International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Credit Card Fraud Detection Using Machine Learning Algorithms: A Comparative Study of Six Models

Joseph J. Assabil[1] and Ibidun Christiana Obagbuwa[2]*

Submitted: 15/05/2024 **Revised:** 25/06/2024 **Accepted:** 08/07/2024

Abstract*:* Credit Card Fraud (CCF) is a significant financial threat where individuals impersonate others to conduct unauthorized financial activities. This study implements a Combined Resampling Technique (CRT) as a specialized new approach for addressing class imbalances and enhancing model performance in credit card fraud (CCF) detection. Additionally, it evaluates the effectiveness of various machine learning models in accurately identifying fraudulent transactions. Performance metrics included cross-validation K-fold, AUPRC score, precision, recall, averages, and F1 scores. The models assessed encompass traditional algorithms like Logistic Regression, Decision Trees, and K-Nearest Neighbors (KNN), along with ensemble methods such as Random Forest, Adaboost, and Xgboost. The dataset utilised was a simulated data set containing credit card transactions spanning January 2019 to December 2020, involving 1000 customers and 800 merchants. The study addresses data imbalance using techniques like SMOTE and employs feature engineering for improved results. Notably, the K-NN algorithm demonstrated superior performance in detecting fraudulent transactions, making it a valuable tool in combating the CCF crisis.

Keywords: *AUPRC-score, Adaboost, Combined Resampling Technique, K- fold Cross Validation, K-NN algorithm, SMOTE, Xgboost.*

1. Introduction

The technological era has witnessed a remarkable increase in financial fraud activities globally. There have been various reasons attributed to the rise in these fraudulent activities. Some of the listed reasons are, increase in e-commerce sites with structured online payments, online banking, global shift in the trade units to the online platform in an attempt to target larger consumer base etc. The easiness at which this type of fraud happens has raised dozens of questions across numerous countries. CCF can be narrowly categorized into Card Present Frauds (CPF) and Card Not Present Frauds (CNPF). CPFs occur when a person steals another person's physical credit card and uses it to carry out fraudulent activities [1]. On the other hand, CNPFs does not overtly require the presence of a physical card. CNPFs are often carried out remotely where a vulnerable individual's credit card details are obtained through phishing scams or data breaches, spywares, or even through the use of software to generate fake credit card numbers [1, 2]. In recent years, the use of machine learning algorithms and other artificial intelligence techniques have become increasingly popular in the fight against credit card fraud, as these techniques can help detect patterns and anomalies in large volumes of data and flag potentially fraudulent transactions in real time. The upsurge of CCF related dilemmas as such, has created a dire need in the field of Artificial Intelli- gence (AI) and Machine Learning (ML) to devise wholesome detective ways to protect the transaction interests of individuals across the globe [3, 4, 5] and the focus of this research is to briefly glance at some of these ML algorithms and to compare their effectiveness towards combating the conundrum at hand. The approach adopted in this paper is nothing of novelty, it just happens to boast of a customised sampling technique in handling the class imbalance issue related to credit card datasets. According to a report by [6, 7], CCF losses reached nearly a staggering $30 billion globally in 2018, with the United States accounting for nearly half of those losses. Another report by [5, 8] explored the credit card fraud contributions of individual countries. Several publications have also made their position clear about the trillions of dollars generated on the online platforms and the high proportion of money lost due to fraudulent transactions as illustrated in Figure 3. This, as such, has created a dire need for machine learning algorithms in combating fraudulent transactions, as well as devising novel detection algorithms using machine learning models, all in bid to swiftly detect fraudulent transactions to save millions of dollars for companies as well as vulnerable individuals. Other publications published by [1, 9, 5], have

[1] *School of Computer Sciences & Applied Mathematics;*
University of the Witwatersrand; South Africa
[2] *Department of Computer Science & Information Technology,*
Sol Plaatjie University, South Africa
Email: ibidun.obagbuwa@spu.ac.za

all made clear their intentions on the need for both novel and pre-existing machine learning detection techniques, as well as neural network establishments, in their quest for detecting fraudulent transactions, all in aid of making the online space, a safer platform for transacting. Most of these publications actually agree to some extent on the success rates achieved by some of these detection algorithms, particularly with a specific Random Forest boasting a high accuracy detection rate for fairly small datasets, while others boast a fair detection rate with different data set ranges [10, 8, 11]. It suffices to say that adequate research is currently being done in the area of deep learning, neural networks and model training [5, 9] to work on preventing fraudulent activities associated with credit cards. As the saying goes, prevention is better than cure, perhaps prevention with deep learning is the way to go in the immediate future. In the research paper, we will dive into some of the detection algorithmic techniques out there. These detection algorithms are classified into three categories viz, the traditional algorithms, the ensemble algorithms, and the deep learning algorithms.

This research ignites a beacon of hope in the relentless fight against financial fraud. By wielding the power of machine learning, we have unearthed a potential champion: the K-Nearest Neighbors (K- NN) algorithm. Within the confines of a simulated dataset, K-NN emerged victorious, demonstrably surpassing its competitors across a spectrum of evaluation metrics (average score, AUC, AUPRC, and F1-score). This dominance signifies its remarkable prowess in deciphering the intricate patterns woven by fraudulent activity, patterns that often defy linear explanation and plague traditional detection methods. K-NN's inherent strengths, particularly its ability to navigate complex, non-linear relationships and thrive in imbalanced datasets where fraudulent transactions are scarce, make it a formidable weapon in this ongoing war. While these findings are confined to the realm of simulated data, they offer a springboard for real-world application. Equipping financial institutions with this potentially transformative tool could revolutionize fraud detection. This research, however, transcends the mere identification of a single champion. It serves as a catalyst for further exploration. The landscape of machine learning algorithms is vast, and models like XGBoost and neural networks hold immense promise. By delving deeper into their capabilities and refining existing models for real-world complexities, we can forge an even more robust arsenal. This research stands as a testament to the transformative power of machine learning in safeguarding financial transactions. It paves the way for a future where sophisticated algorithms become the cornerstone of enhanced detection and prevention capabilities, ultimately securing our financial well- being.

2. Methodology

2.1 Experimental Set-Up

Hardware: The experiments were conducted on a standard personal computer or a cloud-based machine equipped with sufficient processing power and memory, such as a system with 16GB RAM and a multi-core CPU/GPU.

Software: The software environment included the Python programming language, Jupyter Notebook as well as Google Colab notebook for the code execution and interactive computing, machine learning li- braries such as scikit-learn, imbalanced-learn, sklearn.linear_model (for importing logistic regression clas-sifier), sklearn.model_selection (for train, test and splitting dataset), sklearn.ensemble (for importing the Random Forest classifier), sklearn.metrics (for importing the evaluation metrices), sklearn.feature_selection (for importing the feature selection technique), imblearn.over_sampling (for importing the Combined Resampling technique, involving Oversampling, SMOTE and Undersampling), TensorFlow/Keras, and XGBoost as well as data manipulation libraries including pandas and numpy, and visualization tools like matplotlib and seaborn.

Dataset: The study utilized a publicly available credit card fraud detection dataset, specifically the syn-thetic fraud transaction dataset from Kaggle. The link can be found in Kaggle - (https://www.kaggle.com/datase card-frauds-synthetic-dataset)

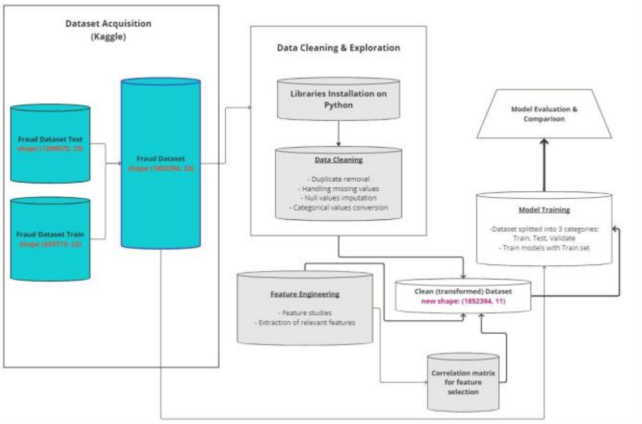

Dataset Acquisition
(Kaggle)
Fraud Dataset Test
shape (1296675, 23)
Fraud Dataset Train
shape (555719, 23)
Fraud Dataset
shape (1852394, 23)
Data Cleaning & Exploration
Libraries Installation on Python
Data Cleaning
- Duplicate removal
- Handling missing values
- Null values imputation
- Categorical values conversion
Model Evaluation & Comparison
Model Training
-Dataset splitted into 3 categories: Train, Test, Validate
- Train models with Train set
Feature Engineering
- Feature studies
- Extraction of relevant features
Clean (transformed) Dataset
new shape: (1852394, 11)
Correlation matrix for feature selection

Figure 1: Architectural diagram for methodology execution.

