University of Fort Hare
Together in Excellence

PREDICTION OF FRAUDULENT TRANSACTIONS USING MACHINE LEARNING TECHNIQUES

By

Given Manyike

(224068619)

Submitted in fulfillment of the requirements for the degree of

Honours in Applied Statistics

Department of Statistics

Faculty of Science and Agriculture

Supervisor: Dr M. Mutambayi

Alice, Eastern Cape, South Africa

DECLARATION

I declare that the study titles “*Prediction of Fraudulent Transactions Using Machine Learning Techniques”* is my original work and has not been submitted for credit towards a degree or examination at another institution of higher learning and that the sources I have used or mentioned have been properly cited and recognized.

Authors signature: [illegible] Date: 2024/10/21

DEDICATION

My research is dedicated to my mother, father and my siblings.

ACKNOWLEDGEMENTS

This study would have not been possible without the outstanding assistance from my supervisor, Dr M, Mutambayi. His knowledge, constant consultation and vast experience helped me in achieving my goals of making this full project possible from the beginning to the end. I am sincerely grateful for the amazing support throughout my project.

I would like to thank my funders, Multichoice in collaboration with Student Village. If not for their financial assistance I would have not been able to achieve this. Furthermore, I would like to specifically acknowledge Zanele, my bursary administrator for giving me a chance and constantly being a great support structure.

Lastly, I would like to thank my family and my classmates who were also there for me throughout this journey.

ABSTRACT

Background: Corporate fraud detection in financial transactions is significant to the financial sector and thus more advanced techniques such as machine learning are necessary. This study explains the drawbacks of the existing methods for detecting fraud in transactions using Random Forest, Logistic Regression, Decision Tree, Support Vector Machines, K-Nearest Neighbor, Artificial Neural Networks and Naive Bayes models.

Methods: The study used financial transactions and applied the available models for financial classifications of fraud. Statistics such as accuracy, Positive Predictive Value method (PPV), Negative Predictive Value (NPV), sensitivity, specificity and F1-score metrics were used to measure each model's performance.

Results: Among the models with the best accuracy were Random Forest models, SVM and KNN which achieved an accuracy of 99.88% for all three models while a balanced accuracy of 91.64% was attained in a reduced Random Forest model, which also had a similar accuracy of 99.88% in comparison to the full model. Furthermore, models like Logistic Regression and Decision Trees yielded 98.32% and 98.76% respectively. On the other hand, ANN only achieved 0.36% accuracy.

Conclusion: In the study's conclusion, the Random Forest model, especially its simple form, is viewed favourably, considering its accuracy and ease of interpretation, for fraud detection. The results underscore the need to choose suitable machine learning models and features in the systems to increase their accuracy and efficiency in detecting fraudulent activities.

Keywords: Fraud Detection, Machine Learning, Random Forest, Logistic Regression, Decision Tree, Support Vector Machines, K-Nearest Neighbor, Artificial Neural Networks, Naïve Bayes.

CONTENTS

FIGURES

TABLES

ABBREVIATIONS

AE-PRF	*– AUTOCODERS PROBABLISTIC RANDOM FOREST*
VIF	*– VECTOR INFLATION FACTOR*
VDP	*– VECTOR DECOMPOSITION PROPORTION*
PPV	*– POSITIVE PRIDICTED VALUE*
NPV	*– NEGATIVE PREDICTED VALUE*
KNN	*– K- NEAREST NEIGHBOUR*
ANN	*– ARTIFICIAL NEURAL NETWORK*
SVM	*– SUPPORT VECTOR MACHINES*
ML	*– MACHINE LEARNING*
CNN	*- CONVOLUTIONAL NEURAL NETWORKS*

CHAPTER 1: INTRODUCTION

To aid comprehension of the study, this part provides a view of the central theme of the project. The subsections are organized as follows: Section 1.1 is an introduction to Machine Learning (ML) algorithms. This is followed by an analysis of the problem statement in Section 1.2, Section 1.3 highlights the aim and objectives of the project; Section 1.4 - outlines hypotheses for testing; the normality, checking for outliers and multicollinearity. Section 1.5 will outline the significance of the study while Section 1.6 discusses the scope and limitations of it; methodologies and approaches are discussed under section 7, and finally, Section 1.8 will detail the organization of the project.

1.1. Introduction to Machine Learning Algorithms

The increasing mobility in transactions digitally comes with its advantages, and one of them is the ease of conducting financial transactions. In the same breath is the increasing concern over fraudulent activities. A strategy essential for deterring fraudulent activities and fraudulent transactions over the systems is preventing such transactions. This research project: 'Predicting fraudulent transactions using machine learning techniques' focuses on examining and developing sophisticated machine learning techniques for improving the speed and detection accuracy of fraud detection systems.

Card frauds are the center of attention in research activities as always because fraudulent activities keep changing. Nevertheless, there is still an ability to improve on the traditional approaches since they cannot address new forms of fraud that will require sophisticated machine-learning techniques. In this project, multiple models including Random Forest, Logistic Regression, Decision Tree, K-Nearst Neighbors (KNN), Artificial Neural Networks (ANN), Support Vector Machine (SVM), and Naive Bayes will be used to predict the fraudulent transactions.

Prior works have stated the need to focus on algorithm selection and the non-uniform distribution of the extreme population of fraud data. In particular, Thennakoon et al. (2019) have enhanced the attempt of credit card fraud detection systems with the use of Logistic Regression, Naive Bayes and SVM by accurate performances on a developed real-time credit card fraud detection system. In the same context, Awoyemi et al. (2017) were able to show that KNN and also Naive Bayes classifiers can be quite effective and efficient in handling very skewed datasets such as those encountered with credit card fraud. Also, as Raghavan and El Gayar (2019) explained, one possible direction is the use of SVM, which could be expanded with CNNs for larger data, but for smaller accentuated datasets it was proposed to combine SVM, Random Forest and KNN ensemble approaches.

Credit card fraud is also easier to combat with the help of algorithms such as Decision Tree and Random Forest according to the study by Dileep et al. (2022). They showed that the models built can classify the transactions as fraudulent or non-fraudulent with a high degree of accuracy by designing the decision trees and forests based on the end users' activities. This not only improves detection efficiency but also establishes a solid strategy for understanding the actual credit card usage transaction data.

Furthermore, Lin and Jiang (2021) have applied a new technique called the autoencoders probabilistic random forest (AE-PRF) for detecting credit card fraud. It utilizes the learning potential of autoencoders for high-dimensionality feature extraction as well as random forests for classifying skewed datasets. The combination is acknowledged as elsewhere more efficient than other unfriendly techniques, which is data resampling, in employer piercing parameters like accuracy and others and doing without most restructuring of the data.

The employment of these other techniques towards solving the problem of changing and complex structure of the fraud transaction is up-to-date. This study also aims to understand how different models for detecting fraud perform, as this will promote the creation of fraud detection systems that are robust.

The goal of combining these different machine learning solutions is to overcome the problem that fraud is an inherently wayward and complicated phenomenon. Aiming at enhancing the present models for fraud detection, this work focuses on comparing systems for a variety of different techniques to be able to detect dissenting models from others that would prove to be the most effective in various aspects.

To summarize, this research project intends to improve the existing literature on the detection of credit card fraud by applying various machine learning techniques as credit card fraud remains a frequently reported crime. This study has the practical aim to add value to the previous scientific outcomes by using novel approaches towards developing better and more efficient fraud detection systems and therefore, increasing the safety and trust of monetary transactions.

1.2. Problem statement

Fraud detection fields depend on understanding the trends of transactions and identifications of fraud to maintain financial integrity. With increasing digital money transactions, there are more chances for fraud thereby calling for rigorous measures that can detect abnormalities and discrepancies efficiently. This research aims to develop techniques that improve accuracy in detecting fraud through the use of machine learning (ML) algorithms analyzing huge amounts of transaction datasets.

The most challenging task is choosing a good Machine Learning model that suits the characteristic features of our dataset. There are strengths and weaknesses to each algorithm; hence, we need to assess their performance in terms of detecting fraud. Types such as logistic regression, k-nearest neighbours, and artificial neural networks offer different ways to divide transactions at varying levels of precision as well as interpretability. Logistic regression works best in its simplicity while it forms a reference point for other complex models.

Our particular interest lies in seeing how complex ML algorithms can accurately classify transactions. Support vector machines (SVM) and decision trees are both powerful classifiers and can be used even for high-dimensional feature spaces. In addition to this, ensemble methods such as random forests enhance performance by aggregating predictions from several models thereby increasing accuracy and reducing overfitting, which commonly occurs when a model is trained with too many redundant features or noise.

In contrast to these ML approaches, our investigation is aimed at identifying the algorithm that achieves the best trade-off between accuracy, efficiency, and interpretability. This will lead to more robust systems of fraud detection that could protect financial institutions as well as their clients better over time. We must gain some new insights through which we could revolutionize fraud detection so that normal transaction processing continues smoothly while illegitimate activities are easily identified and stopped promptly.

1.3. Aim and Objectives

1.3.1. Aim

The aim is of this study is to assess how well some machine learning techniques can classify transactions in the used dataset.

1.3.2. Objectives

The study has the following objectives:

- To Determine which machine learning algorithm works best for fraud detection, in this dataset by considering performance metrics and practical deployment factors.
- To Identify predictors that are related to the detection of fraud.

1.4. Hypotheses

1.4.1. Distribution of the dependent variable – is_fraud

In this section I checked the distribution of the dependent variable using a histogram and concluded that the number of cases of non-fraudulent transactions far exceed the number of fraudulent transactions. The ratio of fraudulent transactions (1) to non-fraudulent transactions (0) is 0.39% vs 99.61%, respectively.

1.4.2. Normal Distribution

In this section I checked the normality of the continuous variables in the fraud dataset. The purpose of checking for normality is to determine how the data is distributed, and it also gives a broad understanding of the covariates. I used the Kolmogorov-Smirnov test which uses histograms and Q-Q plots to check the normality of the continuous variables.

The Hypothesis:

$$H_0: The\ continous\ variables\ are\ normally\ distributed$$

$$H_A: The\ continous\ variables\ are\ not\ normally\ distributed$$

The conclusion was that the continuous variables were not normally distributed since the histograms and the Q-Q plots showed signs of unequal distribution.

1.4.3. Checking for Outliers

In this section I am going to check for outliers. When dealing with large datasets, there is always a probability of finding outliers within the dataset. Checking for outliers is crucial because they can distort the analysis of the data, highlight data errors and reveal variability. Identifying these outliers helps maintain/ensure the accuracy of the machine learning algorithms and gives more reliable results.

The hypothesis:

$$H_0\colon The\ data\ has\ outliers$$

$$H_A\colon The\ data\ has\ no\ outliers$$

Boxplots are going to be used to check for outliers in the dataset.

1.4.4. Checking for Multicollinearity

Kim et.al. (2016) states that Multicollinearity is a condition of high linear interrelationships among the explanatory variables, which produces incorrect results in regression analyses. Variance inflation factor (VIF), condition index and number, condition number, and variance decomposition proportion (VDP) are some of the diagnostic tools for multicollinearity. Checking for multicollinearity is important because it helps reduce inflated standard errors and unstable coefficient estimates.

The Hypothesis:

$$H_0\colon There\ is\ multicollinearity\ in\ the\ dataset$$

$$H_A\colon There\ is\ no\ multicollinearity\ in\ the\ dataset$$

To check for multicollinearity, I will use a correlation matrix and also the VIF value.

1.5. Significance Of the Study

The importance of this study is that it could transform how fraud detection is done in the financial sector. The growth of digital transactions has also led to a rise in complexity as well as the frequency of fraudulent activities. This research addresses this urgent requirement for advanced machine learning solutions capable of quickly and accurately detecting fraudulent transactions, thereby protecting financial institutions and their clients against cases of fraud. Through examining several machine learning algorithms, this investigation is intended to improve the accuracy and efficacy of fraud detection systems. Such findings may result in improved real-time monitoring tools which are not only able to identify fraud but also reduce false positives leading to a seamless transaction experience for legitimate users.

Furthermore, the study enhances understanding of the efficiency of different algorithms in addressing highly imbalanced datasets, which is necessary for fraud detection. For instance, one can use logistic regression, k-nearest neighbours or ensemble techniques such as random forests. This way, we can learn what each method does well and where it fails. It is important to note that these findings have implications for other sectors beyond finance like healthcare, retail and cybersecurity that require anomaly detection. Doing this will not only aid in improving decision-making but also helps build innovation within the various industries by advancing the application of machine learning.

1.6. Scope and Limitations of the Study

1.6.1. Scope

The purpose of this study is to assess the effectiveness of different machine learning models: logistic regression, k-nearest neighbour (k-NN), artificial neural network (ANN), support vector machine (SVM), decision trees, and random forests in spotting fraudulent financial transactions. This work will find out which model has a good combination of being accurate, efficient and interpretable for fraud detection. It includes studying transaction data to build models that can recognize deviations and forecast fraudulence leading to better fraud detection systems.

1.6.2. Limitations of the Study

- Binary classification tasks since the independent(Y) variable is a binary dataset.

Only considering some selected Machine learning Algorithms from the list below:

- Random Forest
- Logistic Regression
- Support vector machines
- Decision tree
- Naïve Bayesian algorithm

- K-nearest neighbour
- Artificial neural network models.

1.7. Methods and Techniques

Several techniques in machine learning are used in this study to handle credit card fraud detection, which includes Random Forest, Logistic Regression, Decision Tree, Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Artificial Neural Network (ANN) and Naïve Bayes. From all the above modes only 3 best-performing algorithms will be chosen for further analysis. A hybrid sampling approach is utilized to deal with the highly imbalanced data set by combining the under-sampling of legitimate transactions and the over-sampling of fraudulent ones. Each technique's performance is measured using metrics like accuracy, sensitivity, specificity, precision, etc.

1.8. Organization of the Study

CHAPTER 1: INTRODUCTION

In this section I introduced Machine Learning algorithms and their relation to fraud detection. The subtopics in this section are, 1.1 Introduction to Machine Learning Algorithms, 1.2 The problem statement, 1.3 The Aim and Objectives, 1.4 Hypothesis testing, 1.5 The significance of the study, 1.6 The scope and limitations of the study and finally, 1.7 which gives a brief description of the methods and techniques that will be used.

CHAPTER 2: LITERATURE REVIEW

In this section I will giving findings from other authors who have done a similar study as this one and share their results they obtained from the machine learning algorithms.

CHAPTER 3: METHODOLOGY

Here I will give detailed definitions of the machine learning algorithms that will be used in the research.

CHAPTER 4: DATA ANALYSIS

In this section, I will check for the distribution of the dependent variable, normality of continuous variables, check for outliers and check for multicollinearity. I will also share extensive results from the machine learning algorithms.

CHAPTER 5: DISCUSSION, CONCLUSION AND RECOMMENDATIONS

In this section I will give my conclusions and recommendations on which machine learning algorithms are best suited for fraud detection

CHAPTER 2: LITERITURE REVIEW

2.1 Introduction

The financial sector faces a significant problem in the name of credit card fraud detection since it needs technologically advanced answers to mitigate threats. The number of transactions being performed is increasing thereby requiring effective, reliable and accurate methods that can be used to identify any form of falsification with ease. One way out might be implementing machine learning algorithms that are designed specifically for anomaly detection in transaction data.

