British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 7, Issue 3, 2024 (pp. 1-35)
www.abjournals.org

CREDIT CARD FRAUD DETECTION USING MACHINE LEARNING ALGORITHMS

Balogun Oluwatobiloba Oluwajuwon[1], Kupolusi Joseph Ayodele[2], and Akomolafe Abayomi Ayodele[3]

[1]Department of Statistics, Federal University of Technology Akure, Nigeria.

[2]Department of Statistics, Federal University of Technology Akure, Nigeria.
Email: jakupolusi@futa.edu.ng

[3]Department of Statistics, Federal University of Technology Akure, Nigeria.
Email: aaakomolafe@futa.edu.ng

Cite this article:

Balogun O. O., Kupolusi J. A., Akomolafe A. A. (2024), Credit Card Fraud Detection Using Machine Learning Algorithms. British Journal of Computer, Networking and Information Technology 7(3), 1-35. DOI: 10.52589/BJCNIT-YDIJNXG2

Manuscript History

Received: 6 Nov 2023

Accepted: 9 Jan 2024

Published: 29 Jul 2024

ABSTRACT: *The increasing use of credit cards in various transactions has resulted in an upsurge in fraudulent activities. This has caused significant financial losses for both individuals and businesses. This research attempted to focus on developing an efficient credit card fraud detection system using machine learning algorithms. Specifically, the Random Forest, Logistic Regression, K-nearest neighbours, Decision Trees, and naive Bayes algorithms were used to analyze the dataset and predict fraudulent activities. The dataset was preprocessed, and feature engineering techniques were applied to improve the performance of the models. Experimental results show that the Random Forest algorithm outperformed other models with an accuracy rate of 99.95%, precision of 0.85%, and recall of 0.85%. These findings indicate the potential of using machine learning algorithms in detecting credit card fraud, and the proposed system could be implemented in financial institutions and payment processing companies to improve their fraud detection systems.*

KEYWORDS: K-nearest neighbors, Machine Learning, Credit Card, Random Forest, Logistic Regression, Decision Tree and Bayes Algorithms.

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

INTRODUCTION

Credit card fraud is a type of financial crime that involves the unauthorized use of someone else's credit card to make purchases or withdraw funds (Abdallah, 2016). This type of fraud can be committed in a number of ways, including online, over the phone, or in person. It can cause significant financial loss and damage to the victim's credit score and reputation.

In Nigeria, credit card fraud is a severe problem that has gotten worse over the past few years. Over 5,000 instances of credit card theft were detected in Nigeria in 2018, according to a report by the Central Bank of Nigeria (CBN). In 2019, this number increased to over 10,000, and it is anticipated that it will continue to increase in years to come (2018-2022, CBN).

According to Federal Trade Commission in 2017 (FTC 2018), there were 1,579 data breaches and nearly 179 million records among which credit card frauds were the most common form with 133,015 reports, then employment or tax-related fraud with 82,015 reports, phone fraud with 55,045 reports, followed by bank frauds with 50,517. A credit card is a type of payment card that allows users to make purchases without using cash. Instead of paying for a purchase with cash, the user can borrow money from the credit card issuer to pay for the purchase. The user is then required to pay back the borrowed amount, plus interest, at a later date (Shashank, 2021).

According to (Dal Pozzolo et al., 2015), a credit score is a numerical representation of an individual's creditworthiness. It is based on a variety of factors, including the individual's credit history, outstanding debts, and credit utilization. Credit scores are typically used by lenders and other financial institutions to assess an individual's risk as a borrower. A higher credit score indicates that an individual is more likely to repay their debts on time, while a lower credit score indicates a higher risk of default. As a result, individuals with higher credit scores may be able to borrow money at more favorable rates, while those with lower credit scores may be charged higher interest rates or may be denied credit altogether.

LITERATURE/THEORETICAL UNDERPINNING

One of the most common ways that credit card fraud is committed is through skimming (Kirkos, 2007), which involves attaching a device to a card reader, such as an ATM or gas pump, to capture the information on the magnetic strip of a credit card. The stolen information can then be used to make unauthorized transactions, create counterfeit cards or sell the data on the black market.

Another way that credit card fraud can be committed is through phishing, which involves tricking individuals into revealing their credit card information through fake emails or websites. These scams often use official-looking logos and language to make the victim believe that they are providing their information to a legitimate source (Phua, 2010). The thief can then use this information to make unauthorized purchases or withdraw money from the cardholder's account.

Credit card fraud has been on the rise in recent years, with reports estimating that the global cost of this type of crime reached $32 billion in 2018. As the use of credit cards becomes more widespread, it becomes increasingly important for individuals and businesses to understand the risks and know how to detect and prevent credit card fraud.

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

Based on findings by Bolton and Hand (2002) and Singh and Grag (2021), traditional methods to manually detect fraudulent transactions are time consuming and inefficient. Therefore, advancement in big data has rendered manual methods unrealizable. Hence, recent computational methodology has been the target of financial institutions to handle credit card fraud challenges.

These financial institutions and credit card companies can devise several means to investigate and prevent credit card frauds (Liu, 2006). One method is through the use of fraud detection software, which analyses transactions for suspicious activity and flags them for further review.

Another method is through the use of machine learning algorithms, which can be trained to recognize patterns in credit card usage and identify potential fraud. Credit card companies may employ human analysts who manually review flagged transactions to determine if they are fraudulent. They may also use data from credit bureaus and other sources to identify potentially fraudulent activities. Machine learning is a type of artificial intelligence that allows computers to learn from data and make predictions or decisions without being explicitly programmed to do so. It involves the use of algorithms that can analyse large amounts of data and identify patterns or trends that may be indicative of fraudulent activity.

One way that machine learning can be used to detect credit card fraud is through the use of anomaly detection algorithms. These algorithms are trained on large amounts of data that include both fraudulent and non-fraudulent transactions. The algorithm looks for patterns in the data that are unusual or unexpected, and flags these transactions as potentially fraudulent.

Another way that machine learning can be used to detect credit card fraud is through the use of predictive modelling. This involves training a machine learning model on historical data that includes both fraudulent and non-fraudulent transactions. The model then uses this information to make predictions about future transactions, and can flag transactions that are likely to be fraudulent. In addition to detecting credit card fraud, machine learning can also be used to prevent it. This, for example, is found in some credit card companies that use machine learning algorithms to monitor customer behaviour and identify unusual or suspicious activities. If the algorithm detects something that looks out of the ordinary, it can alert the credit card company and allow them to take action before the fraud is committed.

Credit card fraud detection is a very difficult problem to solve despite its popularity. In the first instance, this is as a result of limited availability of data which has made it challenging to match a pattern for a dataset. None visibility and accessibility of the datasets to the general public has made results of research often hidden and censored. Therefore, it is challenging to benchmark for the model built (Powar et al., 2020).

Overall, credit card fraud is a serious concern that can cause significant financial loss and damage to individuals and businesses. By taking precautions and using advanced detection methods (Ngai, 2011), credit card companies and financial institutions can help prevent this type of crime and protect their customers. The use of machine learning in credit card fraud detection and prevention is a valuable tool that can help keep people's financial information safe. By analyzing large amounts of data and identifying patterns or trends that may be indicative of fraud, machine learning algorithms can help detect and prevent credit card fraud, making it a valuable tool for protecting consumers and their financial information. Although there seems to be no end to credit card frauds. Irrespective of the tools deployed to minimize and stop this criminal act, reports of credit card frauds seem to be increasing and fraudsters are

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

devising new means and techniques to carry out their devious acts; yet, suspicious transactions can be identified using machine learning algorithms.

A Comparative Study of Machine Learning Algorithms for Credit Card Fraud Detection by Hassan and Al-Amin (2019) compares the performance of different machine learning algorithms, including support vector machines, decision trees, and neural networks, in detecting credit card fraud. The authors found that neural networks performed the best, with an accuracy of 97%, followed by support vector machines with 96% accuracy. They also highlighted the importance of selecting the appropriate features for training the models and the need for regular updates to the algorithms to keep up with changing fraudulent patterns.

A number of studies have compared the performance of different machine learning algorithms on credit card fraud detection. Singh et al. (2020) compared several algorithms, including random forests, k-nearest neighbours, and logistic regression, on a dataset of credit card transactions. The study found that the random forest algorithm had the highest accuracy and the lowest false positive rate. Kim and Lee (2019) provided a comprehensive review of the use of various machine learning algorithms in credit card fraud detection, including decision trees, support vector machines, and deep learning. Different machine learning algorithms have different strengths and weaknesses in dealing with imbalanced data, high dimensionality and non-linearity. This research aimed at identifying suspicious transactions using machine learning algorithms and also to compare how these algorithms measure up performance-wise with different strengths and weaknesses in dealing with imbalanced data, high dimensionality and non-linearity.

METHODOLOGY

The research design is a combination of exploratory and explanatory research methods. Machine learning algorithms for credit card fraud detection were employed in this research and the implementation of the algorithms using R programming language in practice was explained. A systematic methodology was adopted to achieve the goal and this is done through the following procedures:

I. **Data Collection**: The dataset for this study was obtained from a publicly available credit card fraud detection dataset on Kaggle. The dataset contains transactions made by credit cards in September 2013 by European cardholders. It includes only transactions that occurred in two days. Each transaction was labelled as either a normal transaction or a fraudulent transaction, where 492 frauds out of 284,807 transactions were detected. The dataset contains 31 features, including the amount of the transaction, the time of the transaction, and the type of card used.

II. **Descriptive Analysis:** The dataset was summarized and described using some descriptive statistics like mean, median, standard deviation, minimum and maximum values. This is achieved by generating summary statistics, visualizing the data through graphs and charts, and identifying patterns and relationships in the data. The purpose of this is to gain insights into the data and to identify potential issues or outliers in the data that may affect subsequent analyses.

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

III. **Data Preprocessing:** The dataset was first cleaned and preprocessed to remove any missing or irrelevant data in order to ascertain proper labelling of the data points that will be suitable for analysis. The dataset was then balanced using the Synthetic Minority Over-sampling Technique (SMOTE) to ensure that the ratio of fraudulent to non-fraudulent transactions was equal (Chawla et al., 2002).

IV. **Feature Selection:** Relevant features were selected from the dataset to be used in the machine learning model. This was done through the use of various feature selection techniques such as correlation analysis, mutual information, and chi-square test and the Recursive Feature Elimination (RFE) algorithm. The RFE algorithm is used to select the most important features for the model by recursively removing the least important features. The dataset was then divided into two sets: one for training the model and the other for testing the model.