2.2 Dataset

The dataset retrieved from kaggle consisted of a Fraud train and Fraud test. The two (2) datasets were conflated to form the fraud dataset where the relevant Exploratory Data Analysis (EDA) techniques were performed on the dataset (see Figure 1). Due to the arduous issues related to accessing real time fraud data from the financial industries, this research paper rather dealt with a simulated credit card transaction dataset, which contained genuine and fraud transactions that spanned from Jan 2019 - Dec 2020. Simulated in the context that the data in its entirety mimicked a real time data set thus including null values, duplicate entries and any common mistakes easily associated with real time datasets. The dataset covered credit cards of 1000 customers doing transactions with a pool of 800 merchants. The dataset worked with in this research paper was imbalanced and that required some work to be done on the general dataset before the actual model fitting. Some basic Exploratory Data Analysis (EDA) was performed on the dataset to deal with the anomalies encountered with the data. Also since the dataset to deal with the anomalies encountered with the data. Also since the dataset is practically a simulant, there was no need for PCA analysis to hide certain confidential features or to scale any other features since this dataset was specifically designed for such a task of analysis thus erasing any threat to the relevant parties.

2.2.2 Data Attributes

2.2.1 Source of Simulation

The dataset was extracted from Kaggle and was generated using Sparkov Data Generation Github tool, which was created by Brandon Harris. This simulation was run from 1 Jan 2019 to 31 Dec 2020 and the files were conflated and transformed into one data frame using the concatenate function.

Unnamed:	- Unique index
trans_date:	- Transaction date
trans_time:	- Transaction time
cc_num:	- Credit Card number
merchant:	- Type of Merchant
category:	- Category of the transaction
amt:	- Amount of Transaction
first:	- Individual First name
last:	- Individual Surname
amt:	- Amount of Transaction
unix_time:	- Time lapse for Transaction
is_fraud:	- Fraudulent target class denoted by 1 and 0 otherwise.

2.2.3 Dealing with Class Imbalances

In this study, the dataset was highly imbalanced, with fraudulent cases accounting for only 0.5% of the total entries. This significant imbalance posed a challenge, as training machine learning models on such skewed data typically leads to suboptimal performance. Models trained on imbalanced data tend to be biased towards the majority class, resulting in poor detection of the minority class, which in this case isthe fraudulent instances. To address this issue, a technique devised in this paper known as Combined Resampling Technique was employed to balance the dataset.

Oversampling

Oversampling is a technique used to increase the number of instances in the minority class to match themajority class. In this study, oversampling was implemented by increasing the sample size of the minority class (i.e., the 'is_fraud' target feature with a class value of 1) using a probabilistic approach. Specifically, the minority class instances were replicated based on their proportion in the dataset to enhance their representation. This method helps to ensure that the machine learning models have enough instances offraudulent cases to learn from, thereby improving their ability to detect fraud.

Undersampling

Undersampling on the other hand, involves reducing the number of instances in the majority class (i.e.,the 'is_fraud' target feature with a class value of 0). By selectively removing instances from the majority class, the overall dataset becomes more balanced. This approach helps to mitigate the bias towards the majority class and ensures that the models are not overwhelmed by non-fraudulent instances.

Combined Resampling Technique and Proportion Spread Strategy

In this paper, a combined approach of both oversampling and undersampling was used to handle the class imbalance. The strategy adopted involved a proportion spread of (0.8 : 0.008), meaning the minority class was increased by 10,000% while the majority class was reduced by 35%. Specifically, the minority class instances were increased from 9,651 to 965,100, and the majority class instances were decreased from 1,842,743 to 1,206,375. This adjustment resulted in an almost balanced dataset, allowing for more effective training of the machine learning models. By implementing this combined resampling technique, the study aimed to create a balanced dataset that provided the models with an adequate representation of both fraudulent and non-fraudulent cases. This, in turn, enhanced the models' ability to accurately detect and predict fraud, improving the overall effectiveness of the fraud detection system.

2.2.4 Data Cleaning

The dataset was examined for null values using the 'isnull' function on python. The function revealed the absence of null entities in the dataset. The data set was also examined for duplicated entities and there was none across the [1852394 rows 23 columns] data frame. Features in the data set such as transaction time and date, names of the individuals, geographical address, zip code and others were all regarded as irrelevant for purposes of data pre-processing. The relevant features from the data were then extracted based on a correlation with the target feature from a correlation plot.

2.3 Feature Selection Method

Fraud detection relies heavily on the quality and relevance of features used to identify fraudulent transactions [12, 13]. Feature selection techniques play a crucial role in this process by identifying the most informative features that contribute significantly to fraud classification, while discarding irrelevant or redundant ones [13]. Filter methods like Pearson correlation coefficient evaluate the linear correlation between features and the target variable (fraudulent or legitimate). Features with high positive or negative correlation with the target are considered relevant for fraud classification, while features with low correlation are discarded. Additionally, correlation among selected features is minimized to avoid redundancy [13]. Filter methods with correlation coefficients are valuable tools for initial feature selection in fraud detection. It helps eliminate irrelevant or highly correlated features, improving model efficiency. They are efficient for large datasets, computationally inexpensive. It often assumes linear relationships between features, and may overlook non-linear relationships [14, 13]. A correlation plot, usually referred to as a correlation matrix or a heatmap, is a visual representation of the correlation coefficients between multiple variables in a dataset. A Correlation plot thus measures the strength and direction of the linear relationship between two variables. A correlation matrix is thus a very powerful tool employed in the EDA step to understand how the individual variables were related to each other and most importantly, to the target feature. The Correlation matrix uses a correlation coefficient which is a statistical value that ranges from -1 to 1 [15, 16]. It is used as a quantification measure for the degree of association between two variables. Positive values closer to 1, indicate a strong positive linear relationship i.e. when one variable increases, the other tends to increase. Negative values indicate a negative linear relationship (as one variable increases, the other tends to decrease), and values closer to zero indicate weak or no linear relationship and a value of 1 is a reflection of a perfect linear relationship (highly rare) [15]. Thus with the correlation plot, we were able to deduce the relevant features to be extracted in order to fit the our models for more concise results.

2.4 Evaluation Metrices:

2.4.1 AUC-ROC - Score

The ROC curve is a visual tool used to illustrate the balance between the True Positive Rate (TPR) and the False Positive Rate (FPR) across various threshold values. TPR, often referred to as sensitivity or recall, signifies the proportion of correctly predicted positive instances relative to all actual positive instances. On the other hand, FPR denotes the proportion of incorrectly predicted positive instances relative to all actual negative instances [17, 2].

The ROC-AUC score quantifies the area under the ROC curve. This area ranges from 0 to 1, with higher values indicating better model performance. The AUC value can be interpreted as follows:

- **AUC = 0.5:** The model's predictions are equivalent to random guessing

- **AUC > 0.5:** The model is better than random guessing

- **AUC = 1:** The model perfectly distinguishes between the two classes.

Advantages of ROC-AUC:

1. **Scale Invariance:** ROC-AUC is unaffected by the class distribution and is insensitive to changes in the decision threshold. This makes it a robust metric for imbalanced datasets.

2. **Classifier Comparison:** ROC-AUC provides a useful way to compare the performance of different models, even if their predicted probabilities are on different scales.

3. **Threshold Selection:** The ROC-AUC metric is instrumental in helping to pinpoint the ideal classification threshold, enabling one to make informed decisions that strike the right balance between TPR and FPR. Furthermore, ROC-AUC demonstrates its robustness when confronted with imbalanced datasets, a common scenario where one class vastly outnumbers the other. In such situations, where accuracy can be misleading, ROC-AUC steps forward as a dependable measure, providing a more accurate evaluation of the model's performance.

