Blockchain: Research and Applications 5 (2024) 100207

Contents lists available at ScienceDirect

Blockchain: Research and Applications

journal homepage: www.journals.elsevier.com/blockchain-research-and-applications

Research Article

Detecting anomalies in blockchain transactions using machine learning classifiers and explainability analysis

Mohammad Hasan [a], Mohammad Shahriar Rahman [b], Helge Janicke [c,d], Iqbal H. Sarker [c,d,*]

[a] Department of Computer Science and Engineering, Premier University, 4000 Chitagong, Bangladesh
[b] Department of Computer Science and Engineering, United International University, 1212 Dhaka, Bangladesh
[c] Cyber Security Cooperative Research Centre, 6027 Perth, Australia
[d] Centre for Securing Digital Futures, Edith Cowan University, 6027 Perth, Australia

ARTICLE INFO

Keywords:
Anomaly detection
Blockchain
Bitcoin transactions
Data imbalance
Data sampling
Explainable AI
Machine learning
Decision tree
Anomaly rules

ABSTRACT

As the use of blockchain for digital payments continues to rise, it becomes susceptible to various malicious attacks. Successfully detecting anomalies within blockchain transactions is essential for bolstering trust in digital payments. However, the task of anomaly detection in blockchain transaction data is challenging due to the infrequent occurrence of illicit transactions. Although several studies have been conducted in the field, a limitation persists: the lack of explanations for the model's predictions. This study seeks to overcome this limitation by integrating explainable artificial intelligence (XAI) techniques and anomaly rules into tree-based ensemble classifiers for detecting anomalous Bitcoin transactions. The shapley additive explanation (SHAP) method is employed to measure the contribution of each feature, and it is compatible with ensemble models. Moreover, we present rules for interpreting whether a Bitcoin transaction is anomalous or not. Additionally, we introduce an under-sampling algorithm named XGBCLUS, designed to balance anomalous and non-anomalous transaction data. This algorithm is compared against other commonly used under-sampling and over-sampling techniques. Finally, the outcomes of various tree-based single classifiers are compared with those of stacking and voting ensemble classifiers. Our experimental results demonstrate that: (i) XGBCLUS enhances true positive rate (TPR) and receiver operating characteristic-area under curve (ROC-AUC) scores compared to state-of-the-art under-sampling and over-sampling techniques, and (ii) our proposed ensemble classifiers outperform traditional single tree-based machine learning classifiers in terms of accuracy, TPR, and false positive rate (FPR) scores.

1. Introduction

Blockchain, a chain of blocks that contains the history of several transactions or records of other applications in a public ledger, has been considered an emerging technology both in academic and industrial areas since the last decade [1]. Bitcoin, the first digital cryptocurrency, was proposed in 2008 and then successfully implemented by Nakamoto [2]. Although blockchain was created to support the popular Bitcoin currency, transactions of other digital cryptocurrencies (such as Ethereum, Ripple, Litecoin, etc.), health records, transportation, IoT applications, etc. [3], are stored in the blocks in a decentralized manner and are managed without the help of a third party organization. Prominent attributes such as trustworthiness, verifiability, decentralization, and immutability have rendered blockchain an effective integration partner for various information and communication technology (ICT) applications. Nevertheless, this technology remains susceptible to an array of challenges, including security breaches, privacy concerns, energy consumption, regulatory policies, and issues like selfish mining [4].

Despite its growing popularity in digital payments, Bitcoin remains susceptible to a range of attacks, including temporal attacks, spatial attacks, and logical-partitioning attacks [5]. To ensure the effective implementation of blockchain technology, the timely detection of malicious behavior or transactions within the network, or the identification of novel instances in the data, is imperative [6]. Swift and appropriate actions must be taken to mitigate potential risks. Within the context of blockchain systems, anomaly detection assumes paramount importance. This facet aids in the identification and prevention of potential malicious activities, thereby upholding the system's integrity [7]. Nonetheless, the inherent imbalance between normal and anomalous data in blockchain datasets, such as those of Bitcoin or Ethereum

