit : {Counter(Ysmote_train)}")
```

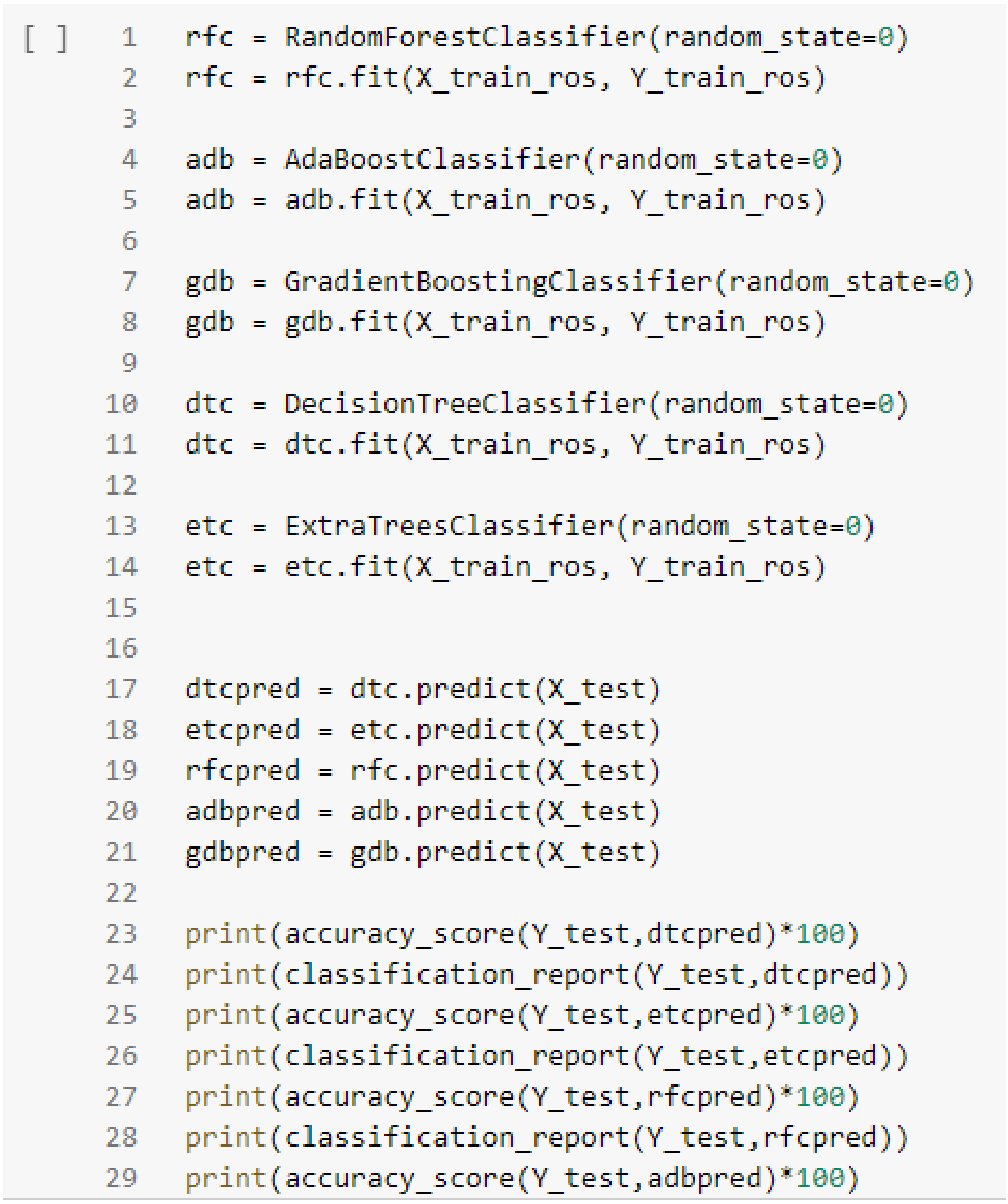
```
The number of classes before fit : Counter({0: 199019, 1: 345})
The number of classes after fit : Counter({0: 199019, 1: 199019})
```

```
SmoteData = {'lab' : ['Not Fraud', 'Fraud'] , 'values': [Counter(Y_train)[0], Counter(Y_train)[1]]}
df2 = pd.DataFrame(data=SmoteData)
df2.plot.bar()
```

