







Article

Unmasking Banking Fraud: Unleashing the Power of Machine Learning and Explainable AI (XAI) on Imbalanced Data

S. M. Nuruzzaman Nobel ¹, Shirin Sultana ¹, Sondip Poul Singha ¹, Sudipto Chaki ¹,
Md. Julkar Nayeem Mahi ², Tony Jan ³, Alistair Barros ⁴ and Md Whaiduzzaman ^{3,4,*}

¹ Department of Computer Science and Engineering, Bangladesh University of Business and Technology, Dhaka 1216, Bangladesh; smnuruzzaman712@gmail.com (S.M.N.N.); shirinsultana596@gmail.com (S.S.); sondippingsingh@gmail.com (S.P.S.); sudiptochakibd@gmail.com (S.C.)

² Department of Software Engineering, Daffodil International University, Dhaka 1207, Bangladesh; mahi.1992@gmail.com

³ Design and Creative Technologies, Torrens University, Brisbane, QLD 4006, Australia; tony.jan@torrens.edu.au

⁴ School of Information Systems, Queensland University of Technology, Brisbane, QLD 4000, Australia; alistair.barros@qut.edu.au

* Correspondence: wzaman@juniv.edu

Abstract: Recognizing fraudulent activity in the banking system is essential due to the significant risks involved. When fraudulent transactions are vastly outnumbered by non-fraudulent ones, dealing with imbalanced datasets can be difficult. This study aims to determine the best model for detecting fraud by comparing four commonly used machine learning algorithms: Support Vector Machine (SVM), XGBoost, Decision Tree, and Logistic Regression. Additionally, we utilized the Synthetic Minority Over-sampling Technique (SMOTE) to address the issue of class imbalance. The XGBoost Classifier proved to be the most successful model for fraud detection, with an accuracy of 99.88%. We utilized SHAP and LIME analyses to provide greater clarity into the decision-making process of the XGBoost model and improve overall comprehension. This research shows that the XGBoost Classifier is highly effective in detecting banking fraud on imbalanced datasets, with an impressive accuracy score. The interpretability of the XGBoost Classifier model was further enhanced by applying SHAP and LIME analysis, which shed light on the significant features that contribute to fraud detection. The insights and findings presented here are valuable contributions to the ongoing efforts aimed at developing effective fraud detection systems for the banking industry.

Keywords: fraud detection; machine learning; logistic regression; decision tree; SVM; XGBoost; oversampling SMOTE; SHAP analysis; LIME analysis



Citation: Nobel, S.M.N.; Sultana, S.; Singha, S.P.; Chaki, S.; Mahi, M.J.N.; Jan, T.; Barros, A.; Whaiduzzaman, M. Unmasking Banking Fraud: Unleashing the Power of Machine Learning and Explainable AI (XAI) on Imbalanced Data. *Information* **2024**, *15*, 298. <https://doi.org/10.3390/info15060298>

Academic Editors: Christos Michalakelis, Mara Nikolaidou and Evangelia Filiopoulou

Received: 1 May 2024
Accepted: 9 May 2024
Published: 23 May 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Banking systems in the current era are extremely important to global economies because they enable financial transactions and supply key services to consumers and businesses alike. However, fraudulent behaviors can result in significant financial losses and destroy public trust in such institutions. Currently, criminals find various ingenious ways to commit fraud [1]. The methods used by fraudsters are constantly evolving with technological advancements, making it difficult for conventional fraud detection systems to keep up. Fraudsters increase the intricacy and sophistication of their schemes by constantly updating their methods to take advantage of new vulnerabilities. Traditional rule-based fraud detection techniques attempt to discover unusual fraud patterns when working with imbalanced datasets. This study was driven by the banking industry's desire for more advanced fraud detection technology. Financial institutions lose revenues due to conventional fraud detection systems' inability to keep up with fraudsters' dynamic nature. We can overcome conventional fraud detection constraints by combining machine learning (ML) and explainable artificial intelligence (XAI).

Some most commonly used methods for reducing bank fraud are artificial intelligence [2], machine learning [3], LSTM [4], XGBoost [5], and so on. This study explores how machine learning can analyze financial transactions. When compared to more conventional rule-based systems, ML models provide a significant improvement in both accuracy and efficiency. In addition, XAI methods improve openness by detailing the reasoning behind ML models' choices and allowing human analysts to verify and analyze the outcomes. The success of machine learning models is highly dependent on their training accuracy and diversity. Skewed results and inefficient fraud detection could result from using inaccurate or missing data. Also, it may be difficult for non-technical stakeholders to completely grasp and accept the generated predictions from traditional ML models due to their inherent complexity.

Using machine learning and explainable AI (XAI) on imbalanced data, we try to overcome the shortcomings of current fraud detection systems. This study introduces a novel approach to enhance the precision of fraud detection models while also ensuring that the models' decision-making processes are transparent and interpretable. Developing robust fraud detection systems driven by machine learning algorithms is crucial, as they adapt to evolving fraud patterns, minimize false negatives, and detect fraudulent behaviors that elude rule-based systems. Machine learning classifiers have evolved into crucial tools for detecting fraudulent transactions, account takeovers, identity theft, and other types of financial crime used to detect fraudulent activity in banking. We have used SHAP and LIME [6] analyses to provide a clear explanation of how the input properties affect the predictions of the model. The main problem with synthetic financial datasets is that they are imbalanced. To handle an imbalanced dataset, we use some data preprocessing techniques, such as oversampling SMOTE [7], MinMaxScaler [8], and LabelEncoder [9], which can boost the efficiency of the machine learning methods.

By using XAI methods, we understand the logic behind how machine learning models arrive at their conclusions. This openness allows investigators and stakeholders to spot any weaknesses, biases, or vulnerabilities in the system, allowing for the constant fine-tuning and development of the fraud detection process. Our findings demonstrate the effectiveness and interpretability of our suggested methods for identifying and explaining fraudulent transactions in asymmetric datasets. Our goal is to provide a comprehensive framework for developing accurate and trustworthy financial fraud detection systems by integrating the strengths of machine learning, XAI, and imbalanced data handling approaches.

Below are listed the contributions of this study:

- Proposal of an efficient algorithm: Using cutting-edge machine learning techniques, this study proposes a reliable and efficient approach for dealing with class imbalance in the banking fraud detection domain.
- Dataset balancing: To mitigate the class imbalance issue, the SMOTE algorithm was used to balance the dataset.
- Evaluation of Machine Learning Algorithms: To determine fraudulent transaction methods, four machine learning techniques, including Logistic Regression, XGBoost Classifier, Decision Tree, and SVM, were analyzed.
- Model explainability: In order to interpret and explain the model's predictions, SHAP and LIME analysis techniques were employed.

This study aims to close the gap between machine learning's benefits and drawbacks in fraud detection by incorporating XAI approaches. With the proposed method, the problems caused by unbalanced data in banking fraud detection can be successfully resolved. The purpose of merging machine learning and XAI is to increase the precision and efficacy of fraud detection models on imbalanced data. This strategy enables financial organizations to more successfully uncover banking fraud by enabling them to make decisions based on predictions produced by the ML-XAI-based model. In addition, by utilizing XAI approaches, we solve the issues related to the black box nature of ML models by providing transparency and interpretability. This raises stakeholder confidence in the system and equips them with the information they need to make decisions based on fraud signs.

The remainder of this paper is organized as follows: Section 2 discusses earlier research to detect fraudulent activity in banking sector. Section 3 provides our state-of-the-art (SOTA) methodologies equipped with ML-XAI techniques. In Section 4, the performance assessment is presented to validate our proposed work. Section 5 highlights the interpretability of our suggested methods with the help of XAI-based algorithms. Finally, Section 6 brings the paper to a conclusion with possible extensions in the future research directions.

2. Literature Review

A multitude of scholars have endeavored to tackle the pervasive issue of banking fraud detection through the application of both ML- and non-ML-based methodologies. Varmedja et al. [10] devised an ML-based method for identifying fraudulent transactions on credit cards. SMOTE was used for data balancing. The RF algorithm has been shown to be superior in performance, with a detection rate of 99.96%. The NB method had a 99.23% success rate, while the MLP method had a 99.93% success rate. However, two days of data on credit card transactions from European consumers may provide some insight. Due to its short-term nature and limited geographic emphasis, it reduces the model's efficiency in detecting fraudulent activity in general. Pech [11] uses machine learning to detect fraud in mobile money transfers by treating it as a binary classification task. Oza et al. [12] used several ML techniques on a labeled dataset of payment transactions. The study [13] provides an overview of the most recent changes in the financial crime scene and highlights the implementation challenges faced by existing and new graph solutions. Some research [14] used only classifier optimization based on an SVM, whereas others [15,16] used Logistic Regression, SVM, Random Forest, and ANN. From those papers [11,12,16], Logistic Regression showed better results than the other methods. In [17,18], a simulator named PaySim was used to evaluate several methods to prevent fraud based on thresholds [19] on a mobile money service. The authors of [20] employed a decentralized data mining methodology to tackle the issues of imbalanced credit card distributions. In [21], we see a fraud detection algorithm that employs a proactive approach to detect fraud without using any fraudulent past cases. A hybrid model that combines supervised and unsupervised techniques to detect fraud was proposed in a study [22], but the results were inconclusive. The authors of [23] suggest an ensemble approach for detecting financial fraud using machine learning algorithms. The study presented a thorough method for identifying financial fraud using machine learning algorithms [24]. However, there is no explanation of the feature selection procedure. An analysis was performed by Khatri et al. [25] to evaluate the efficiency of detecting credit card fraud utilizing various methods, such as DT, KNN, LR, RF, and NB. The experimental findings showed accuracies of 85.11%, 91.11%, 87.5%, 89.77%, and 6.52%, respectively, on a highly imbalanced dataset. In evaluating the ECCFD dataset, Hema [26] utilized RF, LR, and CatBoost but overlooked the issue of imbalanced classes. Despite this, Hema found that RF produced outstanding results with an accuracy of 99.95%. Meanwhile, Kumar et al. [27] achieved 90% accuracy using RF. Awoyemi et al. [1] explored several ML approaches; additionally, a hybrid sample approach was used to deal with the imbalanced data. Accuracy levels of 97.92%, 54.86%, and 97.69% were achieved by the NB, LR, and KNN, respectively. The choice to omit a feature selection technique was made by the authors. A sophisticated payment card fraud detection system was developed by Manjeevan et al. [28] using the GA for feature selection and aggregation. The GA-ANN achieved 81.82% accuracy, the GA-DT achieved 81.97% accuracy, and the GA-RF achieved just 77.95% accuracy. Puh et al. [29] address the issue of imbalanced class distribution in the dataset by applying SMOTE and evaluating the performance of various algorithms, including RF, SVM, and LR. Results showed an AUC of 91.14% for static learning and 91.07% for incremental learning, with corresponding average precision values of 73.37% and 84.13%, respectively.