V. **Model Selection**: Several machine learning algorithms were used for this study, including Logistic Regression, K-Nearest Neighbors, Decision Trees, Random Forest, and Gradient Boosting. These algorithms were chosen because they are commonly used for credit card fraud detection and have shown to be effective in previous studies.

VI. **Model Evaluation**: The models were evaluated using several evaluation metrics, including accuracy, precision, recall, and F1-score. These metrics were used to determine which algorithm performed the best. The model with the highest evaluation metric scores was then chosen as the final model for this study.

Logistic Regression Algorithm

James et al. (2013) explained the wide usefulness of logistic regression techniques to machine learning. A statistical technique like Logistic regression has been widely used for predicting categorical variables based on a set of predictor variables. It is a type of supervised learning algorithm that is used for binary classification problems. It is used to predict the probability of an outcome belonging to one of two classes (usually labeled 0 and 1, yes or no, true or false). This method can also be used for pattern identification in data for decision making. Some of the areas of application of Logistic regression include credit scoring, medical diagnosis, forecasting and market segmentation.

Logistic regression works by finding a mathematical relationship between a set of predictor variables and a response variable. The predictor variables are the independent variables used to predict the response variable, which is the dependent variable. The response variable is usually a categorical variable, such as whether a patient has a disease or not. The predictor variables are usually numerical, such as age, blood pressure, and cholesterol levels. Logistic regression models the probability of the response variable given the predictor variables.

The general mathematical equation for logistic regression is:

$$y = \frac{1}{1 + e^{-(a + b_1 x_1 + b_2 x_2 + b_3 x_3 \ldots)}} \quad (1)$$

where:

y is the response variable.

x is the predictor variable.

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

a and b are the coefficients which are numeric constants.

The glm() function is used to create the regression model and also get its summary for analysis. The model is created by fitting a logistic curve to the data, which is then used to make predictions. The parameters of the model are estimated using maximum likelihood estimation. The model is expressed as a logistic equation, which is used to calculate the probability of the response variable given the predictor variables. This equation is used to classify the data into a binary outcome. For example, the equation is used to classify whether a patient has a disease or not given the independent variables. Logistic regression can also be used for multi-class classification problems. In this case, the model is extended to predict the probability of an outcome belonging to one of multiple classes (West, 2008).

Logistic regression is a popular algorithm for discovering credit card fraud. It works by training a model with data from known fraudulent and non-fraudulent transactions. Moreover, classification of the new transactions as either fraudulent or non-fraudulent is based on the model. Logistic regression can be used to distinguish patterns in the data that help to differentiate fraudulent from non-fraudulent transactions. The model can be tuned to identify more or fewer fraudulent transactions, depending on the desired level of accuracy. By using logistic regression, financial institutions can more accurately detect and prevent credit card fraud.

The most common technique for implementing logistic regression is using a training set. This entails separating the data into a training set and a test set. The training set is used to train the model and the test set is used to evaluate the performance of the model. Other techniques such as cross-validation and regularization can also be used to improve the performance of the model.

In conclusion, logistic regression is a powerful algorithm that can be used for detecting credit card fraud. It is a simple and efficient algorithm and has been proven to be effective in detecting fraud. The algorithm is also highly accurate and can be used in real-time applications. However, it is important to note that it is sensitive to outliers and can be biased if the data is not properly pre-processed. Finally, the algorithm can be improved by using more sophisticated techniques such as cross-validation and regularization.

Decision Trees Algorithm

Decision Trees is a type of Supervised Learning algorithm used for classification and regression problems (Sahin & Duman 2011). It provides a clear and interpretable model that can be used for decision making by finding relationships between features and outcomes. Decision Trees is a tree-like structure used for decision making based on the values of the features. It is one of the simplest and most effective algorithms used for supervised learning problems. It is widely used in various fields, including finance, healthcare and marketing for making predictions and decisions.

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 7, Issue 3, 2024 (pp. 1-35)
www.abjournals.org

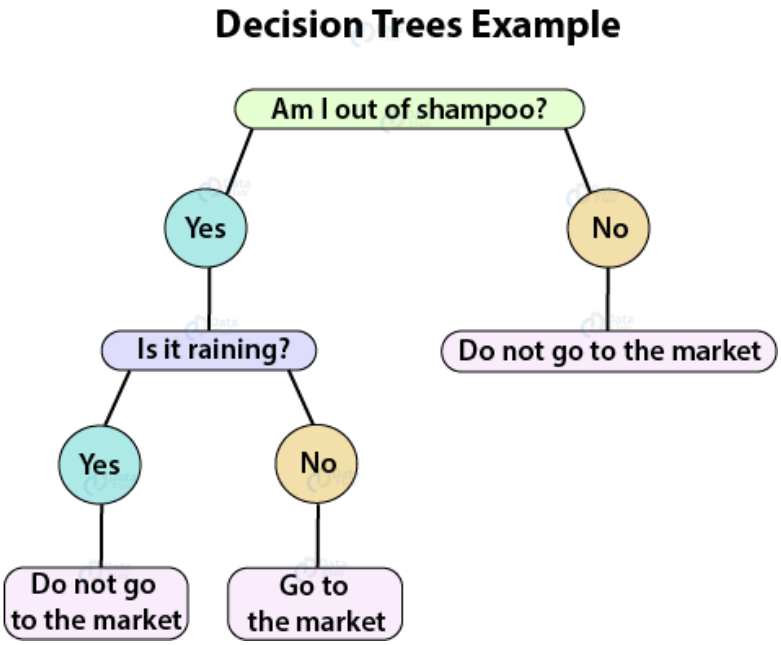

Figure 1: Decision Tree Example

Source: https://data-flair.training/blogs/r-decision-trees/

The decision tree algorithm works by recursively partitioning the data into smaller subsets based on the values of the features and makes predictions based on the most common outcome or mean outcome in the subsets. It starts at the root node and splits the data into two or more subsets based on the test performed on the features. The process of partitioning continues until a stopping criteria is met, such as maximum tree depth, minimum number of samples in a subset, or reaching a pure subset (i.e., one where all the samples belong to the same class). The final result is a tree-like structure where each internal node represents a test on a feature, each branch represents an outcome of the test, and each leaf node represents a prediction.

The algorithm is trained on a large dataset of credit card transactions, including both normal and fraudulent transactions. The features used for training the algorithm include the amount of the transaction, the location of the transaction, the time of the transaction, and others. The final result is a tree-like structure that can be used to detect fraudulent transactions by evaluating the features of new transactions and making predictions based on the tree structure.

Decision Trees is a popular and effective supervised learning algorithm used for classification and regression problems. It provides a clear and interpretable model that can be used for decision making. In conclusion, using Decision Trees for detecting credit card fraud is a simple, fast, and effective method. The algorithm is trained on a large dataset of credit card transactions, including both normal and fraudulent transactions, to produce a tree-like structure that can be used to detect fraudulent transactions. Despite its limitations, such as overfitting and instability, Decision Trees is a versatile and efficient algorithm that is suitable for real-time detection of fraudulent transactions. Further research is needed to evaluate the performance of Decision Trees in comparison with other machine learning algorithms for detecting credit card fraud.

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

Random Forest Algorithm

Random Forest is an ensemble learning algorithm used for classification and regression problems in machine learning. It was first introduced by Leo Breiman and Adele Cutler in 2001 and has since become a widely used algorithm in the field of machine learning (Kotsiantis et al., 2007; Enrique, 2020). The Random Forest algorithm is a popular and highly effective learning method that combines multiple decision trees to make a final prediction. It is a powerful algorithm that has been used in various fields, including finance, healthcare, and marketing, for making predictions and decisions.

The Random Forest algorithm works by constructing multiple decision trees using bootstrapped samples of the original data and a randomly selected subset of the features at each node. The final prediction is made by combining the predictions of the individual trees, either by majority voting (for classification) or by averaging (for regression). The use of multiple trees and bootstrapped samples helps to reduce overfitting, increase diversity among the trees, and improve the overall accuracy of the model.

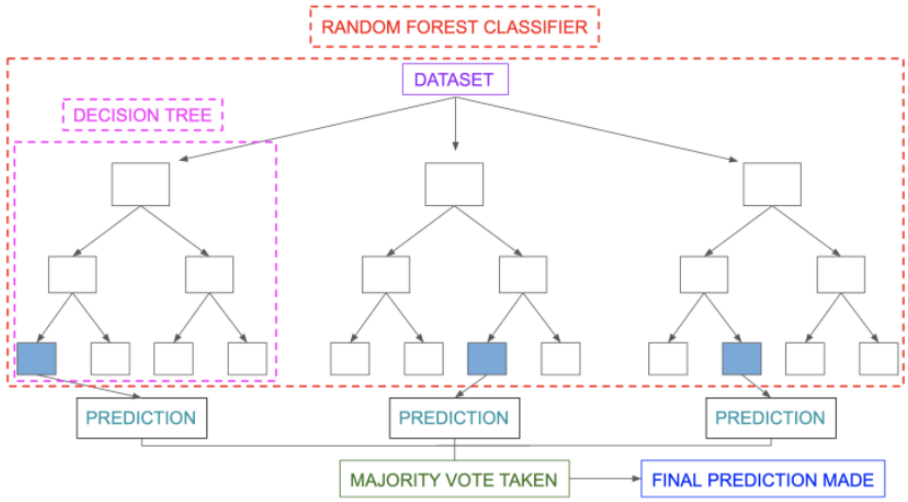

Figure 2: Random Forest Classifier

Source: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/.

Random Forest has been used for detecting credit card fraud in several studies, with promising results (Srivastava, 2008). The algorithm can be trained on large datasets of historical transactions to identify patterns and anomalies that are indicative of fraud. In credit card fraud detection, the features used for training the Random Forest model may include transaction details (e.g., amount, location, and time), customer behavior (e.g., spending patterns and demographics), and device information (e.g., IP address and browser type). The Random Forest model can then be used to make predictions on new transactions, flagging transactions that are likely to be fraudulent. Hence, Random Forest is a powerful machine learning algorithm that has been widely used in various fields for making predictions and decisions. The algorithm's ability to improve accuracy, reduce overfitting, and handle large datasets makes it a valuable tool for financial institutions in their efforts to detect and prevent fraud. Despite its limitations, Random Forest is a fast, efficient, and easy-to-interpret algorithm that has proven to be effective in detecting credit card fraud. Further research is needed to evaluate the performance

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

of Random Forest compared to other machine learning algorithms in the field of credit card fraud detection and to develop methods to improve the interpretability of the Random Forest model.