* Corresponding author.
E-mail address: m.sarker@ecu.edu.au (I.H. Sarker).

https://doi.org/10.1016/j.bcra.2024.100207
Received 5 September 2023; Received in revised form 1 February 2024; Accepted 11 May 2024
Available online 20 May 2024

transactions, presents a substantial challenge for conventional anomaly detection methodologies. In many instances, the frequency of anomalous data points significantly pales in comparison to that of normal data points, thereby yielding imbalanced datasets. This skewed data distribution can exert an adverse impact on the efficacy of anomaly detection algorithms. These algorithms, often biased towards the majority class (normal data), face difficulties in accurately discerning the minority class (anomalous data) [8].

Several over- and under-sampling techniques such as synthetic minority oversampling technique (SMOTE), adaptive synthetic (ADASYN), random under sampling (RUS), NearMiss, etc., have been used to handle imbalanced data in various domains [9]. Under-sampling techniques can alleviate bias towards the majority class and enhance the performance of anomaly detection algorithms [10]. In this study, the number of anomalous Bitcoin transactions is much lower than that of legal transactions. However, it is necessary to identify illicit Bitcoin transactions as well as normal transactions accurately. Since there are only 108 anomalous cases in this dataset, the under-sampling techniques will select the same number of non-anomalous cases. Additionally, the models can learn the positive samples that help to classify the anomalous transactions from the independent test set more correctly. Moreover, the over-sampling methods generate artificial synthetic data to equalize the minority and majority samples [11]. The synthetic over-sampled data minimizes the false positive rates in the test dataset. However, this method can't perform well for highly imbalanced data. For this, variations of generative adversarial network (GAN) based over-sampling techniques have been investigated by researchers for generating artificial data to balance the minority and majority classes [12]. Although there are several under-sampling algorithms, no technique is free from limitations. A major problem in most of the under-sampling algorithms is that significant instances which have a great impact on model training may be missed [13]. Besides, overfitting problems, sensitivity to noises, introducing of bias into the database, etc., can reduce the performance of the machine learning (ML) models [14]. Nevertheless, there is no perfect under-sampling technique proposed as the dataset varies. Considering these issues, we propose an under-sampling technique based on the ensemble method. Our proposed algorithm selects a subset of instances from the majority class which has a significant impact on the performance of ML models. Our algorithm gives importance to all the samples in the dataset by iteratively selecting the subsets, which minimizes the chance of missing the significant samples. Additionally, several combined balancing techniques have been investigated to compare which technique performs better in Bitcoin anomaly detection.

After the data are balanced, selecting a ML classifier is another challenging task. Tree-based ML classifiers have been used in many studies to classify malicious activities [15] since faster training can be performed on tree-based classifiers [16]. However, several studies have shown that the ensemble method can perform better in the case of large-scale data, e.g., Bitcoin transactions, than a single machine learning algorithm [17]. The idea behind the stacked ensemble method is that by combining multiple models, the strengths of each model can be leveraged while mitigating their weaknesses [18]. The stacked ensemble method differs from other ensemble methods, such as bagging and boosting, in that it combines models with different algorithms and/or hyperparameters rather than replicating the same model multiple times. This allows for greater model diversity and can lead to better performance. On the other hand, voting ensemble models predict the output by using the highest majority voting (hard voting) or the highest average value of the individual prediction (soft voting) [19]. Although the complexity and computational costs may increase due to training multiple models in the base classifier, there are several valid reasons for using the stacking ensemble method. First, tree-based models are computationally faster than other ML models such as SVM or KNN. Second, tree-based models can handle the data without any conversion or normalization [20]. Third, a stacking-based ensemble model can increase performance by minimizing the prediction errors caused by the variance components. Moreover, the voting classifier with only tree-based models is a powerful and interpretable ensemble learning technique. Combining the predictions of different tree-based algorithms can provide more accurate and robust predictions, making it an attractive choice for various classification and regression tasks. Additionally, the interpretability of tree-based models remains preserved in the ensemble, which can be a valuable asset in applications where model transparency and understanding are essential. Considering the effectiveness of the combined tree-based methods, both stacked and voting ensemble models are proposed in this study.