Table 1 provides a comprehensive overview of various research studies focused on the identification of fraudulent transactions in the banking sector.

Table 1. This table compares the performance and limitations of existing approaches.

Methods	Contribution	Limitation
RF, NB, MLP [10]	Assessed the three ML algorithms’ credit card fraudulent transactions detection	The short-term nature of the dataset may reduce the generalization ability.
Hybrid Model [22]	A novel hybrid framework that seamlessly integrates supervised and unsupervised methodologies to enhance the accuracy of fraud detection.	
Logistic Regression and XGBoost [23]	An ensemble machine learning framework for detecting fraudulent credit card transactions leveraging an imbalanced dataset.	Lacks an explanation of the feature selection process.
KNN,DT, LR, RF, and NB [25]	Used a severely imbalanced dataset to evaluate ML algorithms for detecting fraudulent transactions with credit cards.	Poor classification performance.
RF, LR, and Category Boosting (CatBoost) [26]	The performance of RF, LR, and Category Boosting (CatBoost) is evaluated to obtain the most efficient model for evaluating the ECCFD dataset.	Did not tackle the class imbalance issue.
NB, LR, and KNN [1]	The imbalanced dataset was addressed through hybrid sampling.	Poor classification performance of LR. Also, did not implement the feature selection method.
GA-RF, GA-ANN, GBT, and GA-DT [28]	Applied Genetic Algorithms (GA) for feature selection and aggregation, along with multiple machine learning methods, to evaluate the approach’s effectiveness.	Consumer transactional aspects must be assessed using data from several regions, although this study concentrates only on Malaysian financial transactions.

3. Research Methodology of Our Framework

The core objective of our work revolves around effectively distinguishing between non-fraudulent and fraudulent financial transactions, leveraging advanced techniques in the field of financial transaction analysis. The diagram depicted in Figure 1 provides an extensive overview of our research process, including the phases of data collection, data preprocessing, data analysis, algorithm implementation, results, and discussion.

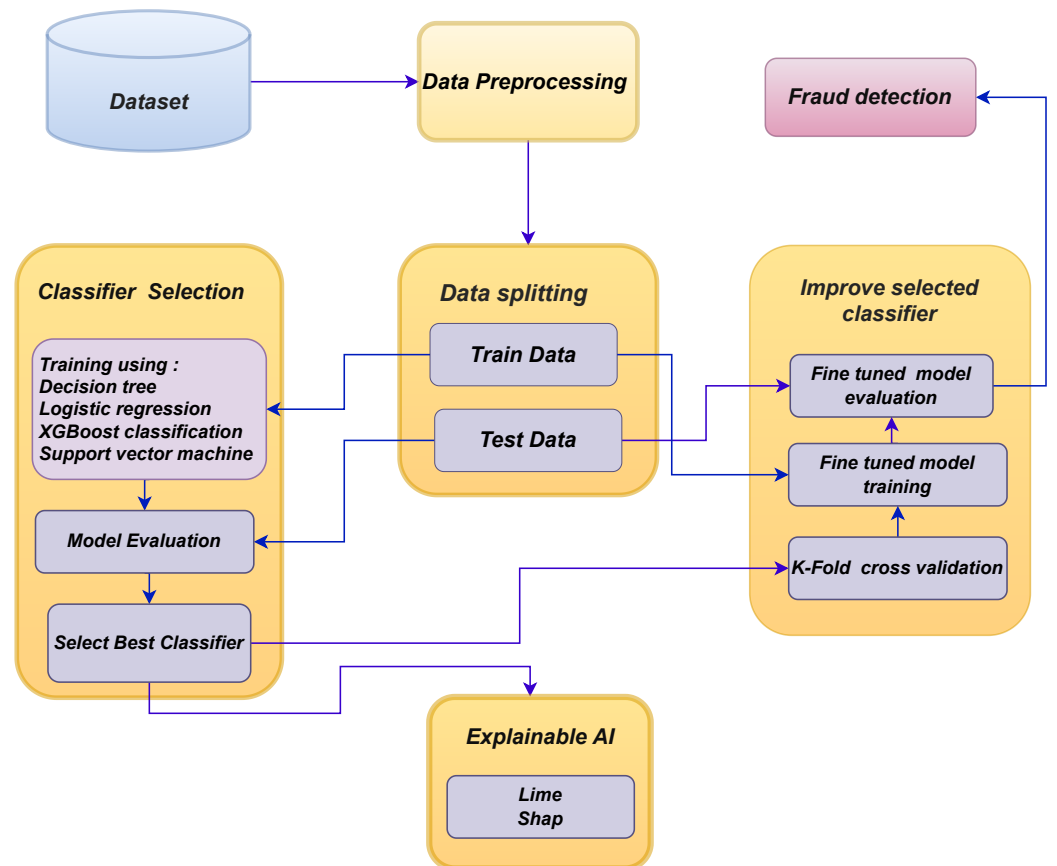


Figure 1. Our proposed framework for banking fraud detection on imbalanced dataset.

3.1. Dataset Explanation

Public financial data needs to have adequate availability, especially in financial transactions. To tackle this issue, we collected a synthetic dataset from Kaggle [30], created with the help of the simulator PaySim. PaySim compiles information from the confidential dataset to create a fake one and then injects fraudulent activity to test how well fraud detection systems perform. Initial logs came from the international company offering the mobile banking service, now available in over 14 countries.

The dataset shows a class imbalance with 6 million transactions, with only 8213 (0.14%) being fraudulent, as shown in Figure 2.

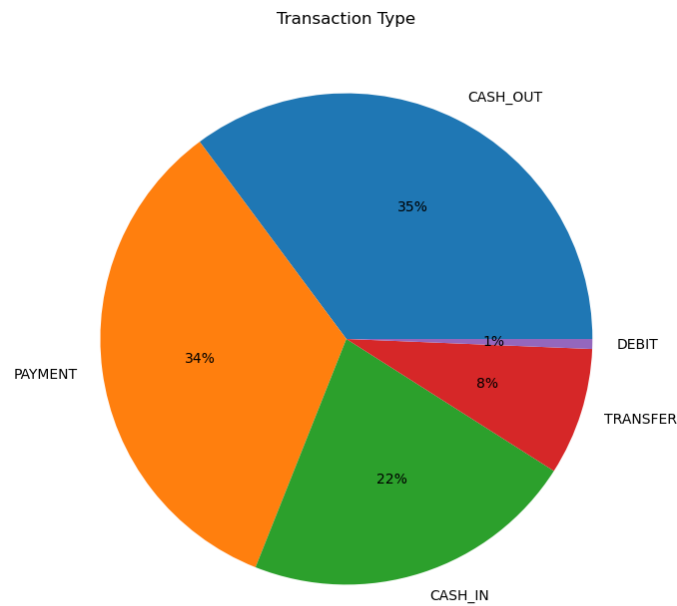


Figure 2. Transaction types.

3.2. Data Preprocessing

Figure 3 illustrates a box plot characterizing the variable “step” distribution across the fraud classifications, with 0 denoting the non-fraudulent class and 1 indicating fraudulent transactions.

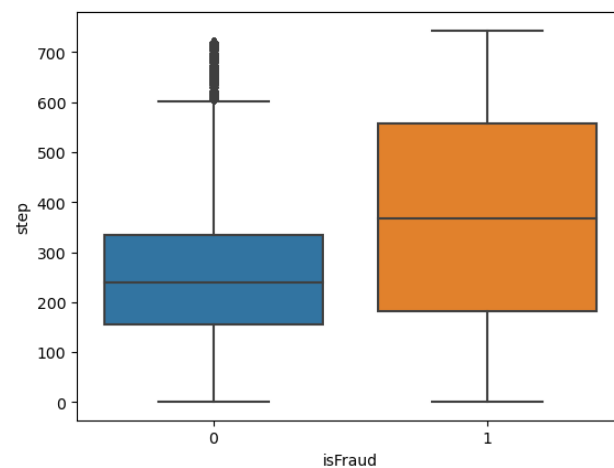


Figure 3. Step vs. fraud classification.

The width of the box represents the range of the “step” variable for all cases that are not fraudulent, as well as the interquartile range. With a shorter range, the values of “step” for non-fraudulent cases are more likely to cluster together.

As the box of 1 is larger than the box of 0, the values of “step” in fraudulent cases are more widely distributed and have a greater range than in non-fraudulent cases. We can infer that the variable “step” tends to have a narrower range and less variability for non-fraudulent instances by comparing the sizes of the two boxes in the box plot. Fraudulent instances demonstrate a more excellent range and variability in the “step” variable. The difference in box sizes implies that the variable “step” may have a distinct distribution or behavior for fraudulent and non-fraudulent cases.

Several preprocessing steps were taken to prepare the data for analysis and modeling. One approach to addressing data imbalance is the sampling technique, which creates false representative samples of minority classes.

Table 2 exhibits the data quantities before and after the implementation of resampling techniques. When undersampling is used, around 75% of the original data is discarded. This poses a significant risk of distorting results, limiting insights, compromising the reliability of any analysis or decision-making process, and resulting in a notable bias within the dataset. The discarded data may include various perspectives or outliers that could impact the overall analysis. On the other hand, utilizing SMOTE includes approximately 50% more data, which effectively balances the dataset (Figure 4). Adding approximately 50% more data can effectively address issues such as class imbalance and enhance the overall performance of machine learning models. By using a more extensive and diverse dataset, models can learn a broader range of patterns, enabling them to make more accurate predictions when faced with new data. As a result, the models become more reliable and robust in their performance. Augmenting the dataset offers a broader range of examples for the model to learn from. This helps to decrease the chances of overfitting and enhances the model’s efficiency. That is why, for this task, the technique employed was SMOTE, to address the imbalanced data issue.