Limitations of ROC-AUC:

• ROC-AUC might not be the best metric when the cost of false positives and false negatives is significantly different.

• It does not provide information about the actual performance of the model in terms of accuracy, precision, or recall.

• It assumes that the model produces predicted probabilities, which might not always be the case for all classification algorithms.

In summary, ROC-AUC is a valuable metric for assessing the overall quality of a binary classification model's predictions. It provides a single value that encapsulates the model's ability to distinguish between classes across different threshold values. Its scale invariance, ability to handle class imbalances, and usefulness in model comparison make it a widely used metric in machine learning.

2.4.2 AUPRC

The AUPRC (Area Under the Precision-Recall Curve) score stands as a vital metric employed to assess the effectiveness of binary classification models. It finds particular relevance in scenarios featuring imbalanced datasets, where one class significantly outnumbers the other. AUPRC centers its evaluation on the delicate balance between precision and recall, shedding light on a model's capacity to accurately identify positive cases while minimizing the occurrence of false positives [17, 18]. Visualized as a precision-recall curve, this metric offers a graphical depiction of the precision-recall trade-off across various threshold values. Precision assesses the fraction of positive predictions that are correct among all instances predicted as positive, whereas recall, also known as sensitivity or the true positive rate, evaluates the fraction of actual positive instances correctly identified among the total positive instances.

AUPRC Interpretation:

The AUPRC score quantifies the area under the precision-recall curve. Similar to ROC-AUC, AUPRC's values range from 0 to 1, where higher values indicate better model performance. The AUPRC value can be interpreted similarly to ROC-AUC:

• **AUPRC = 0.5:** The model's predictions are equivalent to random guessing.

• **AUPRC > 0.5:** The model is better than random guessing.

• **AUPRC = 1:** The model perfectly identifies positive instances.

Advantages of AUPRC:

1. Imbalanced Datasets: AUPRC is especially useful when dealing with imbalanced datasets, where the positive class is much rarer. It provides a more informative evaluation of a model's performance compared to accuracy or ROC-AUC in such cases.

2. Focus on Positive Class: AUPRC prioritizes the positive class, which is crucial in applications where correctly identifying positive instances is more important than overall accuracy.

3. Threshold Selection: Similar to ROC-AUC, AUPRC can assist in choosing an appropriate threshold for class prediction.

4. Model Comparison: Just like with ROC-AUC, higher AUPRC values generally indicate better model performance, making it useful for model selection.

5. Imbalanced Data: When working with imbalanced datasets, AUPRC provides a better understanding of how well a model is identifying positive instances.

Limitations:

AUPRC, like ROC-AUC, does not provide a complete picture of a model's performance and should be considered alongside other metrics like accuracy, precision, and recall. It assumes that the model produces predicted probabilities. In summary, AUPRC is a valuable metric for assessing the performance of binary classification models, especially in cases where the class distribution is imbalanced. It focuses on the precision-recall trade-off and is well-suited for scenarios where correctly identifying positive instances is more critical. When working with imbalanced data or applications with varying class distributions, AUPRC provides insights into the model's ability to perform well in challenging situations.

2.4.3 F1-Score

The F1-score (see equation (3)) is a metric that balances precision (see equation (1)) and recall (refer to equation (2)), providing a single value that represents a model's overall accuracy in identifying positive instances while minimizing false positives and false negatives. It is especially useful when the class distribution is imbalanced.

F1-score Calculation:

$$Precision = \frac{(TruePositive)}{(TruePositive) + (FalsePositive)} \quad (1)$$

$$Recall = \frac{(TruePositive)}{(FalseNegative) + (TruePositive)} \quad (2)$$

$$F1-score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (3)$$

Use and Advantages:

F1-score (see equation (3)) is particularly useful when the cost of false positives and false negatives is important. It is a single metric provision that captures both precision and recall, giving you an overall performance measure [17, 19]. In cases where class imbalance is a concern as in the dataset in this paper, F1-score thus helps you evaluate a model's ability to perform well on both classes.

2.4.4 Macro-average & Weighted-average Scores

The macro and weighted average scores are all evaluation techniques commonly used for multiclass classification tasks where there are multiple classes to predict and sometimes, uniclass classification tasks which this paper focused on.

Macro-Average: In macro-average, you calculate the precision, recall, and F1-score for each class separately and then take the average of these values. This treats each class equally, regardless of class distribution.

Weighted Average: In weighted average, you calculate the precision, recall, and F1-score for each class separately, and then take the weighted average, where the weights are determined by the class distribution. This gives more weight to classes with more samples [5].

Use and Advantages:

Both macro-average and weighted average are suitable when evaluating a model's performance across multiple classes. Macro-average treats all classes equally, providing an unbiased overall performance measure. Weighted average is better when class distributions are imbalanced as with credit card fraud transaction datasets, as it takes into account the prevalence of each class. In summary, the macro-average, and weighted average scores are essential metrics for assessing the performance of classification models across different scenarios [17, 18, 5].

2.4.5 Cross-Validation Score

Cross-validation score is an evaluation metric that provides an aggregated view of how well a model generalizes to new, unseen data across different subsets of the dataset [19]. The cross-validation score is usually calculated by averaging the individual scores whether F1-score, accuracy, ROC-AUC or any other appropriate metric obtained from each fold [19, 20].

Uses:

1. **Reliable Performance Estimate:** Cross-validation provides a more reliable estimate of a model's performance compared to a single train-test split, as it assesses the model on multiple variations of the data.

2. **Model Selection:** Cross-validation helps in selecting the best model among different algorithms or parameter settings.

3. **Identifying Overfitting:** If a model performs well on the training data but poorly on cross validation data, it might be overfitting. Cross-validation helps identify such cases.

4. **Handling Limited Data:** Cross-validation is particularly useful when you have a limited amount of data, as it maximizes the use of available samples for both training and evaluation.

3. Results & Discussion

3.1 Foreseeable Insights from the Data

From Figures 2 & 3, one can notice that in comparing the individual means for the fraudulent and non- fraudulent datasets, the fraudulent datasets have a significantly higher mean amount, larger population size and a longer time of transactions as well as an increased mean for age.

Higher mean amount in the fraudulent samples: Why is this so? Could this be attributed to the fact that fraudsters would want to make the most out of any given opportunity to make a notable sum of money.

Increased transaction time in the fraudulent samples: The average time recorded for the fraudulent sample was significantly higher than the non-fraudulent sample (see Figures 2&3). This reason could be attributed to the need for fraudsters to adequately cover their tracks, the restraint offered by the platforms in attempt to avoid fraud activities, the dire need for the fraudster to bequeath more amount than necessary etc. These could be some of the reasons that can be attributed to the increased average in the fraudulent samples compared to the non-fraudulent samples.

Increased mean age in the fraudulent samples: Figure 4 shows an increased mean age in the fraudulent samples compared to the non-fraudulent samples. This rather is an unusual insight worth discussing. Could this be associated with the fact that only the matured are capable of carrying out fraudulent feats? However this does not hold for an insight, as age is not a definite predictor of fraud related activities since anyone of any particular age group is culpable culprit.

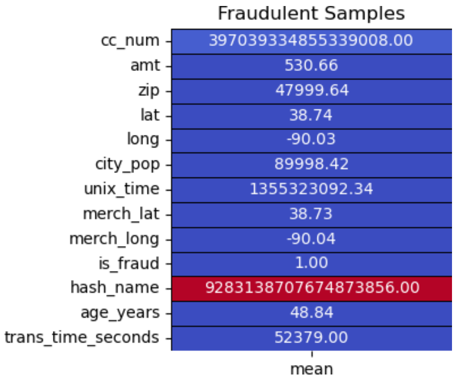
Larger Population size for the fraudulent sample: This insight is widely discussed across multiple disciplines as it was found that crime statistics are strongly correlated with a larger population size.

Some of these insights shared above, could be the gateway for determining and swiftly detecting fraudulent patterns and in the foreseeable future if explored, could add to the list of mitigating arsenals in the weaponry for the combat against credit card fraud activities.