After performing all these great tasks for anomaly detection, a question can arise "Should we trust the prediction of the black-box model?". Explainable artificial intelligence (XAI), is a field of interest for finding answers to this question [21]. This latest artificial intelligence (AI) technique helps to increase the explainability and transparency of black-box AI models by facilitating complex interpretable decisions [22]. Two popular XAI techniques, e.g., 'local interpretable model-agnostic explanations' (LIME) [23] and 'shapely additive explanations' (SHAP) [24] have been used by researchers to prove the explainability and transparency of AI models. SHAP was investigated in our study since it can find interpretable decisions more quickly and accurately.

In this study, an extreme gradient boosting-based clustering (XGBCLUS) under-sampling method is proposed and compared to other state-of-the-art under-sampling techniques to detect anomalous Bitcoin transactions. Besides, two popular over-sampling techniques, e.g., SMOTE and ADASYN, have been used to generate synthetic data points for balancing majority and minority classes and have been compared to the combined sampling methods SMOTE with edited nearest neighbor (ENN) and SMOTE with TOMEK links. A comparative analysis is performed among the down-sampling and over-sampling methods to decide which sampling technique is appropriate for identifying anomalous transactions. To classify nonillicit and illicit Bitcoin transactions, tree-based ML classifiers such as extreme gradient boosting (XGB) [25], random forest (RF) [26], decision tree (DT) [27], gradient boosting (GBoost) [28], and adaptive boosting (AdaBoost) [29] have been used. Although a single ML classifier may work well for anomaly detection on the training dataset, the final prediction on the independent test dataset may perform poorly. Therefore, both stacking and voting ensemble models have been proposed to increase the accuracy by combining the outcomes from individual classifiers. The main reason behind this is that if one of the chosen ML classifiers in the ensemble does not perform well, the risk can be minimized by averaging the outputs of all of them. Cross-validation has been used to avoid overfitting where the beginning training dataset is used to create multiple mini-train-test splits. The 10-fold cross-validation was used by partitioning the data into 10 subsets known as folds. Iteratively the algorithm is trained on nine folds while the remaining fold is kept as the test set. Finally, the correctness of the prediction by the black-box ML models is proven via the XAI technique, i.e., via SHAP analysis. A set of rules is also presented to conduct interpretability analysis for determining whether a Bitcoin transaction is anomalous. The contributions of this study are summarized below.

- We introduce an under-sampling algorithm based on XGB called XGBCLUS, and we compare it with state-of-the-art methods.
- We also explore various over-sampling and combined sampling techniques for classifying Bitcoin transactions.
- Further, we compare the effectiveness of both under-sampling and over-sampling techniques, and we also compare the tree-based ensemble classifiers with the individual ML classifiers for anomaly detection.
- We explain the predictions of the ensemble models using SHAP (an XAI technique) and identify the crucial features that exert the greatest influence on classifying Bitcoin transactions.
- Finally, we present a set of rules derived from a tree-based model to conduct interpretability analysis for anomalous transactions.

The rest of the paper is structured as follows: Section 2 provides a summary of recent research involving both supervised and unsupervised ML techniques. In Section 3, we outline the specifics of the proposed methods. Comparative results are presented in Section 4. Section 5 contains a discussion, and finally, Section 6 concludes the research paper.

2. Related work

Researchers have focused on to detecting or predicting whether a transaction is anomalous or not using the concept of blockchain intelligence [30], i.e., introducing AI for anomaly detection or fraud detection in blockchain transaction data. Several supervised and unsupervised techniques, along with various balancing methods, have been applied to detect fraudulent transactions in blockchain networks.