Table 2. The number of instances before and after sampling

Data Splitting	Original	SMOTE	RandomUnderSampler
Training dataset	5090096	10167051	13140
Test dataset	1272524	2541763	3286

In addition, the categorical variables in the dataset needed to be encoded as numerical values. Categorical features were encoded as integers using the LabelEncoder method for use with a wide range of algorithms. Another preprocessing tool, MinMaxScaler, scales numerical properties to a specified range, often between 0 and 1. This guarantees that all features are roughly the same size and keeps the smaller ones from being overshadowed by their larger counterparts. These data preprocessing methods ensure that numerical and categorical features are handled correctly and translated into an analysis-ready format.

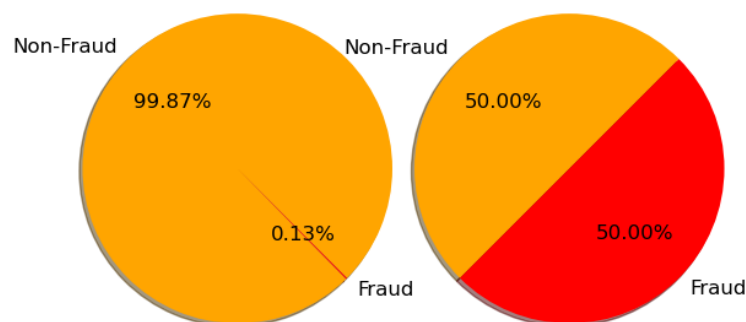


Figure 4. SMOTE analysis based on percentile to detect balance and imbalanced data.

3.2.1. SMOTE-Based Data Balancing

To increase the diversity of the dataset, we apply SMOTE (Synthetic Minority Over-sampling Technique), which interpolates existing samples, creating artificial samples for minority groups that are similar but not identical in the fraud class. This balancing technique is widely recognized to enhance the performance of machine learning algorithms, leading to improved results.

In Algorithm 1, the process of SMOTE is described. The initial two lines depict the input, output, and variable initialization. The operational process of SMOTE for data balancing is demonstrated in lines 1 through 9. The SMOTE technique utilizes the K-nearest neighbor pattern to create artificial data. It selects minority class data points at random and then identifies the K-nearest neighbors within the dataset. It generates synthetic data by selecting the K-nearest neighbors from the randomly chosen data.

Algorithm 1: SMOTE algorithm

Input: M: samples from the minority class; N: number of synthesized samples; K: number of nearest neighbors to consider; n: number of synthesized samples to generate per minority class sample.

Output: S: synthetic minority class samples.

```

1 Initialize S as an empty list;
2 Calculate the n:  $n = N / \text{length}(M)$ ;
3 for  $m_i \in M$  do
4   Find the K-nearest neighbors of  $m_i$  using a distance metric such as Euclidean distance.
5   while  $j \neq n$  do
6     Choose a random neighbor  $m_n$  from the K-nearest neighbors;
7     Produce an artificial sample  $m_s$  through linear interpolation between  $m_i$  and  $m_n$ ;
8      $m_s = m_i + \alpha \times (m_n - m_i)$ .
9   Append  $m_s$  to S.
```

Here, N = number of minority class samples and k = the number of nearest neighbors; then, the time complexity is approximately $\mathcal{O}(N \times k)$.

3.2.2. LabelEncoder

The LabelEncoder is a highly effective method utilized in data preprocessing to transform categorical variables into a numerical format. This is achieved by assigning a distinct numerical label to each category present in the input. The LabelEncoder facilitates seamless conversion between the initial categorical data and its numerical representation. The utilization of a particular methodology is prevalent when handling attributes such as class labels or target variables in the context of classification predicaments. Within our dataset, we have three categorical features that have been converted to numeric data through the use of LabelEncoder. The encoding process is facilitated by utilizing the scikit-learn library [31], a python package that offers the LabelEncoder class.

3.2.3. MinMaxScaler

The MinMaxScaler technique is a widely used approach for feature scaling. Its primary function is standardizing and normalizing numerical attributes within a given dataset. Scaling numerical features of a dataset to a predetermined range, usually within the confines of 0 and 1, is known as re-scaling. This technique is highly advantageous in scenarios where the data exhibit disparate scales and ranges. Scaled data can effectively mitigate the issue of certain features overpowering others in the machine learning algorithm, especially

in cases with a significant disparity in the feature ranges. The formula for MinMaxScaler is as follows:

$$x_{scaledi} = \frac{(x_i - x_{min})}{(x_{max} - x_{min})} \tag{1}$$

where

$x_{scaledi}$ = the scaled value of a feature;

x_i = the feature’s initial value is denoted by this symbol;

x_{min} = dataset’s minimum feature value;

x_{max} = dataset’s maximum feature value.

Our dataset comprises eight distinct numerical features, each exhibiting significant variation in their original values. The MinMaxScaler is employed to effectuate a transformation of these features into a range from 0 to 1. The scikit-learn library [32] is also utilized for the purpose of scaling.

3.3. Correlation Analysis

Correlation analysis is a statistical method that aims to understand the relationship between two or more variables [33]. The computational algorithm performs an analysis that yields a correlation coefficient bounded by -1 and $+1$. The underlying implication is that a positive coefficient signifies a positive correlation between the variables, a negative coefficient implies a negative correlation, and a coefficient of zero signifies the absence of correlation between the variables. Noting that a correlation exists between two variables does not prove that an alteration in one variable results in a modification in the other. Correlation analysis can be utilized to identify the fundamental properties of the dataset. Figure 5 shows the dataset correlation analysis findings.

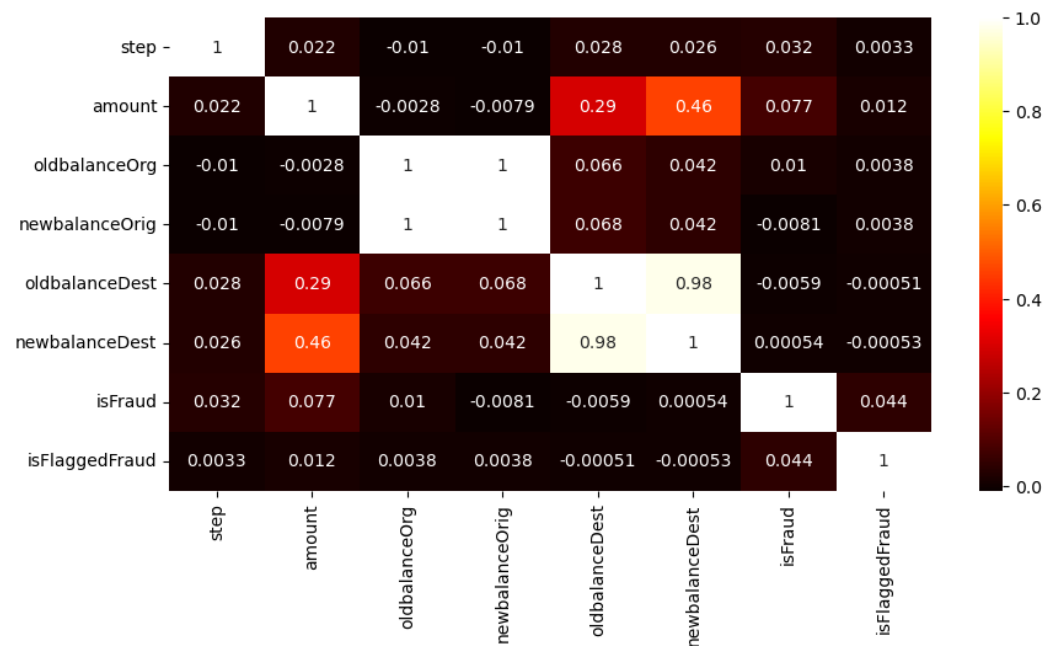


Figure 5. Correlation analysis.

The heatmap serves as a visual depiction of the correlation matrix, wherein each cell symbolizes the correlation existing between two distinct variables. The correlation values are visually represented through a color-coded scheme, encompassing a range from -1 to 1 . This scheme employs distinct colors such as black to denote negative correlation, white to signify positive correlation, and red to indicate the absence of correlation.

3.4. Baseline Architectures

Logistic Regression

Logistic regression predicts binary outcomes (e.g., yes/no, 1/0, true/false) based on multiple features [34]. The posterior probability distribution $P(Y|X)$ represents the target variable and characteristics. Given X , they return a probability distribution over Y . Logistic regression uses a logistic or sigmoid function to imitate binary outcomes. Continuous inputs were converted using a logistic function into probability values between 0 and 1. The logistic function determines the parameters best suited to the nonlinear function, like the sigmoid activation function [1]. The equation for the Logistic Regression is typically expressed as follows:

$$y = \frac{1}{1 + e^{-(z)}} \tag{2}$$

$$z = \beta_0 + \beta_1x_1 + \dots + \beta_nx_n \tag{3}$$

where

y = predicted value;

x = input value;

β_0 = bias term;

β_n = the coefficient of the input variable x_n .

The sigmoid function transforms the linear combination of β_0 and β_nx into a probability value between zero and one. The coefficients are estimated using maximum likelihood estimation or other optimization techniques. A threshold can then be applied to classify the observations into their respective binary classes, typically using 0.5 as the cutoff.

The Logistic Regression’s interpretability helps banks and financial organizations identify fraud-detection factors (See Figure 6). This model can handle big datasets and is computationally efficient, which is significant in banking due to the enormous number of transactions. The Logistic Regression’s ability to handle binary outcomes, combine numerous characteristics, produce interpretable results, generate probabilistic forecasts, and handle large datasets makes it an efficient banking fraud detection method.