Many different methods have been used in machine learning for fraud detection which includes various weaknesses and purposes. These models can find irregularities through which one could pick out fraudulent activities from genuine ones. In this respect, the algorithm choice is crucial as it can remarkably influence both the efficacy and precision of the whole process thus, comprehending the possibilities offered by each method becomes a must.

The latter machine learning algorithms are Random Forest, Logistic Regression, Decision Tree, Support Vector Machines (SVM), K-Nearest Neighbor (KNN), and Artificial Neural Networks (ANN). In terms of classification tasks, these techniques present a range of options which can be employed to boost the detection rate. Their use in addressing fraudulent activities has been established by many studies which have indicated their usefulness in real-world cases.

Therefore, this article will provide an all-inclusive review of these machine learning algorithms by looking at how effective they are and where they can be applied such as credit card fraud detection. The purpose of this study is to inform future advancement in the field of fraud detection based on recent research findings.

2.2 Random Forest

Random Forest is one of the most salient and widely applicable machine learning approaches which, due to its ability to adapt to a wide variety of problems, even such as fraudulent detection, and insurance claims improving, remains in high demand. For example, according to Dileep et al. (2021), forest-based user activities supervised the randomized credit card transaction fraud detection model. The study illustrated that a high level of precision was achieved in identifying fraudulent transactions using the Random Forest model which is applicable in practice. The study states that the accuracy of the Random Forest algorithm was around 85.8% which is quite high given the nature of the task detection of fraud.

Furthermore, Lin and Jiang (2021) reported credit card fraud detection based on the use of the -PRF method, an autoencoder combined with probabilistic random forest. This particular technique uses autoencoders to extract features and random forests for classification to treat dimensionality and imbalanced datasets efficiently. Their results underscored the fact that several performance measures including those referenced mean accuracy are reached concerning AE-PRF even in the absence of significant data resampling. This confirms the flexibility and robustness of the Random Forest in any kind of data setting.

Hussain et al. (2021) have focused on fraud detection using Random Forest and Support Vector Machine in comparison. Random Forest had an edge over other approaches when tested on large data sets, achieving an accuracy of 85.8%. Such features enable the reduction of correlation problems in an algorithm due to the fact that at each split, only a subset of the feature subset is chosen. Due to this characteristic, Random Forest is well adapted to the requirements of the more multi-dimensional predictive modelling, e.g., for the enhancement of fraud detection.

Moreover, Tiwari et al. (2021) also found that Random Forest was the best of all the models employed as it produced more precision and accuracy as compared to Logistic Regression and Decision Trees on any of the Big Data Analytical Frameworks Hadoop applied to German credit card data set. This also emphasizes the ability of the algorithm to process data of this size and increases the effectiveness of present and future fraud detection systems. As Random Forest has been employed in numerous studies with constant high performance in fraud detection, this metric speaks reliability and robustness of the tool.

2.3 Logistic Regression

Logistic Regression is a model employed with great success in machine learning for binary problems such as fraud detection. Its popularity stems from its ease of use interpretability and efficiency when working with expansive data. Thennakoon et al. (2019) drew attention to the effectiveness of Logistic Regression in their credit card fraud detection system where its accuracy in accepting some fraud patterns was 74%. This study also showed that Logistic Regression is an efficient technique for detecting fraudulent transactions, which is beneficial in enhancing the available fraud detection techniques.

Performing similar investigations, Awoyemi et al. (2017) also conducted a performance analysis on the performance of Logistic Regression on extreme skews in credit card fraud data. Their paper recorded an optimal accuracy of Logistic Regression of 54.86% which was quite low compared to other practices such as K-Nearest Neighbors (KNN), and Naive Bayes. Even if it is not very accurate, this approach is still appreciated because it is very easy to use and results are quick to read out without much difficulty. The study highlighted the need for data preparation and resampling strategies for enhancement of working of Logistic Regression on unbalanced datasets.

To support further the effectiveness of Logistic Regression, Itoo et al. (2021) assessed its performance against Naive Bayes and KNN. The study found that Logistic Regression was able to achieve the highest accuracy of 95% among the three models applied to the credit card fraud detection data set. Further, logistic regression outperformed Naive Bayes and KNN in sensitivity, specificity, precision and F-measure. This indicates the strength of the model in classifying whether or not a given transaction is fraudulently obtained.

Lastly, Hussein et al. (2021) applied Logistic Regression as a meta-classifier in a stacking ensemble for credit card fraud detection. The research fused the outputs of a fuzzy-rough nearest neighbour and a sequential minimal optimization classifier and used Logistic Regression to the output as a final prediction. The ensemble model yielded detection rates of 84.90% and 76.30%, thus affirming the ability of Logistic Regression to enhance the effectiveness of fraud detection systems when combined with other classifiers. In this way, it is pointed out that the application of Logistic Regression is very broad and it can add value in improving the performance of systems for detection of fraud through the use of ensemble techniques.

2.4 Decision Tree

Decision Tree algorithms are also popular because of their ease of use and interpretation. This works by employing a tree structure that binary splits the data into branches where each branch is representative of a feature decision. Dileep et al. (2021) also implemented Decision Trees using a constructed tree from user activities to detect credit card fraud. Through the study, it was found that such Decision Tree models achieved high degrees of accuracy at 83.6% for detecting incidences of fraud. This brings out the possibility of this model in designing fraud detection systems which are both accurate and easy to interpret.

Hussain et al. (2021) Compared the effectiveness of Decision tree Algorithms & Random Forest and Support Vector machines (Royal side) in detecting card fraud. The findings indicated that Decision Tree recorded 83.6 % accuracy, which is a bit less than Random Forest but still quite applaudable. Decision Trees are also able to accommodate both types of data: numbers and words and it is also easy to put them into practice which is why, they are extremely helpful in the detection of fraudulent activities.

Itoo et al. (2021) also discussed the results of Decision Trees and other machine-learning algorithms in comparison to one another. Although the paper mainly studied Logistic Regression, Naive Bayes, and KNN algorithms it also highlighted the practicality of Decision Trees Technique, particularly in dealing with the problem of class imbalance using resampling techniques. The study indicated that with the right method of data processing and control measures in place, Decision trees would improve both the effectiveness and efficiency of fraud detection systems.

Tiwari et al. (2021) stated that Decision Trees, for analysing German credit card data, performed effectively with precision and accuracy utilizing a Big Data Analytical Framework with Hadoop. The study emphasized that Decision Tree performance surpassed that of Support Vector Machines in accuracy on the national bank's credit card warehouse dataset. This goes to show once again how effective Decision Trees are in the diverse data space and the potential that they have in enhancing the existing fraud detection systems. The fact that Decision Trees continue to give positive results across different areas and studies makes them a credible technique for fraudulent transaction detection.

2.5 Support Vector Machines (SVM)

Support Vector Machine (SVM) is an efficient supervised classification algorithm used mostly on complex datasets where data has linear and non-linear relationships. According to Thennakoon et al. (2019), SVM models have shown 91% accuracy in some instances of fraud detection patterns being one of the most accurate models in that study. This high level of accuracy speaks to the capability of SVM in the management of fraud cases effectively.

Hussain et al. (2021) also investigated the application of SVM in the case of fraud detection and reported a comparison with the Random Forest method. This finding confirmed that SVM gives an accuracy of 84.1% which indicates the strength of the algorithm in the detection of fraudulent activities. The authors noticed that SVM is good for structured and semi-structured data including text images and trees among others and does not generally overfit the data. This is especially beneficial with SVM for the dataset where there is a boundary between classes.

Raghavan and El Gayar (2019) noted that SVMs have potential when applied with Convolutional Neural Networks (CNN), especially for larger datasets. They found it reasonable to use SVMs integrated with CNNs on fraud detection systems. This integration provides the best of both worlds whereby SVMs are used for the classification and competitive neural networks are used for feature extraction so that the accuracy and reliability of the system become better.

Tiwari et.al.(2021) noted that under the national bank's credit card warehouse dataset that they used Decision trees showed more accuracy than the SVM. However, with high-dimensional data settings, SVMs were still found to be performing well. The researchers reported that the application of SVM on the PagSeguro dataset yielded satisfactory results in terms of precision, recall and economic aspects of the task. This also emphasizes the flexibility and efficiency of SVMs towards different types of data environments and their prospects for enhancing fraud detection systems.

2.6 K-Nearest Neighbor (KNN)

K-Nearest Neighbors (KNN) is one of those classification algorithms which is quite neat and easy to implement for fraud detection. If one is to describe KNN, one would say KNN involves locating the 'k' closest data points (neighbours) in the dataset and predicting based on the class that is mostly represented among these neighbours. Awoyemi et al. (2017) conducted a study which focussed on the efficiency of KNN classification of highly skewed credit card fraud data and the optimal computed accuracy was 97.69%. Such high accuracy indicates how well KNN can be used to detect fraudulent transactions, hence purporting the need to take additional steps such as data preprocessing.

Itoo et al. (2021) focused on credit card fraud detection utilizing the KNN classifier and compared it with the other classifiers. The study found that KNN had a degree of accuracy of 75% which was lower than that obtained by the Logistic Regression but rather quite impressive. This also brings out the fact that KNN being an unsupervised method suffers when applied without the resampling methods aiming at imbalanced data. This shows again how important data pretreatment is for KNN performance improvement in today's fraud detection systems.

To enhance KNN further, Raghavan and El Gayar (2019) noted the advantages of using such approaches for small data sets. Their work indicated that the performance of three types of KNN models and hybrid models based on KNN with SVM, and Random Forest may be very good in practice. In this case, the functionality of several models is used, which contributes to better accuracy and reliability of the results.

According to Tiwari et al. (2021), It was found that KNN was able to achieve good precision and recall in several datasets. All the studies included in this research highlighted KNN with a combination of other approaches such as data resampling techniques SMOTE to address imbalance problems to increase accuracy in fraud detection. This further highlights the applicability and efficiency of KNN in different data situations and opportunities for the enhancement of fraud detection systems.

2.7 Artificial Neural Network (ANN)

The design of Artificial Neural Networks (ANN) follows the biological principle of the human brain and can learn any complex relationship from data. They turn out to be very useful in dealing with large amounts of data and in dealing with non-linear aspects. Thennakoon et al. (2019) did not state specifically ANN in their study however such inferences can be drawn in the research context with regards to fraud detection systems enhancing the systems through the use of advanced computer civilization techniques like ANN. The right neural network architecture given the expansive data sets available enhances the ability the detect fraudulent transactions.

According to Raghavan and El Gayar (2019), the prospects of deep learning models deserve to include ANN types such as Convolutional Neural Networks in the detection of fraud. Their research has demonstrated that novel performance of heat generation could be utilized using the detection of fraud with the help of neural networks and support vector machines. This combination works by taking advantage of both models; CNNs do the heavy feature extraction and SVMs handle the classification which improves the accuracy and reliability at the end.

Tiwari et al. (2021) noticed however that concerning the data granularity, ANN, particularly deep networks, were quite accurate on the German credit card dataset. He added that although

ANNs are faster in identifying fraud, Bayesian Belief Networks were able to identify a larger percentage of fraud. There is a need to compare performing other models for undirected fraud detection as this highlights the strength of ANNs in doing fraud detection, especially for new and rapid attacks.

Asha and KR (2021) considered the relative efficacy of KNN and SVM machine learning algorithms and ANN in the detection of credit card fraud rates in this poster. The study concluded that as high as 100% prediction accuracy may be obtained for ANN models showing that ANN models are the most appropriate for this application. At the same time, this study stressed the effectiveness of other techniques, such as resampling data normalization and data preprocessing to alleviate the potential issues caused by imbalanced data proportions. This brings out the fact that clamping ANNs with suitable data treatment measures can lead to very efficient fraud detection systems.

2.8 Naïve Bayes

Naive Bayes is a classifier based on Bayes' theorem that offers great usability because of its design, especially when classifying data that needs classification such as detecting fraudulent cases. It posits that the features are conditionally independent given the class label, thus facilitating the computation and optimizing it for bigger data sets. Thennakoon et al. (2019) pointed out the applications of Naive Bayes in a real-time credit card fraud detection system where a fraud accuracy of 83% was attained for some fraud patterns. This illustrates the ability of the model to do its work in detecting fraudulent activities effectively.

Awoyemi et al. (2017) adopted a Naive Bayes approach to a rather skewed credit card fraud data set on which they were able to record a maximum accuracy of 97.92%. This success highlights the capacity of Naive Bayes for imbalanced data even without perfect data preprocessing, especially subject to the methods utilized. The study also focused on resampling methods and their applications with an emphasis on improving the performance of Naive Bayes techniques in the detection of fraud.

Itoo et al. (2021) assessed the effectiveness of using Naive Bayes, Logistic Regression, and KNN methods in the application of credit card fraud detection. The study reported that the Naive

Bayes classifier achieved an accuracy of 91%, which is higher than that obtained with KNN, but less than that achieved with the Logistic Regression model. The same was observed in the KNN, where Naïve Bayes outperformed in sensitivity, specificity, precision, and F-measure. This demonstrates the strength as well as the reliability of the model in distinguishing between fraud and non-fraud cases.

Tiwari et al. (2021) examined the importance of applying Naive Bayes on different datasets and found out that it provides considerable recall and precision. The study pointed out that combining Naive Bayesian approaches with SMOTE for resampling imbalanced data sets will increase fraud detection accuracy. Tiwari et al. (2021) this increases the potential of the method in enhancing fraud detection systems further the na iv bayes method will not disappoint an expanding availability of data.

CHAPTER 3: METHODOLOGY

3.1 Introduction

This section describes the methods used by the academics in carrying out the research on fraud detection, with emphasis on the statistical techniques and machine learning strategies. First, a frequency distribution table is employed to assess the distribution of the dependent variable and then Kolmogorov-Smirnov test for normality is carried out. Considering normality, Razali and Wah (2011) state that the Kolmogorov-Smirnov test seeks the degree of mismatching which is defined as the maximum predictive accuracy attained against empirical and theoretical distribution creating images on the backs of ears cutting a slice out of normal.

The next step involves checking for and eliminating the outliers, for instance by applying box plots which help to present the distribution of the data and identify any outliers. Outliers are considered by Hoaglin and Iglewicz(1987) as being any data point which lies beyond one and a half times the inter-quartile range away from the quartiles. The presence of multicollinearity amongst continuous variables are evaluated Nutz (2008) by means of a correlation matrix. Dormann et al (2013) point out the fact that when independent variables are strongly correlated to each other, as measured in terms of Pearson's correlation coefficient, causes the machine learning models to underperform and therefore distort the predictions.

Different algorithms such as Logistic Regression as well as Random Forest have been utilized to detect fraudulent cases. As referred to in the work of Hosmer et al. (2013), logistic regression uses the logistic function to estimate the probability of a binary outcome. As defined by Breiman (2001), Random Forest is a learning paradigm that takes many decision trees and combines the outputs of all of them for better classification and lower overfitting of the models. The performance of these models was assessed using Confusion Matrix's Accuracy, Sensitivity, Specificity and Precision metrics that give a rounded understanding of the models and their ability to detect fraud. These performance metrics add value in determining the most appropriate fraud detection model.

3.2 Test For Normality

3.2.1 Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov test is a non-parametric statistical test that is used to test whether a given sample comes from a specified distribution, or whether two given samples come from the same distribution. Even though the method focuses on assessing the difference between a specific theoretic and an experimental distribution function, or two experimental distribution functions, it is normally used for evaluating two samples.