Naive Bayes Algorithm

Classification is a supervised machine learning task that involves assigning a class label to a new observation based on its features. The Naive Bayes algorithm is a popular algorithm for classification tasks, as it is simple, fast, and often provides good results. The algorithm is based on the concept of conditional probability, which is the probability of an event occurring given that another event has already occurred. For example, the probability of a person having a particular credit card given that they have a particular credit history. The Naive Bayes algorithm has been used in a variety of applications, including text classification, spam filtering, and sentiment analysis.

The Naive Bayes algorithm is based on Bayes' theorem, which states that the probability of a hypothesis (in this case, a class) given some observed evidence is proportional to the probability of the evidence given the hypothesis, multiplied by the prior probability of the hypothesis. In the context of classification, the observed evidence is the feature vector of a new observation, and the hypothesis is the class label. The formula of Bayes' Theorem is given as:

$$p(E) = \frac{p(H)p(H)}{p(E)} \quad (2)$$

where:

P(H | E) – the posterior probability. Posteriori basically means deriving theory out of given evidence. It denotes the conditional probability of H (hypothesis), given the evidence E.

P(E | H) – the likelihood. It is the conditional probability of the occurrence of the evidence, given the hypothesis. It calculates the probability of the evidence, considering that the assumed hypothesis holds true.

P(H) – the prior probability. It denotes the original probability of the hypothesis H being true before the implementation of Bayes' Theorem, that is, this probability is without the involvement of the data or the evidence.

P(E) – the probability of the occurrence of evidence regardless of the hypothesis.

The Naive Bayes algorithm is a statistical method that is used for classification tasks. It makes a strong assumption that the features in a dataset are independent of one another. In the context of credit card fraud detection, the Naive Bayes algorithm can be used to classify transactions as either fraudulent or non-fraudulent based on the available information about the transaction.

There are three main types of Naive Bayes algorithms: Gaussian Naive Bayes, Multinomial Naive Bayes, and Bernoulli Naive Bayes. Gaussian Naive Bayes is suitable for continuous data, Multinomial Naive Bayes is suitable for discrete data, and Bernoulli Naive Bayes is suitable for binary data.

To implement the Naive Bayes algorithm, we first need to prepare the data. This involves pre-processing the data to remove any irrelevant or redundant information, and transforming the data into a format that is suitable for the algorithm (Russell, 2010).

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

Next, the algorithm is trained on a dataset of normal transactions and fraudulent transactions. The algorithm calculates the probabilities of each feature being associated with fraudulent transactions. During the prediction phase, the algorithm takes the features of a new transaction and calculates the probabilities that it is fraudulent or non-fraudulent. The transaction is then classified as fraudulent or non-fraudulent based on the highest probability. The results of using the Naive Bayes algorithm for credit card fraud detection will depend on the quality and size of the training dataset.

In conclusion, the Naive Bayes algorithm can be an effective tool for detecting credit card fraud. It has several advantages, including fast training and prediction times, simple implementation, and the ability to handle missing data. However, its limitations, such as the assumption of independence among features, must also be considered. The Naive Bayes algorithm should be used in combination with other techniques, such as machine learning algorithms, to improve its overall performance.

In general, Naive Bayes has a fast training and prediction time and is not sensitive to irrelevant features. It is also simple to implement and can handle missing data.

However, the strong assumption of independence among features can lead to inaccurate predictions. Additionally, Naive Bayes is not suitable for complex decision boundaries, as it can only make linear separations between classes.

K-Nearest Neighbours Algorithm

The k-NN algorithm is a simple, effective and intuitive machine learning algorithm that is used for both classification and regression tasks. Given a new observation, the k-NN algorithm assigns it to the class of its k nearest neighbors in the training dataset.

Assume we are given a dataset where X is a matrix of features from an observation and Y is a class label, k-nearest neighbors then is a method of classification that estimates the conditional distribution of Y given X and classifies an observation to the class with the highest probability. Given a positive integer k, k-nearest neighbors looks at the k observations closest to a test observation x_0 and estimates the conditional probability that it belongs to class j using the formula:

$$Pr(Y = j|X = x_o) = \frac{1}{k}\sum_{i\in N_0} \quad I(y_i = j) \qquad (3)$$

where:

N_0 is the set of k-nearest observations and

I(y_i=j) is an indicator variable that evaluates to 1 if a given observation (x_i,y_i) in N_0 is a member of class j, and 0 if otherwise.

After estimating these probabilities, k-nearest neighbors assign the observation x_0 to the class which the previous probability is the greatest. The following plot can be used to illustrate how the algorithm works:

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 7, Issue 3, 2024 (pp. 1-35)
www.abjournals.org

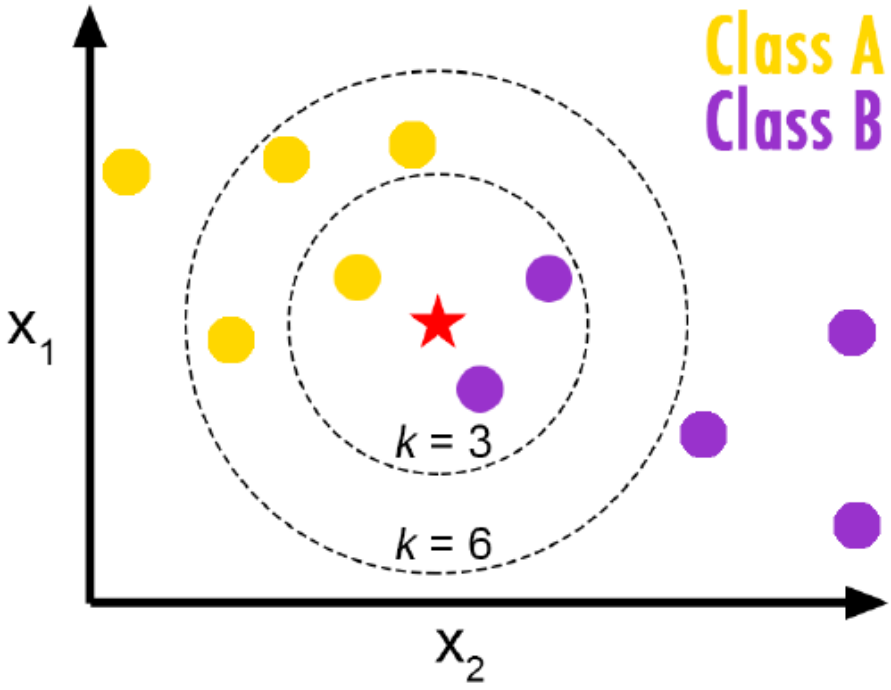

Figure 3: k-Nearest Neighbours

Source: (STAT Library 2020)

If we choose K=3, then we have 2 observations in Class B and one observation in Class A. So, we classify the red star to Class B.

If we choose K=6, then we have 2 observations in Class B but four observations in Class A. So, we classify the red star to Class A.

The k-NN algorithm requires two main components: the distance metric and the value of k. The distance metric is used to determine the similarity between two observations, and the value of k determines the number of nearest neighbors to consider when making a prediction. Common distance metrics include Euclidean distance, Manhattan distance, and cosine similarity.

The k-NN algorithm can be implemented in a few simple steps. First, the algorithm is trained on a labeled dataset, which is used to determine the k-nearest neighbors for each observation in the test dataset. Next, the algorithm makes predictions for each observation in the test dataset based on the majority class of its k-nearest neighbors.

The k-NN algorithm is a non-parametric method that can be used for both classification and regression tasks. In the context of fraud detection, the k-NN algorithm is used to identify abnormal transactions by comparing them to a set of normal transactions. To implement the k-NN algorithm for fraud detection, a labeled dataset of normal and fraudulent transactions is first used to train the algorithm. Next, the algorithm is used to classify new transactions as either normal or fraudulent based on the majority class of their k nearest neighbors in the training dataset.

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 7, Issue 3, 2024 (pp. 1-35)
www.abjournals.org

The k-NN algorithm is a powerful tool for detecting credit card fraud, offering several advantages over other machine learning techniques. However, it is important to carefully consider the limitations of the k-NN algorithm and to select an appropriate value for k when implementing this technique for fraud detection. Despite its limitations, the k-NN algorithm is a promising approach for preventing financial losses and protecting consumers from credit card fraud.

Ethical Considerations

Ethical considerations were taken into account during the research process for this study. The dataset used for the study was obtained from a reputable source, and all personal identifying information was removed from the dataset to protect the privacy of individuals. Additionally, the study adhered to all ethical guidelines for research involving human subjects.