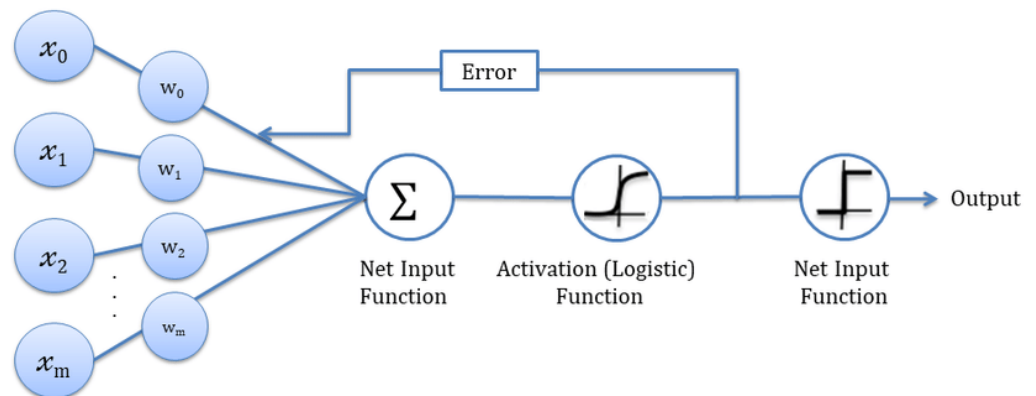


Figure 6. Logistic regression model architecture [35].

3.5. XGBoost

The XGBoost (Extreme Gradient Boosting) algorithm is a variation of the gradient boosting framework, where multiple weak prediction models are trained and combined to form a robust prediction model [36], which was developed by Chen et al. in 2016 [37]. Each model aims to lower the residual error and reduce the preceding model’s error. Thus, each subsequent sequential model receives an updated residual error value for boosting. A visual representation of the XGBoost architecture is depicted in Figure 7.

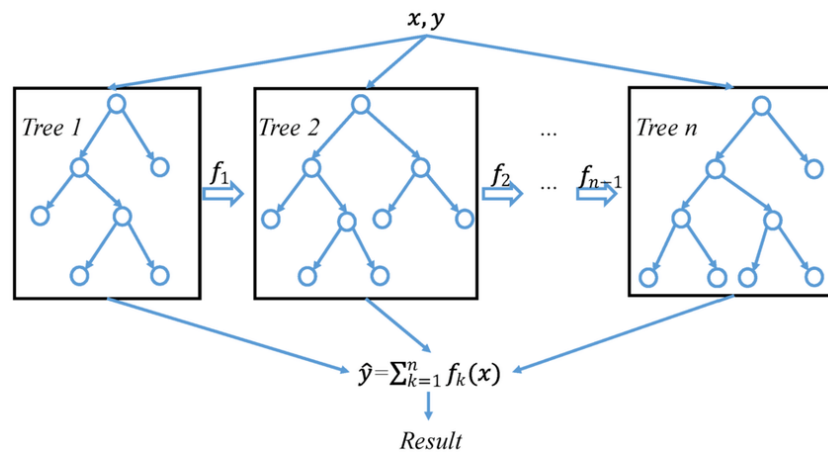


Figure 7. An overview of the XGBoost architecture [38].

Algorithm 2 demonstrates how XGboost operates on the provided dataset. The first two lines of this algorithm indicate the inputs and variable initialization. The dataset is split into testing, training, and validation datasets in lines 2 to 9. Lines 10 to 17 demonstrate the deployment of the model.

Algorithm 2: Detecting fraud with XGboost

```

Input: D: balanced dataset
1 Initialize parameters (e.g., learning rate = 0.001, maximum depth (max_depth) = 7,
  n = number of estimators (n_estimators) = 341, subsample = 0.9,  $\gamma = 0.5$ );
2 Divide the dataset into a testing set and a training set;
3  $X_{train}$  : input variables of training dataset;
4  $X_{test}$  : input variables of testing dataset;
5  $Y_{train}$  : target variables of training dataset;
6  $Y_{test}$  : target variables of testing dataset;
7 Split the training set split into training and validation sets;
8  $X_{val}$ : input variables of validation dataset;
9  $Y_{val}$  : target variables of validation dataset;
10 Model = XGBClassifier (n_estimators);
11 Model = Model.fit( $X_{train}, Y_{train}$ );
12  $Y_{pred} = \text{Model.pred}(X_{test})$ ;
13 if  $Y_{pred} == 0$  then
14 |   transactions = legitmate;
15 else
16 |   transactions = malicious.
    
```

Here, N = total number of samples, t = total number of trees, d = total height of the trees, and x = total number of non-missing entries in the training set; then, the time complexity is roughly $\mathcal{O}(t \times d \times x \times (\log n))$. $\mathcal{O}(t \times d)$ is the time required to predict a fresh sample.

XGBoost is well-suited for banking fraud detection, as it effectively handles imbalanced datasets by penalizing misclassification errors more heavily for the minority class, allowing it to accurately predict fraudulent cases. Additionally, its parallelization capabilities and optimized tree-building process facilitate fast and efficient model training, making it feasible to process large volumes of banking transaction data. XGBoost’s ability to handle missing values and outliers further enhances its accuracy in detecting fraudulent activities.

3.6. Decision Tree

For both classification and regression issues, decision trees (DTs) are a go-to standard supervised machine learning model [39]. It is a model like a tree, with nodes and branches for selecting the predictor variable that will lead to the most consistent possible subsets of data concerning the target variable [40,41]. DTs are well-known for their ability to classify data with minimal effort and high dependability. Producing a DT requires a calculation of entropy and information gain (IG) [42]. In this work, the C4.5 algorithm developed by Quinlan [43] was utilized as a classifier. The algorithm recursively operates until a stopping requirement is met. To calculate the entropy of decision tree T , use the equation below:

$$H(T) = - \sum_{i=1}^{|Y|} p_i \log_2 p_i \tag{4}$$

In this equation:

$|Y|$ indicates the quantity of classes present in the dataset;

p_i indicates the proportion of data points belonging to a specific (i) class.

The information gain of decision tree T for attribute a can be calculated using the following equation:

$$IG(T, a) = H(T) - \sum_{v \in V} \frac{|T_v|}{|T|} H(T_v) \tag{5}$$

In this equation:

$IG(T, a)$ represents the information gain of decision tree T ;

and $|T_v|$ and $|T|$ indicate the number of instances in T_v and T , respectively.

3.7. SVM

Vapnik et al. are credited with the original implementation of Support Vector Machines (SVM) [44]. The core idea behind SVM is that data may be optimally classified by plotting the set on a hyperplane. The SVM method finds the hyperplane that separates the two groups the most, also called the maximum-margin hyperplane. A set of support vectors, or margin points, determines where exactly the hyperplane will lie [45].

By mapping the input data to a higher-dimensional space, kernel functions allow for more accurate linear separation. A visual representation of the structure of SVM is shown in Figure 8.

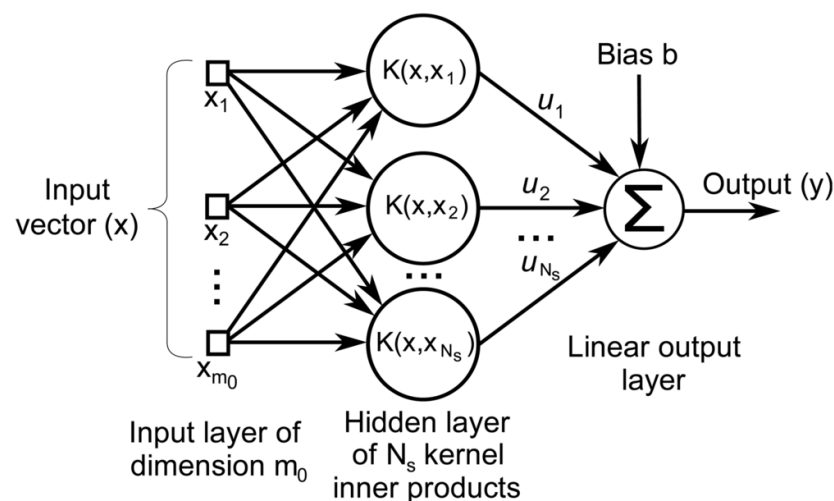


Figure 8. An overview of the SVM architecture [46].

The binary linear SVM classification algorithm is designed to determine the optimal hyperplane decision boundary by leveraging a training dataset. Depending on whether or

not the flawless classification of the training dataset is possible and desirable, optimality may be interpreted in the following ways:

- A hard margin optimality may be employed if the classes in the training dataset can be perfectly separated. Here, the decision boundary of the hyperplane is set such that it is maximally far away from the closest data point in the training set.
- A soft marginal optimality is implemented if completely accurate classification is not desired. Here, the choice of hyperplane is an adjustable compromise between increasing the distance to the next correctly identified training point and decreasing the misclassification rate.

The decision boundary is solely determined by the Support Vectors, which comprise a subset of the input vectors used during training. The input vector's allocated class is determined by the side of the decision boundary on which it lies. Classification using a linear Support Vector Machine is shown graphically in Figure 9 for both linearly and non-linearly separable classes.

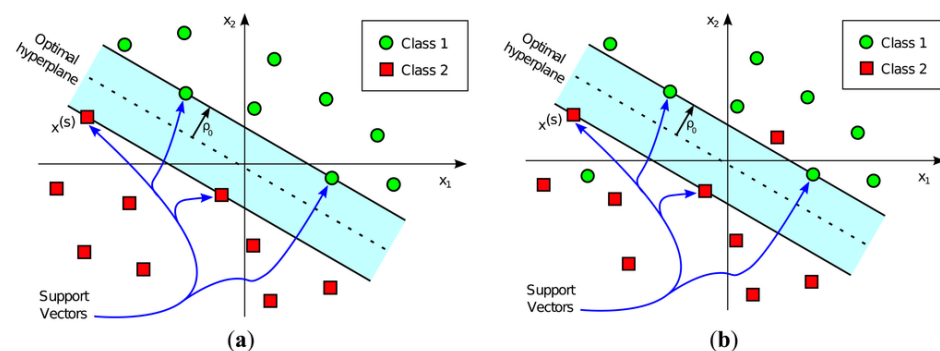


Figure 9. Support Vector Machine (SVM) classifier representations for (a) a pattern that can be linearly separated, where a hyperplane completely divides the space into green circles and red squares, and (b) a pattern that cannot be linearly divided, where no hyperplane divides the space into green circles and red squares [46].