```
SmoteData = {'lab' : ['Not Fraud', 'Fraud'] , 'values': [Counter(Ysmote_train)[0], Counter(Ysmote_train)[1]]}
df2 = pd.DataFrame(data=SmoteData)
df2.plot.bar()
```

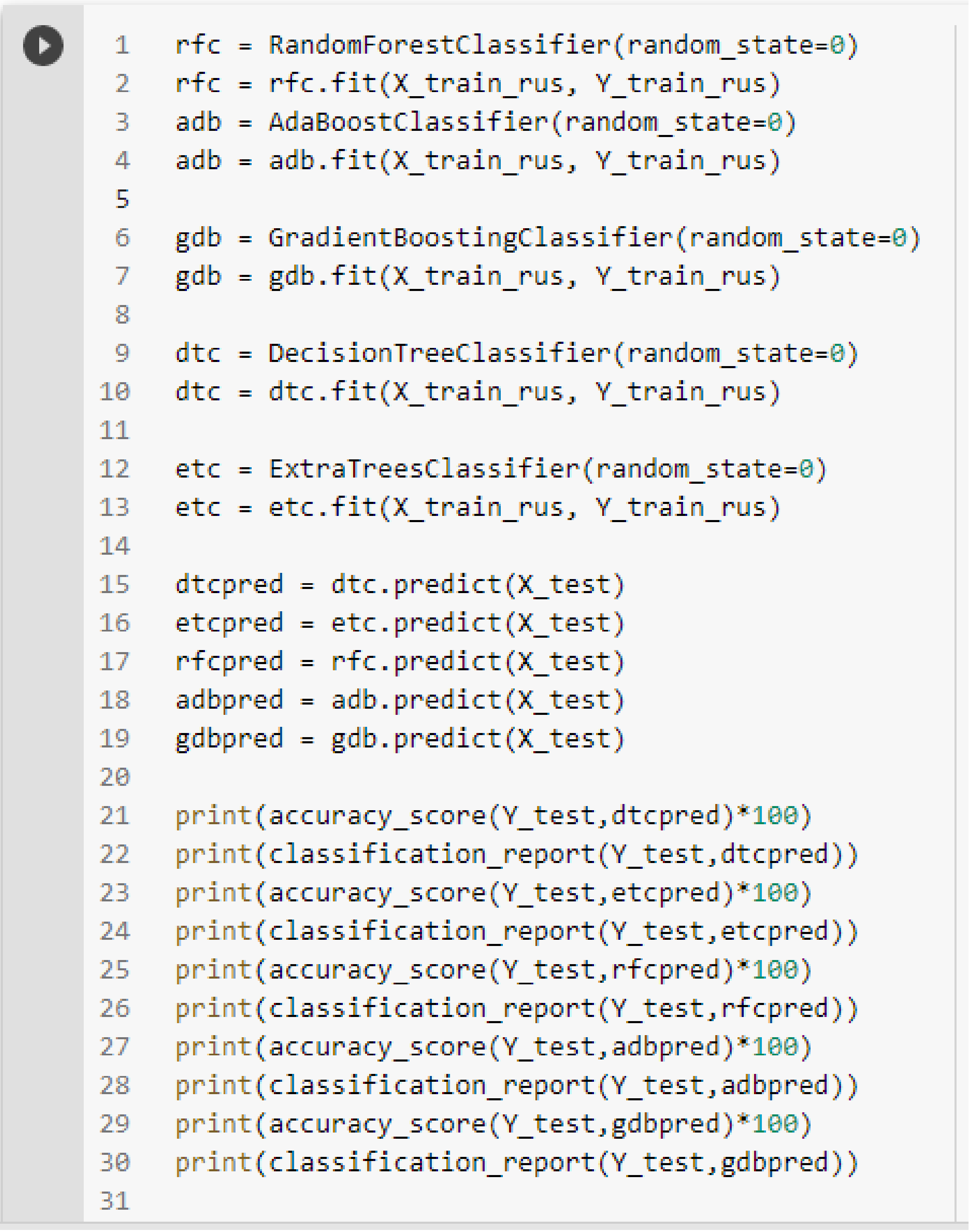
Appendix D

Code for training and testing classifiers using RUS, ROS and SMOTE



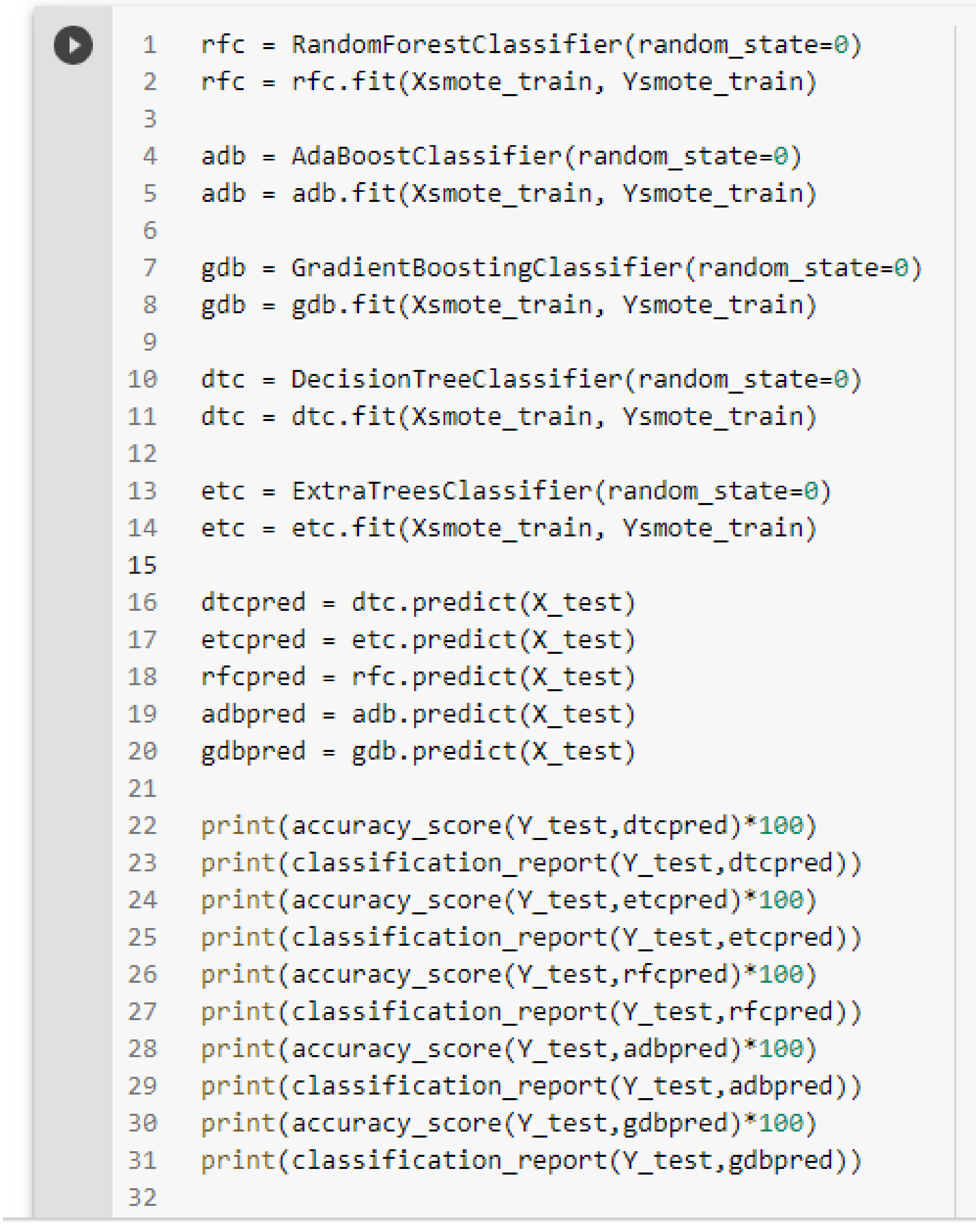
```python
rfc = RandomForestClassifier(random_state=0)
rfc = rfc.fit(X_train_ros, Y_train_ros)

adb = AdaBoostClassifier(random_state=0)
adb = adb.fit(X_train_ros, Y_train_ros)

gdb = GradientBoostingClassifier(random_state=0)
gdb = gdb.fit(X_train_ros, Y_train_ros)

dtc = DecisionTreeClassifier(random_state=0)
dtc = dtc.fit(X_train_ros, Y_train_ros)

etc = ExtraTreesClassifier(random_state=0)
etc = etc.fit(X_train_ros, Y_train_ros)


dtcpred = dtc.predict(X_test)
etcpred = etc.predict(X_test)
rfcpred = rfc.predict(X_test)
adbpred = adb.predict(X_test)
gdbpred = gdb.predict(X_test)

print(accuracy_score(Y_test,dtcpred)*100)
print(classification_report(Y_test,dtcpred))
print(accuracy_score(Y_test,etcpred)*100)
print(classification_report(Y_test,etcpred))
print(accuracy_score(Y_test,rfcpred)*100)
print(classification_report(Y_test,rfcpred))
print(accuracy_score(Y_test,adbpred)*100)
```

RANDOM UNDER-SAMPLING