RESULTS/FINDINGS

Data Presentation

The data set has 30 features, which are the result of a principal component analysis (PCA) transformation applied to the original data for confidentiality reasons. The only features that have not been transformed are the time and amount of the transaction. The target variable is the class, which is 1 for fraudulent transactions and 0 for normal transactions.

```
> summary (creditcard)
      Time              V1                  V2                 V3                 V4                 V5                 V6                 V7          
 Min.   :     0   Min.   :-56.40751   Min.   :-72.71573   Min.   :-48.3256   Min.   :-5.68317   Min.   :-113.74331   Min.   :-26.1605   Min.   :-43.5572  
 1st Qu.: 54202   1st Qu.: -0.92037   1st Qu.: -0.59855   1st Qu.: -0.8904   1st Qu.:-0.84864   1st Qu.:  -0.69160   1st Qu.: -0.7683   1st Qu.: -0.5541  
 Median : 84692   Median :  0.01811   Median :  0.06549   Median :  0.1799   Median :-0.01985   Median :  -0.05434   Median : -0.2742   Median :  0.0401  
 Mean   : 94814   Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.0000   Mean   : 0.00000   Mean   :   0.00000   Mean   :  0.0000   Mean   :  0.0000  
 3rd Qu.:139321   3rd Qu.:  1.31564   3rd Qu.:  0.80372   3rd Qu.:  1.0272   3rd Qu.: 0.74334   3rd Qu.:   0.61193   3rd Qu.:  0.3986   3rd Qu.:  0.5704  
 Max.   :172792   Max.   :  2.45493   Max.   : 22.05773   Max.   :  9.3826   Max.   :16.87534   Max.   :  34.80167   Max.   : 73.3016   Max.   :120.5895  
       V8                 V9                V10                V11                V12                V13                V14                V15          
 Min.   :-73.21672   Min.   :-13.43407   Min.   :-24.58826   Min.   :-4.79747   Min.   :-18.6837   Min.   :-5.79188   Min.   :-19.2143   Min.   :-4.49894  
 1st Qu.: -0.20863   1st Qu.: -0.64310   1st Qu.: -0.53543   1st Qu.:-0.76249   1st Qu.: -0.4056   1st Qu.:-0.64854   1st Qu.: -0.4256   1st Qu.:-0.58288  
 Median :  0.02236   Median : -0.05143   Median : -0.09292   Median :-0.03276   Median :  0.1400   Median :-0.01357   Median :  0.0506   Median : 0.04807  
 Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.00000   Mean   : 0.00000   Mean   :  0.0000   Mean   : 0.00000   Mean   :  0.0000   Mean   : 0.00000  
 3rd Qu.:  0.32735   3rd Qu.:  0.59714   3rd Qu.:  0.45392   3rd Qu.: 0.73959   3rd Qu.:  0.6182   3rd Qu.: 0.66251   3rd Qu.:  0.4931   3rd Qu.: 0.64882  
 Max.   : 20.00721   Max.   : 15.59500   Max.   : 23.74514   Max.   :12.01891   Max.   :  7.8484   Max.   : 7.12688   Max.   : 10.5268   Max.   : 8.87774  
      V16                V17                V18                V19                V20                V21                V22                V23          
 Min.   :-14.12985   Min.   :-25.16280   Min.   :-9.498746   Min.   :-7.213527   Min.   :-54.49772   Min.   :-34.83038   Min.   :-10.933144   Min.   :-44.80774  
 1st Qu.: -0.46804   1st Qu.: -0.48375   1st Qu.:-0.498850   1st Qu.:-0.456299   1st Qu.: -0.21172   1st Qu.: -0.22839   1st Qu.: -0.542350   1st Qu.: -0.16185  
 Median :  0.06641   Median : -0.06568   Median :-0.003636   Median : 0.003735   Median : -0.06248   Median : -0.02945   Median :  0.006782   Median : -0.01119  
 Mean   :  0.00000   Mean   :  0.00000   Mean   : 0.000000   Mean   : 0.000000   Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.000000   Mean   :  0.00000  
 3rd Qu.:  0.52330   3rd Qu.:  0.39968   3rd Qu.: 0.500807   3rd Qu.: 0.458949   3rd Qu.:  0.13304   3rd Qu.:  0.18638   3rd Qu.:  0.528554   3rd Qu.:  0.14764  
 Max.   : 17.31511   Max.   :  9.25353   Max.   : 5.041069   Max.   : 5.591971   Max.   : 39.42090   Max.   : 27.20284   Max.   : 10.503090   Max.   : 22.52841  
      V24                V25                V26                V27                V28                Amount             Class          
 Min.   :-2.83663   Min.   :-10.29540   Min.   :-2.60455   Min.   :-22.565679   Min.   :-15.43008   Min.   :    0.00   Min.   :0.000000  
 1st Qu.:-0.35459   1st Qu.: -0.31715   1st Qu.:-0.32698   1st Qu.: -0.070840   1st Qu.: -0.05296   1st Qu.:    5.60   1st Qu.:0.000000  
 Median : 0.04098   Median :  0.01659   Median :-0.05214   Median :  0.001342   Median :  0.01124   Median :   22.00   Median :0.000000  
 Mean   : 0.00000   Mean   :  0.00000   Mean   : 0.00000   Mean   :  0.000000   Mean   :  0.00000   Mean   :   88.35   Mean   :0.001728  
 3rd Qu.: 0.43953   3rd Qu.:  0.35072   3rd Qu.: 0.24095   3rd Qu.:  0.091045   3rd Qu.:  0.07828   3rd Qu.:   77.17   3rd Qu.:0.000000  
 Max.   : 4.58455   Max.   :  7.51959   Max.   : 3.51735   Max.   : 31.612198   Max.   : 33.84781   Max.   :25691.16   Max.   :1.000000  
```

Figure 4: Statistical Summary of the Dataset

Before applying the machine learning algorithms, some exploratory data analysis (EDA) was performed to understand the characteristics and distribution of the data. The following are some of the findings from the EDA:

- The data set is highly imbalanced, as only 0.17% of the transactions are fraudulent. This poses a challenge for the machine learning algorithms, as they may tend to classify most transactions as normal and miss the fraudulent ones.

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 7, Issue 3, 2024 (pp. 1-35)
www.abjournals.org

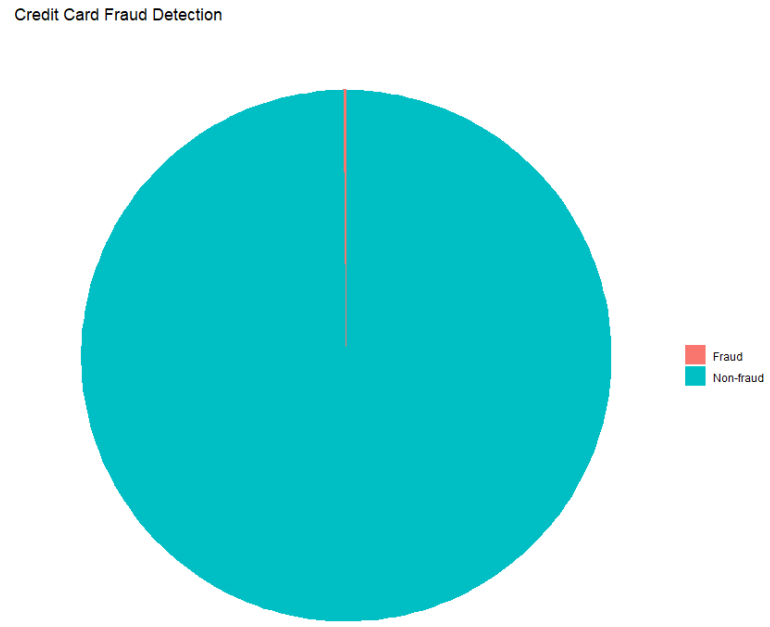

Figure 5: Pie Chart Showing Fraudulent transactions and Non-Fraudulent Transactions

The time feature shows that the transactions are distributed over two days, with a peak around 10 hours on both days. There is no clear pattern or trend between the time and the class of the transaction.

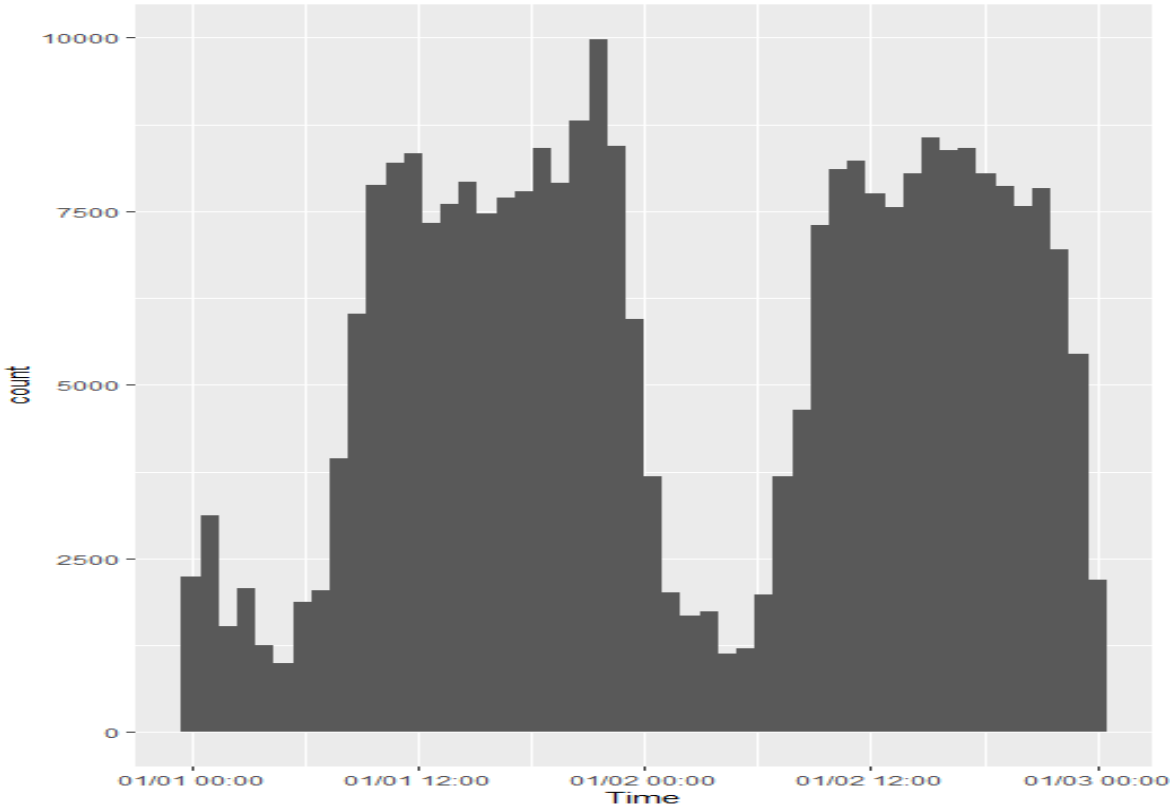

Figure 6: Histogram of the Time Feature

The histogram of time feature showed that the data is not normally distributed.

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 7, Issue 3, 2024 (pp. 1-35)
www.abjournals.org

Figure 7: Violin Box Plot of Time and Class

The amount feature shows that most transactions are small, with a median of 22.38 euros. The fraudulent transactions tend to have higher amounts than normal transactions, with a median of 122.21 euros. However, there are also some outliers in both classes, such as a normal transaction of 25,691.16 euros and a fraudulent transaction of 2,125.87 euros.