4. Results and Discussion

4.1. Experimental Setup

Intending to obtain the best possible results from the training procedure, the experimental setup encompasses several orchestrated steps. We utilized SMOTE to balance the dataset to address the class imbalance issue. By creating synthetic samples by interpolating between real ones, the SMOTE method improves the representation of the minority class. After that, we made use of LabelEncoder to quantify categorical variables. By undergoing this transformation, machine learning algorithms are able to process and comprehend the categorical features, allowing for their incorporation into the fraud detection model. To further standardize the numerical characteristics of the dataset, we used MinMaxScaler. By bringing all feature values into the same range, this scaling method ensures that no one feature's larger values overshadow the training of the model.

The dataset was divided into three segments so that we could assess our model's efficacy. A proportion of 80% of the dataset was placed in the training set to ensure that the model could learn as much as possible from the data. Model generalizability was checked and fine-tuned using a validation set that included 10% of the training data. The accuracy of the model was then evaluated in a controlled setting using 20% of the data adopted as a testing set.

The specific hyperparameters for different classifiers are provided in Table 3. These hyperparameters were selected via fine-tuning, which likely involved a combination of manual exploration and automated techniques such as random search or grid search. The process of fine-tuning machine learning models entails the systematic adjustment of various hyperparameters in order to optimize their performance.

Specifying the random state as 42 in the Logistic Regression model ensures that the randomization process employed within the algorithm will yield the same results consistently across multiple iterations. XGBoost uses 341 estimators, with a max depth of 7 and a subsample value of 0.9. The learning rate is 0.001, and a gamma value of 0.5 controls minimum loss reduction. To split an internal node in a decision tree, a minimum of 69 samples is required. This algorithm uses entropy as a criterion to accurately measure the purity of each split, providing valuable insights for decision-making. In addition, the strategy employed to choose the optimal split at each node is determined by the splitter value of 42. SVM uses a linear kernel to create a decision boundary between classes. A C value of 1.0 balances margin maximization with classification error minimization.

Table 3. Model parameters.

Model	Details of Parameters
Logistic Regression	random state = 42.
XGBoost	n_estimators = 341 max_depth = 7, subsample = 0.9, learning rate = 0.001, $\gamma = 0.5$.
Decision Tree	min samples split = 69 criterion = entropy splitter = 42
SVM	kernel = linear, C = 1.0

Figure 10 illustrates the significance of each feature in classifying fraudulent transactions. The figure clearly shows that the feature labeled as “type” is significantly more important than all the other features. The features named “newbalanceOrig” and “amount” also have significant importance. “Step” and “oldbalanceorg” contribute little importance to the decision-making process. The feature named “OldBalancedest” holds minor significance, while the “nameDest” feature does not contribute to the classification of fraudulent transactions.

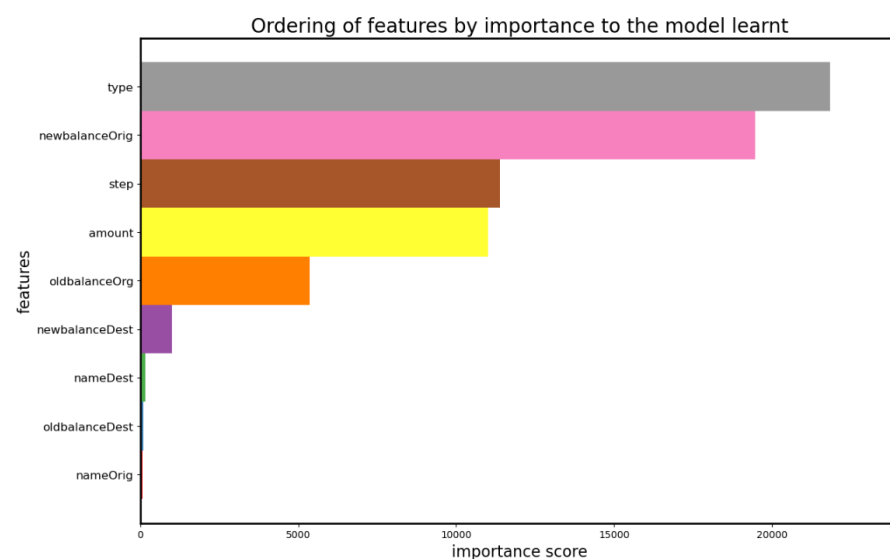


Figure 10. Ordering of features by importance to the model learned.

4.2. Evaluation Measures

Evaluation measures (performance metrics or scoring criteria) are required to assess model quality, efficacy, and performance accurately. These parameters offer quantifiable, objective standards for evaluating predictions against actual results. The F1-score, accuracy, precision, and recall metrics assess a model's effectiveness. Accuracy in classification is the percentage of samples with the correct labels. The proportion of actually positive elements to all positively marked items is referred to as precision. The proportion of correct identifications relative to the sum of correct and incorrect identifications is known as recall. The F1-score is a measure that takes into account both accuracy and recall. A higher score reflects a better performance. The equations are as follows:

$$\text{Accuracy} = \frac{T_P + T_N}{T_P + F_P + F_N + T_N} \quad (6)$$

$$\text{Precision} = \frac{T_P}{T_P + F_P} \quad (7)$$

$$\text{Recall} = \frac{T_P}{T_P + F_N} \quad (8)$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (9)$$

The true positive (T_P) rate measures the total amount of correctly identified samples. F_P , or false positive, represents the total number of positively classified negative instances. True negative, or T_N , represents the total amount of negative instances that are adequately classified by the model. The false negative (F_N) metric indicates how many truly positive examples were classified as negative by the model.

4.3. Result Analysis

Table 4 shows the results of the four machine learning models. Logistic Regression had a precision of 0.92 and a recall of 0.86, with a well-balanced F1-score of 0.89. Its high accuracy rate of 98.99% means it can correctly classify most data points.

Table 4. Comparisons of model efficiency.

Model	Precision	Recall	F1-Score	Accuracy
Logistic Regression	0.92	0.86	0.89	98.99%
XGBClassifier	0.96	0.88	0.92	99.88%
Decision Tree	0.87	0.85	0.86	98.96%
SVM	0.64	0.36	0.48	96.91%

The XGBoost model obtains 0.96 precision, 0.88 recall, and a 0.92 F1-score, which is impressive. The model's 99.88% success rate in classification is indicative of its superior overall performance.

Decision Tree model has an F1-score of 0.86, a recall of 0.85, and a precision of 0.87. This model has 98.96% accuracy, which is a little lower than the preceding models but still decent.

SVM model, on the other hand, performs poorly compared to the others. The SVM model has an F1-score of 0.48 due to its low precision (0.64) and recall (0.36). Although the SVM model classifies the majority of instances correctly with an accuracy of 96.91%, its performance lags significantly behind the other models.

In Table 4, the XGBoost model demonstrates superior performance compared to the other three candidate models. The Logistic Regression and Decision Tree models demonstrate commendable performance, albeit with slightly inferior metrics in comparison to XGBoost. Conversely, the Support Vector Machine (SVM) model exhibits significantly inferior performance, indicating poor classification performance.

Table 5 presents an overview of an analysis of the utilized machine learning model evaluations using various sampling strategies. Looking at the performance of the Logistic Regression model, it appears that by using the oversampling technique, SMOTE attained a remarkable accuracy score of 98.99%. Without the sampling technique, it resulted in a slightly lower accuracy of 96.89%, while the undersampling technique yielded the lowest accuracy score of 81.71%.

The XGBoost Classifier performed well, with an accuracy score of 95.97%. The SMOTE sampling technique resulted in the highest accuracy score of 99.88%. Interestingly, the undersampling technique also achieved a high accuracy score of 97.41%, indicating that reducing the majority of class instances did not impact the model’s performance.

The Decision Tree model achieved the highest accuracy score of 98.96% when using SMOTE. The initial dataset resulted in an accuracy rate of 97.87%, whereas the undersampling technique produced a similarly good result with an accuracy score of 96.74%. These results indicate that the performance of the Decision Tree model is relatively similar across all three sampling techniques. The SVM model achieved 96.91% accuracy with SMOTE sampling. The accuracy without sampling was 92.74%, and undersampling decreased it to 88.89%.

Table 5. Performance of different models with different sampling techniques.

Model	SMOTE	Normal	Undersampling
Logistic Regression	98.99%	96.89%	81.71%
XGBClassifier	99.88%	95.97%	97.41%
Decision Tree	98.96%	97.87%	96.74%
SVM	96.91%	92.74%	88.89%

The Receiver Operating Characteristic (ROC) curve is a powerful visualization tool used to assess the performance of binary classification models. The ROC curve displays the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) for various classification models in Figure 11. The closer a model’s ROC curve is to the top-left corner of the plot, the better its performance in correctly classifying positive and negative instances. The XGBoost model, with its curve-hugging the top-left corner, emerges as the best-performing model among those presented in the image. The coloured curves depict the performance of different classification models. The Logistic Regression model (blue curve) with an Area Under the Curve (AUC) of 0.94 performs reasonably well. In contrast, the Decision Tree model (orange curve) with an AUC of 0.84 performs less than the others. The Random Forest model (green curve) with an AUC of 0.96 demonstrates the second-best performance, closely followed by the XGBoost model (red curve) with an impressive AUC of 1.00, indicating its superior ability to achieve a high True Positive Rate while maintaining a low False Positive Rate.

The dashed line represents the random classifier or the baseline model, where the True Positive Rate and False Positive Rate are equal across all threshold values. This line serves as a reference point, and any model that falls below this line is considered to perform worse than random guessing.

Based on the analysis, it is clear that the XGBoost model with the highest accuracy has the most potent discriminative power out of the models evaluated, with an AUC of 1.00. The model with the second highest accuracy, the Decision tree model, has weaker performance, with an AUC value of 0.84.

In Figure 12, the bar plot, along with the macro average and weighted average, provides a thorough overview of the precision, recall, and F1-score metrics for each class (non-fraudulent or fraudulent). The macro average provides an overall evaluation of performance without taking into account class imbalances by representing an unweighted average of precision, recall, and F1-score across all classes. In contrast, the utilization of a weighted average as a performance measure accounts for class imbalances by assigning weights to the average based on the frequency of instances within each class. Notably,

it is clear from the bar plot that the bar representing the weighted average is taller than the bar indicating the macro average. This suggests that, in comparison to the macro average, the weighted average, which takes into consideration the effect of class distribution, produces a higher performance measure result. These two bars' different heights highlight the adverse effects of class imbalance on the overall performance assessment.