```
rfc = RandomForestClassifier(random_state=0)
rfc = rfc.fit(X_train_rus, Y_train_rus)
adb = AdaBoostClassifier(random_state=0)
adb = adb.fit(X_train_rus, Y_train_rus)

gdb = GradientBoostingClassifier(random_state=0)
gdb = gdb.fit(X_train_rus, Y_train_rus)

dtc = DecisionTreeClassifier(random_state=0)
dtc = dtc.fit(X_train_rus, Y_train_rus)

etc = ExtraTreesClassifier(random_state=0)
etc = etc.fit(X_train_rus, Y_train_rus)

dtcpred = dtc.predict(X_test)
etcpred = etc.predict(X_test)
rfcpred = rfc.predict(X_test)
adbpred = adb.predict(X_test)
gdbpred = gdb.predict(X_test)

print(accuracy_score(Y_test,dtcpred)*100)
print(classification_report(Y_test,dtcpred))
print(accuracy_score(Y_test,etcpred)*100)
print(classification_report(Y_test,etcpred))
print(accuracy_score(Y_test,rfcpred)*100)
print(classification_report(Y_test,rfcpred))
print(accuracy_score(Y_test,adbpred)*100)
print(classification_report(Y_test,adbpred))
print(accuracy_score(Y_test,gdbpred)*100)
print(classification_report(Y_test,gdbpred))
```

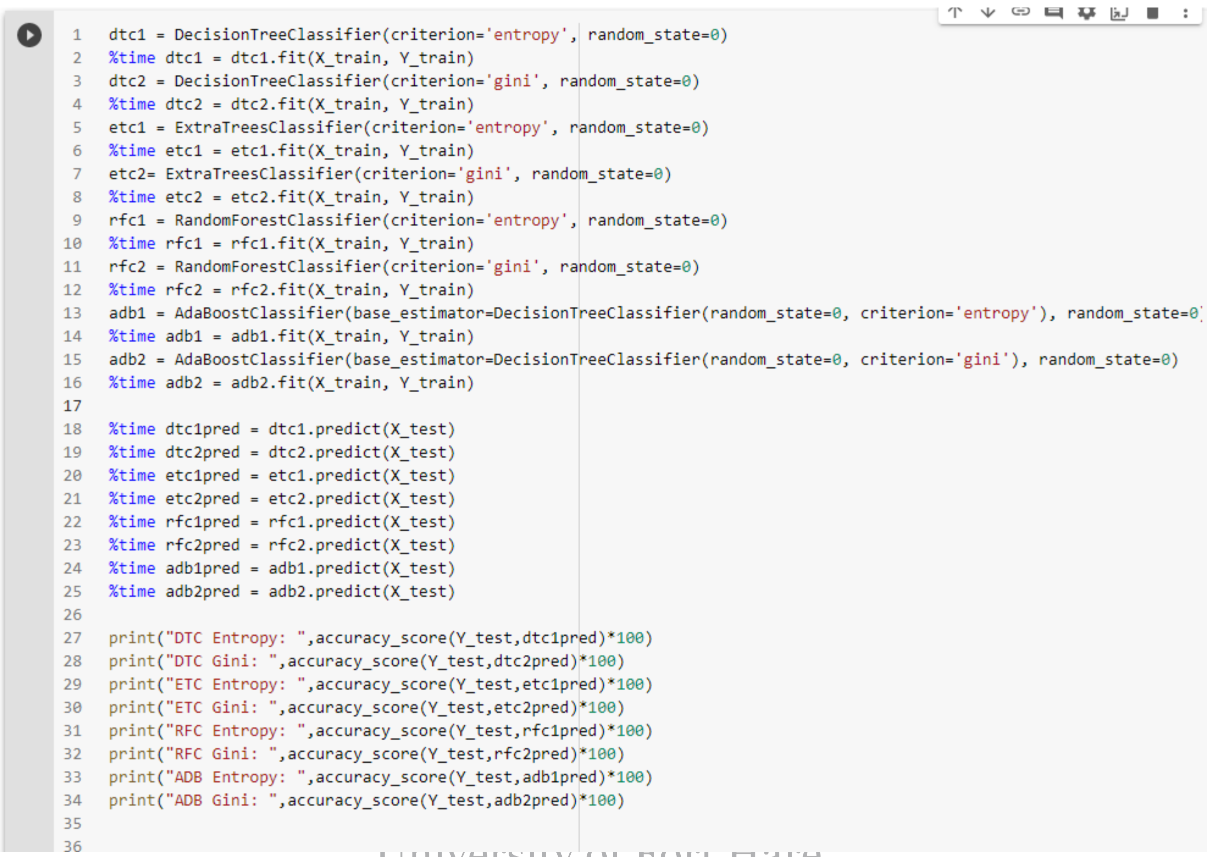
SMOTE CODE



```python
rfc = RandomForestClassifier(random_state=0)
rfc = rfc.fit(Xsmote_train, Ysmote_train)

adb = AdaBoostClassifier(random_state=0)
adb = adb.fit(Xsmote_train, Ysmote_train)

gdb = GradientBoostingClassifier(random_state=0)
gdb = gdb.fit(Xsmote_train, Ysmote_train)

dtc = DecisionTreeClassifier(random_state=0)
dtc = dtc.fit(Xsmote_train, Ysmote_train)

etc = ExtraTreesClassifier(random_state=0)
etc = etc.fit(Xsmote_train, Ysmote_train)

dtcpred = dtc.predict(X_test)
etcpred = etc.predict(X_test)
rfcpred = rfc.predict(X_test)
adbpred = adb.predict(X_test)
gdbpred = gdb.predict(X_test)