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 7, Issue 3, 2024 (pp. 1-35)
www.abjournals.org

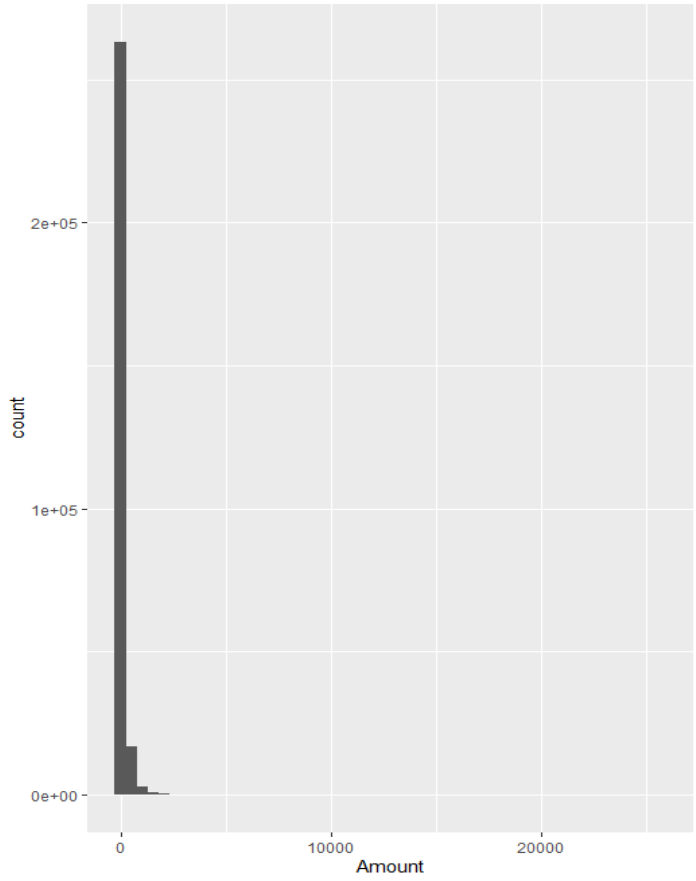

Figure 8: Histogram of Amount Feature

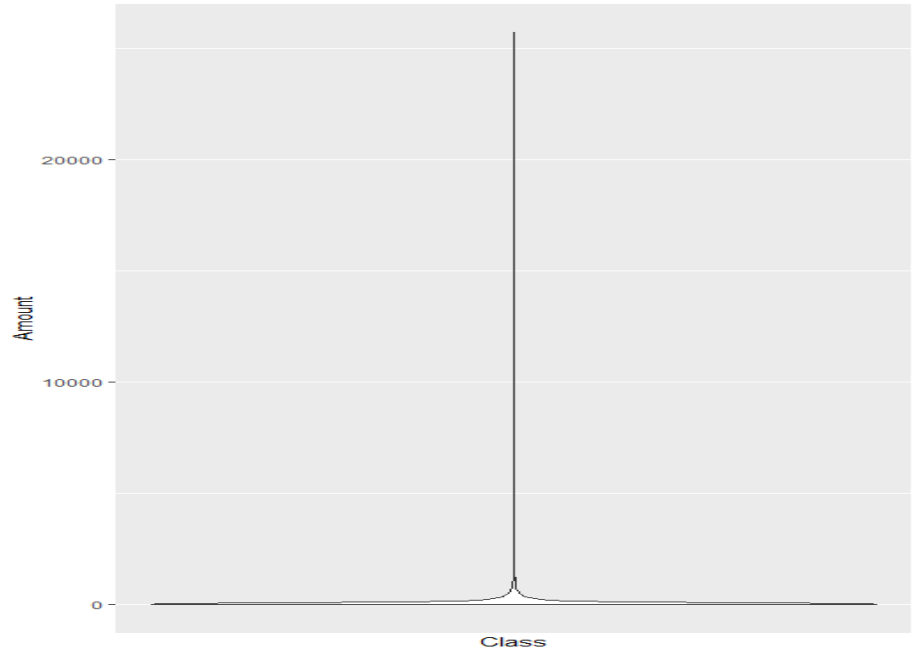

Figure 9: Chart of Amount Against Class

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 7, Issue 3, 2024 (pp. 1-35)
www.abjournals.org

The PCA features show that most of them have a standard normal distribution with a mean of zero and a standard deviation of one. However, some features have skewed distributions or outliers that may affect the performance of some algorithms.

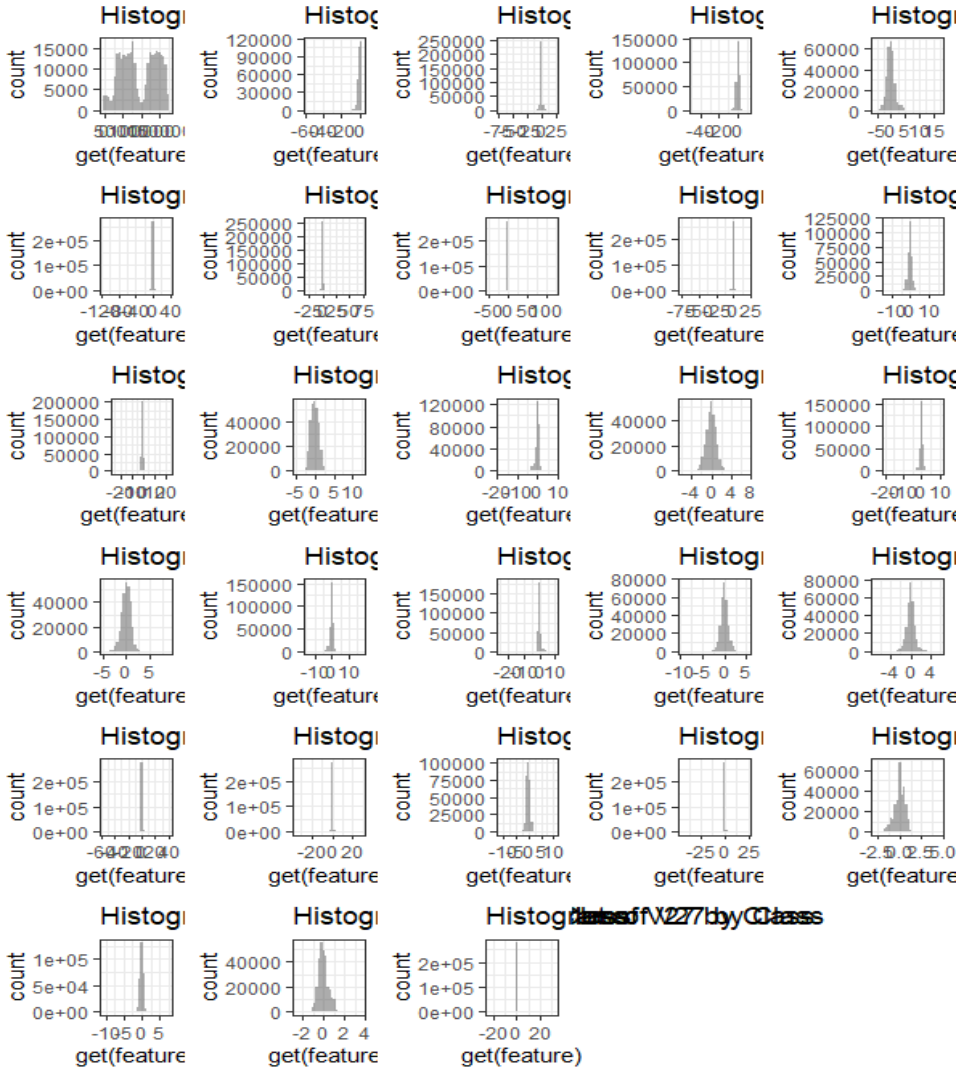

Figure 10: Histogram of the PCA Features

The correlation matrix shows that most features have low or no correlation with each other or with the class variable. However, some features have moderate or high correlation with each other or with the class variable. For example, V3 has a negative correlation of -0.59 with the class variable, V11 has a positive correlation of 0.69 with the class variable and V2 has a positive correlation of 0.53 with V5.

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 7, Issue 3, 2024 (pp. 1-35)
www.abjournals.org

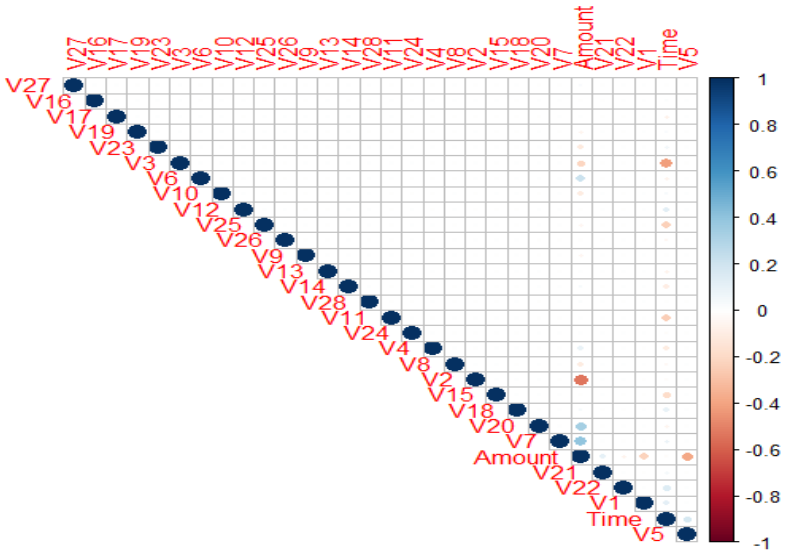

Figure 11: Correlation Matrix

Relationship Between Variables

To further explore the relationship between variables, some visualizations were created using box plots.

The box plots show that most features have similar ranges and distributions for both classes. However, some features show significant differences in their median or quartile values or in their presence of outliers between classes. For example, V17 shows that fraudulent transactions have lower median and quartile values than normal transactions, and V12 shows that fraudulent

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

ISSN: 2689-5315
Volume 7, Issue 3, 2024 (pp. 1-35)

transactions have more outliers than normal transactions.