3.2.2 One-Sample Kolmogorov-Smirnov Test

The one-sample Kolmogorov-Smirnov test assesses the chances of a single continuous data sample being drawn from a particular form of the theoretical distribution. This test is used especially when the theoretical distribution is known, and it is required to find out whether the obtained data follows that distribution or not. Here are the steps followed to test the specific continuous variable for normality:

1. Define Hypothesis

$$H_0: The\ Continous\ variable\ follows\ a\ Normal\ distribution.$$

$$H_1: The\ continous\ variable\ does\ not\ follow\ a\ Normal\ Distribution$$

2. The Empirical Cumulative Distribution Function (ECDP):

For each continuous variable $X_1, X_2, \ldots\ldots, X_n$, the empirical cumulative distribution function $F_n(x)$ is calculated. It is defined as:

$$F_n(x) = \frac{Number\ of\ Observations\ X_i \leq x}{n} \tag{3.1}$$

3. Specify the Reference CDF:

$F(x)$ is the reference cumulative distribution function in the normal distribution that is tested against. For value of x, the CDF $F(x)$, gives the probability that a value from the theoretical distribution is $\leq x$.

4. Kolmogorov-Smirnov Statistic:

$$D_n = sup_x |F_n(x) - F(x)| \tag{3.2}$$

It measures the greatest deviation between the observed data and the expected theoretical distribution.

5. Compare K-S statistic with Critical Value:

Critical Value is obtained based on the sample size n and level of significance α. Alternatively, the p-value method can be used.

6. Conclusion

If the K-S statistic $D_n \geq Critcal\ Value, or\ the\ p - value\ \leq\ \alpha(0.05)$ we reject the null hypothesis.

For every continuous variable, such as the transaction amounts in my dataset, the one-sample K-S test will show whether the variable under concern has any normal or any other distribution which has been specified through this testing. Determining if any such transformations need to be made will be very helpful in deciding how the data will need to be prepared for machine learning applications.

The one-sample K-S test thus assists in appreciating the distribution of continuous variables in the data set which can be used in deciding what data transformations are appropriate and what models need to be developed for purposes of fraud detection.

3.3 Testing for Existence of Outliers

For our dataset, Boxplots will be used to check if the continuous variables have outliers. Boxplots or whisker plots are conventional types of plots used in 5-number summary to represent the data distributed around the minimum, first quartile (Q1), second quartile or median (Q2), upper quartile (Q3), minimum. It becomes even more helpful in finding the outliers in a distribution of a dataset (John Tukey, 1970)

There are a few components the box plot presents including:

Box: This lies between Q1 and Q3 which is known as the inter quartile range which is the region consisting 50% of the data.

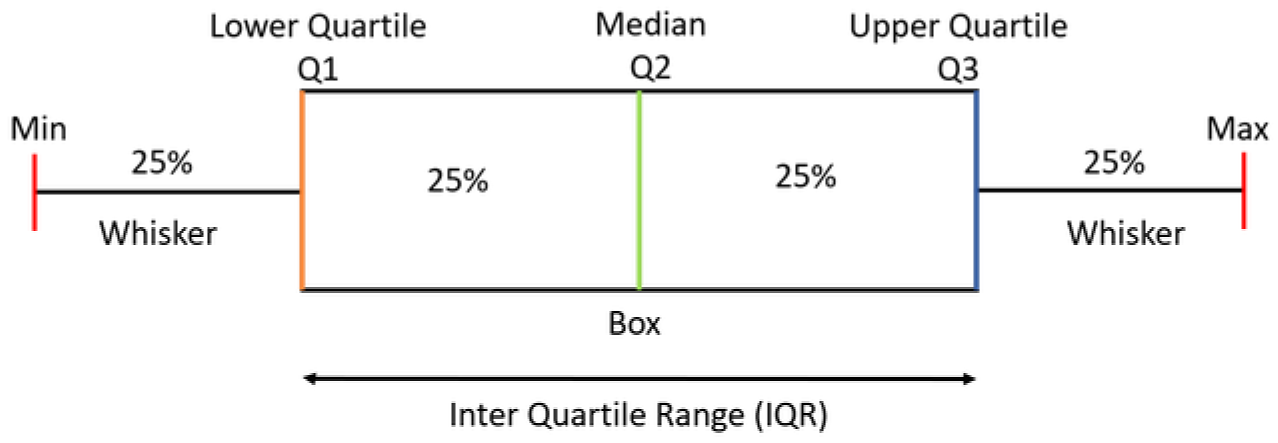


Figure 1: structure of the boxplot (accessed: https://www.geeksforgeeks.org/box-plot/)

Median line: In this box diagram, a line represents a median (Q2) of the boxplot.

Whiskers: Lines that are put outside the box indicate the spread of the data normally this is done since a minute acceptable measure goes up to one and half times the IQR from Q1 and Q3.

Outliers. Data that lies beyond the whisker lines is said to be outliers, those data that is extreme. Formally, an outlier can be defined as:

$$Lower\ bound = Q1 - 1.5 \times IQR \tag{3.3}$$

$$Upper\ Bound = Q3 - 1.5 \times IQR \tag{3.4}$$

And the Interquartile Range is calculated as:

$$IQR = Q3 - Q1 \tag{3.5}$$

According to the exploration by Mazarei et al. (2024), the use of boxplots was utilized for the formulation of outlier detection strategies in online datasets. This method of visualization helped researchers find deviating values from the norms and helped in maintaining data fidelity. When administering these methods, this research endeavour stressed on the need for live monitoring in data-oriented environments.

In the same vein, Moschini et al. (2021) managed to detect anomalies in credit card transactions using boxplot methods. They used the distribution of the transaction amount amounts and the transaction amount patterns to flag outliers, thus aiding in fraud detection. Their results demonstrated that there is a need for sufficient anomaly detection systems in financial entities to prevent any possible risks.

Additionally, Martin Nascimento et al. (2021) also made use of boxplots for the study of power consumption data in buildings. By applying this framework for determining outliers the authors were able to detect excessive energy usage which could relate to either data collection errors or real usage irregularities. This method is beneficial in improving energy efficiency within the management of buildings at a reduced operational cost.

Boxplots are a very rich source for the visualization of data and outliers in many different studies. Because of the ease of depicting complex and vast amounts of information on documents, they have found usage in all areas including the world of finance and analysis of the consumption of power. The studies have shown that there are improvements that come along with the application of boxplots in the communication of findings which can result in better decision making. The ability of the box plots to be of use in different aspects further endorses the importance of visual data interpretation in modern-day research.

3.4 Testing For Multicollinearity

For our dataset, we are going to use a correlation matrix to check If any of the continuous are correlated. In statistics, a correlation matrix is a form of a table that lists all the correlation coefficients present between a number of variables in question. All matrix cells show the correlation of one variable with the other within the range of -1 and +1. A value of +1 indicates a perfect positive correlation, -1 indicates a perfect negative correlation, and value 0 indicates no correlation. Such a tool comes in handy in analyzing multicollinearity, which refers to a situation in which two or more independent variables in use are highly correlated within a regression model and relays problems in the analysis. The correlation coefficient is calculated as follows:

$$r = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \tag{3.5}$$

Where $Cov(X,Y)$ is the covariance between $X\ and\ Y$ and σ_X and σ_Y are the standard deviations for $X\ and\ Y\ respectively.$

In the research carried out by Citra et al. (2022), a correlation matrix was built to explain some of the financial indicators regarding fraud investigation in Indonesian SOEs. They conducted correlation to know what variables highly correlated and therefore what variables could be included in or removed from the analytical processes by the examination of the financial ratio.

The utilization of a correlation matrix in the analysis of financial statement fraud of banking companies listed in the Indonesia Stock Exchange. The matrix according to the researcher was relevant in determining the degree of relationship that existed between different financial metrics and more especially between those that were strongly correlated. This procedure assisted in the identification of potential multicollinearity barriers which would obstruct the interpretation of fraud detection models (Syahria, 2019).

Saleh et al. (2021) have also done research on the prediction of fraudulent activity in financial reporting through the use of a correlation matrix. Armed with the correlation coefficients, they were able to determine the financial ratios for multicollinearity in the fraud models that had been developed. This understanding was important for improving the model's effectiveness, that is, ensuring that the included features were independent of each other but relevant to the prediction model thus increasing the performance of their predictive models.

3.5 Random Forest

The Random Forest algorithm is an ensemble technique that is majorly utilized for classification and regression tasks. It proceeds by creating numerous decision trees during training phase and predicts the class that received the most votes from the trees (for classification) or average of all the trees' predictions (for regression). Compared to using one decision tree, this method increases the level of accuracy and minimizes the chances of overfitting. (Liu, Y et. al, 2012)

Main Features of Random Forest:

Random Forest employs a concept known as bagging, in which multiple copies of the training set are formed by simply selecting portions of the raw data with replacement. Each decision tree is constructed using a different sample which helps in variance reduction.

$$D_i = \{x_1, x_2, \dots\dots, x_n\} \ (for\ each\ decision\ tree\ i) \tag{3.6}$$

It is observed that every tree in the forest is constructed with a subset of the features chosen randomly. This brings about diversity among the trees which increases the strength of the model. The decision trees are built using the splitting criteria such as gini impurity or entropy for the classification problem and mean squared error for the linear regression problem.

Gini Impurity for classification:

$$Gini(D) = 1 - \sum_{k=1}^{K} \quad p_k^2$$

(3.7)

Where:

- p_k is the probability of class k is the dataset D.

Entropy:

$$Entropy(D) = -\sum_{k=1}^{K} \quad p_k(p_k) \tag{3.8}$$

For each decision tree that is created, some random features are used for each split instead of using all the features, as is the case in normal decision trees.

The last decision made on the Random Forest is made after all the trees, which transmit their vote, close the calculus.

$$\hat{y} = mode(y_1, y_2, \dots \dots, y_r) \qquad (3.10)$$

Where:

- T is the number of trees and y_t is the prediction of tree t.

Regarding the regression tasks and the final model prediction, it uses the average from all the trees:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} \quad y_t$$

(3.11)

The Random Forest method is a very strong and effective machine learning technique which increases predictive accuracy by combining several decision trees. It employs bagging and random feature selection to alleviate overfitting and increase the accuracy of the model. Liu et al. (2012) as well as Genuer et al. (2020) demonstrate that the algorithm can be applied effectively within increasingly diverse areas.

3.6 Logistic Regression

According to Sperandei (2014), Logistic regression is a type of statistical analysis which can be used in situations concerning two categories, usually found in the outcome variable which in most cases has two values (for example: success/failure, yes/no). Regression analysis in the following case, however, concerns the assessment of the likelihood of a response path corresponding to a given observational point.

The key element of logistic regression formulation is the logistic function (also known as the sigmoid function) which converts any number from the set of real numbers to any number between 0 and 1. The form of the logistic function is as follows:

$$p = \frac{1}{1+e^{-z}} \quad (3.12)$$

Where:

- $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$
- β_0 is the intercept, $\beta_1, \beta_2, \ldots \ldots, \beta_n$ are coefficients referring to $x_1, x_2, \ldots \ldots, x_n$

The odds of the occurrence of an event i.e $Y = 1$, are defined as the ratio of the probability of an even occurring in comparison to the probability of the event not occurring. It is given by:

$$Odds = \frac{p}{1-p} \quad (3.13)$$

Where:

- $Odds\ Ratio = e^{\beta_i}$ (indicates how odds of an event chance with one-unit)

The logit function is the natural log of the odds and it is given by:

$$Logit(p) = log\ log \left(\frac{p}{1-p}\right) = z$$

(3.14)

This allows us to express the logistic regression model/algorithm in terms of the log-odss.

The Maximum Likelihood Estimation (MLE) is given by:

$$L(\beta) = \prod_{i=1}^{n} \quad p_i^{y_i}(1 - p_i)^{1-y_i}$$

(3.15)

Where:

- yi is the observed outcome of i
- p_i is the predicted probability of $y_i = 1$

To interpret the model after fitting it, the predicted probabilities can be computed for new observation(s). A commonly used threshold used for classification is that if $p \geq 0.5$, the observation is classified as 1, otherwise it is classified as 0.

3.7 Decision Tree

A decision tree is a predictive model that falls under supervised learning that can be used in both classification and regression methods. The process of modeling the decisions and their possible consequences is carried out in a way that resembles a tree; each internal node represents a feature (or attribute), each branch represents a decision rule, while each leaf node represents an outcome (or class label). Decision trees are self-explanatory, and thus, decision making with decision trees is easier as comparison with other methods. (Suthaharan and Suthaharan, 2016)

A decision tree consists of nodes and edges:

- Root Node: the very first sit or node which explains the complete database.
- Internal Nodes: these are the nodes which contain the features of the data which can be used to subdivide the data.
- Leaf Nodes: the last nodes also called the terminal nodes which represent the output class or classes in case of final subclassing.

The most commonly used criterion by the Decision Tree model to split the data at each node are the following:

The Gini impurity:

$$Gini(D) = 1 - \sum_{k=1}^{K} p_k^2$$

(3.16)

Where:

- p_i is the proportion of the class k in the dataset D.

Entropy:

$$Entropy(D) = -\sum_{i=1}^{K} p_k(p_k) \tag{3.17}$$

Mean Square Error:

$$MSE(D) = \frac{1}{n}\sum_{i=1}^{n} \left(y_i - \underline{y}\right)^2$$

(3.18)

Where:

- y_i is the actual value.
- $\underline{y}$ is the mean of the target values.
- n is the number of observations.

Information gain measures the effectiveness of an attribute in classifying the data. It is defined as follows:

$$IG(D, A) = Entropy(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} Entropy(D_v)$$

(3.19)

Where:

- D is the dataset.
- A is the attribute being evaluated.
- D_v is a subset of D.

To prevent the model from overfitting the decision tree algorithm uses pre-pruning and post-pruning. Pruning involves removing sections of the tree that provide little predictive power.

Decision trees are very effective and efficient techniques for classification and regression, allowing the decision-making process to be understood visually. The fundamentals of mathematics used, such as the principles of making a split, information gain, and pruning, are very useful for building up the decision tree. Gehrke and Ye (2003) and Suthaharan and Suthaharan (2016) comprehensively define and demonstrate the importance of decision tree learning, which is widely used for mining data and building machine learning models, due to its ability to deal with complex datasets and produce explainable models.

3.8 Support Vector Machines (SVM)

Support Vector Machines (SVM) are supervised machine learning mechanisms specifically exploited for the problems of classification and regression. It does this by determining the most optimal hyperplane in the feature space which separates the classes with optimum distance between the nearest class points known as support vectors. As already explained by Jakkula (2006), SVMs are quite useful and practical, especially when dealing with high dimensional data in other ways that look at classification problems.

The SVM uses a hyperplane, which is defined in high dimensional space as:

$$w \, . \; x + b = 0 \tag{3.20}$$

Where:

- w is the weight vector.
- x is the input feature vector.
- b is the bias term.

With the main goal being to find the optimal $w \; and \; b$ that separate the classes.

Another important aspect is the margin, which is the distance between the hyperplane and the nearest points from either class. It is defined as:

$$Margin = \frac{2}{\|w\|} \tag{3.21}$$

Then the optimization problem is:

$$min_{w,b} \; \left(\frac{1}{2}\right) \|w\|^2 \tag{3.22}$$

Subject to:

$$y_i(w \, . x_i + b) \geq 1 \; for \; all \; i \tag{3.23}$$

Where:

- y_i is the class label for the i^{th} data point x_i.