print(accuracy_score(Y_test,dtcpred)*100)
print(classification_report(Y_test,dtcpred))
print(accuracy_score(Y_test,etcpred)*100)
print(classification_report(Y_test,etcpred))
print(accuracy_score(Y_test,rfcpred)*100)
print(classification_report(Y_test,rfcpred))
print(accuracy_score(Y_test,adbpred)*100)
print(classification_report(Y_test,adbpred))
print(accuracy_score(Y_test,gdbpred)*100)
print(classification_report(Y_test,gdbpred))

```

Appendix E

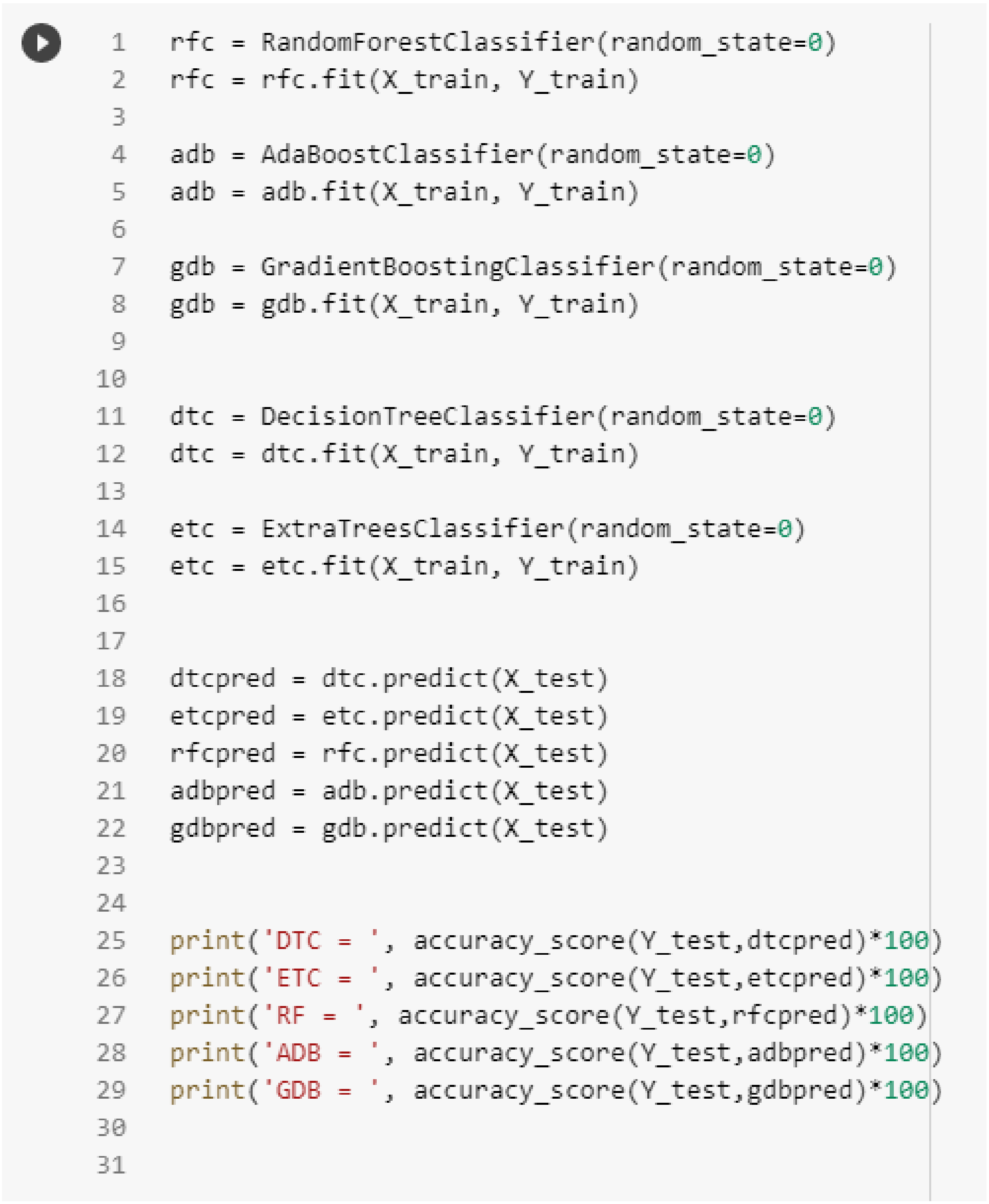
Code for training and testing classifiers using Gini index and Entropy as split Criteria.



```python
dtc1 = DecisionTreeClassifier(criterion='entropy', random_state=0)
%time dtc1 = dtc1.fit(X_train, Y_train)
dtc2 = DecisionTreeClassifier(criterion='gini', random_state=0)
%time dtc2 = dtc2.fit(X_train, Y_train)
etc1 = ExtraTreesClassifier(criterion='entropy', random_state=0)
%time etc1 = etc1.fit(X_train, Y_train)
etc2= ExtraTreesClassifier(criterion='gini', random_state=0)
%time etc2 = etc2.fit(X_train, Y_train)
rfc1 = RandomForestClassifier(criterion='entropy', random_state=0)
%time rfc1 = rfc1.fit(X_train, Y_train)
rfc2 = RandomForestClassifier(criterion='gini', random_state=0)
%time rfc2 = rfc2.fit(X_train, Y_train)