3.2 Machine Learning Models

Logistic Regression Model

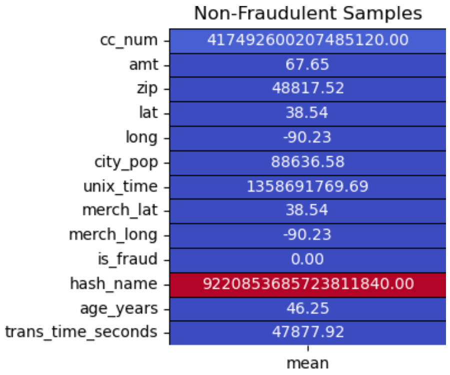
The logistic regression algorithm is a binary classification algorithm that makes use of a logistic function in order to predict the probability that an input belongs to a particular class. The Logistic regression model also falls under supervised learning algorithm and as a binary classification system model, it is trained to classify any given data into one of two categories for example, yes or no, true or false [18]. It works by modeling the probability of an input belonging to a particular class as a function of the input features. The output generated by the logistic regression model is usually a probability distribution which has values lying between 0 and 1, and can be threshold to make a binary classification decision. Logistic regression is often simple in its implementations and interpretations, and can thus be useful for understanding the relationship between input features and the target variable. However, it may not perform well on datasets with non-linear relationships between the input features and the target variable [18, 10].



Fraudulent Samples

	mean
cc_num	397039334855339008.00
amt	530.66
zip	47999.64
lat	38.74
long	-90.03
city_pop	89998.42
unix_time	1355323092.34
merch_lat	38.73
merch_long	-90.04
is_fraud	1.00
hash_name	9283138707674873856.00
age_years	48.84
trans_time_seconds	52379.00

Figure 2: Analysis into the fraudulent datasets: Mean of respective features captured showed that the average amount (amt) and transaction time (unix_time_) spent under the fraud- ulent transactions were higher than the average amount spent under the non-fraudulent sam- ples.



Non-Fraudulent Samples

	mean
cc_num	417492600207485120.00
amt	67.65
zip	48817.52
lat	38.54
long	-90.23
city_pop	88636.58
unix_time	1358691769.69
merch_lat	38.54
merch_long	-90.23
is_fraud	0.00
hash_name	9220853685723811840.00
age_years	46.25
trans_time_seconds	47877.92

Figure 3: Analysis into the non-fraudulent datasets: Mean of respective features captured showed that the average amount (amt) and transaction time (unix_time_) spent under the fraudulent transactions were higher than the average amount spent under the non-fraudulent samples.

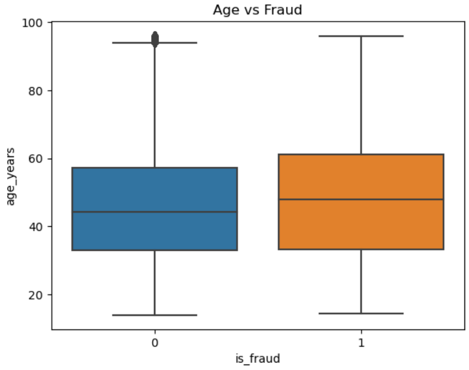


Figure 4: Age driven analysis of the dataset where non-fraudulent categories are indicated with 0 and the fraudulent categories indicated with 1.

K - Nearest Neighbour

KNN is an instance-based algorithm that makes predictions based on the similarity of an input to the training data. KNNs falls under a category of lazy learning algorithms used for classification and regression analysis [18, 21, 22]. It works by finding the K nearest neighbors in the training data to a given input and making a classification or regression decision based on the average or weighted average of their values. KNN can be useful for non-parametric and non-linear classification problems and can handle datasets with complex decision boundaries. However, the only disadvantage it has is that it has high sensitivity to the choice of K and the distance metric used, and may not perform well in general on high-dimensional datasets [21, 22, 19].

Decision Tree Classifier (DTC)

The Decision Tree Classifier algorithm builds a tree-like structure to make decisions based on certain input features. It has advantages like interpretability and non-linearity handling, but it also usually suffers from over-fitting and instability. Careful tuning and techniques like pruning usually help mitigate these issues.

Here's how a Decision tree classifier works:

1. Tree Construction: The algorithm starts with the entire dataset at the root node and selects the best feature to split the data based on a certain criterion (e.g., Gini impurity, entropy, information gain). The dataset is divided into subsets based on the chosen feature's values.

2. Node Expansion: The process is recursively applied to the subsets, creating child nodes. Each node represents a subset of the data and corresponds to a decision rule based on a feature value. The algorithm continues to split nodes until a stopping criterion is met (e.g., maximum depth, minimum samples per leaf).

3. Leaf Nodes (Decision): The final nodes in the tree are called leaf nodes or terminal nodes. Each leaf node is associated with a class label that represents the predicted output. When new data is input into the tree, it traverses the tree from the root to a leaf node based on the feature values, and the class label associated with that leaf node becomes the predicted output.

Random Forest

Random forests are ensemble methods that works with a combination of numerous decision trees in order to make predictions [19, 22]. Each tree is trained on a randomly derived subset of the features and a random subset of the training data, which in turn helps to reduce over-fitting and bolster the overall accuracy of the model.

- Random Forest is robust, handles a mix of feature types, and typically requires less hyperparameter tuning compared to individual decision trees. It is suitable for a wide range of classification tasks and can handle large datasets.

- The main drawback is that this model's predictions might be harder to interpret than a single decision tree. Also, while it reduces overfitting compared to a single decision tree, it can still overfit in some cases.

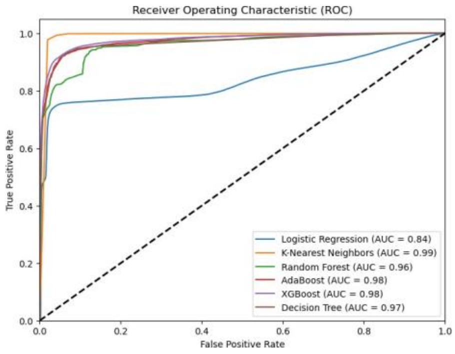


Figure 5: Results summary for models indicating various AUC scores

Adaboost

Adaboost is also an ensemble learning algorithm that focuses on improving the performance of weak learners (classifiers with modest accuracy) by combining them into a strong classifier. Adaboost performs multiple iterations. In each iteration, it assigns higher weights to the misclassified examples from the previous iteration. This focuses the learner's attention on harder-to-classify examples. Adaboost is able to continuously corrects its mistakes by assigning higher importance to misclassified examples, effectively adjusting the model's focus on the difficult cases.

Xgboost

Extreme Gradient Boosting commonly known as XGBoost is a sophisticated gradient boosting algorithm designed for both regression and classification tasks. It enhances the boosting approach by adding regularization and handling of missing values. XGBoost uses a custom objective function that combines a loss function with regularization terms. This helps in both minimizing the errors and controlling the model's complexity.

3.3 Results from this Work

As illustrated in Table 1 and Figure 2, the K-Nearest Neighbour classification algorithm demonstrated the highest accuracy amongst the six (6) models compared in this study. It came out atop in terms of average score, AUC score, AUPRC and F1 scores. Based on the metric specific insights, its paramount to notice that the AUC metric measured the model's ability to distinguish between the fraudulent and legitimate transactions. All models except the Logistic Regression model performed extremely well as seen in Figure 5 - Figure 18 with their AUC scores. The average accuracy score metric represents the overall proportion of correctly classified transactions. Under this metric, Xgboost, KNN, and Adaboost achieved high accuracy, indicating their effectiveness in making correct predictions. The AUPRC metric considers both true positives and false positives, focusing on the model's ability to correctly identify fraudulent transactions while minimizing false alarms. The K-NN classifier again outperformed (see Figures 5, 9, 10 & 11) the rest of the models, highlighting its balance between precision and recall. The F1-score metric unlike the rest of the metrics combines precision and recall into a single metric, providing a balanced view of the model's performance. Similar to AUPRC, K-NN demonstrated the best balance between identifying true positives and avoiding false positives. Based on these analysis, K-NN followed by Xgboost, stands out as the most effective model for fraud detection in the simulated dataset. Its superior performance across all metrics suggests its robustness and generalizability. Some noteworthy instances to consider regarding the overall poor performance of the Logistic classifier in this study can be attributed to one or more of the following points discussed below:

- The Logistic regression classifier usually assumes a linear relationship between the independent variables and the dependent variable (fraudulent or legitimate). However, fraud patterns might not always exhibit such linear relationships. Complex interactions and non-linear dependencies might exist between various factors contributing to fraud, which Logistic Regression struggles to capture.