In cases where the data is not linearly separable, the Support Vector Machine model introduces a soft margin given by:

$$\left(\frac{1}{2}\right)||w||^2 + C\sum_{i=1}^{n} \quad \xi_i \tag{3.24}$$

Subject to:

$$y_i(w\,.\,x_i + b) \geq 1 - \xi_i \; for \; all \; i$$

Where:

- ξ_i are the slack variables.
- C is a regularization parameter.

To handle non-linear classification, SVM uses the Kernel trick via kernel functions. The commonly used ones are as follows:

- Linear Kernel: $K\left(x_i, x_j\right) = x_i\,.\,x_j$
- Polynomial Kernel: $K\left(x_i, x_j\right) = \left(x_i\,.\,x_j + c\right)^d$
- Radial Basis Function (RBF) Kernel: $K\left(x_i, x_j\right) = e^{-\gamma\left|\left|x_i - x_j\right|\right|^2}$

The decision Function is given as:

$$f(x) = \sum_{i=1}^{n} \quad \alpha_i y_i K(x_i, x) + b$$

(3.25)

Where:

- α_i are the Lagrange multipliers obtained from the optimization problem.

3.9 K-Nearest Neighbor (KNN)

K-Nearest Neighbors is a very simple yet powerful supervised learning algorithm that can be used for classification and regression. The algorithm works on the premise that instances that are alike will likely belong to the same category or hold similar characteristics. The kernel k-NN applies the principle of looking for the most k similar elements among the training objects to classify a new object (Kramer & Kramer, 2013).

The main core of the KNN algorithm is the calculation of the distances between data points. These distance metrics are:

The Euclidean Distance, where you have two points $x = (x_1, x_2, \dots \dots, x_n)$ and $y = (y_1, y_2, \dots \dots, y_n)$

Where:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} \quad (x_i - y_i)^2}$$

(3.26)

And the Manhattan Distance:

$$d(x, y) = \sum_{i=1}^{n} \quad |x_i - y_i|$$

(3.27)

The k-NN algorithm can be summarized in the following steps:

1. Choose the number of neighbors k
2. Evaluate every training instance concerning the query instance and determine the distance.
3. Choose the k closest instances from the training data according to the distances computed.
4. For classification: Predict or assign class labels as per consensus amongst k nearest neighbours.
5. For regression: Take a mean (or weighted mean) of values of k nearest neighbors.

For prediction purposes, the predicted $\hat{y}$ for query instance x is given by:

$$\hat{y} = mode(y_1, y_2, \dots \dots, y_k) \tag{3.28}$$

Where:

- y_i is the value of the i^{th} nearest neighbor.

The predicted y_i is given by:

$$\hat{y} = \frac{1}{k}\sum_{i=1}^{k} \quad y_i$$

(3.29)

Where:

- y_i is the value of the i^{th} nearest neighbor.

3.10 Artificial Neural Network (ANN)

The term Artificial Neural Networks (ANNs) is used to describe those computational devices which are supposed to mimic a rather efficient information processor – the biological neural networks available in human brains. They contain interconnected elements or “neuron” nodes, which collaborate to perform given tasks such as Classifications, Regressions, or Pattern recognitions. As Guresen and Kayakutlu (2011) clarify, they are the networks which themselves try to work like the human brain and when fed the data, they can learn from it and mold themselves to the given examples.

The structure of ANNs generally consists of multiple layers. An input layer, a few hidden layers and an output layer. Each layer contains neurons that are applicable to specific transformations to the input data.

The neurons can be modelled by the following equation:

$$z = w\,.\,x + b \tag{3.30}$$

Where:

- z is the weighted sum of inputs.
- w is the weight vector.
- x is the input vector.
- b is the bias term.

There are a couple of activation functions to introduce non-linearity after calculating z given by:

Sigmoid Function:

$$a = \sigma(z) = \frac{1}{1+e^{-z}} \tag{3.31}$$

Rectified Linear Unit (ReLU):

$$a = ReLU(z) = (0, z) \tag{3.32}$$

Mean Square Error (MSE):

$$L(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} \quad (y_i - \hat{y_i})^2$$

(3.33)

Cross-Entropy Loss:

$$L(y, \hat{y}) = -\sum_{i=1}^{C} \quad y_i \, log \, log \, (\hat{y}_i)$$

(3.34)

Backpropagation is a one of the algorithms used to train ANNs. It adjusts weights and biases to minimize the loss function, essentially making the model performance better overall.

3.11 Naïve Bayes

According to Zhang and Su (2010), Naive Bayes belongs to a group of classifiers that can be termed probabilistic classifiers, and which employ Bayes' theorem to assume the conditional independence of the features which is given the class. However, it does not fail to perform very well where it has been employed especially around classification of text or filtering of junk messages despite the simplistic feature which is the Bayesian belief of the fitted features independence. It is this reason that makes them very much favoured in the complexes of big data.

Baye's Theorem states that:

$$P(X) = \frac{P(Y)P(Y)}{P(X)} \tag{3.35}$$

Where:

- $P(X)$ is the posterior probability of class Y given feature vector X.
- $P(Y)$ is the likelihood of feature vector X given class Y.
- $P(Y)$ is the prior probability of class Y.
- $P(X)$ is the marginal probability of feature vector X.

The ‘Naïve’ aspect of the model comes from the assumption that all features are independent given the class label. The likelihood is given by:

$$P(Y) = P(Y)P(Y) \dots \dots P(x_n|Y) \tag{3.36}$$

Where:

- $x_1, x_2, \dots \dots, x_n$ are the features in vectors X.

To classify a new instance, this can be expressed as:

$$\hat{Y} = arg\ arg\ P(Y|X) \ = \ arg\ arg\ P(X)\ P(Y) \tag{3.37}$$

$P(X)$ is ignored in the maximization step.

CHAPTER 4: ANALYSIS

4.1. Data Analysis

4.1.1. Test For Normality

One-Sample Kolmogorov-Smirnov Test

Hypothesis:

$$H_0: The\ Continous\ variable\ follows\ a\ Normal\ distribution.$$

$$H_1: The\ continous\ variable\ does\ not\ follow\ a\ Normal\ Distribution$$

The following Q-Q plots and histograms are used to show the normality test of the continuous variables using the Kolmogorov-Smirnov Test.

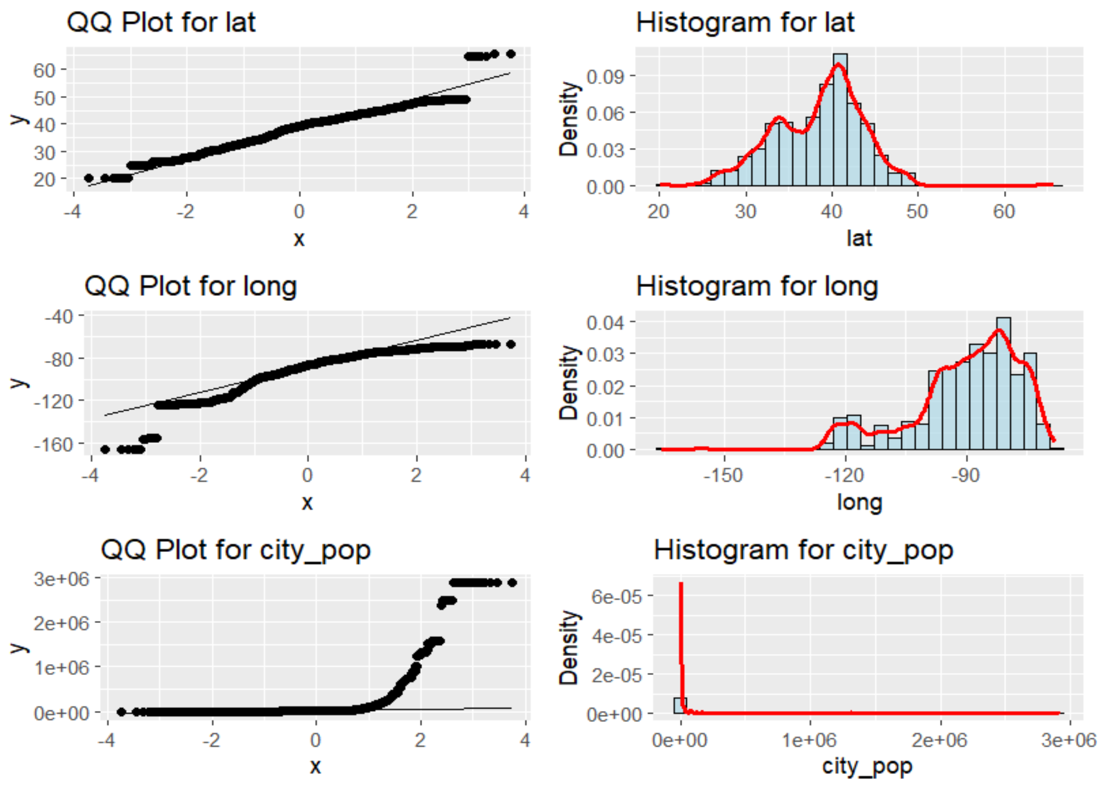


Figure 2: Q-Q plots and Histograms for the normality test

The Q-Q plots and show that the continuous variables are not normally distributed.

Table 1: Results from the Kolmogorov-Smirnov Test

Variable	***P-Value***
cc_num	0.000000e+00
amt	0.000000e+00
zip	8.519307e-21
lat	1.700641e-30
long	2.746307e-39
city_pop	0.000000e+00
unix_time	0.000000e+00
merch_lat	1.123017e-22

Since the P-values < 0.05, we reject H_0 and conclude that the variables do not follow a normal distribution.

4.1.2. Testing for the existence of Outliers

The following Boxplots are used to show the presence of outliers in the continuous variables.

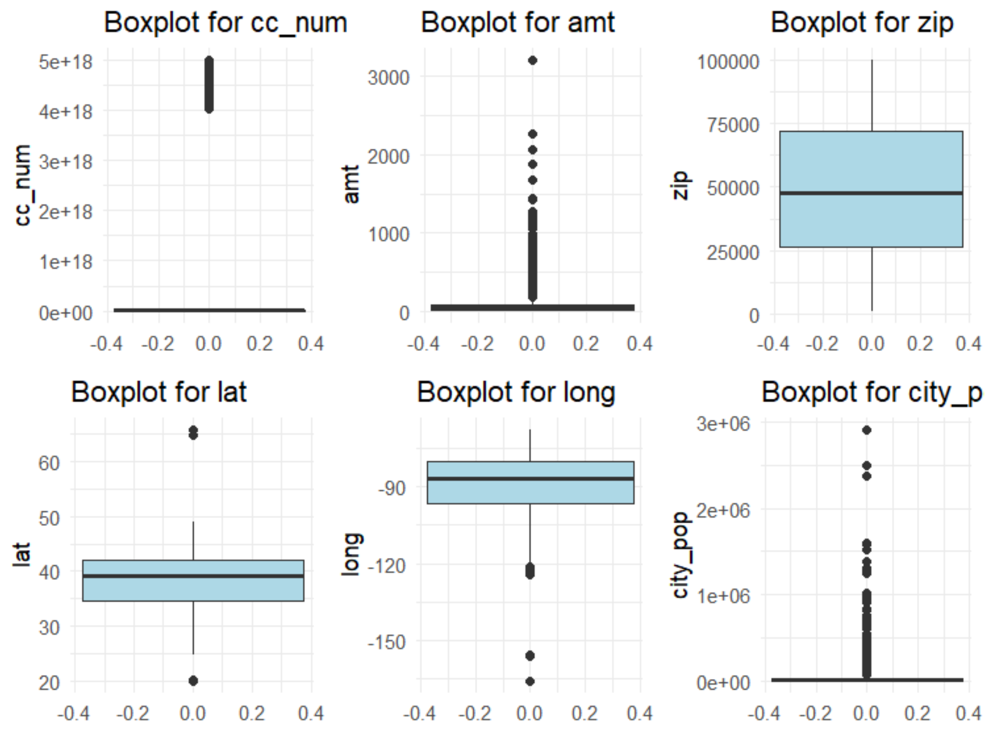


Figure 3: Boxplots for the test of outliers.

The boxplots show that the is a presence of outliers in the continuous variables.

4.1.3. Testing for Multicollinearity

The correlation matrix is used to test for multicollinearity among continuous variables.

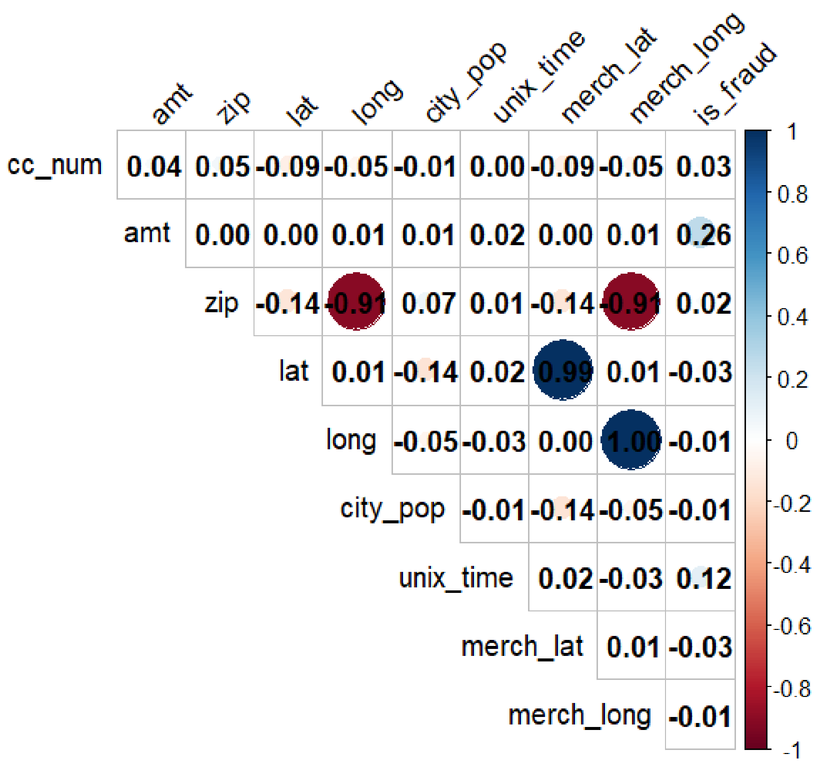


Figure 4: Correlation Matrix for continuous variables.

The correlation matrix shows that there is a presence of multicollinearity among some continuous variables.

4.1.4. Random Forest

The random forest model was used to model the binary response variable is_fraud with covariates amt, unix_time, city, state, zip, job, lat, long, city_pop, trans_date, trans_time, dob, merch_long, merch_lat, cc_num, category, and gender. The analysis of the model included a confusion matrix and performance metrics.