2.1. Supervised techniques

Chen et al. [31] employed supervised ML classifiers, including RF, AdB, MLP, SVM, and KNN, to detect bitcoin theft. RF exhibited the best performance with an F1 value of 0.952, surpassing that of the other unsupervised algorithms. Another study [32] explored the Bitcoin ecosystem, categorizing illegal activities and using bagging and GBoost for classification. While visualizing results, a notable limitation was the absence of proper balancing techniques for model tuning. Singh et al. [33] utilized SVM, DT, and RF for Ethereum network anomaly detection, detecting anomalous transactions with some limitations in dataset coverage. Active learning tools were applied in a previous study [34] for illegal activity detection in Bitcoin transactions, claiming superiority over unsupervised methods. A comparative study [35] favored ensemble-based methods for classifying non-anomalous and anomalous Bitcoin transactions based on accuracy and F1-score metrics.

2.2. Unsupervised techniques

In their analysis [36], the authors used Bitcoin transaction data to create two graphs for users and transactions, aiming to detect anomalies. Employing unsupervised methods such as SVM, K-means clustering, and Mahalanobis distance, they identified two anomalous users and one anomalous transaction out of 30 cases. However, the method struggled to identify maximum positive cases. Another study [37] utilized the K-means algorithm for anomaly detection in blockchain electronic transactions, also incorporating OSVM to find outliers. This approach suffered from high false positive rates. In a comparative study [38], various unsupervised learning algorithms, including IForest, one class SVM, two phase clustering, and multivariate Gaussian, were evaluated, with the multivariate Gaussian algorithm showing the highest F1-score. Evaluating the trimmed K-means clustering algorithm, Monamo et al. [39] successfully identified 5 anomalous activities out of 30 cases related to illicit transactions in the Bitcoin network. An encoder-decoder-based deep learning model was designed to detect anomalous activities in the Ethereum transaction network, claiming to detect illicit activities in the Ethereum network for the first time [40]. Lastly, clustering and role detection methods were applied in a study [41] to identify suspicious users in Bitcoin transaction data, using K-means for clustering and RoIX for role detection.

2.3. Balancing techniques

Researchers, as highlighted in Ref. [42], have proposed various undersampling and over-sampling methods to enhance evaluation metrics. For instance, in a comparative study [43], customized nearest-neighbor under-sampling achieved 99% accuracy, outperforming various SMOTE-based over-sampling techniques in analyzing Bitcoin and Ethereum transaction data. In the realm of bank transactions, ensemble-based classifiers, particularly SVM SMOTE dataset balancing with the RF classifier, were found effective in detecting fraudulent activities [44]. Similarly, authors in Ref. [45] proposed an under-sampling technique using fuzzy C-means clustering and similarity checks for identifying illicit activities in credit card transactions. Additionally, SMOTE oversampling has been employed in studies such as Ref. [46] and Ref. [47] to classify illegal activities in credit card transactions, while various under-sampling techniques were investigated in Ref. [48] and Ref. [49] to discern normal and illegal activities. Notably, for handling highly imbalanced datasets, Ahmed et al. [50] explored several under-sampling techniques for early product back-order prediction.

3. Methodology

The overall methodology of this study is illustrated in Fig. 1. Initially, we collected the dataset and conducted necessary preprocessing. Subsequently, the dataset was divided into training and test data. Data sampling was exclusively applied to the training data, while the test data remained independent. The sampled data was employed to train both individual and ensemble ML classifiers. Finally, the independent test data was utilized to validate the models, employing various evaluation metrics including accuracy, true positive rate (TPR), false positive rate (FPR), and receiver operating characteristic-area under curve (ROC-AUC) score. Additionally, we conducted a comparative analysis, combined with XAI and rules from DT, which is discussed in Section 4.