- Inability to handle complex data: Fraudulent activities often involve intricate schemes and evolving tactics. Logistic Regression thus in this wise, struggled to effectively model such complex (see Figure 6) data compared to more flexible models like XgBoost and decision trees that can handle non-linear patterns and feature interactions. The exact complexity handling problem aligns with the work of [23] where the Logistic regression model equally struggled in like manner.

• Sensitivity to outliers: Fraudulent transactions can often present as outliers in the data. Logistic Regression can be sensitive to these outliers, potentially skewing the model's coefficients and impacting its generalizability [24].

• Limited feature selection: Logistic Regression primarily relies on feature coefficients to distinguish between classes. In scenarios with numerous features as in this study, it might not effectively identify the most relevant ones for accurate classification.

It is important to note that the choice of the optimal model ultimately depends on the specific context and priorities of the fraud detection task. While the likes of K-NN and the boosting algorithms excel in this scenario, other models might prove more suitable depending on factors like size of the dataset, interpretability, computational efficiency, or sensitivity to imbalanced data.

Table 1: Comparison of the various classification algorithms on the chosen dataset revealed that the KNN algorithm to be the most efficient with an F1-score of 97%.

Models	**Average(%)**		**AUC score(%)**	**AUPRC score(%)**	**F1-score(%)**
	Macro	**Weighted**			
Lr	86	86	84	82	83
K-NN	**97**	**97**	**99**	**95**	**97**
DTC	93	93	97	89	92
Rf	89	90	96	85	90
Adaboost	93	93	97	89	93
Xgboost	94	94	98	90	94

3.4 Comparison of this Work with Existing Studies

While this paper evaluated similar algorithms as [17], direct comparisons are limited due to inherent differences in data, evaluation metrics, and experimental settings. However, a broader analysis encompassing works by [25], [26] and [27] reveals valuable insights. K-Nearest Neighbors (K-NN) consistently emerges as the top performer across studies, highlighting its effectiveness in this specific task. While Xgboost demonstrates strong performance, slight variations may occur depending on the study context. Conversely, Logistic Regression consistently exhibits the lowest performance across various studies, suggesting its limitations for this specific data and task type. Finally, Decision Tree Classifier (DTC), Adaboost, and Random Forest (see Figures 13-14, 18-20 and 21-23) exhibit variable performance across studies, potentially attributable to differing experimental settings. A publication by [25] in similar works, noted a yet overwhelming performance by the Logistic regression classifier. In their publication, Sensitivity and Precision scores were the evaluation metrics. The publication evaluated five (5) algorithms which were; Decision Tree, K-Nearest Neighbour, Logistic Regression, Random Forest and Naive Bayes. Out of all the classifiers, the K-NN classifier stood out once again with sensitivity and precision scores respectively of (81.19% and 91.11%) (85.86% F1-score). Followed closely by the Random Forest Classifier with an F1-score of 83.60%, 82.05% for the Decision tree classifier and 77.35% for the Logistic regression classifier. This confirms the limitations with the Logistic regression model discussed in this paper in section 3.4. [26] in their paper, also noted the following results with an imbalanced dataset where five (5) models were evaluated for their accuracy on a similar dataset. The models evaluated in the study were the Random forest algorithm, Decision Tree, Xgboost and Logistic regression. The ROC scores obtained for these models were: 100% for all the classifiers and 66% for the logistic regression classifier indicating the sub-minimal performance of this classifier and its behavior in handling imbalanced datasets. [27] in their comparative work on neural networks and ML algorithms in fraud detection obtained the following accuracy results for the models: K-NN (99.13%) (refer to Figures 9-11), Logistic regression (96.27%) (see Figures 6-8), Decision Tree (96.40%) (see Figures 18-20). It is paramount to note that their study compared four (4) ML algorithms with neural networks. It was also noted in this publication that the K-NN classifier was the top performing classifier with the highest accuracy and specificity scores and the Logistic regression, the least performing classifier in terms of accuracy and specificity scores. This is consistent with the performance of the classifiers compared in this study.

This study revealed K-Nearest Neighbors (K-NN) as the champion for fraud detection in this simulated dataset, topping the charts across average score, AUC, AUPRC, and F1-score. This stellar performance can be attributed to K-NN's inherent strengths: its flexibility in capturing non-linear fraud patterns, its ability to handle imbalanced datasets where fraudulent transactions are rare [28], and potentially less sensitivity to outliers compared to models like Logistic Regression. Logistic Regression, while simpler to interpret, might struggle with the complexities of fraud due to its assumptions of linearity and limitations in handling intricate data. While other models like XGBoost also showed promise, K-NN's potential for handling complex relationships between features without extensive pre-selection makes it a standout choice for this specific fraud detection task [29]. However, it's crucial to remember that these results are specific to the simulated data and chosen metrics. Real-world data and business priorities might influence the optimal model selection, so further evaluation is recommended before deployment. While the likes of K-NN and Xgboost emerge as the frontrunners, and the Logistic Regression classifier consistently underperforms in the studies discussed, further analysis considering specific application requirements is crucial for optimal model selection. Direct comparisons are limited due to inherent study variations, but broader insights can be gained from considering multiple studies.

4. Conclusion

Our investigation into machine learning's efficacy for fraud detection yielded promising results. K-Nearest Neighbors (K-NN) stood out as the most adept algorithm, demonstrably surpassing its competitors in accurately identifying fraudulent transactions within the simulated dataset (refer to Table 1). This dominance across evaluation metrics, including average score, AUC, AUPRC, and F1-score, suggests K- NN's superior ability to learn and classify complex patterns characteristic of fraudulent behavior. While limited by simulated data, this research offers a valuable stepping stone for real-world applications. Future endeavors will focus on validating K-NN's performance with real-world data and exploring other models to ensure generalizability in the continuous fight against fraud. This paves the way for machine learning to become a cornerstone of enhanced detection and prevention capabilities.

5. Author Contributions

Study conception and design, analysis and interpretation of results, and draft manuscript preparation: JJA and ICO. Data collection: JJA. All authors reviewed the results and approved the final version of the manuscript.

Funding

Many thanks to the NEPTTP (e-science) foundation for funding this research work

6. Acknowledgement

I would like to extend my most profound appreciation to the e-Science foundation for their financial support and unwavering assistance in this academic journey. This research would not have been possible without their generous funding, which enabled me to dedicate the necessary time and resources to the project.

7. Data Availability Statement

The datasets for this study can be found in Kaggle - (https://www.kaggle.com/datasets/maheshyaadav/credit-card-frauds-synthetic-dataset)

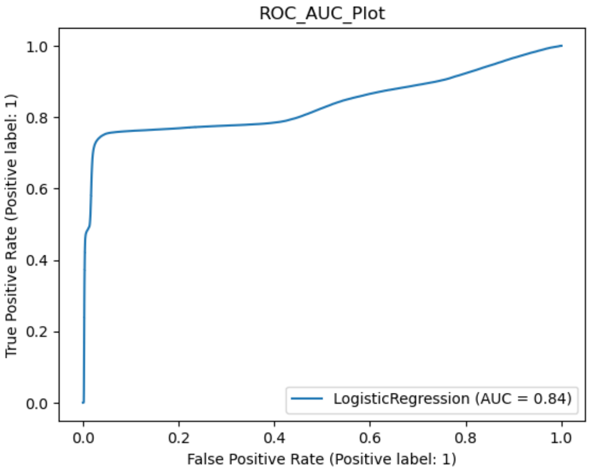


Figure 6: ROC curve reflecting a score of 84% for the logistic regression model.