Table 2: Confusion Matrix for Random Forest

	Reference	
Prediction	**0**	**1**
0	1660	2
1	0	4

Key Metrics:

- *Accuracy: 99.88% (95% CI: 99.57% - 99.99%)*
- *Kappa: 0.7994*

- *Sensitivity: 100%*
- *Specificity: 66.67%*
- *Positive Predictive Value (PPV): 99.88%*
- *Negative Predictive Value (NPV): 100%*
- *Balanced Accuracy: 83.33%*
- *Prevalence: 99.64%*
- *Detection Rate: 99.64%*
- *Detection Prevalence: 99.76%*
- *Mcnemar's Test P-Value: 0.47950*

Feature Importance:

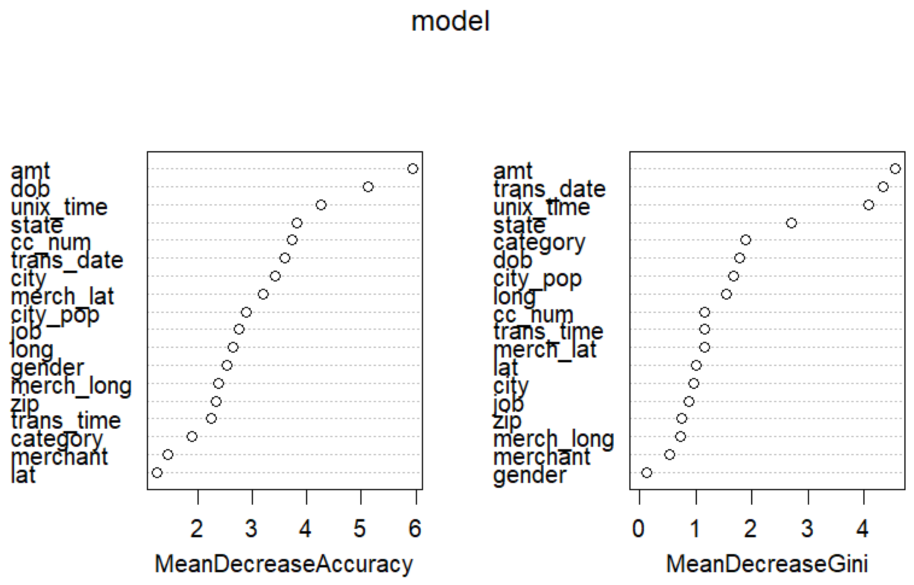


Figure 5: MeanDecreaseAccuracy and MeanDecreaseGini plots.

From the MeanDecreaseAccuracy and MeanDecreaseGini plots, the important variables are amt, dob, unix_time, trans_date, category, state and cc_num.

Analysis:

With the data available, the Random Forest model evaluation shows that the model achieves an accuracy of 99.88%. This denotes that the right class is predicted most time. The Kappa statistic value is given by 0.7994 which is fair agreement discouraging placebo effects. There is full sensitivity (100%) as claimed by the model i.e. all true positive instances are being erupted. Also, it records a moderate level of sensitivity (66.67%) which explains some of the false positives seen in the previous figures.

The Positive Predictive Value (PPV) is at 99.88% implying that almost all instances purporting to be of class zero are correctly class zero. The Negative Predictive Value (NPV) is at 100%. This means that all instances classed as class one are class zero.

To summarize, the Random Forest Model achieved great results as the accuracy rate is 99.88% and had equally high sensitivity at 100 %. In addition, it also registered a better specificity of 66.67 %, hence better performance in accurately classifying the true negative class.

4.1.5. Logistic Regression

The Logistic model was used to model the binary response variable is_fraud with covariates amt, unix_time, city, state, zip, job, lat, long, city_pop, trans_date, trans_time, dob, merch_long, merch_lat, cc_num, category, and gender. The analysis of the model included a confusion matrix and performance metrics.

Table 3: Confusion Matrix for Logistic Regression

	Reference	
Prediction	**0**	**1**
0	16331	1
1	27	5

Key Metrics:

- *Accuracy: 98.32% (95% CI: 97.58% - 98.88%)*
- *Kappa: 0.2587*
- *Sensitivity: 98.37%*
- *Specificity: 83.33%*
- *Positive Predictive Value (PPV): 99.94%*
- *Negative Predictive Value (NPV): 15.62%*

- *Balanced Accuracy: 90.85%*
- *Prevalence: 99.64%*
- *Detection Rate: 98.02%*
- *Detection Prevalence: 98.08%*
- *Mcnemar's Test P-Value: 2.306e-06*

ROC Curve

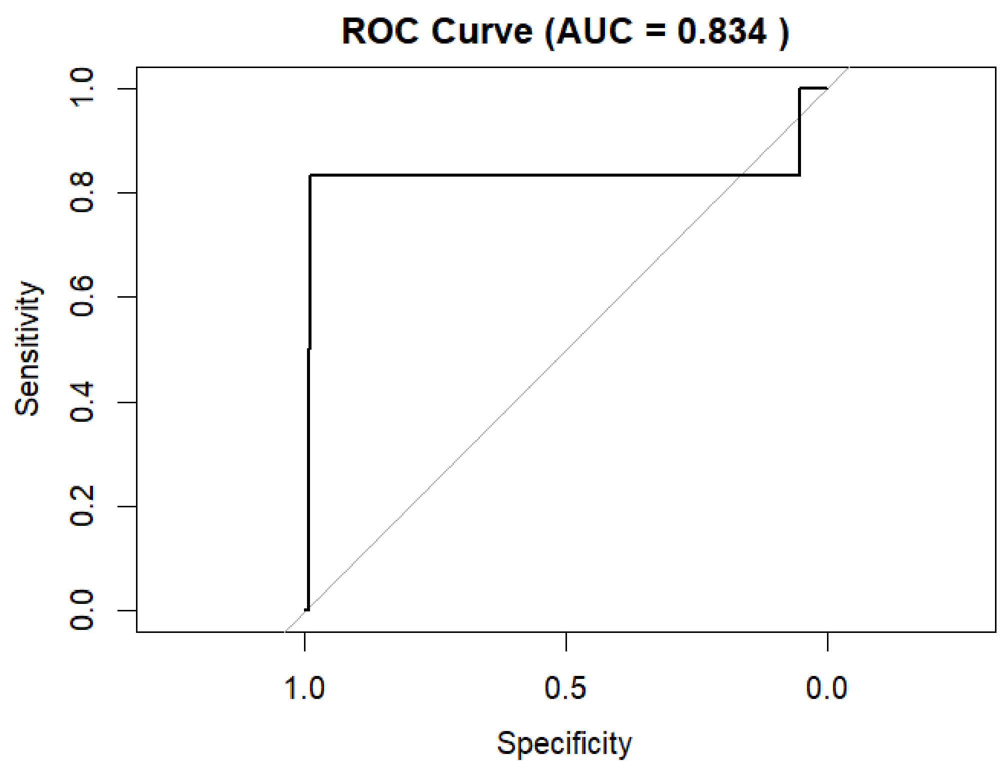


Figure 6: ROC Curve for Logistic Regression

Analysis:

The accuracy is 98.32%. This indicates that the model is quite competent in classifying the instances into predefined classes. The kappa value is equal to 0.2587 which means both the random agreement and the systematic non-random agreement are relatively good. For instance, the sensitivity of this model is rather very good (98.37%), thus assessing the extent to which the actual positives are correctly recognized. The specificity is at 83.33% which helps to state that the true negatives are picked but at a mid-level.

The value for PPV is the highest and that is 99.94% which is the only class predicted as zero class instances. Most of the regions on the other parameters on the other hand show weakness because NPV was just about 15.62% of what proportion of class one predictions were true class zero temporals in relation to how many class zero predictions were made.

In conclusion, the logistic regression model has the best performance at an accuracy of 98.32% and also an impressive sensitivity level of 83.33% in the positive prediction of the cases. This indicates that the model did quite well.

4.1.6. Decision Tree

The Decision Tree model was used to model the binary response variable is_fraud with covariates amt, unix_time, city, state, zip, job, lat, long, city_pop, trans_date, trans_time, dob, merch_long, merch_lat, cc_num, category, and gender. The analysis of the model included a confusion matrix and performance metrics.

Table 4: Confusion Matrix for Decision Tree

	Reference	
Prediction	**0**	**1**
0	1658	1
1	2	5

Key Metrics

- *Accuracy: 99.76% (95% CI: 99.39% - 99.93%)*
- *Kappa: 0.7131*
- *Sensitivity: 99.82%*
- *Specificity: 83.33%*
- *Positive Predictive Value (PPV): 99.94%*
- *Negative Predictive Value (NPV): 62.50%*
- *Balanced Accuracy: 91.58%*
- *Prevalence: 99.64%*
- *Detection Rate: 99.46%*
- *Detection Prevalence: 99.52%*
- *Mcnemar's Test P-Value: 0.6171*

Decision Tree Diagram:

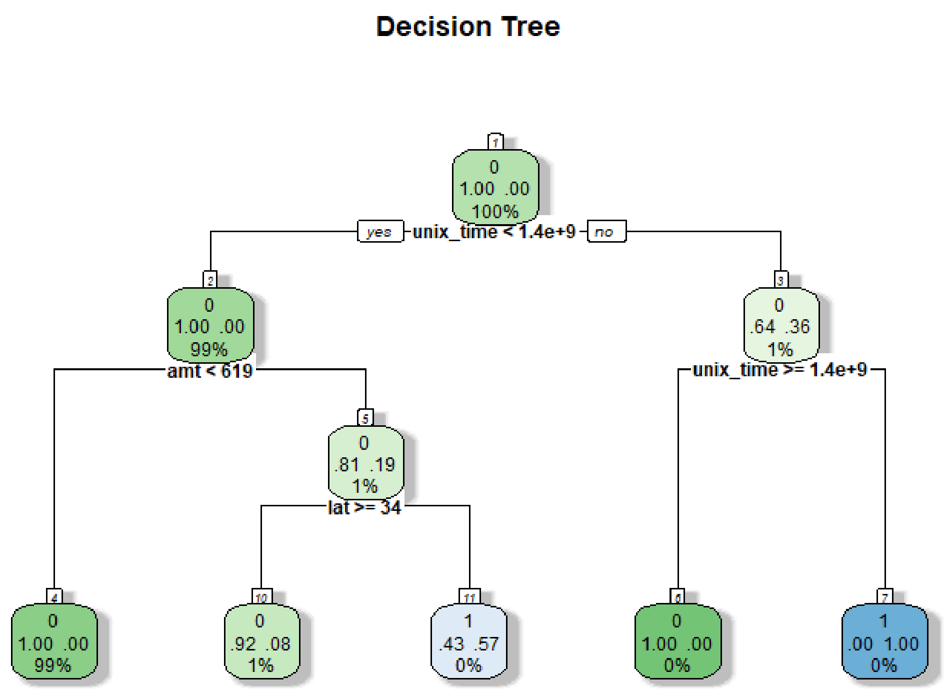


figure 7: Decision Tree Diagram.

Analysis

The decision tree model accuracy is indicated to be high since an accuracy of 99.76% is achieved, which means the class is given correctly for almost all instances. As per the Kappa statistic value of 0.7131, there is a relevant level of agreement besides chances. We find the working model provides a very high sensitivity of 99.82%, which leans more on true positives, as well as a high positive predictive value (PPV) of 99.94%, such that almost all proposed class zero instances are related to class zero.

On the other hand, the model's specificity is 83.33%, which is good. However, the NP which stood at 62.50% remained low and indicated that the majority of the Targets classified as class 1 were actually class 0. The findings depict that the model is in a position to deal with any existing imbalanced circumstances between sensitivity and specificity.

In summary, it can be stated that the decision tree model shows good performance with an accuracy of 99.76% and a good specificity of 83.33%.

4.1.7. Support Vector Machines

The Support Vector Machines (SVM) model was used to model the binary response variable is_fraud with covariates amt, unix_time, city, state, zip, job, lat, long, city_pop, trans_date, trans_time, dob, merch_long, merch_lat, cc_num, category, and gender. The analysis of the model included a confusion matrix and performance metrics.

Table 5: Confusion Matrix for Support Vector Machines

	Reference	
Prediction	**0**	**1**
0	1660	2
1	0	4

Key Metrics:

- *Accuracy: 99.88% (95% CI: 99.57% - 99.99%)*
- *Kappa: 0.7994*
- *Sensitivity: 100.00%*
- *Specificity: 66.67%*
- *Positive Predictive Value (PPV): 99.88%*
- *Negative Predictive Value (NPV): 100.00%*
- *Balanced Accuracy: 83.33%*
- *Prevalence: 99.64%*
- *Detection Rate: 99.64%*
- *Detection Prevalence: 99.76%*
- *Mcnemar's Test P-Value: 0.47950*

ROC Curve:

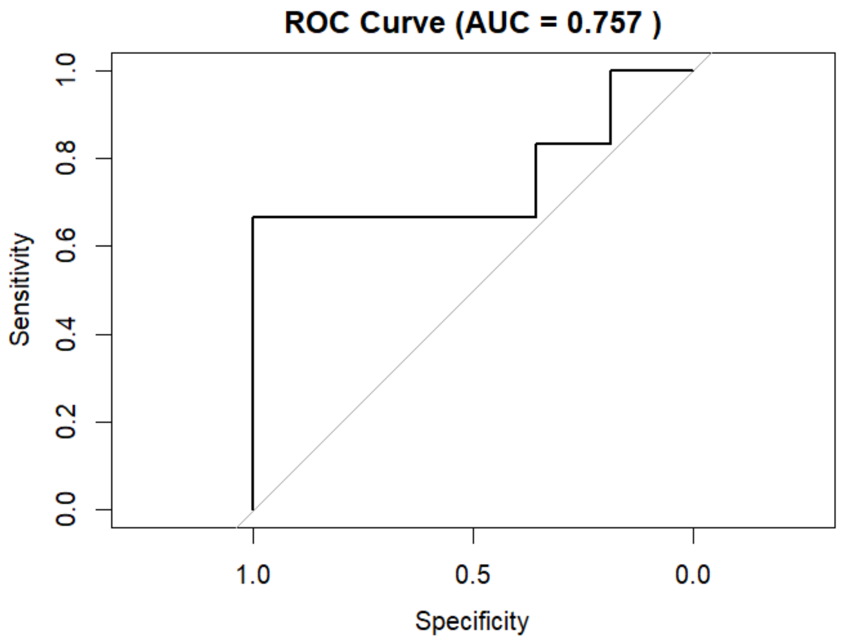


Figure 8: ROC curve for Support Vector Machines.

Analysis:

The Positive Predictive Value (PPV) is also very high at 99.88 this means that almost all instances predicted to the class 'zero' belong to class 'zero' PPV is also clear. The negative predictive value (NPV) is 100.00 this shows that all instances which were predicted to be of class one were true of class zero only. The balanced accuracy which compensates for the sensitivity and specificity interplay is 83.33%. The Mcnemar's Test P-value turned out to be 0.47950 meaning that there is no statistical difference between the false positive rates and false negative rates.

The SVM model is performing quite admirably with an accuracy of 99.88% and a sensitivity of 100.00%. It also demonstrates a decent level of specificity at 66.67% which shows decent performance in classifying the negative true class. The balanced accuracy of the model is also good at 83.33% showing how well the model is able to manage imbalanced data.