3.1. Dataset

The Bitcoin transaction data has been collected from the IEEE Data Portal[1]. It contains a total of 30,248,134 samples where 30,248,026 have been labeled as negative samples, i.e., the non-malicious transactions. On the other hand, there are only 108 malicious samples. So it is clear that the dataset is highly imbalanced. Exploratory data analysis (EDA) has been performed to reveal the dataset insights. There exist 12 attributes with a label to indicate whether a Bitcoin transaction is anomalous or not. An anomalous transaction is labeled by 1 and 0 stands for non-anomalous transactions. A correlation matrix of the features is depicted in Fig. 2. Features such as in_btc, out_btc, total_btc, mean_in_btc, and mean_out_btc exhibit strong positive correlations. Conversely, the indegree and outdegree features display weak correlation. While in_malicious, out_malicious, is_malicious, and all_malicious features are notably correlated with the output feature out_and_tx_malicious, no substantial correlation is observed between these features and indegree, outdegree, in_btc, out_btc, total_btc, mean_in_btc, and mean_out_btc.

Furthermore, we conducted a hypothesis test for feature selection employing the T-test to examine correlations between positive (anomalous Bitcoin transactions) and negative (non-anomalous Bitcoin transactions) samples. The T-test determines whether a significant difference exists between the means of the positive and negative samples. T-statistic values and corresponding p-values are computed for each attribute. These values are presented in Table 1. Notably, all attributes demonstrate significance with p-values below 0.01, except for outdegree. Additionally, it's worth noting that all features (in_malicious, out_malicious, is_malicious, and all_malicious) share the same value range as the target feature out_and_tx_malicious. Consequently, this similarity poses a challenge for ML classifiers in effectively discerning Bitcoin transactions, leading to the exclusion of these four features and outdegree. Ultimately, a set of seven features, including the target feature, is selected for classification. A summary of the selected features is shown in Table 2.

Following the feature selection process, the negative and positive samples were segregated, and duplicate entries were eliminated exclusively from the negative samples. To mitigate computational complexity, we opted to retain only 200,000 negative samples, accompanied

[1] https://ieee-dataport.org/open-access/bitcoin-transactions-data-2011-2013

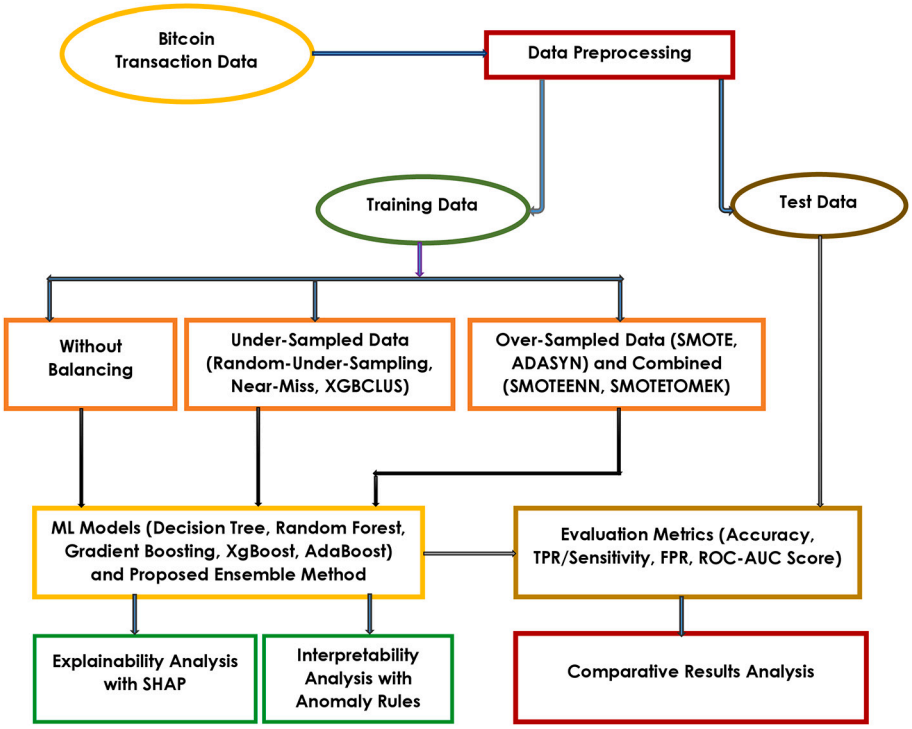


Fig. 1. Methodology of this study.