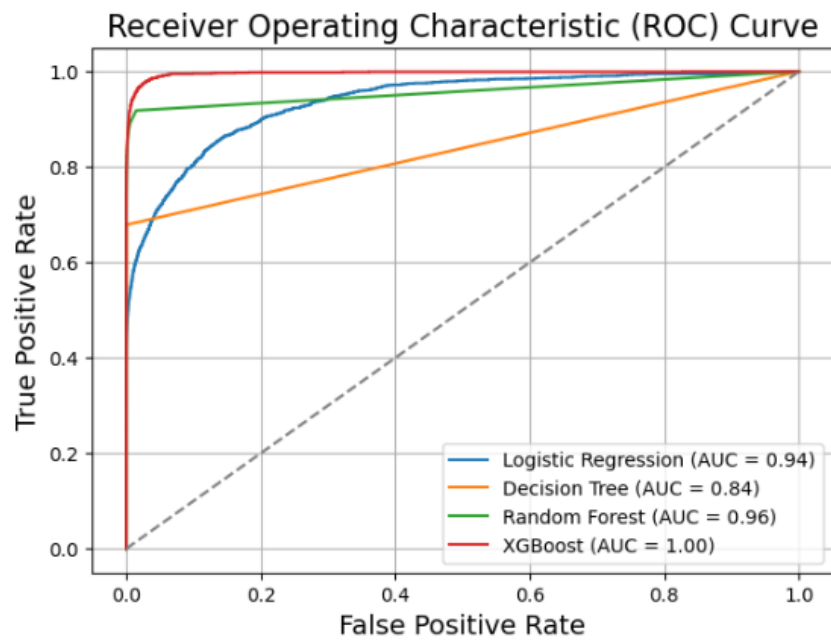


Figure 11. A comparison of the models' ROC curves.

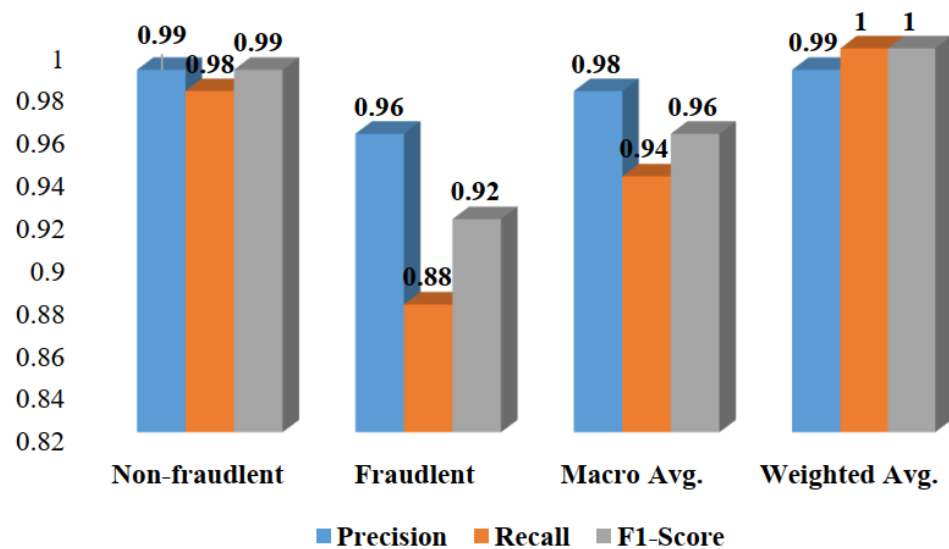


Figure 12. Performance of the XGBoost Classifier.

4.3.1. K-Fold Cross-Validation

One way to determine if a machine learning model is overfitting is by using K-fold cross-validation. This technique evaluates the model's performance on both the training and validation datasets. If the model performs well on the training data but needs help with unseen data, it is likely overfitted. K-fold cross-validation is a reliable way to detect this issue.

K-fold cross-validation involves splitting a dataset into K folds, using one as the validation set and the rest as the training sets. This is repeated K times to evaluate the

model's performance and prevent overfitting. Consistent performance across all folds indicates strong generalization.

Our research utilized a rigorous ten-fold cross-validation approach to meticulously analyze the dataset. The dataset was meticulously divided into ten subsets called "folds". We trained the model using nine out of ten folds to ensure optimal training results [47], while the remaining fold was reserved for validation. This meticulous process was rigorously repeated ten times, with each iteration utilizing a different fold from the dataset for validation. The training accuracy values consistently ranged between 99.979% and 99.982%, conclusively demonstrating that the model consistently achieves an exceptionally high level of accuracy when trained on the data.

Based on the data in Table 6, the training and validation accuracy exhibit minimal differences across all K values. The observed data do not exhibit any notable variations or significant differences in the accuracy values between the training and validation [48] sets across different folds. The observed differences between them are relatively inconsequential, indicating an absence of significant overfitting in the model. The model exhibits an excellent ability to generalize to new instances [49], as shown by the consistently high validation accuracy. The validation accuracy [50] slightly decreases compared to the corresponding training accuracy values, which is typical of a well-generalized model.

Table 6. Performance of ten-fold cross-validation of our proposed model.

Value of K	Train_acc (%)	Valid_acc (%)
Fold = 1	99.991	99.973
Fold = 2	99.991	99.952
Fold = 3	99.991	99.984
Fold = 4	99.991	99.983
Fold = 5	99.991	99.982
Fold = 6	99.992	99.979
Fold = 7	99.991	99.980
Fold = 8	99.992	99.984
Fold = 9	99.991	99.980
Fold = 10	99.993	99.984

4.3.2. Comparison with Current State-of-the-Art Methods

Table 7 compares various methods and models based on their achieved banking fraud detection accuracy.

Table 7. Comparison of proposed model's performance against other models' accuracy levels.

Reference	Dataset	Accuracy (%)
[10]	Credit card fraud detection	RF = 99.96, NB = 99.23 MLP = 99.93
[23]	Transaction records: 18,060	Logistic Regression + XGBOOST = 98.523
[26]	Credit card transactions conducted by European cardholders.	Logistic Regression = 99.88 Random Forest = 99.95 CatBoost = 99.93
[1]	Credit card transactions Total transaction records: 284,807	NB = 97.92 KNN = 97.69 LR = 54.86
[28]	Statlog (Australian Credit)	Gradient Boosted Trees (GBT) = 96.7
Proposed Method	Synthetic Financial Datasets	99.88

5. Explainability Analysis

Understanding the factors or features that impact a model’s predictions is crucial for its interpretability and for identifying limitations or improvement areas. Examining patterns, correlations, or anomalies associated with fraudulent activity can provide invaluable insights to prevent future fraud. Therefore, explainability is of the utmost importance in model analysis.

In this study, two well-known methods for explainability analysis—SHAP (Shapley Additive Explanations) and LIME (Local Interpretable Model-Agnostic Explanations)—are utilized. These methods aim to shed light on how the model makes its predictions, but they employ distinct approaches. While SHAP offers a global perspective on the model’s behavior and feature importance, LIME explains individual predictions, providing a local perspective. By using both methods, transparency can be ensured, making it easier to detect and prevent fraud.

5.1. Model’s Interpretability Using SHAP Analysis

The SHAP method utilizes cooperative game theory to evaluate the role of each feature in a prediction by analyzing all possible feature combinations. By assessing all possible coalitions of features, SHAP values determine the average impact of each feature on all possible predictions. For fraud detection purposes, SHAP analysis is beneficial in identifying which features affect the model’s determination of whether a transaction is fraudulent. It pinpoints the significant indicators of fraudulent transactions based on their influence on the model’s predictions. Mean SHAP may aid in determining which characteristics have the greatest influence on fraud detection, allowing for more targeted feature engineering and model development [51,52].

In Figure 13, we present global explanations of the XGBoost model. The selection of the XGBoost model for SHAP analysis was based on its proven superiority in fraud detection, as mentioned earlier. SHAP uses colors to find low and high feature values. The red pixels represent strong feature values that are important in allocating a particular class. The blue pixels represent low feature values.

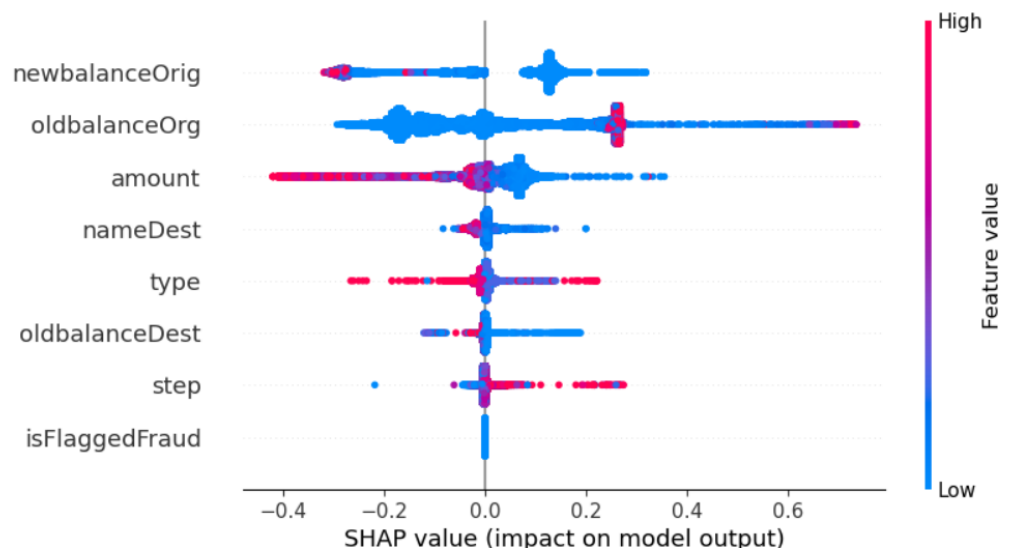


Figure 13. XGBoost model’s global explanations.