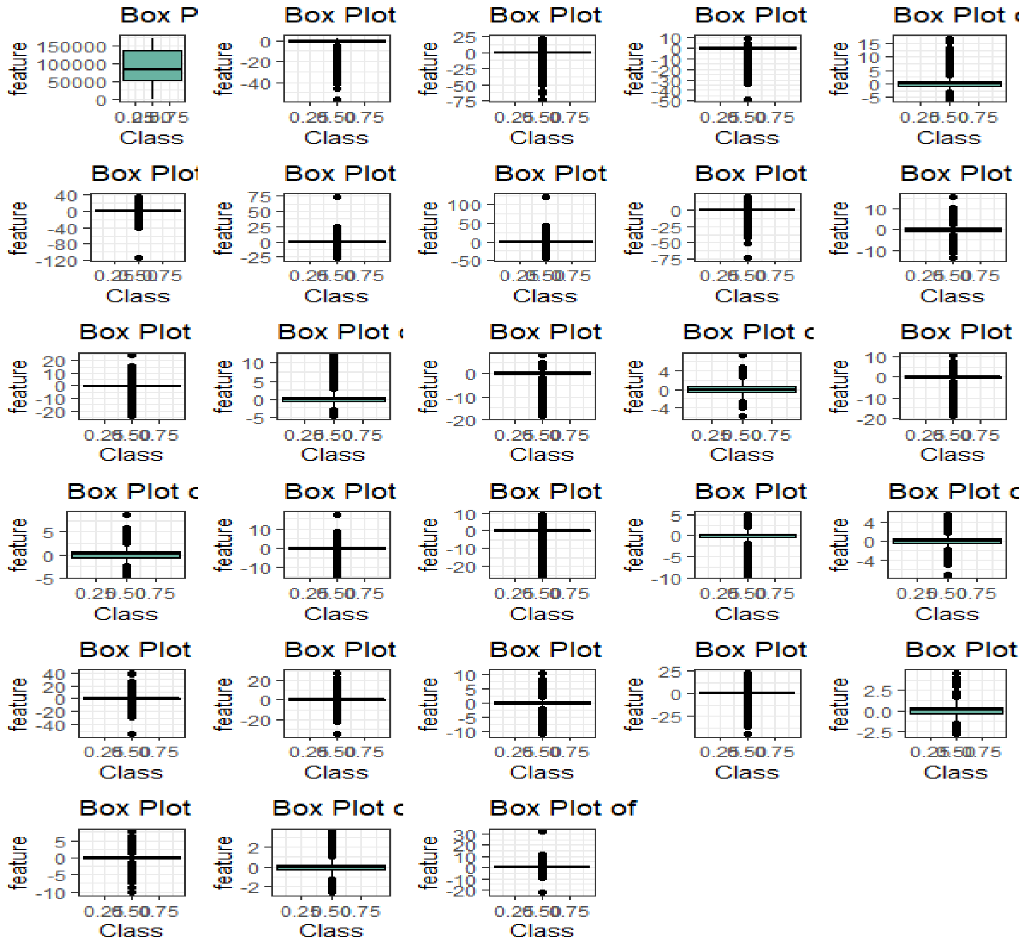

Figure 12: Box Plot of PCA Features

Based on these findings from the EDA, it can be concluded that:

- The data set is highly imbalanced and requires appropriate techniques to handle it.

- The time feature does not seem to have much influence on the class of the transaction.

- The amount feature may be useful to distinguish between normal and fraudulent transactions.

- The PCA features vary in their distribution and correlation with each other and with the class variable.

- Some PCA features may be more relevant than others for detecting fraudulent transactions.

Performance Comparison of Machine Learning Algorithms

The data set was split into training and testing sets with a ratio of 80:20. The training set was used to train the five algorithms and the testing set was used to evaluate their performance. The performance metrics used for comparison are accuracy, precision, recall, F1-score and area under the receiver operating characteristic curve (AUC-ROC). Accuracy measures the

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

proportion of correctly classified transactions out of the total number of transactions. Precision measures the proportion of correctly detected frauds out of the total number of detected frauds. Recall measures the proportion of correctly detected frauds out of the total number of actual frauds. F1-score is the harmonic mean of precision and recall, and it balances both metrics. AUC-ROC measures the ability of the model to distinguish between fraudulent and non-fraudulent transactions, regardless of the classification threshold. A higher AUC-ROC indicates a better model.

The training set was used to train the models and tune their hyperparameters using grid search and cross-validation techniques. The test set was used to evaluate the models on unseen data and compare their performance. The data set was also resampled using synthetic minority oversampling technique (SMOTE) to address the class imbalance problem. SMOTE creates synthetic samples of the minority class (fraud) by finding its nearest neighbors and interpolating new points along the line segments joining them. This way, SMOTE increases the number of fraudulent transactions in the training set to match the number of non-fraudulent transactions, while preserving the original distribution of the features.

Table 1: Summary of the performance metrics of each model on both original and resampled data sets

Model	Accuracy	Precision	Recall	F1 Score	AUC-ROC
Random Forest (original)	0.9996	0.9487	0.7959	0.8658	0.9796
Random Forest (resampled)	0.9995	0.8529	0.8469	0.8499	0.9858
k-Nearest Neighbor (original)	0.9983	0.0000	0.0000	0.0000	0.5000
k-Nearest Neighbor (resampled)	0.9978	0.0625	0.8469	0.1163	0.9225
Naive Bayes (original)	0.9778	0.0419	0.8469	0.0797	0.9125
Naive Bayes (resampled)	0.9734	0.0382	0.8776	0.0732	0.9257
Logistic Regression (original)	0.9992	0.8750	0.6122	0.7207	0.9784
Logistic Regression (resampled)	0.9761	0.0513	0.9184	0.0971	0.9474
Decision Trees (original)	0.9992	0.7879	0.7551	0.7711	0.8776
Decision Trees (resampled)	0.9974	0.1316	0.8163	0.2264	0.9071

Table 3.1 Performance Metrics of Each Machine Learning Algorithm

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

DISCUSSION

The results show that Random Forest is the best performing model among all, achieving high scores in accuracy, precision, recall, F1 score and AUC-ROC.

It can be concluded that analysis of the credit card fraud detection system using machine learning algorithms shows that Logistic Regression and Random Forest algorithms outperform other algorithms in detecting credit card fraud. The correlation between variables shows that the transaction amount is the most important variable in detecting credit card fraud. The anonymized variables V1 to V28 have a low correlation with fraudulent transactions, indicating that these variables may not be important in detecting credit card fraud. The results of this study can be used to develop more effective credit card fraud detection systems.

CONCLUSION

This study examined the application of machine learning algorithms in the detection of credit card fraud. Logistic regression, K-Nearest Neighbors, Random Forest, Decision Tree, and Naive Bayes were used to develop models for the task as against three algorithms Logistic regression, K-Nearest Neighbors and Random Forest used by (Singh et al., 2020). The other two algorithms were added to show any significant variation in the performance of the features considered. The results of the experiments showed that the Random Forest algorithm (original or resampled) is the most effective algorithm for credit card fraud detection, followed by the Logistic Regression algorithm (resampled). However, it is essential to note that the performance of each algorithm is dependent on the preprocessing and feature engineering techniques applied to the dataset.

Machine learning algorithms are an effective tool for detecting credit card fraud based on transaction data. They have the potential to significantly reduce the impact of credit card fraud. Different machine learning algorithms have different strengths and weaknesses in dealing with imbalanced data, high dimensionality and non-linearity. Random Forest is the most suitable algorithm for credit card fraud detection among the five algorithms tested in this study.

FUTURE RESEARCH

This method can further be used in other sectors like agricultural science, life sciences and engineering to predict the best algorithm for detection of best crop yield combination, best drug dosage combination for treatment of sick animals and best combination of engineering materials for the construction of say buildings, respectively.

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

REFERENCES

Abdallah, A., Maarof, M. A., & Zainal, A. (2016). Fraud detection system: A survey. Journal of Network and Computer Applications, 68, 90-113.

Bolton, R. J., & Hand, D. J. (2002). Statistical fraud detection: A review. Statistical Science, 17(3), 235-249.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority over-Sampling Technique. Journal of Artificial Intelligence Research, 16, 321-357.

Dal Pozzolo, A., Caelen, O., Le Borgne, Y.A., Waterschoot, S., & Bontempi, G. (2015). Learned lessons in credit card fraud detection from a practitioner perspective. Expert Systems with Applications, 41(10), 4915-4928.

Enrique, A.D.(2020). Random Forest Machine Learning Model Implementation on Detecting Fraudulent Credit Cards. *Makalah IF2120 Matematika Diskrit – Sem*

Hassan, M. H. and Al-Amin, M. A. (2019). A Comparative Study of Machine Learning Algorithms for Credit Card Fraud Detection. *IEEE Access*

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning:

With applications in R. *Springer*.

Kaggle: Credit Card Fraud Detection Dataset https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Kim J. and Lee Y. (2019). Machine Learning for Credit Card Fraud Detection. *Journal of Information Processing Systems*.

Kirkos, E., Spathis, C., & Manolopoulos, Y. (2007). Data mining techniques for the detection of fraudulent financial statements. Expert Systems with Applications, 32(4), 995-1003.

Kotsiantis, S., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. Informatica (Slovenia), 31(3), 249-268.

Liu, Y., & Zhou, Z.-H. (2006). The influence of class imbalance on cost-sensitive learning: An empirical study. In Proceedings of the Sixth International Conference on Data Mining (pp. 970-974). IEEE.

Ngai E.W.T., Hu Y., Wong Y.H., Chen Y., & Sun X.(2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. Decision Support Systems ,50(3),559–569.

Phua C., Lee V.C.S., Smith K., & Gayler R.(2010). A comprehensive survey of data mining-based fraud detection research. Artificial Intelligence Review ,14(1),1–14.

Russell S.J., & Norvig P.(2010). Artificial intelligence: A modern approach (3rd ed.). Pearson Education.

Sahin Y., & Duman E.(2011). Detecting credit card fraud by decision trees and support vector Machines. *Proceedings of International Multi-conference of Engineers and Computer Scientist* Munich Personal RePEc Archive Paper No .35865.

Shashank Singh 2Meeenu Grag (2021): Credit Card Fraud Detection System. *International Journal of creative research thoughts*, 9(6), 312-316

Singh M., (2020). A Comparative Study of Machine Learning Algorithms for Credit Card Fraud

Detection. *IEEE Symposium Series on Computational Intelligence (SSCI)*

Srivastava A.N., Kundu A., Sural S., & Majumdar A.(2008). Credit card fraud detection using hidden Markov model. IEEE Transactions on Dependable and Secure Computing ,5(1),37–48.