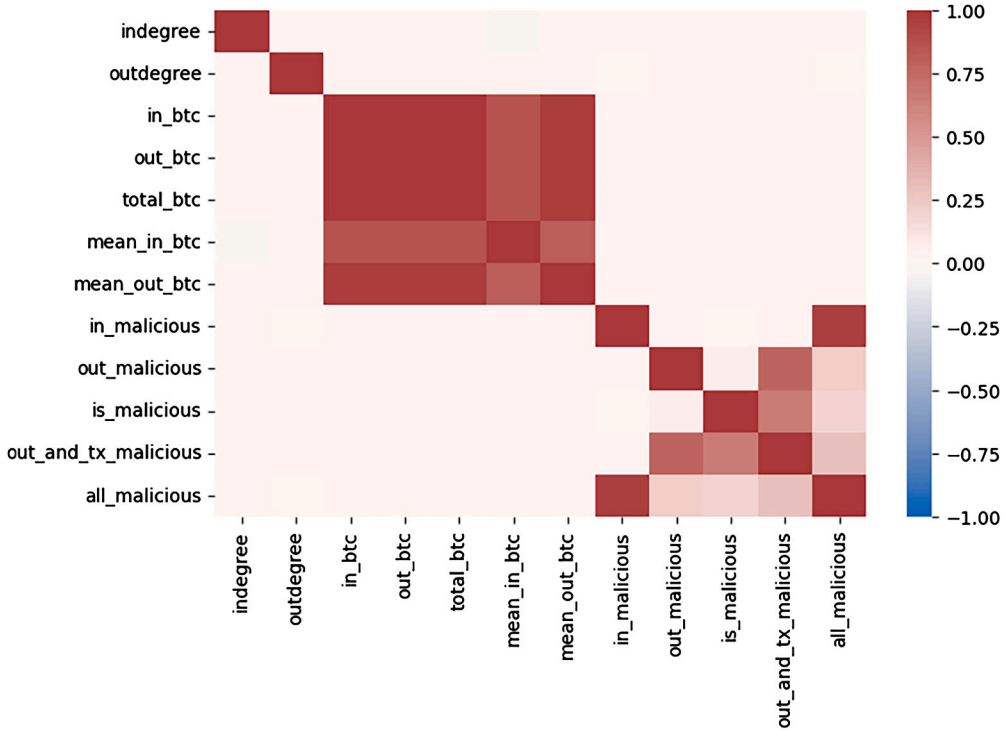


Fig. 2. Correlations among the features using heatmap.

by 108 positive samples. Nevertheless, the resulting imbalance ratio remains considerably high as shown in Fig. 3.

3.2. Imbalanced data handling

In Bitcoin transactions, the number of illicit transactions is significantly lower than that of normal transactions, leading to an imbalanced dataset. Consequently, ML classifiers tend to exhibit bias toward the majority class [51]. While classification accuracy might appear satisfactory in many cases, a notable discrepancy between TPR and FPR values often arises—indicating that the models struggle to accurately classify anomalies. To address this, it is crucial to balance the dataset using built-in or customized sampling techniques prior to training the classification models. In scenarios involving anomaly detection, fraud identification, or money laundering, positive cases are typically scarce. As such, under-sampling techniques can prove effective in rebalancing the dataset while prioritizing accurate identification of positive cases.

Table 1
The *t*-values and *p*-values for all attributes.