adb1 = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(random_state=0, criterion='entropy'), random_state=0
%time adb1 = adb1.fit(X_train, Y_train)
adb2 = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(random_state=0, criterion='gini'), random_state=0)
%time adb2 = adb2.fit(X_train, Y_train)

%time dtc1pred = dtc1.predict(X_test)
%time dtc2pred = dtc2.predict(X_test)
%time etc1pred = etc1.predict(X_test)
%time etc2pred = etc2.predict(X_test)
%time rfc1pred = rfc1.predict(X_test)
%time rfc2pred = rfc2.predict(X_test)
%time adb1pred = adb1.predict(X_test)
%time adb2pred = adb2.predict(X_test)

print("DTC Entropy: ",accuracy_score(Y_test,dtc1pred)*100)
print("DTC Gini: ",accuracy_score(Y_test,dtc2pred)*100)
print("ETC Entropy: ",accuracy_score(Y_test,etc1pred)*100)
print("ETC Gini: ",accuracy_score(Y_test,etc2pred)*100)
print("RFC Entropy: ",accuracy_score(Y_test,rfc1pred)*100)
print("RFC Gini: ",accuracy_score(Y_test,rfc2pred)*100)
print("ADB Entropy: ",accuracy_score(Y_test,adb1pred)*100)
print("ADB Gini: ",accuracy_score(Y_test,adb2pred)*100)
```

Appendix F

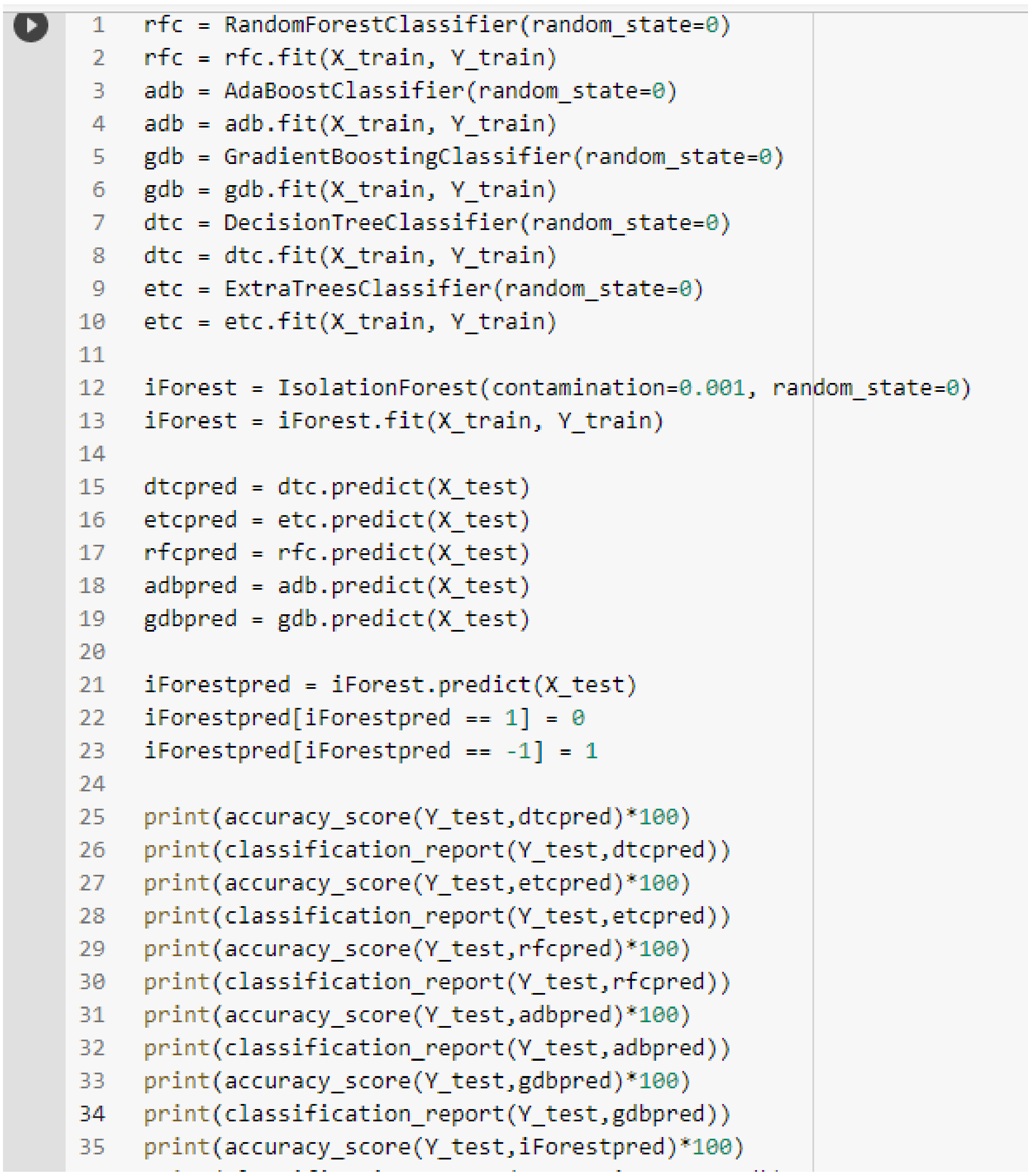
Code for training and testing classifiers by varying sample size