In Figure 14, we present the bar plot of XGBoost. The first SHAP value is “newblanceOrgi”, which obtained +0.14. The next is “oldblanceOrg”, which is +0.13. The lowest values are “newblanceOrgi”, “step”, and “oldblanceDest”, respectively.

The mean SHAP values are presented, which are computed based on their respective influence on the output in accordance with established principles.

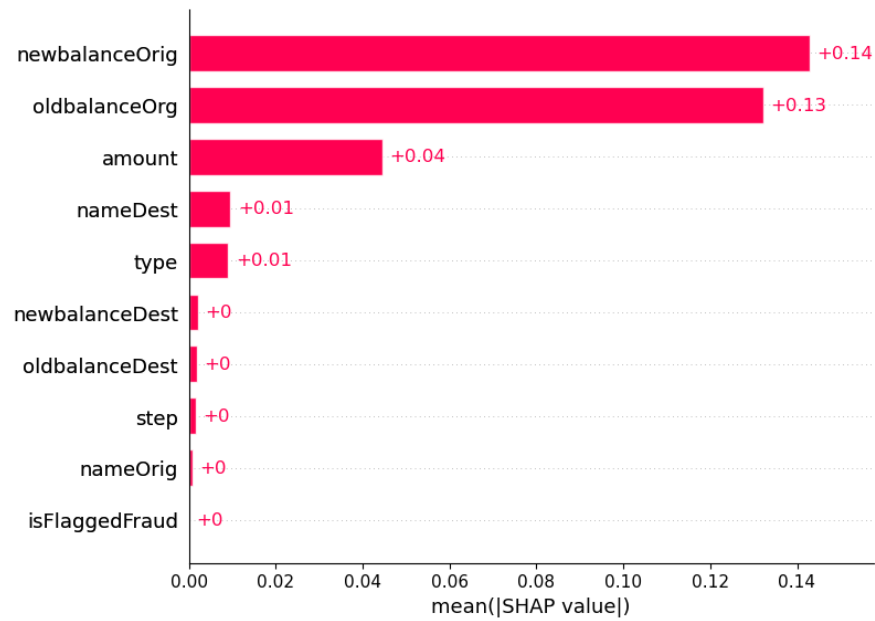


Figure 14. For a conventional bar plot, the mean absolute value of the SHAP values for each feature is required.

5.2. Model’s Interpretability Using LIME

The XGBoost Classifier algorithm’s predictions for identifying fraud were interpreted using LIME. The LIME model may provide an explanation for machine learning model inferences [53] by performing a local approximation of the inference point. The algorithm constructs a linear regression close to a chosen conclusion. Characteristics having a high positive weight lend credence to the prediction choice in the linear regression approximation, whereas characteristics with a high negative weight argue against it. Its goal is to make it easier for people in different locations to understand the reasoning behind a decision to label a transaction as fraudulent or non-fraudulent. The feature significance weights that result from LIME’s explanation generation allow it to do this.

Figure 15 shows the XGBoost model’s LIME explanation. It explains why a transaction was detected as fraudulent and why it was not. The orange bars provide factors that provide significance to the model’s prediction, while the blue bars show counterarguments. According to the description, the quantity and frequency of transactions, as well as the mismatch between the old and new balance’s intended recipients, are among the essential factors in the prediction. Conversely, our explainable machine learning model (XGBoost with LIME) identifies these factors as essential indicators of fraudulent activity in this specific transaction analysis.

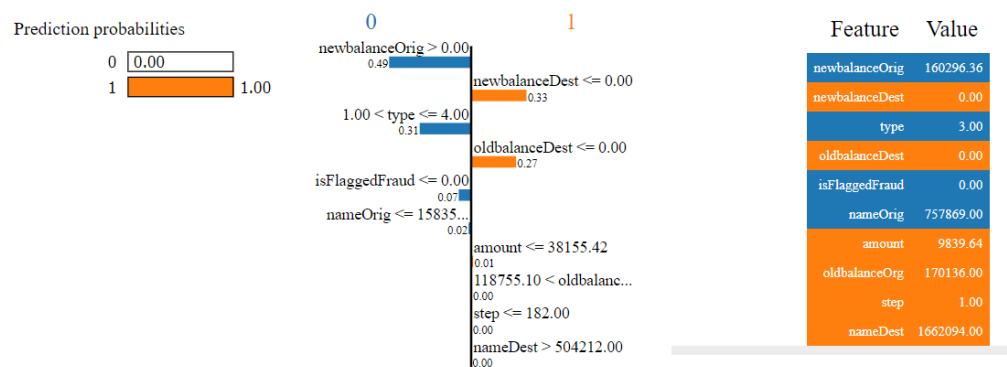


Figure 15. LIME for XGBoost Classifier of prediction on a specific transaction instance. Red indicates features contributing to non-fraud prediction; green indicates contributing to fraud prediction.

6. Conclusions and Future Works

This research has laid a solid foundation for the detection of financial fraud through meticulous preprocessing, robust algorithm selection, and insightful interpretability methods. The exceptional accuracy of the XGBoost Classifier underscores its efficacy in identifying fraudulent activities, even within imbalanced datasets. Our findings employing SHAP and LIME for model interpretation add transparency to the decision-making process and demonstrate promising results; however, there remains room for further exploration. Exploring alternative sampling techniques and delving into the realm of deep learning methods will contribute to refining fraud detection capabilities. The real-world validation of these approaches on diverse financial datasets is crucial for ensuring their practical applicability. By continuing to innovate and refine our methodologies, we can enhance the resilience of financial systems against fraudulent behavior, ultimately fostering greater trust and security within the industry.

Author Contributions: Conceptualization, S.M.N.N. and S.C.; Methodology, S.S. and S.C.; Software, S.M.N.N., S.S. and S.P.S.; Validation, S.S. and M.J.N.M.; Formal analysis, S.M.N.N.; Investigation, T.J., A.B. and M.W.; Resources, S.C. and M.J.N.M.; Data curation, S.S., S.P.S. and S.C.; Writing—original draft, S.M.N.N. and S.P.S.; Writing—review & editing, M.J.N.M. and M.W.; Visualization, S.P.S. and M.W.; Supervision, S.C. and M.J.N.M.; Project administration, M.J.N.M.; Funding acquisition, T.J., A.B. and M.W. All authors have read and agreed to the published version of the manuscript.

Funding: The authors declare this project as self-funded.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The majority of the data presented in this research article are available within the manuscript. Any additional data, if required, can be obtained by contacting the corresponding author upon request.

Conflicts of Interest: The authors declared no conflict of interest.

References

- Awoyemi, J.O.; Adetunmbi, A.O.; Oluwadare, S.A. Credit card fraud detection using machine learning techniques: A comparative analysis. In Proceedings of the 2017 International Conference on Computing Networking and Informatics (ICCNI), Lagos, Nigeria, 29–31 October 2017; IEEE: New York, NY, USA, 2017; pp. 1–9.
- Mytnyk, B.; Tkachyk, O.; Shakhovska, N.; Fedushko, S.; Syerov, Y. Application of Artificial Intelligence for Fraudulent Banking Operations Recognition. *Big Data Cogn. Comput.* **2023**, *7*, 93. [\[CrossRef\]](#)
- Yee, O.S.; Sagadevan, S.; Malim, N.H.A.H. Credit card fraud detection using machine learning as data mining technique. *J. Telecommun. Electron. Comput. Eng. (JTEC)* **2018**, *10*, 23–27.
- Raval, J.; Bhattacharya, P.; Jadav, N.K.; Tanwar, S.; Sharma, G.; Bokoro, P.N.; Elmorsy, M.; Tolba, A.; Raboaca, M.S. RaKShA: A Trusted Explainable LSTM Model to Classify Fraud Patterns on Credit Card Transactions. *Mathematics* **2023**, *11*, 1901. [\[CrossRef\]](#)
- Irénée, M.; Wang, Y.; Hei, X.; Song, X.; Turiho, J.C.; Nyesheja, E.M. XTS: A Hybrid Framework to Detect DNS-Over-HTTPS Tunnels Based on XGBoost and Cooperative Game Theory. *Mathematics* **2023**, *11*, 2372. [\[CrossRef\]](#)
- Hasib, K.M.; Tanzim, A.; Shin, J.; Faruk, K.O.; Al Mahmud, J.; Mridha, M. BMNet-5: A novel approach of neural network to classify the genre of Bengali music based on audio features. *IEEE Access* **2022**, *10*, 108545–108563. [\[CrossRef\]](#)
- Hasib, K.M.; Iqbal, M.; Shah, F.M.; Mahmud, J.A.; Popel, M.H.; Showrov, M.; Hossain, I.; Ahmed, S.; Rahman, O. A survey of methods for managing the classification and solution of data imbalance problem. *arXiv* **2020**, arXiv:2012.11870.
- Maitra, S.; Hossain, T.; Hasib, K.M.; Shishir, F.S. Graph theory for dimensionality reduction: A case study to prognosticate parkinson's. In Proceedings of the 2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 4–7 November 2020; IEEE: New York, NY, USA, 2020; pp. 134–140.
- Jahan, S.; Islam, M.R.; Hasib, K.M.; Naseem, U.; Islam, M.S. Active Learning with an Adaptive Classifier for Inaccessible Big Data Analysis. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; IEEE: New York, NY, USA, 2021; pp. 1–7.
- Varmedja, D.; Karanovic, M.; Sladojevic, S.; Arsenovic, M.; Anderla, A. Credit card fraud detection-machine learning methods. In Proceedings of the 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH), Novi Sad, Serbia, 20–22 March 2019; IEEE: New York, NY, USA, 2019; pp. 1–5.
- Pech, R. *Fraud Detection in Mobile Money Transfer as Binary Classification Problem*; Eagle Technologies Inc Publ: Arlington, VA, USA, 2019; pp. 1–15.