West J.(2008). Data mining for computer security.In C.C.Agarwal & H.Wang (Eds.), Managing

Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

and mining graph data (pp .289–314). Springer
STAT Library powered by NICE CXone Export and are supported by the Department of Education
Open Textbook Pilot Project, the UC Davis office of the Provost, the UC Davis Library, the California State University Affordable Learning Solutions Program, and Merlot
Imposter Scams Top Complaints Made to FTC in 2018 | Federal Trade Commission

APPENDIX

R Source File

```
#installing packages

install.packages("tidyverse")

install.packages("caret")

install.packages("ROSE")

install.packages("dplyr")

install.packages("ggplot2")

install.packages("glm2")

install.packages("pROC")

install.packages("corrplot")

install.packages("readr")

install.packages("tidyr")

install.packages("ROCR")

install.packages("glmnet")

install.packages("caretEnsemble")

install.packages("data.table")

install.packages("ggcorrplot")


#reading data

library(readr)

creditcard <- read.csv(file="creditcard.csv")

View(creditcard)
```


Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

```
#Exploratory Data Analysis
library(data.table)
library(ggplot2)
library(reshape2)
library(dplyr)
library(GGally) # for scatterplot matrix
str(creditcard) #data structure
dim(creditcard) #dimension
summary(creditcard) # provides summary statistics for all columns
glimpse(creditcard) # look at the data
names(creditcard) #
head(creditcard)
tail(creditcard)
data.table(creditcard)
sapply(creditcard, sd) # calculates standard deviation for all columns
sapply(creditcard, var) # calculates variance for all columns
creditcard$Class = as.factor(creditcard$Class) # make Class a factor
summary(creditcard$Class) #provides summary statistics for the Class column
table(creditcard$class) #checking Class imbalance
summary(creditcard$Amount) #provides summary statistics for the Amount column
sd(creditcard$Amount)
IQR(creditcard$Amount) #interquartile range
var(creditcard$Amount)
ggplot(creditcard, aes(x = Class, y = Amount)) + geom_boxplot() #boxplot of Class and Amount
ggplot(creditcard, aes(x = Class, fill = Class)) + geom_bar() #bar plot of Class
ggplot(creditcard, aes(x = V1)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v1 histogram
```


Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

```
ggplot(creditcard, aes(x = V1)) + geom_density(fill = "steelblue", alpha = 0.5) #v1 density plot

ggplot(creditcard, aes(x = Class, y = V1, fill = Class)) + geom_tile() #v1 heatmap

ggplot(creditcard, aes(x = V2)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v2 histogram

ggplot(creditcard, aes(x = V2)) + geom_density(fill = "steelblue", alpha = 0.5) #v2 density plot

ggplot(creditcard, aes(x = Class, y = V2, fill = Class)) + geom_tile() #v2 heatmap

ggplot(creditcard, aes(x = V3)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v3 histogram

ggplot(creditcard, aes(x = V3)) + geom_density(fill = "steelblue", alpha = 0.5) #v3 density plot

ggplot(creditcard, aes(x = Class, y = V3, fill = Class)) + geom_tile() #v3 heatmap

ggplot(creditcard, aes(x = V4)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v4 histogram

ggplot(creditcard, aes(x = V4)) + geom_density(fill = "steelblue", alpha = 0.5) #v4 density plot

ggplot(creditcard, aes(x = Class, y = V4, fill = Class)) + geom_tile() #v4 heatmap

ggplot(creditcard, aes(x = V5)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v5 histogram

ggplot(creditcard, aes(x = V5)) + geom_density(fill = "steelblue", alpha = 0.5) #v5 density plot

ggplot(creditcard, aes(x = Class, y = V5, fill = Class)) + geom_tile() #v5 heatmap

ggplot(creditcard, aes(x = V6)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v6 histogram

ggplot(creditcard, aes(x = V6)) + geom_density(fill = "steelblue", alpha = 0.5) #v6 density plot

ggplot(creditcard, aes(x = Class, y = V6, fill = Class)) + geom_tile() #v6 heatmap

ggplot(creditcard, aes(x = V7)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v7 histogram

ggplot(creditcard, aes(x = V7)) + geom_density(fill = "steelblue", alpha = 0.5) #v7 density plot

ggplot(creditcard, aes(x = Class, y = V7, fill = Class)) + geom_tile() #v7 heatmap

ggplot(creditcard, aes(x = V8)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v8 histogram

ggplot(creditcard, aes(x = V8)) + geom_density(fill = "steelblue", alpha = 0.5) #v8 density plot

ggplot(creditcard, aes(x = Class, y = V8, fill = Class)) + geom_tile() #v8 heatmap
```


Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

```
ggplot(creditcard, aes(x = V9)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v9 histogram

ggplot(creditcard, aes(x = V9)) + geom_density(fill = "steelblue", alpha = 0.5) #v9 density plot

ggplot(creditcard, aes(x = Class, y = V9, fill = Class)) + geom_tile() #v9 heatmap

ggplot(creditcard, aes(x = V10)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v10 histogram

ggplot(creditcard, aes(x = V10)) + geom_density(fill = "steelblue", alpha = 0.5) #v10 density plot

ggplot(creditcard, aes(x = Class, y = V10, fill = Class)) + geom_tile() #v10 heatmap

ggplot(creditcard, aes(x = V11)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v11 histogram

ggplot(creditcard, aes(x = V11)) + geom_density(fill = "steelblue", alpha = 0.5) #v11 density plot

ggplot(creditcard, aes(x = Class, y = V11, fill = Class)) + geom_tile() #v11 heatmap

ggplot(creditcard, aes(x = V12)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v12 histogram

ggplot(creditcard, aes(x = V12)) + geom_density(fill = "steelblue", alpha = 0.5) #v12 density plot

ggplot(creditcard, aes(x = Class, y = V12, fill = Class)) + geom_tile() #v12 heatmap

ggplot(creditcard, aes(x = V13)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v13 histogram

ggplot(creditcard, aes(x = V13)) + geom_density(fill = "steelblue", alpha = 0.5) #v13 density plot

ggplot(creditcard, aes(x = Class, y = V13, fill = Class)) + geom_tile() #v13 heatmap

ggplot(creditcard, aes(x = V14)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v14 histogram

ggplot(creditcard, aes(x = V14)) + geom_density(fill = "steelblue", alpha = 0.5) #v14 density plot

ggplot(creditcard, aes(x = Class, y = V14, fill = Class)) + geom_tile() #v14 heatmap

ggplot(creditcard, aes(x = V15)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v15 histogram

ggplot(creditcard, aes(x = V15)) + geom_density(fill = "steelblue", alpha = 0.5) #v15 density plot
```


Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

```
ggplot(creditcard, aes(x = Class, y = V15, fill = Class)) + geom_tile() #v15 heatmap

ggplot(creditcard, aes(x = V16)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v16 histogram

ggplot(creditcard, aes(x = V16)) + geom_density(fill = "steelblue", alpha = 0.5) #v16 density plot

ggplot(creditcard, aes(x = Class, y = V16, fill = Class)) + geom_tile() #v16 heatmap

ggplot(creditcard, aes(x = V17)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v17 histogram

ggplot(creditcard, aes(x = V17)) + geom_density(fill = "steelblue", alpha = 0.5) #v17 density plot

ggplot(creditcard, aes(x = Class, y = V17, fill = Class)) + geom_tile() #v17 heatmap

ggplot(creditcard, aes(x = V18)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v18 histogram

ggplot(creditcard, aes(x = V18)) + geom_density(fill = "steelblue", alpha = 0.5) #v18 density plot

ggplot(creditcard, aes(x = Class, y = V18, fill = Class)) + geom_tile() #v18 heatmap

ggplot(creditcard, aes(x = V19)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v19 histogram

ggplot(creditcard, aes(x = V19)) + geom_density(fill = "steelblue", alpha = 0.5) #v19 density plot

ggplot(creditcard, aes(x = Class, y = V19, fill = Class)) + geom_tile() #v19 heatmap

ggplot(creditcard, aes(x = V20)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v20 histogram

ggplot(creditcard, aes(x = V20)) + geom_density(fill = "steelblue", alpha = 0.5) #v20 density plot

ggplot(creditcard, aes(x = Class, y = V20, fill = Class)) + geom_tile() #v20 heatmap

ggplot(creditcard, aes(x = V21)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v21 histogram

ggplot(creditcard, aes(x = V21)) + geom_density(fill = "steelblue", alpha = 0.5) #v21 density plot

ggplot(creditcard, aes(x = Class, y = V21, fill = Class)) + geom_tile() #v21 heatmap

ggplot(creditcard, aes(x = V22)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v22 histogram
```


Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

```
ggplot(creditcard, aes(x = V22)) + geom_density(fill = "steelblue", alpha = 0.5) #v22 density plot

ggplot(creditcard, aes(x = Class, y = V22, fill = Class)) + geom_tile() #v22 heatmap

ggplot(creditcard, aes(x = V23)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v23 histogram

ggplot(creditcard, aes(x = V23)) + geom_density(fill = "steelblue", alpha = 0.5) #v23 density plot

ggplot(creditcard, aes(x = Class, y = V23, fill = Class)) + geom_tile() #v23 heatmap

ggplot(creditcard, aes(x = V24)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v24 histogram

ggplot(creditcard, aes(x = V24)) + geom_density(fill = "steelblue", alpha = 0.5) #v24 density plot

ggplot(creditcard, aes(x = Class, y = V24, fill = Class)) + geom_tile() #v24 heatmap

ggplot(creditcard, aes(x = V25)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v25 histogram

ggplot(creditcard, aes(x = V25)) + geom_density(fill = "steelblue", alpha = 0.5) #v25 density plot

ggplot(creditcard, aes(x = Class, y = V25, fill = Class)) + geom_tile() #v25 heatmap

ggplot(creditcard, aes(x = V26)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v26 histogram

ggplot(creditcard, aes(x = V26)) + geom_density(fill = "steelblue", alpha = 0.5) #v26 density plot

ggplot(creditcard, aes(x = Class, y = V26, fill = Class)) + geom_tile() #v26 heatmap

ggplot(creditcard, aes(x = V27)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v27 histogram

ggplot(creditcard, aes(x = V27)) + geom_density(fill = "steelblue", alpha = 0.5) #v27 density plot

ggplot(creditcard, aes(x = Class, y = V27, fill = Class)) + geom_tile() #v27 heatmap

ggplot(creditcard, aes(x = V28)) + geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") #v28 histogram

ggplot(creditcard, aes(x = V28)) + geom_density(fill = "steelblue", alpha = 0.5) #v28 density plot

ggplot(creditcard, aes(x = Class, y = V28, fill = Class)) + geom_tile() #v28 heatmap
```


Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

```
#Visualing Fraud Vs Mon-Fraud
fraudcounts <- table(creditcard$Class) # Count the number of frauds and non-frauds
# Create a data frame with the counts and labels
frauddata <- data.frame(
  fraud = c("Non-fraud", "Fraud"),
  count = c(fraudcounts[1], fraudcounts[2])
)
# Create a pie chart
ggplot(frauddata, aes(x = "", y = count, fill = fraud)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar(theta = "y") +
  labs(title = "Credit Card Fraud Detection", fill = "") +
  theme_void()
#Visualizing Physical Features
physically_imp_features <- c("Time", "Amount", "V1", "V2", "V3") # Select the physically important features
creditcardsubset <- creditcard[, c(physically_imp_features, "Class")] # Subset the data frame to include only the important features and the fraud flag
creditcardmelt <- melt(creditcardsubset, id.vars = "Class") # Melt the data frame to long format
# Create a heatmap
ggplot(creditcardmelt, aes(x = variable, y = Class, fill = value)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "red") +
  facet_wrap(~Class) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
pairs(creditcard) #scatterplot
ggpairs(creditcard)


#Data Preprocessing
```


Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

```
library(caret)
library(dplyr)
library(tidyr)
library(mice)
library(tidyverse)
library(ROSE)
library(DMwR2)
library(smotefamily)
creditcard <- creditcard %>%
  select(-Time) %>% # removing the time column
  mutate_if(is.character, as.factor) %>% # converting character columns to factors
  mutate_all(funs(ifelse(is.na(.), median(., na.rm=TRUE), .))) # handling missing values by imputing with median
creditcard$Class <- as.numeric(creditcard$Class == "fraud") # Encode categorical variables
creditcard[,1:29] <- scale(creditcard[,1:29]) # Standardize numerical variables
boxplot(creditcard) # Remove outliers
creditcard <- creditcard %>%
  filter(creditcard$Amount > -3 & creditcard$Amount < 3)
creditcard <- creditcard %>% # Balancing Using Smote
  mutate(Class = as.factor(Class))
set.seed(123)
creditcard$Class = factor(creditcard$Class)
creditcard_balanced <- SMOTE(creditcard[,1:29], creditcard$Class, K = 5)


# Feature Selection
library(caret)
library(caretEnsemble)
library(e1071)
```


Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

```
library(corrplot)
library(ggplot2)
library(ggcorrplot)
library(caTools)
# Correlation analysis
correlations <- cor(creditcard[, -31]) #correlation matrix
corrplot::corrplot(correlations, type = "upper", order = "hclust")
corrplot(correlations, method = "circle") #correlation plot
# Mutual Information Feature Selection
mi <- information.gain(Class ~ ., creditcard_balanced)
mi <- mi[order(-mi),]
top_features <- names(mi)[1:10]
creditcard <- creditcard_balanced[, c("Class", top_features)]
#chi-square test
chisq_test <- chisq.test(creditcard[, -31], creditcard$Class)
p_values <- chisq_test$p.value
significant_features <- names(which(p_values < 0.05))
creditcard <- creditcard[, c("Class", significant_features)]
# Recursive Feature Elimination
control <- rfeControl(method = "cv", number = 5)
model <- rfe(creditcard[, -31], creditcard$Class, sizes = c(1:5), rfeControl = control)
selected_features <- predict(model, creditcard[, -31])
creditcard <- creditcard[, c("Class", colnames(creditcard[, -31])[selected_features])]
# Splitting the Data into Training & Testing Sets
set.seed(123)
train_index <- createDataPartition(y = creditcard$Class, p = 0.7, list = FALSE)
creditcard_train <- creditcard[train_index, ]
credicard_test <- creditcard[-train_index, ]
```


Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

```
dim(creditcard_train)

dim(creditcard_test)

selected_features <- grep("^V", names(creditcard_train), value = TRUE)[rfe_model$optVariables] #Select features

#Plot features

ggplot(creditcard_train, aes(x = selected_features, y = Class)) +

  geom_boxplot() +

  stat_summary(fun.y = mean, geom = "point")

cor(creditcard_train[selected_features], train_data$Class, use = "pairwise.complete.obs") #Check correlation


# Logistic Regression

library(glm2)

library(glmnet)

library(pROC)

log_model <- glm(Class ~ ., data = creditcard_train, family = "binomial") #fit the logistic regression model

summary(lr_model) #provides summary statistics of the logistic regression model

plot(lr_model) #plots the regression model

# Predict on Test Data

X_test <- creditcard_test %>% select(-Class)

y_test <- creditcard_test$Class

creditcard_test$Class <- predict(lr_model, newdata = creditcard_test, probabilty = TRUE)

# Model Evaluation

library(pROC)

library(caret)

conf_mat <- confusionMatrix(pred, y_test) # confusion matrix

conf_mat$table

roc_curve <- roc(y_test, pred) # roc curve
```


Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

```
auc <- auc(roc_curve) #auc
plot(roc_curve, main = paste("ROC Curve (AUC =", round(auc, 3), ")"))


#Random Forest
# Fit a random forest model
rf_model <- randomForest(Class ~ ., data = creditcard_train, ntree = 100, mtry = 5)
# Predict on the test data
predictions <- predict(rf_model, newdata = creditcard_test)
# Evaluate the model using F1 score, accuracy, precision, recall, AUC, and ROC
f1_score <- F1_Score(predictions, creditcard_test$Class)
accuracy <- confusionMatrix(predictions, creditcard_test$Class)$overall["Accuracy"]
precision <- confusionMatrix(predictions, creditcard_test$Class)$byClass["Precision"]
recall <- confusionMatrix(predictions, creditcard_test$Class)$byClass["Recall"]
auc <- as.numeric(performance(prediction(predictions, creditcard_test$Class), "auc")@y.values)
roc <- plot(performance(prediction(predictions, creditcard_test$Class), "tpr", "fpr"), main = "ROC Curve for Random Forest Model")
# Print the evaluation metrics
cat("F1 Score: ", f1_score, "\n")
cat("Accuracy: ", accuracy, "\n")
cat("Precision: ", precision, "\n")
cat("Recall: ", recall, "\n")
cat("AUC: ", auc, "\n")
# Save the ROC plot as an image file
png("roc_plot.png")
print(roc)
dev.off()


#Naive Bayes
```


Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

```
# Fit a Naive Bayes model

nb_model <- naiveBayes(Class ~ ., data = creditcard_train)

# Predict on the test data

predictions <- predict(nb_model, newdata = creditcard_test)

# Evaluate the model using F1 score, accuracy, precision, recall, AUC, and ROC

f1_score <- F1_Score(predictions, creditcard_test$Class)

accuracy <- confusionMatrix(predictions, creditcard_test$Class)$overall["Accuracy"]

precision <- confusionMatrix(predictions, creditcard_test$Class)$byClass["Precision"]

recall <- confusionMatrix(predictions, creditcard_test$Class)$byClass["Recall"]

auc <- as.numeric(performance(prediction(predictions, creditcard_test$Class), "auc")@y.values)

roc <- plot(performance(prediction(predictions, creditcard_test$Class), "tpr", "fpr"), main = "ROC Curve for Naive Bayes Model")

# Print the evaluation metrics

cat("F1 Score: ", f1_score, "\n")

cat("Accuracy: ", accuracy, "\n")

cat("Precision: ", precision, "\n")

cat("Recall: ", recall, "\n")

cat("AUC: ", auc, "\n")

# Save the ROC plot as an image file

png("roc_plot.png")

print(roc)

dev.off()


#Decision Trees

# Fit a decision tree model

dt_model <- rpart(Class ~ ., data = train, method = "class")

# Predict on the test data

predictions <- predict(dt_model, newdata = test, type = "class")
```


Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

```
# Evaluate the model using F1 score, accuracy, precision, recall, AUC, and ROC

f1_score <- F1_Score(predictions, test$Class)

accuracy <- confusionMatrix(predictions, creditcard_test$Class)$overall["Accuracy"]

precision <- confusionMatrix(predictions, creditcard_test$Class)$byClass["Precision"]

recall <- confusionMatrix(predictions, creditcard_test$Class)$byClass["Recall"]

auc <- as.numeric(performance(prediction(predictions, creditcard_test$Class), "auc")@y.values)

roc <- plot(performance(prediction(predictions, creditcard_test$Class), "tpr", "fpr"), main = "ROC Curve for Decision Tree Model")

# Print the evaluation metrics

cat("F1 Score: ", f1_score, "\n")

cat("Accuracy: ", accuracy, "\n")

cat("Precision: ", precision, "\n")

cat("Recall: ", recall, "\n")

cat("AUC: ", auc, "\n")

# Plot the decision tree

rpart.plot(dt_model)

# Save the ROC plot as an image file

png("roc_plot.png")

print(roc)

dev.off()


#k-Nearest Neighbours

# Fit a k-nearest neighbors model

knn_model <- knn(train[, -31], test[, -31], train$Class, k = 5)


# Predict on the test data

predictions <- as.numeric(knn_model)
```


Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2

```
# Evaluate the model using F1 score, accuracy, precision, recall, AUC, and ROC
f1_score <- F1_Score(predictions, creditcard_test$Class)
accuracy <- confusionMatrix(predictions, creditcard_test$Class)$overall["Accuracy"]
precision <- confusionMatrix(predictions, creditcard_test$Class)$byClass["Precision"]
recall <- confusionMatrix(predictions, creditcard_test$Class)$byClass["Recall"]
auc <- as.numeric(performance(prediction(predictions, creditcard_test$Class), "auc")@y.values)
roc <- plot(performance(prediction(predictions, creditcard_test$Class), "tpr", "fpr"), main = "ROC Curve for KNN Model")
# Print the evaluation metrics
cat("F1 Score: ", f1_score, "\n")
cat("Accuracy: ", accuracy, "\n")
cat("Precision: ", precision, "\n")
cat("Recall: ", recall, "\n")
cat("AUC: ", auc, "\n")
# Save the ROC plot as an image file
png("roc_plot.png")
print(roc)
dev.off()
```


Article DOI: 10.52589/BJCNIT-YDIJNXG2
DOI URL: https://doi.org/10.52589/BJCNIT-YDIJNXG2