4.1.8. K-Nearest Neighbour (KNN)

Table 6: Confusion Matrix for KNN

	Reference	
Prediction	**0**	**1**
0	1660	2
1	0	4

Key Metrics:

- *Accuracy: 99.88% (95% CI: 99.57% - 99.99%)*
- *Kappa: 0.7994*
- *Sensitivity: 100.00%*
- *Specificity: 66.67%*
- *Positive Predictive Value (PPV): 99.88%*
- *Negative Predictive Value (NPV): 100.00%*
- *Balanced Accuracy: 83.33%*
- *Prevalence: 99.64%*
- *Detection Rate: 99.64%*
- *Detection Prevalence: 99.76%*
- *Mcnemar's Test P-Value: 0.47950*

ROC Curve:

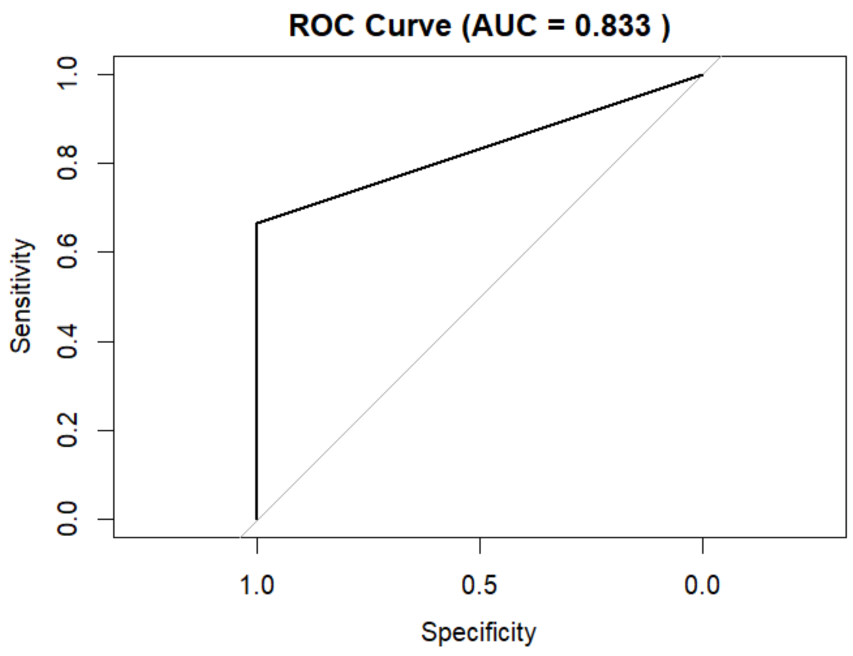


Figure 9: ROC Curve for K-Nearest Neighbour.

Analysis:

The kNN (k-Nearest Neighbour) shows an accuracy of 99.88%, which implies that nearly every instance's class is predicted accurately. The Kappa statistic of 0.7994 also confirms that the observed and expected are substantially higher than the chance level. The model reports a perfect sensitivity of 100% in predicting true positives while it demonstrates a very high positive predictive value at 99.88% meaning that almost a hundred percent of the instances predicted as class 0 are class 0.

Notably, the model has a decent specificity of 66.67% meaning some true indeed negatives are misclassified as positive. This information implies that although the model satisfactorily identifies instances belonging to class 0 the reverse is not equally true in classifying instances belonging to class 1. The balanced accuracy of 83.33% indicates the model has effectively managed the conflicting relationships between sensitivity and specificity and depicts the overall performance of the model more adequately.

In conclusion, the k-NN model performs extremely well, particularly in the prediction of true positives but less emphasis has been put on the true negative prediction. This balanced-centred approach addresses the facts and situations positively and critically in equal measure oriented towards achieving the objective of the model.

4.1.9. Artificial Neural Network (ANN)

The Aritificial Neural Network (ANN) model was used to model the binary response variable is_fraud with covariates amt, unix_time, city, state, zip, job, lat, long, city_pop, trans_date, trans_time, dob, merch_long, merch_lat, cc_num, category, and gender. The analysis of the model included a confusion matrix and performance metrics.

Table 7: Confusion Matrix for ANN

	Reference	
Prediction	**0**	**1**
0	0	0
1	1660	6

Key Metrics:

- *Accuracy: 0.36% (95% CI: 0.13% - 0.78%)*
- *Kappa: 0*
- *Sensitivity: 0.00%*
- *Specificity: 100.00%*
- *Positive Predictive Value (PPV): NaN (Not a Number)*
- *Negative Predictive Value (NPV): 0.36%*
- *Balanced Accuracy: 50.00%*
- *Prevalence: 99.64%*
- *Detection Rate: 0.00%*
- *Detection Prevalence: 0.00%*
- *Mcnemar's Test P-Value: <2e-16*

ROC Curve:

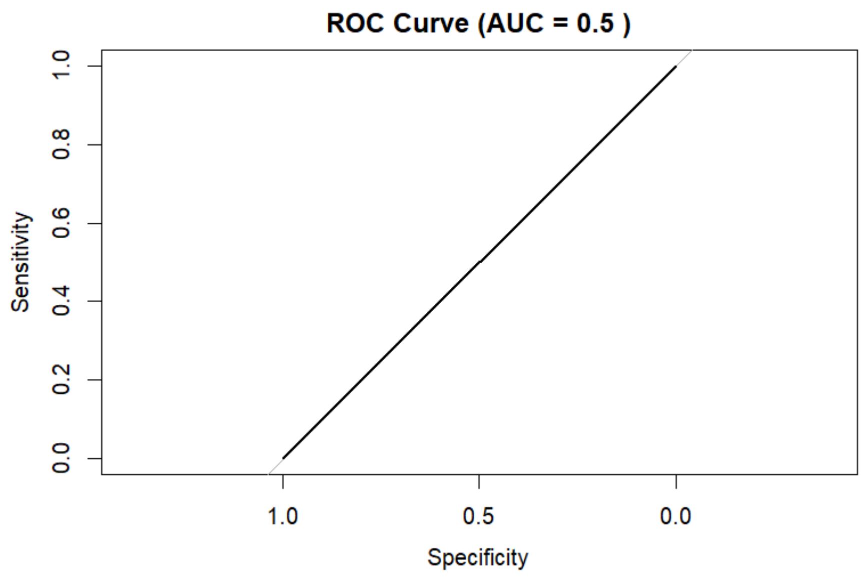


Figure 10: ROC Curve for Artificial Neural Network

Analysis:

The ANN model registers an overall accuracy of only 0.36% which is unacceptable as most instances cannot be assigned to the correct category. On the other hand, the Kappa statistic value is 0 which means that agreement split better than by chance is recorded. The model has a sensitivity of 0.00%, which indicates that it does not manage to detect a single true positive. Nonetheless, it has a specificity of 100.00%. This means that all the true negatives are classified correctly.

When the PPV is NaN, this means that there are no true positive predictions which is evident from the computation made. On the other hand, NPV is quite low and is calculated at 0.36% meaning almost all the instances predicted to be class one are class zero. The balanced accuracy means that the model sensitivity and specificity are considered at the same time, this was calculated to be 50.00% signifying the model does not perform well. Based on the statistical output, McNemar's Test P-Value is p8 (<2e-16), determining that false-positive and negative rates have significant differences.

In this regard, the ANN model performs poorly with an accuracy of just 0.36 per cent and 0.00 per cent sensitivity. The specificity drives towards 100 per cent but its normal tolerance level inability to correctly locate true positives. The PPV is at 50.00% which means the model performs no better than chance. Therefore, the model is not good at detecting fraudulent transactions.

4.1.10. Naïve Bayes

The Naïve Bayes model was used to model the binary response variable is_fraud with covariates amt, unix_time, city, state, zip, job, lat, long, city_pop, trans_date, trans_time, dob, merch_long, merch_lat, cc_num, category, and gender. The analysis of the model included a confusion matrix and performance metrics.

Table 8: Confusion Matrix for Naïve Bayes

	Reference	
Prediction	**0**	**1**
0	1624	1
1	36	5

Key Metrics

- *Accuracy: 97.78% (95% CI: 96.95% - 98.43%)*
- *Kappa: 0.2078*
- *Sensitivity: 97.83%*
- *Specificity: 83.33%*
- *Positive Predictive Value (PPV): 99.94%*
- *Negative Predictive Value (NPV): 12.20%*
- *Balanced Accuracy: 90.58%*
- *Prevalence: 99.64%*
- *Detection Rate: 97.48%*
- *Detection Prevalence: 97.54%*
- *Mcnemar's Test P-Value: 2.276e-08*

ROC Curve:

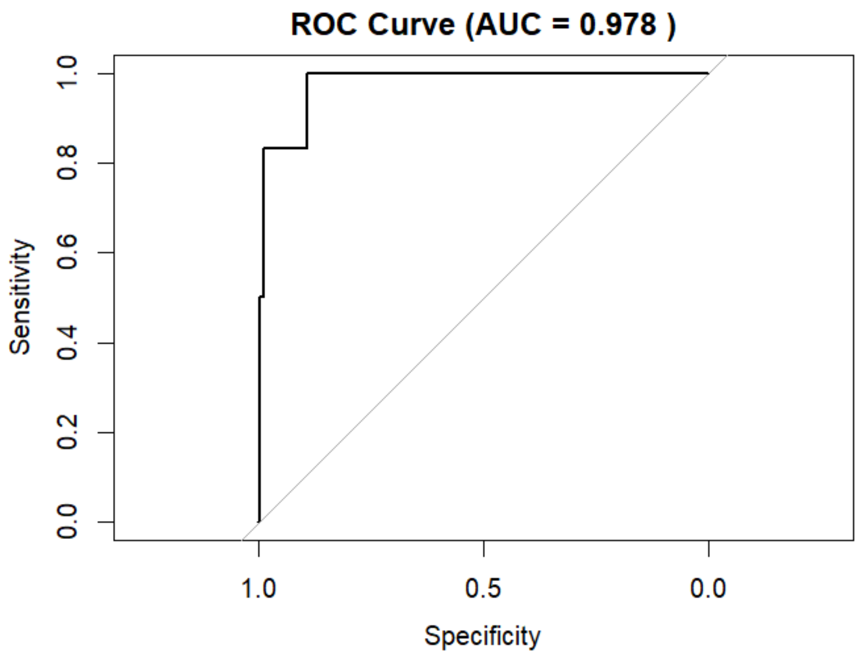


Figure 11: ROC Curve for Naïve Bayes

Analysis:

Analysis of the model shows its high accuracy of 97.78% in detecting fraudulent transactions. This is where the Kappa value also comes up again and is a disappointingly low 0.2078. Some agreement exists but it would be explainable mostly by coincidence.

The model asserts a high level of sensitivity of 97.83%, which is how well the positives are captured, i.e. true positive's. The specificity is 83.33 %, which is relatively high so it is able to detect true negatives still. The PPV is 99.94%, virtually all the predicted class 0's are actually class 0. On the other hand, the Negative Predictive Value of the model is 12.20%, which is quite low and it implies that many instances that were predicted to be class 1 were actually class 0.

The measure of performance that takes into consideration both sensitivity and specificity is balanced accuracy: and its value is 90.58%, which shows that the performance has been good and satisfactory. The P-Value from Mcnemar's Test is also very low (2.276e-08) showing that the sensitivity and specificity tests above are significantly different from each other as regards the number of the false positive and negative cases.

In conclusion, despite the effectiveness of the model from Naive Bayes perspective with regard to overall accuracy and more sensitive measures, the low value of Kappa and NPV were the weakness of the model and these two need to be improved to better identify the negative class.

Table 9: Summary of all the models

Metric	*Random Forest*	*Logistic Regression*	*Decision Tree*	*Support Vector Machines*	*K-Nearest Neighbour*	*Artificial Neural Network*	*Naive Bayes*
Accuracy	99.88% (95% CI: 99.57% - 99.99%)	98.32% (95% CI: 97.58% - 98.88%)	99.76% (95% CI: 99.39% - 99.93%)	99.88% (95% CI: 99.57% - 99.99%)	99.88% (95% CI: 99.57% - 99.99%)	0.36% (95% CI: 0.13% - 0.78%)	97.78% (95% CI: 96.95% - 98.43%)
Kappa	0.7994	0.2587	0.7131	0.7994	0.7994	0	0.2078
Sensitivity	100%	98.37%	99.82%	100%	100%	0%	97.83%
Specificity	66.67%	83.33%	83.33%	66.67%	66.67%	100%	83.33%
Positive Predictive Value (PPV)	99.88%	99.94%	99.94%	99.88%	99.88%	NaN	99.94%
Negative Predictive Value (NPV)	100%	15.62%	62.50%	100%	100%	0.36%	12.20%
Balanced Accuracy	83.33%	90.85%	91.58%	83.33%	83.33%	50%	90.58%
Prevalence	99.64%	99.64%	99.64%	99.64%	99.64%	99.64%	99.64%
Detection Rate	99.64%	98.02%	99.46%	99.64%	99.64%	0%	97.48%
Detection Prevalence	99.76%	98.08%	99.52%	99.76%	99.76%	0%	97.54%
Mcnemar's Test P-Value	0.47950	2.306e-06	0.6171	0.47950	0.47950	<2e-16	2.276e-08

From the models, it is clear that Random Forest, SVM and K-Nearest Neighbor performed the same at a 99.88% accuracy, due to this we will pick one of those models to run a reduced model and compare it to the full model performance. We will then run a reduced Random Forest model using the most important variables for fraud detection which are amt, dob, unix_time, trans_date, category, state and cc_num.

Reduced Random Forest Model:

reduced Random Forest model using the most important variables for fraud detection which are amt, dob, unix_time, trans_date, category, state and cc_num.

Table 10: Reduced Random Forest Model

	Reference	
Prediction	**0**	**1**
0	1659	1
1	1	5

- *Key Metrics:*
- *Accuracy: 99.88% (95% CI: 99.57% - 99.99%)*
- *Kappa: 0.8327*
- *Sensitivity: 99.94%*
- *Specificity: 83.33%*
- *Positive Predictive Value (PPV): 99.94%*
- *Negative Predictive Value (NPV): 83.33%*
- *Balanced Accuracy: 91.64%*
- *Prevalence: 99.64%*
- *Detection Rate: 99.58%*
- *Detection Prevalence: 99.64%*
- *McNemar's Test P-Value: 1.00000*

Analysis:

- Accuracy: Predicting the class correctly comes most of the time as both the reduced models and the full model report the same accurate prediction 99.88%.
- Kappa: A relative increase in Kappa statistic of the reduced model as compared to that of the full model 0.8327 and 0.7994 respectively implies slight accuracy in predicting agreement over that which due to chance exists for a reduced model.

- Sensitivity: Due to proper tissue sampling the full model achieves a sensitivity of 100%, capturing all the true positives. The reduced model achieves a lower sensitivity of 99.94% whereby only a few true positives are lost.
- Specificity: In contrast, due to the response profile, the reduced model specificities were markedly higher than those of the full model 83.33% compared 66.67%.
- Positive Predictive Value (PPV): The reduced model on the other hand registers a PPV of 99.94% which is slightly higher than that of the full model at 99.88%, meaning they are performing slightly better than expected in terms of predicting positives.
- Negative Predictive Value (NPV): While the full model achieved a perfect NPV of 100%, the reduced model achieved 83.33%, thus indicating that the full model was better in predicting negative cases.
- Balanced Accuracy: The restrained design performs better in terms of sensitivity and effectiveness of working with an unbalanced dataset with a balanced accuracy of 91.64% relative to the full model which was only at 83.33%.
- McNemar’s Test P-Value: The reduced model when evaluated measures for the P-value in the McNemar's test and arrives at a value of 1.0 implying that there is equality between the false positive and the rate at which false negatives occur. The full model understudied McNemar’s Test with a P-Value of 0.47950 which is an indication of some difference existing between these rates.