Attribute	*t* value	*p* value
indegree	-14.013838	0.000000
outdegree	0.842249	0.399648
in_btc	-17.229753	0.000000
out_btc	-16.469202	0.000000
total_btc	-16.864202	0.000000
mean_in_btc	-8.727102	0.000000
mean_out_btc	-16.014732	0.000000
in_malicious	-68.869826	0.000000
out_malicious	-5432.702805	0.000000
is_malicious	-3878.622465	0.000000
all_malicious	-1866.899584	0.000000

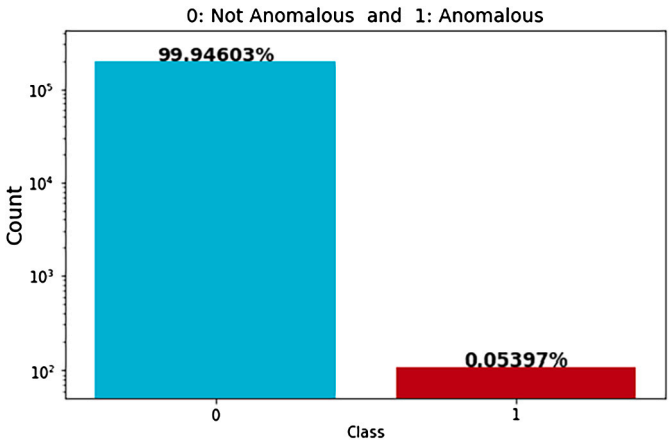


Fig. 3. Class ratio.

However, in instances where the number of positive cases or anomalies in the minority class is exceedingly low, ML classifiers may be trained on a limited dataset generated by the under-sampling technique. On the other hand, over-sampling methods aim to increase the instance count of the minority class to match that of the majority class. Despite generating artificial data based on a combination of majority and minority samples, these methods can be effective. In our study, we introduce an under-sampling algorithm named XGBCLUS, and we also investigate established under-sampling techniques such as random under sampling (RUS) and NearMiss. Furthermore, we explore popular over-sampling techniques including SMOTE, ADASYN, as well as combined approaches like SMOTEENN and SMOTETOMEK. We present a comparison between over-sampling and under-sampling methods in Section 4.

3.2.1. Under-sampling techniques

We have investigated two under-sampling methods, e.g., RUS and NearMiss along with our proposed under-sampling method which is described in Section 3.2.2. RUS is a simple technique used to handle class imbalance in datasets. It randomly selects a subset of instances from the majority class. The size of this subset is determined based on the desired balance ratio between the minority and majority classes. The balance ratio α_{u_s} is defined by Equation (1).

$$\alpha_{u_s} = \frac{N_m}{N_{r_m}} \tag{1}$$

where N_m is the number of samples in the minority class and N_{r_m} is the number of samples in the majority class after resampling. Then, the instances from both the minority class and the randomly selected subset of the majority class instances are combined to form a balanced dataset. Another under-sampling technique namely NearMiss is also used for balancing the dataset. It reduces the imbalance by retaining a subset of instances from the majority class that are close to instances from the minority class. This down-sampling technique selects instances based on their proximity to the minority class, making it possible to preserve important samples. For each instance in the minority class, NearMiss calculates its distances to all instances in the majority class using various distance metrics such as Euclidean distance or Manhattan distance. After that, this algorithm identifies the k instances from the majority class that are closest to each instance in the minority class. The value of k is typically set as a hyperparameter and determines the degree of under-sampling. We set the value as 1 and hence we call it NearMiss-1. Finally, it combines the instances from both the minority class and the selected instances from the majority class to form an under-sampled dataset.

3.2.2. XGBCLUS algorithm

XGBCLUS operates by merging clusters (accomplished by selecting random instances from the majority class in the training data, equal in number to the positive instances from the minority class) and the XGB algorithm. The algorithm starts with splitting the whole dataset into training and test data. The test data is kept independent and the number of positive samples, P, are counted from the training set. Then, the number of iterations, k, is calculated by dividing the total number of negative samples in the training data by the number of positive samples, P.

Algorithm 1 XGBCLUS algorithm.