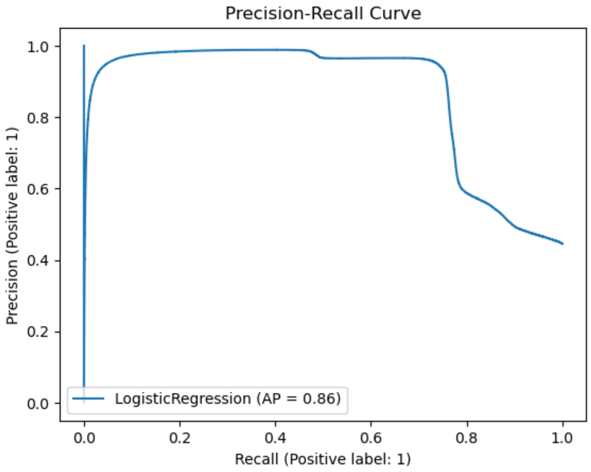


Figure 7: Precision recall curve for Lr showing a score of 85% reflecting its performance on the chosen dataset.

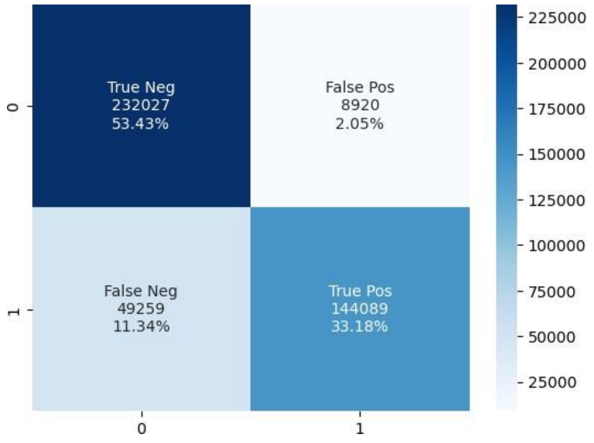


Figure 8: Confusion matrix output of Lr reflecting 58.4%, 33.18%, 11.34% and 2.05% respectively for TN,TP,FN and FP.

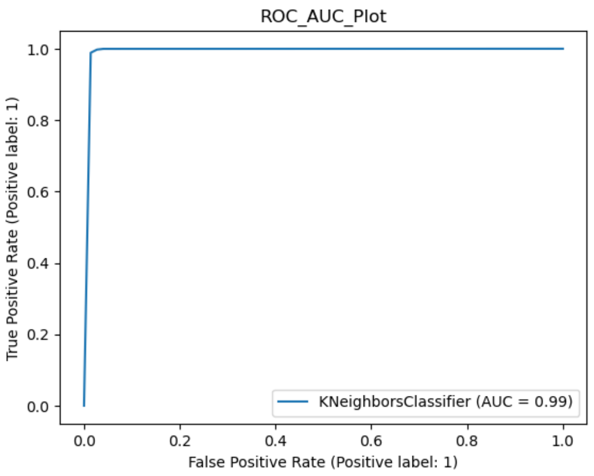


Figure 9: ROC curve reflecting a score of 99% for the K-Nearest Neighbour model.

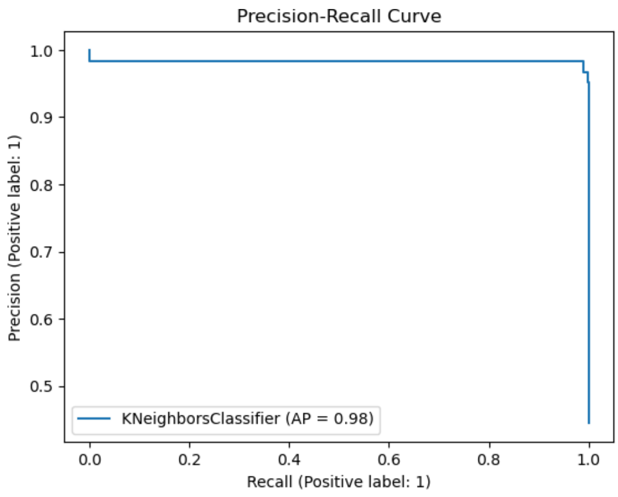


Figure 10: Precision recall curve for KNN showing a score of 98% reflecting its performance on the chosen dataset.

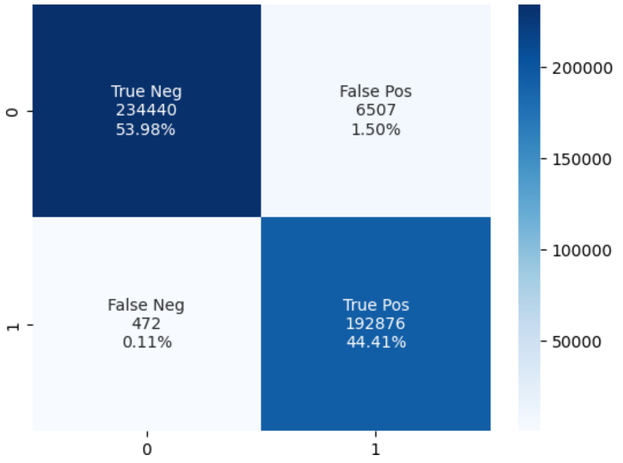


Figure 11: Confusion matrix output of KNN reflecting 53.96%, 44.41%, 0.11% and 1.50% respectively for TN,TP,FN and FP.

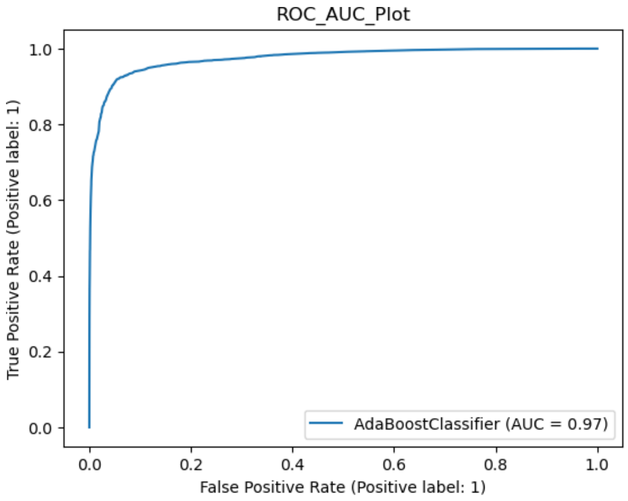


Figure 12: ROC curve reflecting a score of 97% for the Adaboost classification model.

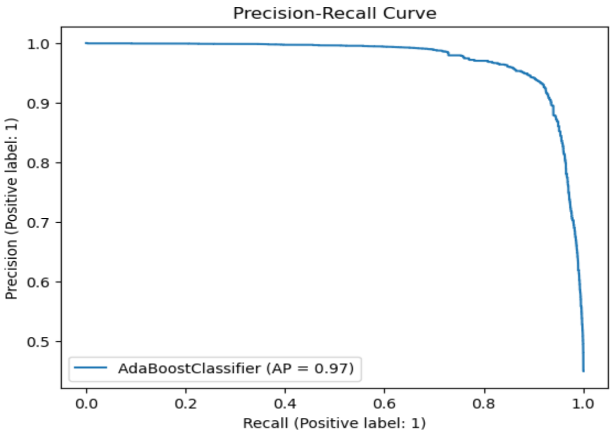


Figure 13: Recision recall curve for Adaboost showing a score of 97% reflecting its performance on the chosen dataset.

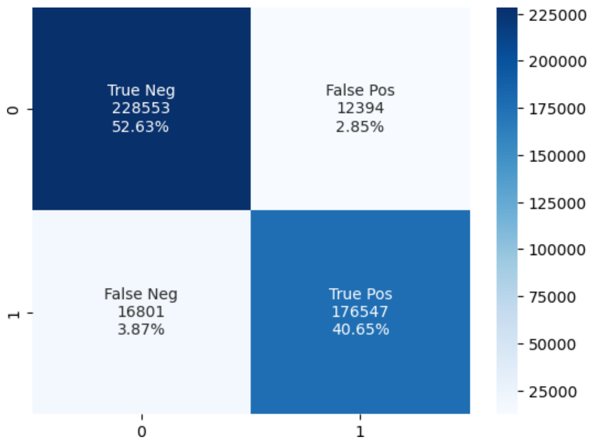


Figure 14: Confusion matrix output of Adaboost Al gorithm reflecting 52.63%, 40.65%, 3.87% and 2.85% respectively for TN,TP,FN and FP.