12. Oza, A. Fraud detection using machine learning. *Transfer* **2018**, *528812*, 532909.
13. Kurshan, E.; Shen, H.; Yu, H. Financial crime & fraud detection using graph computing: Application considerations & outlook. In Proceedings of the 2020 Second International Conference on Transdisciplinary AI (TransAI), Irvine, CA, USA, 21–23 September 2020; IEEE: New York, NY, USA, 2020; pp. 125–130.
14. Pambudi, B.N.; Hidayah, I.; Fauziati, S. Improving money laundering detection using optimized support vector machine. In Proceedings of the 2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), Yogyakarta, Indonesia, 5–6 December 2019; IEEE: New York, NY, USA, 2019; pp. 273–278.
15. Zhang, Y.; Trubey, P. Machine learning and sampling scheme: An empirical study of money laundering detection. *Comput. Econ.* **2019**, *54*, 1043–1063. [[CrossRef](#)]
16. Raiter, O. Applying supervised machine learning algorithms for fraud detection in anti-money laundering. *J. Mod. Issues Bus. Res.* **2021**, *1*, 14–26.
17. Lopez-Rojas, E.A.; Barneaud, C. Advantages of the PaySim simulator for improving financial fraud controls. In *Intelligent Computing: Proceedings of the 2019 Computing Conference, Volume 2*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 727–736.
18. Besenbruch, J. Fraud Detection Using Machine Learning Techniques. Research Paper Business Analytics. 2018. Available online: <https://vu-business-analytics.github.io/internship-office/papers/paper-besenbruch.pdf> (accessed on 30 April 2024).
19. Kuppa, A.; Le-Khac, N.A. Adversarial XAI methods in cybersecurity. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 4924–4938. [[CrossRef](#)]
20. Ngai, E.W.; Hu, Y.; Wong, Y.H.; Chen, Y.; Sun, X. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decis. Support Syst.* **2011**, *50*, 559–569. [[CrossRef](#)]
21. Saia, R.; Carta, S. *Evaluating Credit Card Transactions in the Frequency Domain for a Proactive Fraud Detection Approach*; SECRYPT: Berlin, Germany, 2017; pp. 335–342.
22. Carcillo, F.; Le Borgne, Y.A.; Caelen, O.; Kessaci, Y.; Oblé, F.; Bontempi, G. Combining unsupervised and supervised learning in credit card fraud detection. *Inf. Sci.* **2021**, *557*, 317–331. [[CrossRef](#)]
23. Zhao, Z.; Bai, T. Financial Fraud Detection and Prediction in Listed Companies Using SMOTE and Machine Learning Algorithms. *Entropy* **2022**, *24*, 1157. [[CrossRef](#)] [[PubMed](#)]
24. Nascita, A.; Montieri, A.; Aceto, G.; Ciuonzo, D.; Persico, V.; Pescapé, A. Improving performance, reliability, and feasibility in multimodal multitask traffic classification with XAI. *IEEE Trans. Netw. Serv. Manag.* **2023**, *20*, 1267–1289. [[CrossRef](#)]
25. Khatri, S.; Arora, A.; Agrawal, A.P. Supervised machine learning algorithms for credit card fraud detection: A comparison. In Proceedings of the 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 29–31 January 2020; IEEE: New York, NY, USA, 2020; pp. 680–683.
26. Hema, A. Machine Learning methods for Discovering Credit Card Fraud. *IRJCS Int. Res. J. Comput. Sci.* **2020**, *III*, 1–6.
27. Kumar, M.S.; Soundarya, V.; Kavitha, S.; Keerthika, E.; Aswini, E. Credit card fraud detection using random forest algorithm. In Proceedings of the 2019 3rd International Conference on Computing and Communications Technologies (ICCCT), Chennai, India, 21–22 February 2019; IEEE: New York, NY, USA, 2019; pp. 149–153.
28. Seera, M.; Lim, C.P.; Kumar, A.; Dhamotharan, L.; Tan, K.H. An intelligent payment card fraud detection system. *Ann. Oper. Res.* **2021**, *334*, 445–467. [[CrossRef](#)]
29. Puh, M.; Brkić, L. Detecting credit card fraud using selected machine learning algorithms. In Proceedings of the 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 20–24 May 2019; IEEE: New York, NY, USA, 2019; pp. 1250–1255.
30. Lopez-Rojas, E.; Elmir, A.; Axelsson, S. PaySim: A financial mobile money simulator for fraud detection. In Proceedings of the 28th European Modeling and Simulation Symposium, EMSS, Larnaca, Cyprus, 26–28 September 2016; Dime University of Genoa: Genoa, Italy; 2016; pp. 249–255.
31. sklearn.Preprocessing.LabelEncoder—scikit-learn.org. Available online: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html> (accessed on 1 July 2023).
32. Sklearn.Preprocessing.MinMaxScaler—scikit-learn.org. Available online: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html> (accessed on 1 July 2023).
33. Islam, M.T.; Hasib, K.M.; Rahman, M.M.; Tusher, A.N.; Alam, M.S.; Islam, M.R. Convolutional Auto-Encoder and Independent Component Analysis Based Automatic Place Recognition for Moving Robot in Invariant Season Condition. *Hum. Centric Intell. Syst.* **2022**, *3*, 13–24. [[CrossRef](#)]
34. Hasnat, F.; Hasan, M.M.; Nasib, A.U.; Adnan, A.; Khanom, N.; Islam, S.M.; Mehedi, M.H.K.; Iqbal, S.; Rasel, A.A. Understanding Sarcasm from Reddit texts using Supervised Algorithms. In Proceedings of the 2022 IEEE 10th Region 10 Humanitarian Technology Conference (R10-HTC), Hyderabad, India, 6–18 September 2022; IEEE: New York, NY, USA, 2022; pp. 1–6.
35. Hossain, M.S.; Arefin, M.S. Development of an Intelligent Job Recommender System for Freelancers using Client’s Feedback Classification and Association Rule Mining Techniques. *J. Softw.* **2019**, *14*, 312–339. [[CrossRef](#)]
36. Jullum, M.; Løland, A.; Huseby, R.B.; Ånonsen, G.; Lorentzen, J. Detecting money laundering transactions with machine learning. *J. Money Laund. Control* **2020**, *23*, 173–186. [[CrossRef](#)]
37. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.

38. Wang, Y.; Pan, Z.; Zheng, J.; Qian, L.; Li, M. A hybrid ensemble method for pulsar candidate classification. *Astrophys. Space Sci.* **2019**, *364*, 139. [[CrossRef](#)]
39. Cody, C.; Ford, V.; Siraj, A. Decision tree learning for fraud detection in consumer energy consumption. In Proceedings of the 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 9–11 December 2015; IEEE: New York, NY, USA, 2015; pp. 1175–1179.
40. Javed Mehedi Shamrat, F.; Ranjan, R.; Hasib, K.M.; Yadav, A.; Siddique, A.H. Performance evaluation among id3, c4. 5, and cart decision tree algorithm. In *Pervasive Computing and Social Networking: Proceedings of ICPCSN 2021*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 127–142.
41. Nobel, S.N.; Sultana, S.; Tasir, M.A.M.; Rahman, M.S. Next Word Prediction in Bangla Using Hybrid Approach. In Proceedings of the 2023 26th International Conference on Computer and Information Technology (ICCIT), Cox's Bazar, Bangladesh, 13–15 December 2023; pp. 1–6.
42. Quinlan, J.R. Induction of decision trees. *Mach. Learn.* **1986**, *1*, 81–106. [[CrossRef](#)]
43. Salzberg, S.L. *C4. 5: Programs for Machine Learning*; Quinlan, J.R., Ed.; Morgan Kaufmann Publishers, Inc.: San Francisco, CA, USA, 1993.
44. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [[CrossRef](#)]
45. Salehi, A.; Ghazanfari, M.; Fathian, M. Data mining techniques for anti money laundering. *Int. J. Appl. Eng. Res.* **2017**, *12*, 10084–10094.
46. Ruiz-Gonzalez, R.; Gomez-Gil, J.; Gomez-Gil, F.J.; Martínez-Martínez, V. An SVM-based classifier for estimating the state of various rotating components in agro-industrial machinery with a vibration signal acquired from a single point on the machine chassis. *Sensors* **2014**, *14*, 20713–20735. [[CrossRef](#)] [[PubMed](#)]
47. Li, D.; Jiang, M.R.; Li, M.W.; Hong, W.C.; Xu, R.Z. A floating offshore platform motion forecasting approach based on EEMD hybrid ConvLSTM and chaotic quantum ALO. *Appl. Soft Comput.* **2023**, *144*, 110487. [[CrossRef](#)]
48. Hossen, R.; Whaiduzzaman, M.; Uddin, M.N.; Islam, M.J.; Faruqui, N.; Barros, A.; Sookhak, M.; Mahi, M.J.N. Bdps: An efficient spark-based big data processing scheme for cloud fog-iot orchestration. *Information* **2021**, *12*, 517. [[CrossRef](#)]
49. Whaiduzzaman, M.; Sakib, A.; Khan, N.J.; Chaki, S.; Shahrier, L.; Ghosh, S.; Rahman, M.S.; Mahi, M.J.N.; Barros, A.; Fidge, C.; et al. Concept to Reality: An Integrated Approach to Testing Software User Interfaces. *Appl. Sci.* **2023**, *13*, 11997. [[CrossRef](#)]
50. Achar, S.; Faruqui, N.; Whaiduzzaman, M.; Awajan, A.; Alazab, M. Cyber-physical system security based on human activity recognition through IoT cloud computing. *Electronics* **2023**, *12*, 1892. [[CrossRef](#)]
51. Ekanayake, I.; Meddage, D.; Rathnayake, U. A novel approach to explain the black-box nature of machine learning in compressive strength predictions of concrete using Shapley additive explanations (SHAP). *Case Stud. Constr. Mater.* **2022**, *16*, e01059. [[CrossRef](#)]
52. Ahmed, S.; Nobel, S.N.; Ullah, O. An effective deep CNN model for multiclass brain tumor detection using MRI images and SHAP explainability. In Proceedings of the 2023 International Conference on Electrical, Computer and Communication Engineering (ECCE), Chittagong, Bangladesh, 23–25 February 2023; pp. 1–6.
53. Khedkar, S.; Subramanian, V.; Shinde, G.; Gandhi, P. Explainable AI in healthcare. In Proceedings of the 2nd International Conference on Advances in Science & Technology (ICAST), Mumbai, India, 8–9 April 2019.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.