```python
rfc = RandomForestClassifier(random_state=0)
rfc = rfc.fit(X_train, Y_train)

adb = AdaBoostClassifier(random_state=0)
adb = adb.fit(X_train, Y_train)

gdb = GradientBoostingClassifier(random_state=0)
gdb = gdb.fit(X_train, Y_train)


dtc = DecisionTreeClassifier(random_state=0)
dtc = dtc.fit(X_train, Y_train)

etc = ExtraTreesClassifier(random_state=0)
etc = etc.fit(X_train, Y_train)


dtcpred = dtc.predict(X_test)
etcpred = etc.predict(X_test)
rfcpred = rfc.predict(X_test)
adbpred = adb.predict(X_test)
gdbpred = gdb.predict(X_test)


print('DTC = ', accuracy_score(Y_test,dtcpred)*100)
print('ETC = ', accuracy_score(Y_test,etcpred)*100)
print('RF = ', accuracy_score(Y_test,rfcpred)*100)
print('ADB = ', accuracy_score(Y_test,adbpred)*100)
print('GDB = ', accuracy_score(Y_test,gdbpred)*100)


```

Appendix G

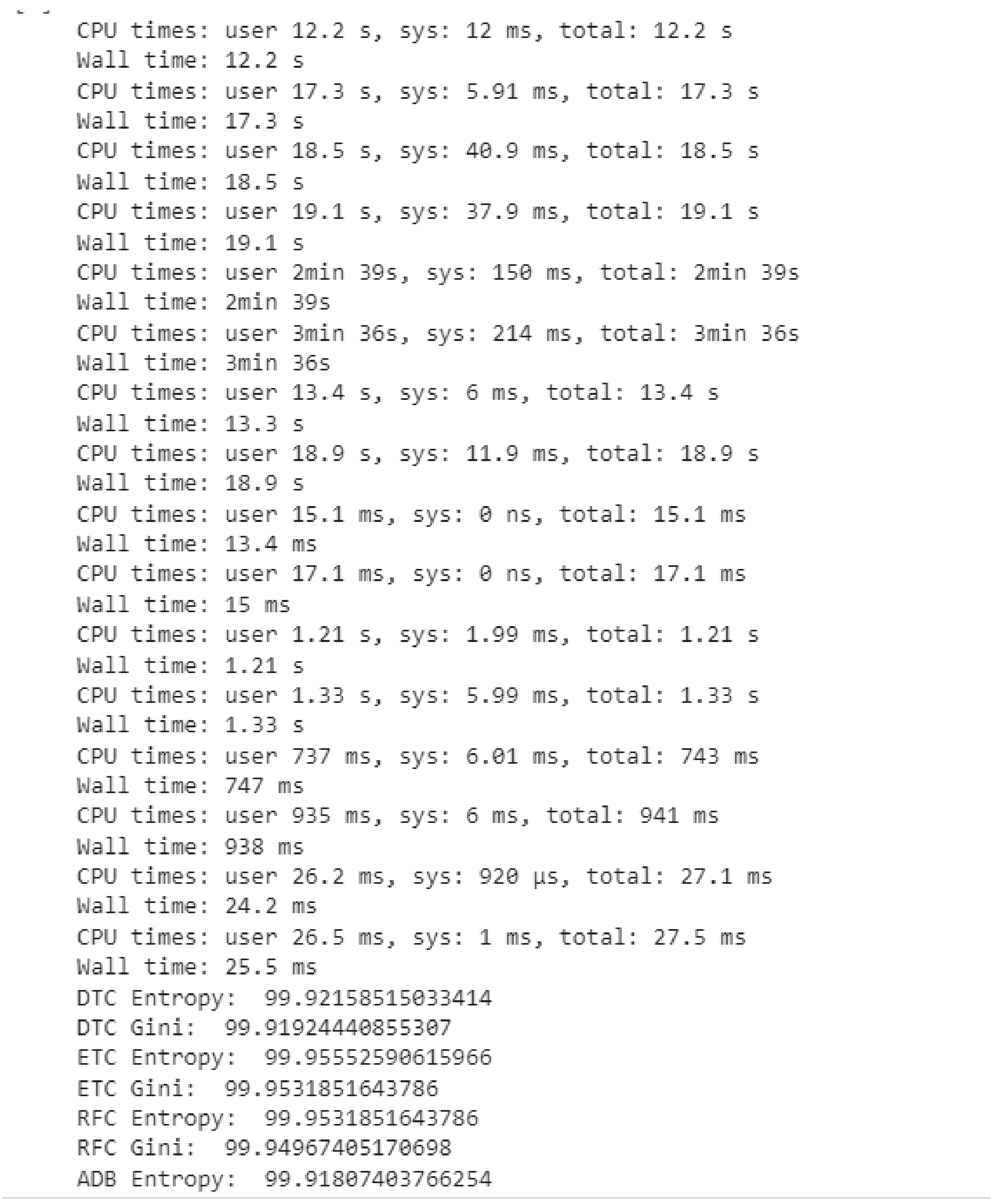
Code for training and testing classifiers that use supervised and unsupervised learning to compare accuracy.



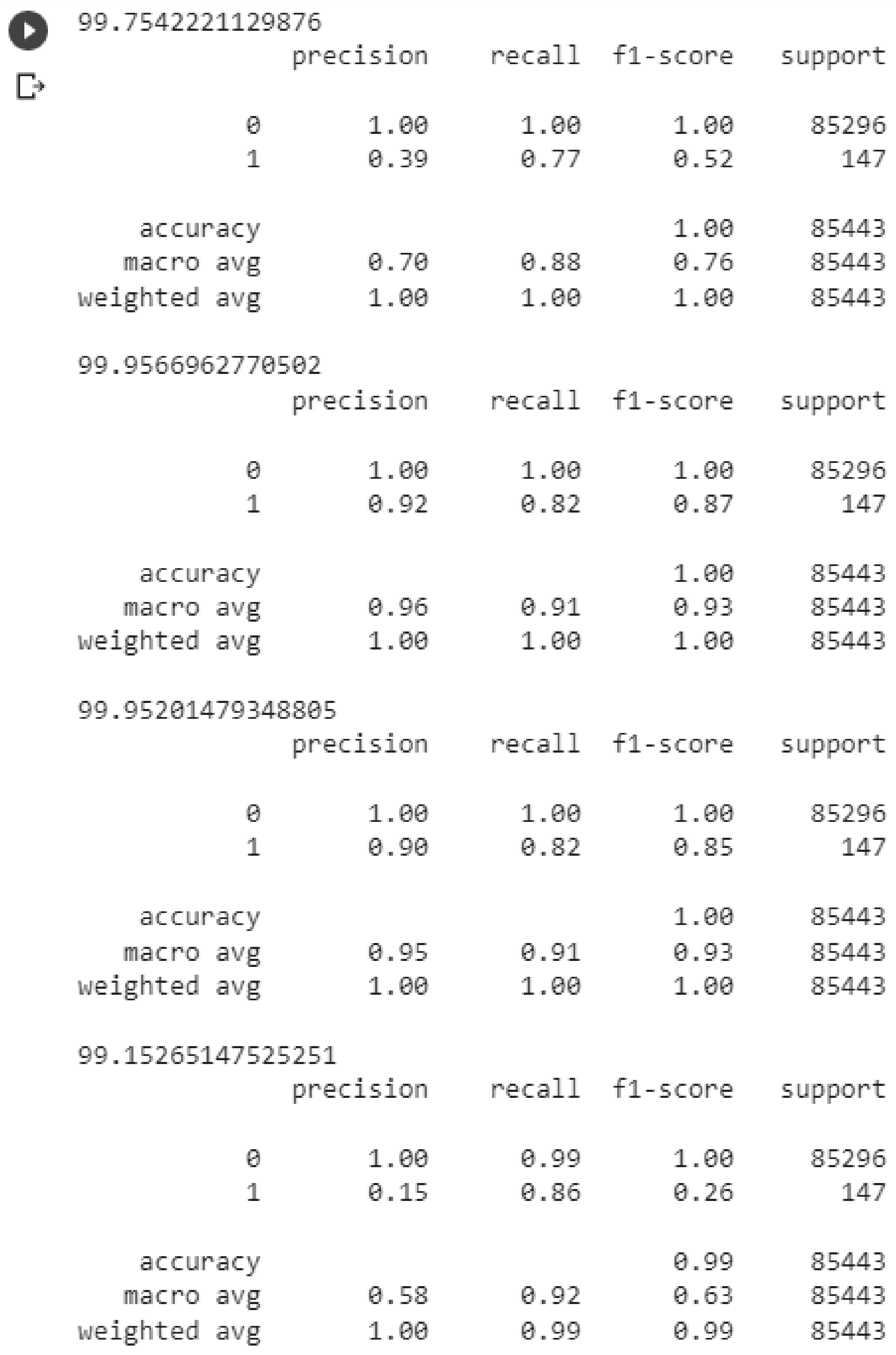
```
rfc = RandomForestClassifier(random_state=0)
rfc = rfc.fit(X_train, Y_train)
adb = AdaBoostClassifier(random_state=0)
adb = adb.fit(X_train, Y_train)
gdb = GradientBoostingClassifier(random_state=0)
gdb = gdb.fit(X_train, Y_train)
dtc = DecisionTreeClassifier(random_state=0)
dtc = dtc.fit(X_train, Y_train)
etc = ExtraTreesClassifier(random_state=0)
etc = etc.fit(X_train, Y_train)

iForest = IsolationForest(contamination=0.001, random_state=0)
iForest = iForest.fit(X_train, Y_train)

dtcpred = dtc.predict(X_test)
etcpred = etc.predict(X_test)
rfcpred = rfc.predict(X_test)
adbpred = adb.predict(X_test)
gdbpred = gdb.predict(X_test)

iForestpred = iForest.predict(X_test)
iForestpred[iForestpred == 1] = 0
iForestpred[iForestpred == -1] = 1