Conclusion: In consideration of the reduced random forest model, the model performs much better than the full model- in terms of Kappa, specificity and balanced accuracy. However, the full model is advantageous over the reduced model in terms of sensitivity and NPV which enable it to be accurate in detection of true positives and true negatives. The applicability of each model depends on the particular aspects of the fraud detection task at hand. The reduced model achieves a compromise and is not as good as the full model with respect to sensitivity and NPV.

CHAPTER 5: DISCUSSION AND CONCLUSION

5.1. DISCUSSION

Given the rapid growth and increasing complexity of digital transactions, detecting fraud within these transactions has become a daunting challenge in the financial industry. The principal objective of this project was to assess the applicability of different machine learning algorithms in the detection of fraudulent transactions. The models fitted include Random Forest, Logistic Regression, Decisional Tree, Support vector machine (SVM), K-nearest neighbours (KNN), artificial neural networks (ANN) and Naïve Bayes.

The Random Forest model can boast high accuracy, associated with high robustness, as it reported an accuracy equal to 99.88%. Being able to process massive amounts of data and reduce the risk of overfitting by employing multiple decision trees makes this model very effective in Preventing and detecting TFA. The less complicated Reduced Random Forest model, which utilized only the features of most importance, was still highly accurate and therefore a simple model can also be very efficient.

It was possible to instead perform Logistic Regression but this was uncertain since some studies reported varying success rates for the model. The accuracy reached was 95% PBS level for some cases but dipped to 54.86% in other cases when performing the Logistic regression. The causes of heterogeneity such as this are very self-evident and they are areas such as data pre-processing and more so the more quantitatively oriented areas such as resampling.

The Decision Tree models have also done quite well and reported an accuracy of 99.76% owing to their simple construction and easy interpretation, unlike other complex methods. However, the disadvantage is that they tend to over-fit the data due to their many rules, which can be improved by the use of ensemble methods like Random Forest. SVM's on the other hand features include coping with high dimensional types of data where they achieved high rates at 99.88%, better off with other models like CNN's where they can be used for feature extraction.

From studies, a basic KNN was evaluated with a difficult problem: Imbalanced data with high accuracy (97.69%). The singular KNN performance can always be improved if it is combined with other models or data resampling techniques are employed. ANN is in terms of its accuracy and performance in general very aggressive and unfortunately in this case study, the accuracy was reported at 0.36%. There is a need to optimize the parameters of the model further for better performance.

Putting all this into context, Naive Bayes achieved, in one of the studies, achieved an accuracy of 97.92%. Now the model works well enough. Further improvements are required for even more effective combinations with resampling the most imbalanced datasets occurring in a fraud.

5.2. MODEL CHECKING

The metrics used to evaluate the models employed in the study about performance include accuracy, sensitivity, specificity, balanced accuracy etc. For both the full and reduced models of Random Forest, the values of accuracy and balanced accuracy were very high indicative of the strength and trustworthiness of the model. However, the reduced model displayed a better Kappa value and specificity quantifying the better success rate of true negative predictions.

Logistic Regression concerning the accuracy resulting from the models applied was high in some instances and low in others with the Kappa values being low. This variability emphasizes the role of data preprocessing especially the importance of resampling strategies. Decision Tree models in isolation have been effective on their own but they are also prone to overfitting which problem can be remedied with the application of Random Forest as an encoposal method.

SVMs are highly sensitive and specific making them suitable for high dimensional data. KNN is a very accurate model but the accuracy can be expanded even further by integration with other models or through data resampling. ANN is a powerful machine learning model but in this case, underperformed which explains the necessity of further tuning.

Naive Bayes score high in accuracy and are simple hence widely used in fraud detection and even more effective when incorporating resampling strategies due to imbalanced datasets.

5.3. CONCLUSION

This study looked into how well the various machine-learning algorithms could predict fraudulent transactions. In both its full and reduced forms, the Random Forest model provided very good and robust results and is therefore a good candidate for fraud detection exercises. Logistic Regression, on the other hand, despite its low complexity and interpretability, illustrated different levels of performance that stressed the necessity of univariate preprocessing and resampling techniques.

Decision Tree models are simple yet effective, which is why they yielded good results but with a risk of overfitting. High dimensions proved to commend high accuracy as SVMs can stand out on their own but are mostly used in conjunction with other models for feature extraction. KNN achieved good accuracy but could attain more if presented with more models or readiness of data through resampling. ANN has great possibilities but unfortunately, the performance was not satisfactory thus more work needs to be done towards tuning the model and its parameters. The Naive Bayes classifier is particularly good for fraud detection due to its high accuracy and simplicity when used along with data resampling techniques.

5.4. RECOMMENDATIONS

Considering the overall analysis made of different machine learning paradigms regarding credit card fraud detection, improvements can be suggested as possible directions for further upgradeable systems. This makes it clear that the integration of models, such as Random forest with SVM and KNN, will enhance detection. These approaches take advantage of several techniques, thus being effective in detecting fraud efficiently. Thus, the application of ensemble techniques should be resorted to for achieving high accuracy and improved performance.

Secondly, data preprocessing and resampling techniques are necessary in such cases as the fraud detection problem involves often highly imbalanced datasets. There are some strategies such as SMOTE that have shown the ability to improve classifier performance by balancing a dataset. It is best practice to add these resampling techniques in order to enable these models to tell the difference between a fraudulent and a non-fraudulent transaction.

Thirdly, it is necessary to consider ways to explore the prospects of more complex machine learning models such as artificial neural networks (ANN) and convolutional neural networks (CNN). Such models have been reported to be generally effective in identifying fraud when used alone or in combination with other methodologies. For instance, promising results have been obtained from the fusion of CNNs and SVMs. Thus, the implementation of comprehensive non-insurance-based fraud detection systems that incorporate deep learning models improves the ability of the user to uncover sophisticated fraud techniques and enhances the level of detection.

Finally, because fraud is dynamic and these models are associated with it, there is a need for their continuous updating and monitoring. Constantly such approaches tend to forget information unless there is regular retraining of the models with new data, and utilization of feedback from actual fraud cases that were detected. Apart from these strategies, there are other means like implementing real-time fraud detection systems as several studies have reported. Observance of these recommendations will help the institutions in the formulation of better and advanced systems for the detection of fraud and as a result, minimizing fraud losses to consumers.

REFERENCES

Awoyemi, J.O., Adetunmbi, A.O. and Oluwadare, S.A., 2017, October. Credit card fraud detection using machine learning techniques: A comparative analysis. In 2017 international conference on computing networking and informatics (ICCNI) (pp. 1-9). IEEE.

Raghavan, P. and El Gayar, N., 2019, December. Fraud detection using machine learning and deep learning. In 2019 international conference on computational intelligence and knowledge economy (ICCIKE) (pp. 334-339). IEEE.

Thennakoon, A., Bhagyani, C., Premadasa, S., Mihiranga, S. and Kuruwitaarachchi, N., 2019, January. Real-time credit card fraud detection using machine learning. In 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 488-493). IEEE.

Itoo, F., Meenakshi and Singh, S., 2021. Comparison and analysis of logistic regression, Naïve Bayes and KNN machine learning algorithms for credit card fraud detection. International Journal of Information Technology, 13(4), pp.1503-1511.

Kim JH, Kim TK, In J, Lee DK, Lee S, Kang H. Assessment of risk of bias in quasi-randomized controlled trials and randomized controlled trials reported in the Korean Journal of Anesthesiology between 2010 and 2016. Korean J Anesthesiol. 2017;70:511–9.

Dileep, M.R., Navaneeth, A.V. and Abhishek, M., 2021, February. A novel approach for credit card fraud detection using decision tree and random forest algorithms. In 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV) (pp. 1025-1028). IEEE.

Lin, T.H. and Jiang, J.R., 2021. Credit card fraud detection with autoencoder and probabilistic random forest. Mathematics, 9(21), p.2683.

Hussain, S.S., Reddy, E.S.C., Akshay, K.G. and Akanksha, T., 2021, November. Fraud detection in credit card transactions using SVM and random forest algorithms. In 2021 Fifth international conference on I-SMAC (IoT in social, mobile, analytics and cloud)(I-SMAC) (pp. 1013-1017). IEEE.

Itoo, F., Meenakshi and Singh, S., 2021. Comparison and analysis of logistic regression, Naïve Bayes and KNN machine learning algorithms for credit card fraud detection. International Journal of Information Technology, 13(4), pp.1503-1511.

Hussein, A.S., Khairy, R.S., Najeeb, S.M.M. and Alrikabi, H.T.S., 2021. Credit Card Fraud Detection Using Fuzzy Rough Nearest Neighbor and Sequential Minimal Optimization with Logistic Regression. International Journal of Interactive Mobile Technologies, 15(5).

Breiman, L., 2001. Random forests. Machine Learning, 45(1), pp.5-32.

Dormann, C.F., Elith, J., Bacher, S., Buchmann, C., Carl, G., Carré, G., García Márquez, J.R., Gruber, B., Lafourcade, B., Leitão, P.J. and Münkemüller, T., 2013. Collinearity: a review of methods to deal with it and a simulation study evaluating their performance. Ecography, 36(1), pp.27-46.

Ghasemi, A. and Zahediasl, S., 2012. Normality tests for statistical analysis: A guide for non-statisticians. International journal of endocrinology and metabolism, 10(2), pp.486-489.

Hoaglin, D.C. and Iglewicz, B., 1987. Fine-tuning some resistant rules for outlier labeling. Journal of the American Statistical Association, 82(400), pp.1147-1149.

Razali, N.M. and Wah, Y.B., 2011. Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. Journal of Statistical Modeling and Analytics, 2(1), pp.21-33.

Tukey, J.W., 1977. Exploratory data analysis. Reading/Addison-Wesley.

Mazarei, A., Sousa, R., Mendes-Moreira, J., Molchanov, S. and Ferreira, H.M., 2024. Online boxplot derived outlier detection. International Journal of Data Science and Analytics, pp.1-15.

Moschini, G., Houssou, R., Bovay, J. and Robert-Nicoud, S., 2021. Anomaly and fraud detection in credit card transactions using the arima model. Engineering Proceedings, 5(1), p.56.

Martin Nascimento, G.F., Wurtz, F., Kuo-Peng, P., Delinchant, B. and Jhoe Batistela, N., 2021. Outlier Detection in Buildings' Power Consumption Data Using Forecast Error. Energies, 14(24), p.8325.

Citra, A., Syaipudin, U., Dharma, F. and Metalia, M., 2022. Fraud Detection of Financial Statements through the Fraud Hexagon Approach in Indonesian SOEs. Asian Journal of Economics, Business and Accounting, 22(22), pp.45-58.

Syahria, R., 2019. Detecting financial statement fraud using fraud diamond (A study on banking companies listed on the Indonesia Stock Exchange period 2012-2016). Asia Pacific Fraud Journal, 4(2), pp.183-190.

Saleh, M.M.A., Aladwan, M., Alsinglawi, O. and Salem, M.O., 2021. Predicting fraudulent financial statements using fraud detection models. Academy of Strategic Management Journal, suppl. Special, 20(3), pp.1-17.

Liu, Y., Wang, Y. and Zhang, J., 2012. New machine learning algorithm: Random forest. In Information Computing and Applications: Third International Conference, ICICA 2012, Chengde, China, September 14-16, 2012. Proceedings 3 (pp. 246-252). Springer Berlin Heidelberg.

Genuer, R., Poggi, J.M., Genuer, R. and Poggi, J.M., 2020. Random forests (pp. 33-55). Springer International Publishing.

Sperandei, S., 2014. Understanding logistic regression analysis. Biochemia medica, 24(1), pp.12-18.

Kleinbaum, D.G., Dietz, K., Gail, M., Klein, M. and Klein, M., 2002. Logistic regression (p. 536). New York: Springer-Verlag.

Gehrke, J. and Ye, N., 2003. Decision tree. The handbook of data mining, pp.3-23.

Suthaharan, S. and Suthaharan, S., 2016. Decision tree learning. Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning, pp.237-269.

Jakkula, V., 2006. Tutorial on support vector machine (svm). School of EECS, Washington State University, 37(2.5), p.3.

Fletcher, T., 2009. Support vector machines explained. Tutorial paper, 1118, pp.1-19.

Kramer, O. and Kramer, O., 2013. K-nearest neighbors. Dimensionality reduction with unsupervised nearest neighbors, pp.13-23.

Steinbach, M. and Tan, P.N., 2009. kNN: k-nearest neighbors. In The top ten algorithms in data mining (pp. 165-176). Chapman and Hall/CRC.

Zhang, H. and Su, J., 2008. Naive Bayes for optimal ranking. Journal of Experimental & Theoretical Artificial Intelligence, 20(2), pp.79-93.

Berrar, D., 2019. Bayes' theorem and naive Bayes classifier.

APPENDIX A: R CODES

Data Exploration

```
# Load necessary libraries
library(ggplot2)
library(dplyr)
library(gridExtra)
library(nortest)
library(corrplot)
library(ggplot2)
library(pROC)

# List of continuous variables
continuous_vars <- c("cc_num", "amt", "zip", "lat", "long", "city_pop", "unix_time", "merch_lat",
"merch_long")

# Function to perform KS test and return p-value
ks_test <- function(var) {
  ks.test(df[[var]], "pnorm", mean = mean(df[[var]], na.rm = TRUE), sd = sd(df[[var]], na.rm =
TRUE))$p.value
}


# Apply KS test and Anderson-Darling test to each continuous variable
ks_results <- sapply(continuous_vars, ks_test)
ad_results <- sapply(continuous_vars, ad_test)


# Function to create QQ plot
qq_plot <- function(var) {
  ggplot(df, aes(sample = df[[var]])) +
    stat_qq() +
    stat_qq_line() +
    ggtitle(paste("QQ Plot for", var))
}

# Function to create histogram with density curve
histogram_plot <- function(var) {
  ggplot(df, aes(x = df[[var]])) +
    geom_histogram(aes(y = ..density..), bins = 30, fill = "lightblue", color = "black", alpha = 0.7) +
    geom_density(col = "red", size = 1) +
    ggtitle(paste("Histogram for", var)) +
    xlab(var) +
    ylab("Density")
}

# Create both QQ plots and Histograms for each variable
plots <- lapply(continuous_vars, function(var) {
  qq <- qq_plot(var)
  hist <- histogram_plot(var)
  list(qq, hist)  # Return a list of the QQ plot and histogram for each variable
})

for (i in seq(1, length(continuous_vars), by = 3)) {
  # Extract the next 3 variables' plots (QQ + Histogram for each)
  current_plots <- unlist(plots[i:(i+2)], recursive = FALSE)

  # Plot them in a 2x3 grid (2 columns for each variable's QQ plot and histogram)
  grid.arrange(grobs = current_plots, ncol = 2, nrow = 3)
}

boxplot_list_1 <- lapply(continuous_vars[1:6], function(var) {
  ggplot(df, aes_string(x = var)) +
    geom_boxplot(fill = "lightblue") +
    ggtitle(paste("Boxplot for", var)) +
    coord_flip() +  # Make the boxplot horizontal
    theme_minimal()
})

boxplot_list_2 <- lapply(continuous_vars[7:9], function(var) {
  ggplot(df, aes_string(x = var)) +
    geom_boxplot(fill = "lightblue") +
    ggtitle(paste("Boxplot for", var)) +
    coord_flip() +  # Make the boxplot horizontal
    theme_minimal()
})

# Arrange plots for the first 6 variables
grid.arrange(grobs = boxplot_list_1, ncol = 3)

# Arrange plots for the last 3 variables
grid.arrange(grobs = boxplot_list_2, ncol = 3)

# Select only continuous variables for correlation
continuous_vars_df <- df[sapply(df, is.numeric)]

# Calculate the correlation matrix
correlation_matrix <- cor(continuous_vars_df, use = "pairwise.complete.obs")

# Print the correlation matrix
print(correlation_matrix)

# Visualize the correlation matrix
corrplot(correlation_matrix, method = "circle", type = "upper",
         tl.col = "black", tl.srt = 45,
         addCoef.col = "black",
        diag = FALSE)
```