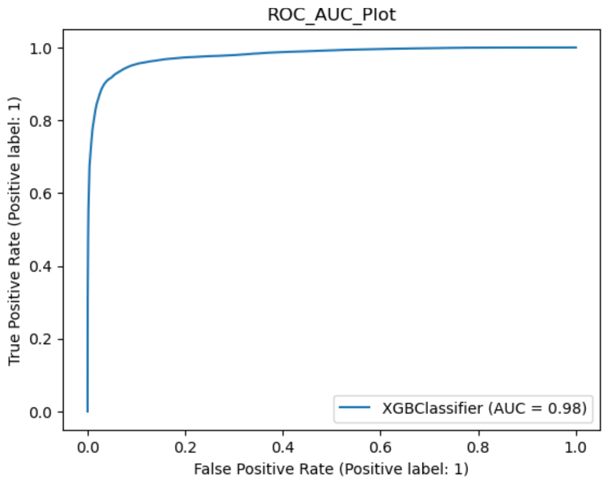


Figure 15: ROC curve reflecting a score of 98% for the Xgboost classification model.

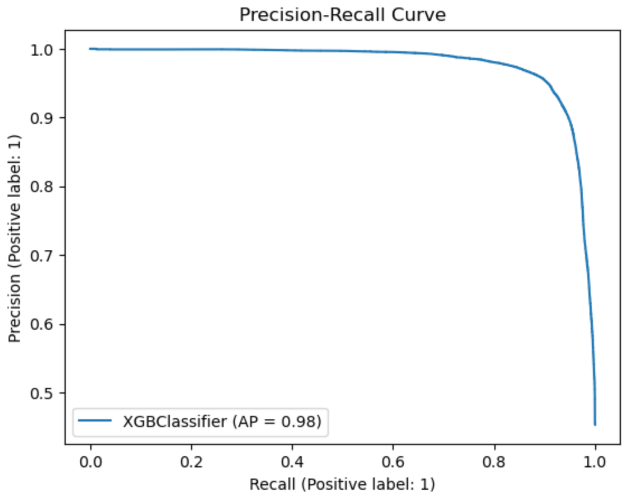


Figure 16: Precision recall curve for Xgboost show- ing a score of 98% reflecting its performance on the chosen dataset.

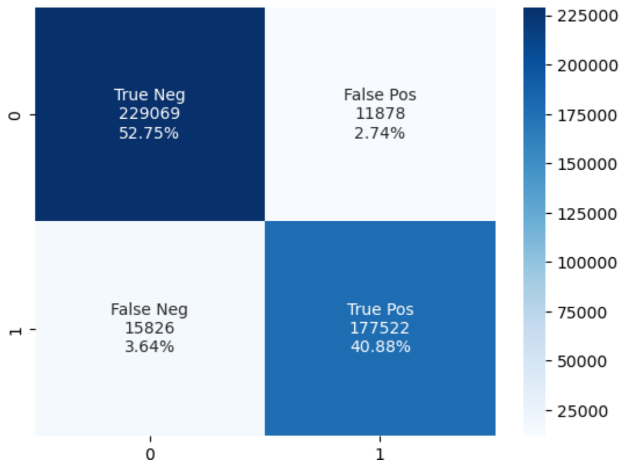


Figure 17: Confusion matrix output of Xgboost reflecting 52.75%, 40.88%, 3.64% and 2.74% respec- tively for TN,TP,FN and FP.

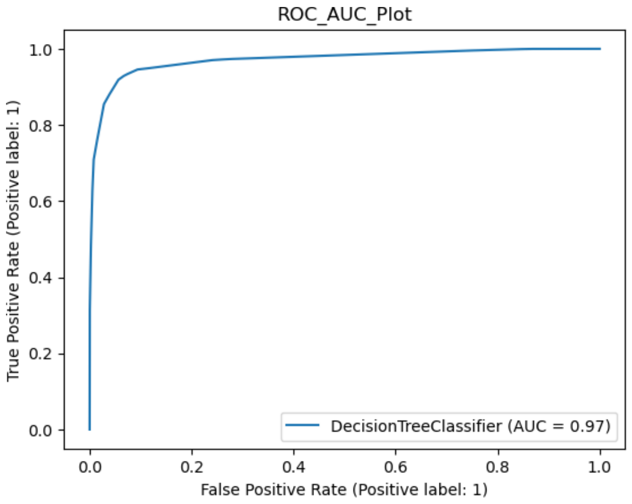


Figure 18: ROC curve reflecting a score of 97% forthe Decision Tree classification model.

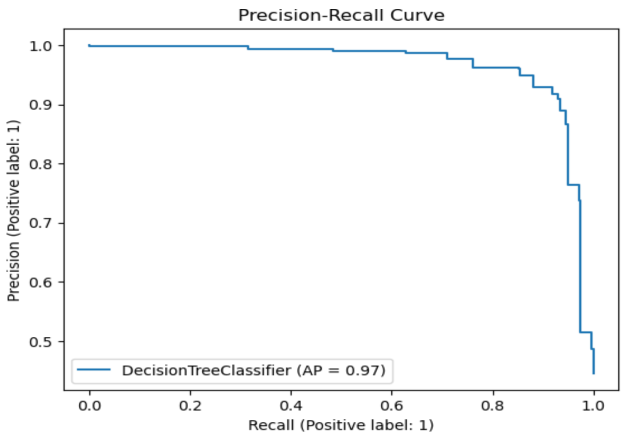


Figure 19: Precision recall curve for Decision Tree showing a score of 97% reflecting its performance on the chosen dataset.

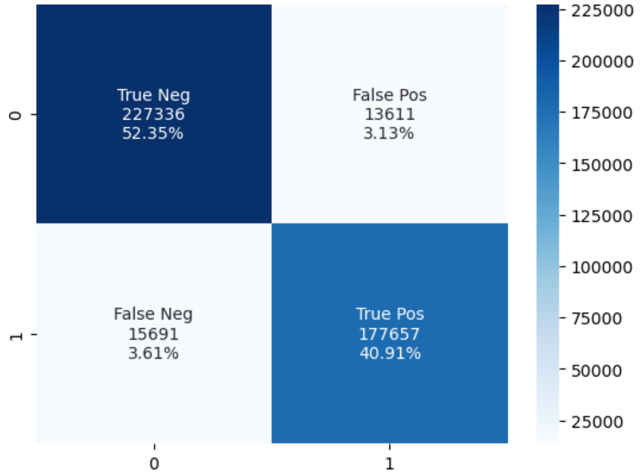


Figure 20: Confusion matrix output of a Decision Tree Classifier reflecting 52.35%, 40.91%, 3.61% and 3.31% respectively for TN,TP,FN and FP.

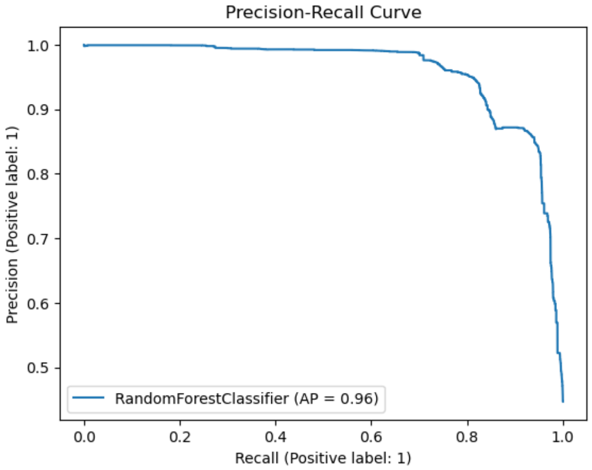


Figure 22: Precision recall curve for Rf showing a score of 96% reflecting its performance on the chosen dataset.