print(accuracy_score(Y_test,dtcpred)*100)
print(classification_report(Y_test,dtcpred))
print(accuracy_score(Y_test,etcpred)*100)
print(classification_report(Y_test,etcpred))
print(accuracy_score(Y_test,rfcpred)*100)
print(classification_report(Y_test,rfcpred))
print(accuracy_score(Y_test,adbpred)*100)
print(classification_report(Y_test,adbpred))
print(accuracy_score(Y_test,gdbpred)*100)
print(classification_report(Y_test,gdbpred))
print(accuracy_score(Y_test,iForestpred)*100)
```

Appendix H

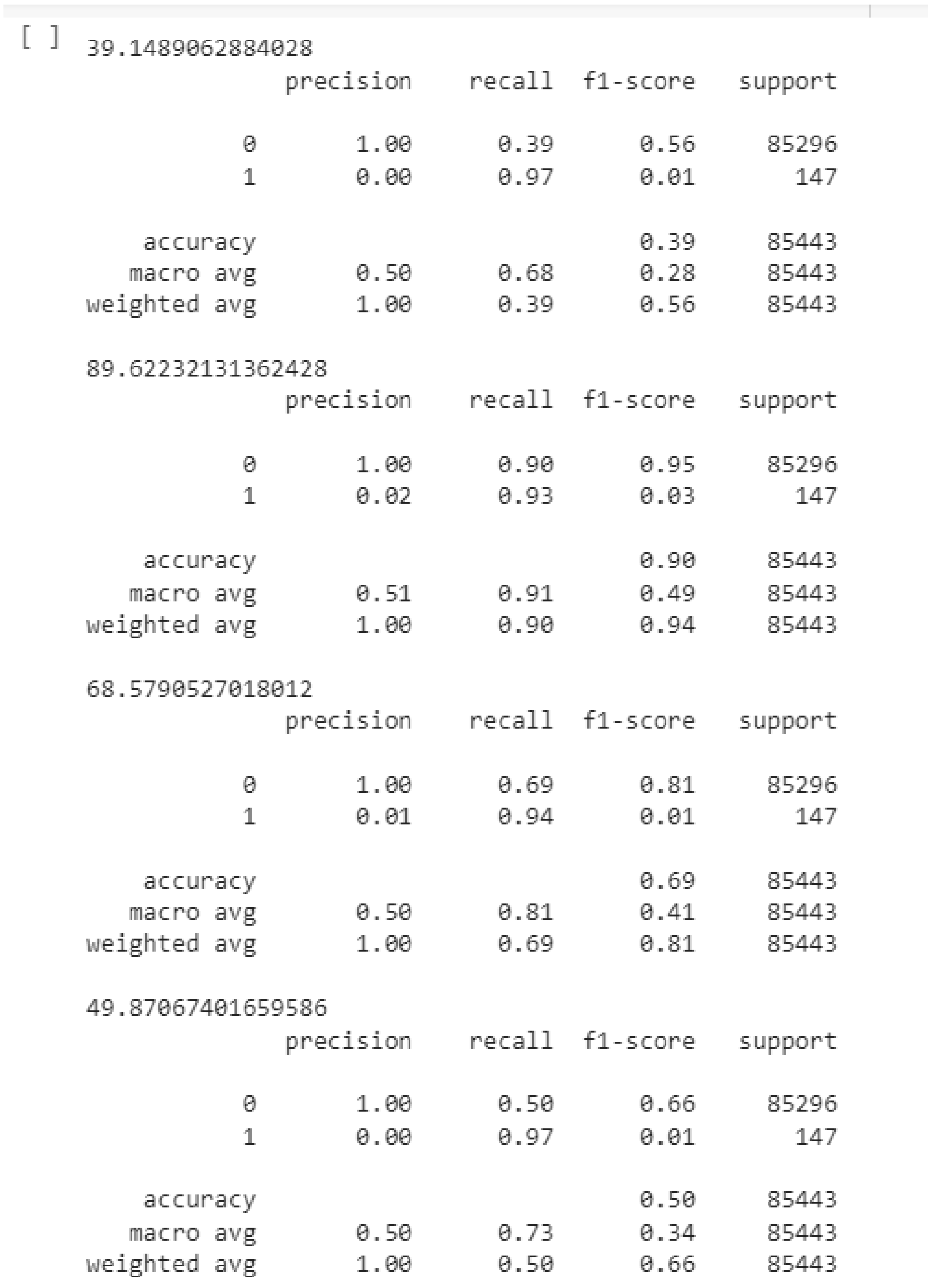
Results.



```
CPU times: user 12.2 s, sys: 12 ms, total: 12.2 s
Wall time: 12.2 s
CPU times: user 17.3 s, sys: 5.91 ms, total: 17.3 s
Wall time: 17.3 s
CPU times: user 18.5 s, sys: 40.9 ms, total: 18.5 s
Wall time: 18.5 s
CPU times: user 19.1 s, sys: 37.9 ms, total: 19.1 s
Wall time: 19.1 s
CPU times: user 2min 39s, sys: 150 ms, total: 2min 39s
Wall time: 2min 39s
CPU times: user 3min 36s, sys: 214 ms, total: 3min 36s
Wall time: 3min 36s
CPU times: user 13.4 s, sys: 6 ms, total: 13.4 s
Wall time: 13.3 s
CPU times: user 18.9 s, sys: 11.9 ms, total: 18.9 s
Wall time: 18.9 s
CPU times: user 15.1 ms, sys: 0 ns, total: 15.1 ms
Wall time: 13.4 ms
CPU times: user 17.1 ms, sys: 0 ns, total: 17.1 ms
Wall time: 15 ms
CPU times: user 1.21 s, sys: 1.99 ms, total: 1.21 s
Wall time: 1.21 s
CPU times: user 1.33 s, sys: 5.99 ms, total: 1.33 s
Wall time: 1.33 s
CPU times: user 737 ms, sys: 6.01 ms, total: 743 ms
Wall time: 747 ms
CPU times: user 935 ms, sys: 6 ms, total: 941 ms
Wall time: 938 ms
CPU times: user 26.2 ms, sys: 920 µs, total: 27.1 ms
Wall time: 24.2 ms
CPU times: user 26.5 ms, sys: 1 ms, total: 27.5 ms
Wall time: 25.5 ms
DTC Entropy:  99.92158515033414
DTC Gini:  99.91924440855307
ETC Entropy:  99.95552590615966
ETC Gini:  99.9531851643786
RFC Entropy:  99.9531851643786
RFC Gini:  99.94967405170698
ADB Entropy:  99.91807403766254
```



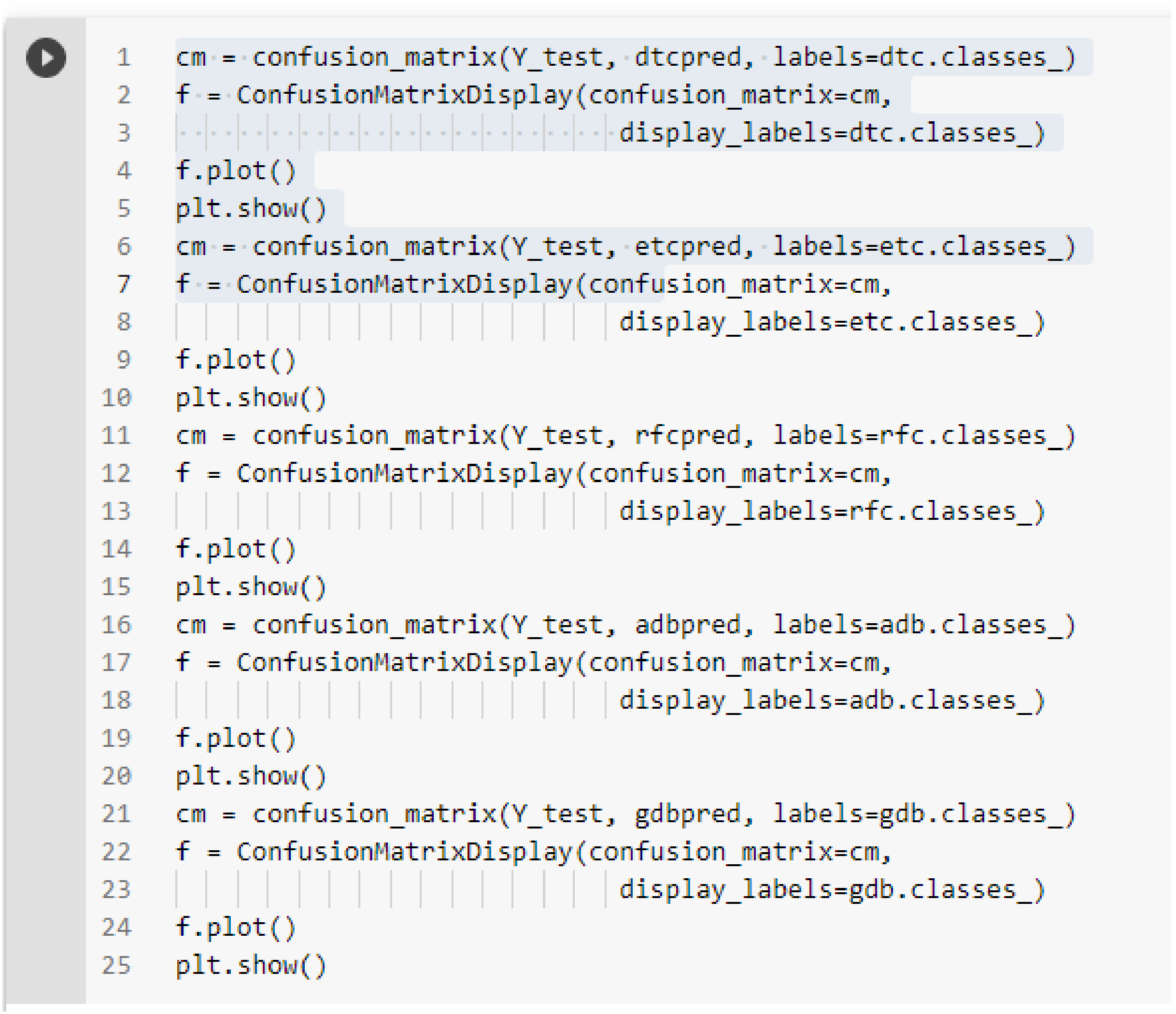
```
99.7542221129876
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85296
           1       0.39      0.77      0.52       147