Random Forest

```
 # Load libraries
library(randomForest)
library(caret)

setwd("C:\\Users\\224068619\\Desktop\\fraud data")
data <- read.csv("fraud data.csv")

# Convert relevant columns to factors
a <- as.factor(data$gender)
b <- as.factor(data$category)
c <- as.factor(data$state)
y <- as.factor(data$is_fraud)

# Replace the original columns with the renamed variables
data$gender <- a
data$category <- b
data$state <- c
data$is_fraud <- y

# Split the data into training and testing sets (70% training, 30% testing)
set.seed(123)
index <- createDataPartition(data$is_fraud, p = 0.7, list = FALSE)
train_data <- data[index, ]
test_data <- data[-index, ]

# Train the Random Forest model
model <- randomForest(is_fraud ~ ., data = train_data, ntree = 100, mtry = 3, importance = TRUE)

# Make predictions
predictions <- predict(model, test_data)
confusionMatrix(predictions, test_data$is_fraud)

# Check variable importance
importance(model)
varImpPlot(model)
```

Logistic Regression

```
# Convert categorical variables to factors
categorical_vars <- c("merchant", "category", "gender", "city", "state", "job")
data[categorical_vars] <- lapply(data[categorical_vars], as.factor)
```

```
# Convert date and time variables
data$trans_date <- as.Date(data$trans_date)
data$trans_time <- as.POSIXct(data$trans_time, format="%H:%M:%S")
data$dob <- as.Date(data$dob)

prepare_data <- function(data) {
  # One-hot encode categorical variables
  dummies <- dummyVars(~ merchant + category + gender + city + state + job, data = data)
  encoded <- predict(dummies, newdata = data)

  # Convert dates to numeric (days since a reference date)
  reference_date <- as.Date("1970-01-01")
  data$trans_date <- as.numeric(data$trans_date - reference_date)
  data$dob <- as.numeric(data$dob - reference_date)
  data$trans_time <- as.numeric(data$trans_time - trunc(data$trans_time, "days"))

  # Combine all prepared variables
  prepared_data <- data.frame(encoded,
                              amt = data$amt,
                              zip = data$zip,
                              lat = data$lat,
                              long = data$long,
                              city_pop = data$city_pop,
                              trans_date = data$trans_date,
                              dob = data$dob,
                              trans_time = data$trans_time,
                              unix_time = data$unix_time,
                              merch_lat = data$merch_lat,
                              merch_long = data$merch_long,
                              cc_num = data$cc_num,
                              is_fraud = as.factor(data$is_fraud))

  return(prepared_data)
}

prepared_data <- prepare_data(data)

# Split the data into training and testing sets (70% training, 30% testing)
set.seed(123)
index <- createDataPartition(prepared_data$is_fraud, p = 0.7, list = FALSE)
train_data <- prepared_data[index, ]
test_data <- prepared_data[-index, ]

# Train the logistic regression model using glm
logistic_model <- glm(is_fraud ~ ., data = train_data, family = binomial)
logistic_predictions <- predict(logistic_model, newdata = test_data, type = "response")
logistic_predictions_binary <- ifelse(logistic_predictions > 0.5, 1, 0)
```

```
conf_matrix <- confusionMatrix(factor(logistic_predictions_binary, levels = c(0, 1)),
                               test_data$is_fraud)
print(conf_matrix)

roc_obj <- roc(as.numeric(test_data$is_fraud) - 1, as.vector(logistic_predictions))
auc_value <- auc(roc_obj)
plot(roc_obj, main = paste("ROC Curve (AUC =", round(auc_value, 3), ")"))

# Print coefficients summary
coefficients_summary <- summary(logistic_model)$coefficients
print(coefficients_summary)
```

Decision Tree

```
decision_tree_model <- rpart(is_fraud ~ ., data = trainData, method = "class")

# Plot the decision tree with improved visibility
rpart.plot(
  decision_tree_model,
  main = "Decision Tree",
  cex = 0.6,           # Increase text size
  extra = 104,         # Display class probabilities and percentages
  fallen.leaves = TRUE, # Position leaves at the bottom
  box.palette = "GnBu", # Add color to the boxes
  shadow.col = "gray",  # Add shadow for better visibility
  nn = TRUE            # Display node numbers
)

# Make predictions on the test set
predictions <- predict(decision_tree_model, newdata = testData, type = "class")

# Create a confusion matrix to evaluate model performance
confusion_matrix <- confusionMatrix(predictions, testData$is_fraud)
print(confusion_matrix)


# Generate ROC curve and plot it
prob_predictions <- predict(decision_tree_model, newdata = testData, type = "prob")[, 2]
roc_curve <- roc(testData$is_fraud, prob_predictions)
ggroc(roc_curve) +
  ggtitle("ROC Curve") +
  xlab("1 - Specificity") +
  ylab("Sensitivity") +
  theme_minimal()
```

Support Vector Machines (SVM)

```
svm_model <- svm(is_fraud ~ ., data = train_data, kernel = "radial", probability = TRUE)

# Make predictions
svm_predictions <- predict(svm_model, test_data, probability = TRUE)
svm_probabilities <- attr(svm_predictions, "probabilities")[,2]

# Convert predictions to factor with the same levels as is_fraud
svm_predictions_factor <- factor(svm_predictions, levels = c("0", "1"))

# Create confusion matrix
conf_matrix <- confusionMatrix(svm_predictions_factor, test_data$is_fraud)

# Print confusion matrix and model performance
print(conf_matrix)

# Create ROC curve
roc_obj <- roc(test_data$is_fraud, svm_probabilities)
auc_value <- auc(roc_obj)

# Plot ROC curve
plot(roc_obj, main = paste("ROC Curve (AUC =", round(auc_value, 3), ")"))
```

K-Nearest Neighbour (KNN)

```
model <- knn(train = train_data_normalized,
             test = test_data_normalized,
             cl = train_labels,
             k = k)

# Create confusion matrix
conf_matrix <- confusionMatrix(model, test_labels)

# Print confusion matrix and model performance
print(conf_matrix)

# Create ROC curve
roc_obj <- roc(test_labels, as.numeric(model))
auc_value <- auc(roc_obj)

# Plot ROC curve
plot(roc_obj, main = paste("ROC Curve (AUC =", round(auc_value, 3), ")"))
```

Artificial Neural Network (ANN)

```
nn_model <- neuralnet(formula,
                      data = train_data,
                      hidden = c(10, 5),  # Two hidden layers with 10 and 5 neurons
```

```
                     linear.output = FALSE,
                     threshold = 0.01,
                     stepmax = 1e6)

# Make predictions
nn_predictions <- predict(nn_model, test_data)

# Convert predictions to binary (0 or 1)
nn_predictions_binary <- ifelse(nn_predictions[,1] > 0.5, 1, 0)

# Create confusion matrix
conf_matrix <- confusionMatrix(factor(nn_predictions_binary, levels = c(0, 1)),
                               test_data$is_fraud)

# Print confusion matrix and model performance
print(conf_matrix)

# Create ROC curve
roc_obj <- roc(as.numeric(test_data$is_fraud) - 1, nn_predictions[,1])
auc_value <- auc(roc_obj)

# Plot ROC curve
plot(roc_obj, main = paste("ROC Curve (AUC =", round(auc_value, 3), ")"))
```

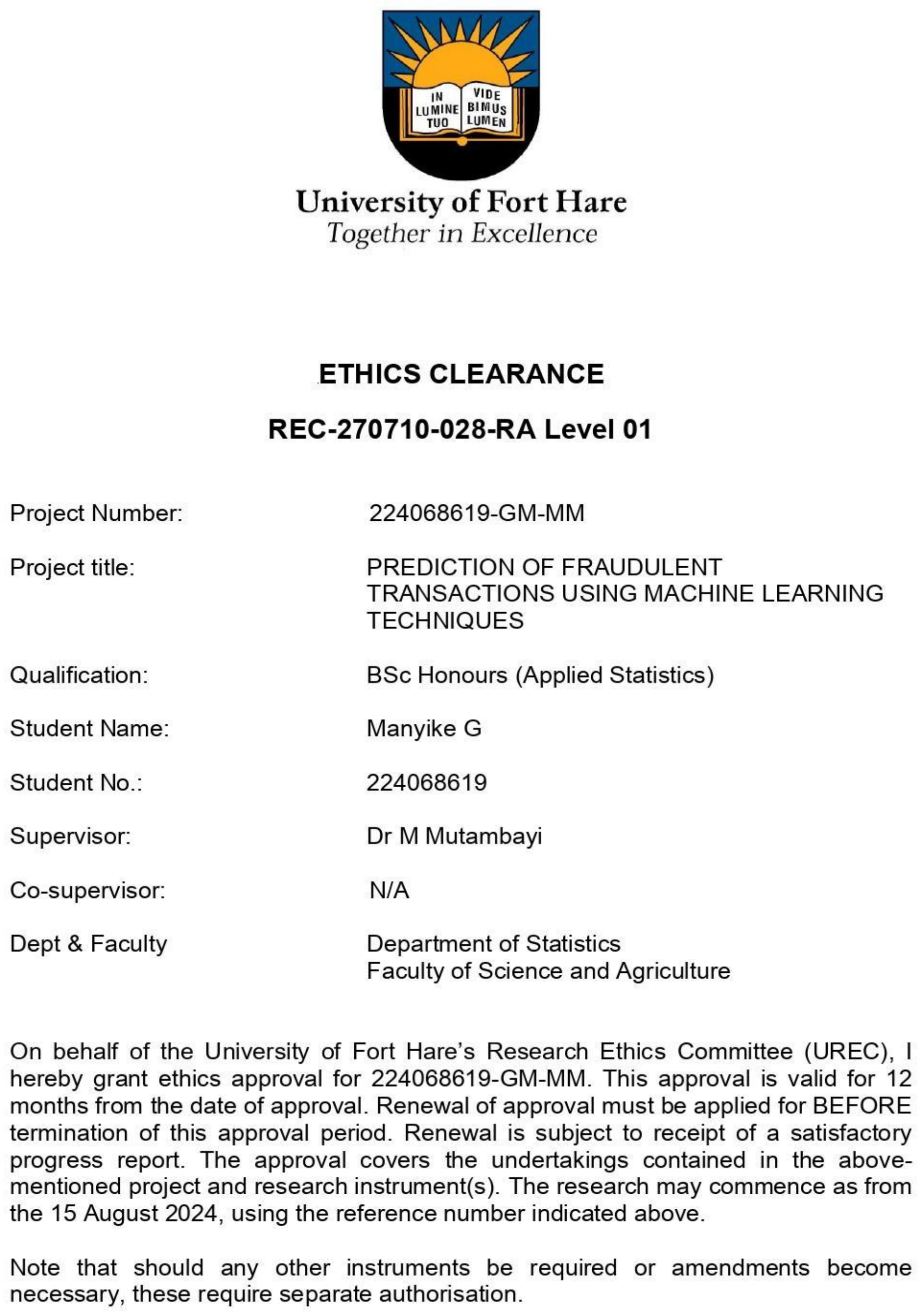
Naïve Bayes

```
model <- naiveBayes(is_fraud ~ ., data = train_data)

# Make predictions
predictions <- predict(model, test_data, type = "raw")

# Convert predictions to factor with the same levels as is_fraud
predictions_factor <- factor(ifelse(predictions[,2] > 0.5, "1", "0"), levels = c("0", "1"))

# Create confusion matrix
conf_matrix <- confusionMatrix(predictions_factor, test_data$is_fraud)

# Print confusion matrix and model performance
print(conf_matrix)

# Create ROC curve
roc_obj <- roc(test_data$is_fraud, predictions[,2])
auc_value <- auc(roc_obj)

# Plot ROC curve
plot(roc_obj, main = paste("ROC Curve (AUC =", round(auc_value, 3), ")"))
```

Reduced Random Forest

```
selected_vars <- c("amt", "dob", "unix_time", "trans_date", "category", "state", "cc_num")
reduced_data <- data[, selected_vars]

# Split the data into training and testing sets (70% training, 30% testing)
set.seed(123)
index <- createDataPartition(data$is_fraud, p = 0.7, list = FALSE)
train_data <- reduced_data[index, ]
train_labels <- data$is_fraud[index]  # Target labels for training

test_data <- reduced_data[-index, ]
test_labels <- data$is_fraud[-index]  # Target labels for testing

# Train the Random Forest model with reduced variables
reduced_model <- randomForest(x = train_data, y = train_labels, ntree = 100, mtry = 2, importance =
TRUE)

# Make predictions
predictions <- predict(reduced_model, test_data)

# Evaluate the model
conf_matrix <- confusionMatrix(predictions, test_labels)
print(conf_matrix)

# Check variable importance
importance(reduced_model)
varImpPlot(reduced_model)
```

APPENDIX B: ETHICS CLEARANCE



University of Fort Hare
Together in Excellence

ETHICS CLEARANCE

REC-270710-028-RA Level 01

Project Number:	224068619-GM-MM
Project title:	PREDICTION OF FRAUDULENT TRANSACTIONS USING MACHINE LEARNING TECHNIQUES
Qualification:	BSc Honours (Applied Statistics)
Student Name:	Manyike G
Student No.:	224068619
Supervisor:	Dr M Mutambayi
Co-supervisor:	N/A
Dept & Faculty	Department of Statistics Faculty of Science and Agriculture

On behalf of the University of Fort Hare's Research Ethics Committee (UREC), I hereby grant ethics approval for 224068619-GM-MM. This approval is valid for 12 months from the date of approval. Renewal of approval must be applied for BEFORE termination of this approval period. Renewal is subject to receipt of a satisfactory progress report. The approval covers the undertakings contained in the above-mentioned project and research instrument(s). The research may commence as from the 15 August 2024, using the reference number indicated above.

Note that should any other instruments be required or amendments become necessary, these require separate authorisation.

Please note that UREC must be informed immediately of

- Any material changes in the conditions or undertakings mentioned in the document;
- Any material breaches of ethical undertakings or events that impact upon the ethical conduct of the research.

The Principal Researcher/Student must report to UREC in the prescribed format, where applicable, annually, and at the end of the project, in respect of ethical compliance.

UREC retains the right to

- Withdraw or amend this approval if
    - Any unethical principal or practices are revealed or suspected;
    - Relevant information has been withheld or misrepresented;
    - Regulatory changes of whatsoever nature so require;
    - The conditions contained in the Certificate have not been adhered to.

- Request access to any information or data at any time during the course or after completion of the project.

Your compliance with Department of Health 2015 guidelines and any other applicable regulatory instruments and with UREC ethics requirements as contained in UREC policies and standard operating procedures, is implied.

UREC wishes you well in your research.

Yours sincerely

[illegible]

Dr Sifiso Mdletshe

Chair: Faculty of Science and Agriculture Research Ethics Committee
15 August 2024