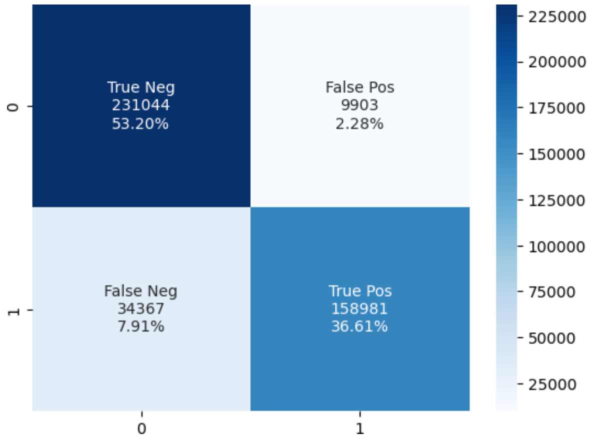


Figure 23: Confusion matrix output of Rf reflecting 53.20%, 36.61%, 7.91% and 2.28% respectively for TN, TP, FN and FP.

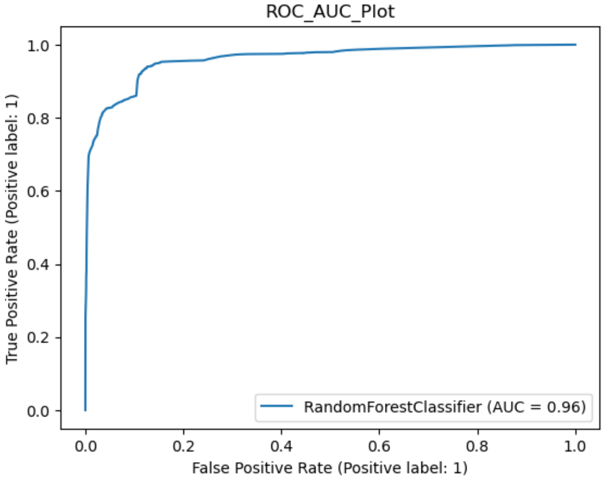


Figure 21: ROC curve reflecting a score of 96% for the Random Forest classification model.

List Of Abbreviations

Adaboost - Adaptive Boosting, AI - Artificial Intelligence, AUC - Area Under Curve, AUPRC - Area Under Precision Recall Curve, CCF - Credit Card Fraud , CPF - Card Present Fraud, CNPF - Card Not Present Fraud, EDA - Exploratory Data Analysis, PCA - Principal Component analysis, SVM - Support Vector Machine, ML - Machine Learning, ROC AUC-Score - Receiver Operating Characteristic & Area Under Curve, K-NN - K- Nearest Neighbour, Xgboost - Extreme Gradient Boost, Lr - Logistic Regression, DTC - Decision Tree Classifier, SMOTE - Synthetic Minority Oversampling Technique, CRT – Combined Resampling Technique.

References

[1] Dornadula, V.N. and Geetha, S., 2019. Credit card fraud detection using machine learning algorithms. Procedia computer science, 165, pp.631-641.

[2] Taghiyev, K.R., Rustamov, T.H. and Hasanzade, A.A., 2021. Analysis Of Payment Cards Fraud Transactions And Measures To Prevent Them. Economic innovations, 23(2 (79)), pp.172-184.

[3] Maniraj, S.P., Saini, A., Ahmed, S. and Sarkar, S., 2019. Credit card fraud detection using machine learning and data science. International Journal of Engineering Research, 8(9), pp.110-115.

[4] Shirgave, S., Awati, C., More, R. and Patil, S., 2019. A review on credit card fraud detection using machine learning. International Journal of Scientific & technology research, 8(10), pp.1217-1220.

[5] Mbakwe, A.B. and Adewale, S.A., Machine Learning and Applications: An International Journal (MLAIJ) Vol.9, No.4, December 2022

[6] Azhan, M. and Meraj, S., 2020, December. Credit card fraud detection using machine learning and deep learning techniques. In 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS) (pp. 514-518). IEEE.

[7] Saheed, Y.K., Baba, U.A. and Raji, M.A., 2022. Big data analytics for credit card fraud detection using supervised machine learning models. In Big data analytics in the insurance market (pp. 31-56). Emerald Publishing Limited.

[8] Bhavsar, A. and Patil, T., “Exploration of Various Machine Learning Algorithms”. In: Pratibha: International Journal of Science, Spirituality, Business And Technology (Ijssbt) (2021).

[9] Chen, J.I.Z. and Lai, K.L., 2021. Deep convolution neural network model for credit-card fraud detection and alert. Journal of Artificial Intelligence, 3(02), pp.101-112.

[10] Alfaiz, N.S. and Fati, S.M., 2022. Enhanced credit card fraud detection model using machine learning. Electronics, 11(4), p.662.

[11] Trivedi, N.K., Simaiya, S., Lilhore, U.K. and Sharma, S.K., 2020. An efficient credit card fraud detection model based on machine learning methods. International Journal of Advanced Science and Technology, 29(5), pp.3414-3424.

[12] Pudjihartono, N., Fadason, T., Kempa-Liehr, A.W. and O'Sullivan, J.M., 2022. A review of feature selection methods for machine learning-based disease risk prediction. Frontiers in Bioinformatics, 2, p.927312.

[13] Chandrashekar, G. and Sahin, F., 2014. A survey on feature selection methods. Computers & electrical engineering, 40(1), pp.16-28.

[14] He, S., Guo, F. and Zou, Q., 2020. MRMD2. 0: a python tool for machine learning with feature ranking and reduction. Current Bioinformatics, 15(10), pp.1213-1221.

[15] Zhao, Z., Anand, R. and Wang, M., 2019, October. Maximum relevance and minimum redundancy feature selection methods for a marketing machine learning platform. In 2019 IEEE international conference on data science and advanced analytics (DSAA) (pp. 442-452). IEEE.

[16] Tamir, A., Watson, E., Willett, B., Hasan, Q. and Yuan, J.S., 2021. Crime prediction and forecasting using machine learning algorithms. International Journal of Computer Science and Information Technologies, 12(2), pp.26-33.

[17] Singh, A., Ranjan, R.K. and Tiwari, A., 2022. Credit card fraud detection under extreme imbalanced data: a comparative study of data-level algorithms. Journal of Experimental & Theoretical Artificial Intelligence, 34(4), pp.571-598.

[18] Jain, Y., Tiwari, N., Dubey, S. and Jain, S., 2019. A comparative analysis of various credit card fraud detection techniques. International Journal of Recent Technology and Engineering, 7(5), pp.402-407.

[19] Wong, T.T. and Yeh, P.Y., 2019. Reliable accuracy estimates from k-fold cross validation. IEEE Transactions on Knowledge and Data Engineering, 32(8), pp.1586-1594.

[20] Kovács, G., 2019. An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets. Applied Soft Computing, 83, p.105662.

[21] Bagga, S., Goyal, A., Gupta, N. and Goyal, A., 2020. Credit card fraud detection using pipeling and ensemble learning. Procedia Computer Science, 173, pp.104-112.

[22] Wang, P., Fan, E. and Wang, P., 2021. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. Pattern recognition letters, 141, pp.61-67.

[23] Tsangaratos, P. and Ilia, I., 2016. Comparison of a logistic regression and Naïve Bayes classifier in landslide susceptibility assessments: The influence of models complexity and training dataset size. Catena, 145, pp.164-179.

[24] Pranckevičius, T. and Marcinkevičius, V., 2017. Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification. Baltic Journal of Modern Computing, 5(2), p.221.

[25] Khatri, S., Arora, A. and Agrawal, A.P., 2020, January. Supervised machine learning algorithms for credit card fraud detection: a comparison. In 2020 10th international conference on cloud computing, data science & engineering (confluence) (pp. 680-683). IEEE.

[26] Ileberi, E., Sun, Y. and Wang, Z., 2021. Performance evaluation of machine learning methods for credit card fraud detection using SMOTE and AdaBoost. IEEE Access, 9, pp.165286-165294.

[27] Dighe, D., Patil, S. and Kokate, S., 2018, August. Detection of credit card fraud transactions using machine learning algorithms and neural networks: A comparative study. In 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA) (pp. 1-6). IEEE.

[28] García, V., Mollineda, R.A. and Sánchez, J.S., 2008. On the k-NN performance in a challenging scenario of imbalance and overlapping. Pattern Analysis and Applications, 11, pp.269-280.