    accuracy                           1.00     85443
   macro avg       0.70      0.88      0.76     85443
weighted avg       1.00      1.00      1.00     85443

99.9566962770502
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85296
           1       0.92      0.82      0.87       147

    accuracy                           1.00     85443
   macro avg       0.96      0.91      0.93     85443
weighted avg       1.00      1.00      1.00     85443

99.95201479348805
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85296
           1       0.90      0.82      0.85       147

    accuracy                           1.00     85443
   macro avg       0.95      0.91      0.93     85443
weighted avg       1.00      1.00      1.00     85443

99.15265147525251
              precision    recall  f1-score   support

           0       1.00      0.99      1.00     85296
           1       0.15      0.86      0.26       147

    accuracy                           0.99     85443
   macro avg       0.58      0.92      0.63     85443
weighted avg       1.00      0.99      0.99     85443
```



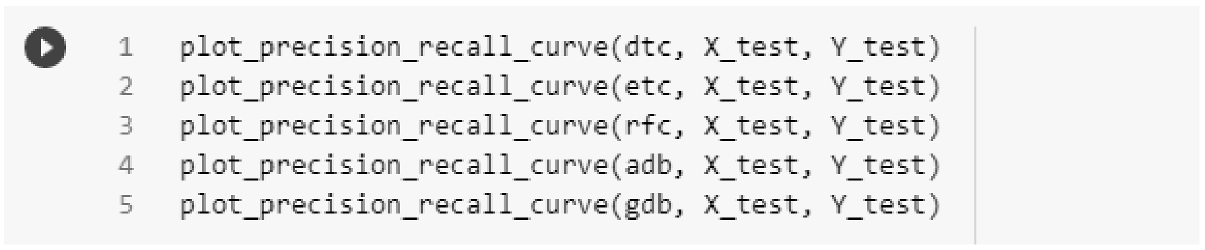
```
[ ]  39.1489062884028
                   precision    recall  f1-score   support

               0       1.00      0.39      0.56     85296
               1       0.00      0.97      0.01       147

        accuracy                           0.39     85443
       macro avg       0.50      0.68      0.28     85443
    weighted avg       1.00      0.39      0.56     85443

     89.62232131362428
                   precision    recall  f1-score   support

               0       1.00      0.90      0.95     85296
               1       0.02      0.93      0.03       147

        accuracy                           0.90     85443
       macro avg       0.51      0.91      0.49     85443
    weighted avg       1.00      0.90      0.94     85443

     68.5790527018012
                   precision    recall  f1-score   support

               0       1.00      0.69      0.81     85296
               1       0.01      0.94      0.01       147

        accuracy                           0.69     85443
       macro avg       0.50      0.81      0.41     85443
    weighted avg       1.00      0.69      0.81     85443

     49.87067401659586
                   precision    recall  f1-score   support

               0       1.00      0.50      0.66     85296
               1       0.00      0.97      0.01       147

        accuracy                           0.50     85443
       macro avg       0.50      0.73      0.34     85443
    weighted avg       1.00      0.50      0.66     85443
```

Appendix I

Code for performance measures.



```
cm = confusion_matrix(Y_test, dtcpred, labels=dtc.classes_)
f = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=dtc.classes_)
f.plot()
plt.show()
cm = confusion_matrix(Y_test, etcpred, labels=etc.classes_)
f = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=etc.classes_)
f.plot()
plt.show()
cm = confusion_matrix(Y_test, rfcpred, labels=rfc.classes_)
f = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=rfc.classes_)
f.plot()
plt.show()
cm = confusion_matrix(Y_test, adbpred, labels=adb.classes_)
f = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=adb.classes_)
f.plot()
plt.show()
cm = confusion_matrix(Y_test, gdbpred, labels=gdb.classes_)
f = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=gdb.classes_)
f.plot()
plt.show()
```



```
plot_precision_recall_curve(dtc, X_test, Y_test)
plot_precision_recall_curve(etc, X_test, Y_test)
plot_precision_recall_curve(rfc, X_test, Y_test)
plot_precision_recall_curve(adb, X_test, Y_test)
plot_precision_recall_curve(gdb, X_test, Y_test)
```

Appendix J

x	a data point
X	a data set of n instances
n	number of data points in a data set, $n = \lvert X \rvert$
m	index of data point x_m, $m \in \{0, ..., n-1\}$
Q	a set of attributes
d	number of attributes, $d = \lvert Q \rvert$
q	an attribute
T	a tree or a node
t	number of trees
$h(x)$	returns the path length of x
$hlim$	evaluation height limit
ψ	subsampling size
l	a possible path length
s	an anomaly score or function which returns an anomaly score
k	the number of nearest neighbours
a	contamination level of anomalies in a data set
cl	number of anomaly clusters

Appendix K

Feature name	Description	Type
Time	Number of seconds elapsed between the current transaction and the first transaction in the dataset	Continuous
V1	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V2	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V3	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V4	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V5	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V6	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V7	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V8	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V9	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V10	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V11	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V12	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V13	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V14	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V15	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V16	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V17	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V18	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V19	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V20	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V21	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous

V22	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V23	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V24	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V25	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V26	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V27	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
V28	Might be a result of a PCA Dimensionality reduction to protect user identities and sensitive features	Continuous
Amount	Amount spend on the transaction	Double/Float
Class	1 if the transaction is fraud or 0 if the transaction is not fraud	Discrete