1: **Input:** Imbalanced training samples, *DATA*; Number of iterations, *k*; Number of positive samples, *P*; The independent test data and the XGB algorithm.
2: **Output:** Selected under-sampled data
3: Initialize TMAX and FMIN
4: Initialize an empty set *Selected_Samples*
5: **for** $i = 1$ to k **do**
6: Select n negative samples arbitrarily equaling to P and prepare the training data
7: Train the model and predict using the test samples
8: Calculate true positive (TP) and false positive (FP) values
9: **if** TP > TMAX and FP < FMIN **then**
10: Set TMAX = TP and FMIN = FP
11: Update current n samples in *Selected_Samples*
12: **end if**
13: **end for**
14: **if** *Selected_Samples* is empty **then**
15: **Goto** step 3 and repeat steps 3 – 13 after changing TMAX and FMIN values
16: **else**
17: **Return** *Selected_Samples*
18: **end if**

In each iteration, the algorithm arbitrarily selects n negative samples equal to P and a new training set is prepared to train the Xgboost model. Using the independent test set, the model predicts the true positive (TP) and false positive (FP) values. The TP and FP values are compared with TMAX and FMIN, respectively. TMAX and FMIN are initialized with arbitrary values where TMAX represents the maximum true positive value and the minimum false positive value is defined by FMIN. If the TP value is greater than the TMAX value and the value of FP is less than the FMIN value, then both TMAX and FMIN are updated with the TP and FP values respectively. At the same time, current n samples are updated in the *Selected_Samples* set. Otherwise, no changes are made in the current iteration.

After the k iterations are finished, the *Selected_Samples* set is checked. If the set is empty, the algorithm should be run again with new TMAX and FMIN values. Otherwise, the samples in the *Selected_Samples* set are the under-sampled data returned by the algorithm. The XGBCLUS algorithm is shown in Algorithm 1.

Table 2
Summary of the selected features.

Feature name	Description
indegree	No. of inputs for a given transaction
in_btc	No. of Bitcoins on each incoming edge to a given transaction
out_btc	No. of Bitcoins on each outgoing edge from a given transaction
total_btc	Total number of Bitcoins for a given transaction
mean_in_btc	Average number of Bitcoins on each incoming edge to a given transaction
mean_out_btc	Average number of Bitcoins on each outgoing edge from a given transaction
out_and_tx_malicious	Status of a given transaction if it is malicious or not

3.2.3. Over-sampling techniques

We have also investigated two popular over-sampling and two combined-sampling strategies for handling the class imbalance in Bitcoin transaction data. Among them, SMOTE is a widely used technique to handle the class imbalance problem, especially in Bitcoin transactions for detecting anomalies. It aims to balance the class distribution by generating synthetic examples of the minority class, thereby mitigating the bias and improving model generalization. For each minority sample X_i, it selects the k nearest neighbors randomly from the same class. A new sample X_n is generated using one of the nearest neighbors X_{zi} from k and the new sample is generated using equation (2).

$$X_n = X_i + \lambda * (X_{zi} - X_i) \tag{2}$$

where λ is a random number between 0 and 1. Thus, a new synthetic instance is generated in the feature space. This process is repeated until the samples in both majority and minority classes are the same. At last, the original minority instances are combined with the newly generated synthetic instances to form a balanced dataset.

ADASYN also uses the same formula for generating the new sample except for the selection of X_i. It increases the density of synthetic instances in the regions that are harder to classify, which provides a more refined way of handling class imbalance. For each minority instance, it first calculates the number of its k nearest neighbors that belong to the majority class to get an idea of how close the minority sample is to the majority class. Then, it computes the imbalance ratio α_{o_s} for each minority instance using equation (3).

$$\alpha_{o_s} = \frac{N_{r_m}}{N_m} \tag{3}$$

where N_{r_m} is the number of samples in the minority class after resampling and N_m is the number of samples in the majority class. This imbalance ratio is used to calculate the desired number of synthetic instances to be generated for the current minority instance. A difficulty ratio is also calculated to discover the hard level. If the ratio is high, it considers more neighbors for generating synthetic instances. By interpolating feature values between the minority instance and its selected neighbors, a new synthetic instance is generated. This process is repeated for all minority instances until the samples in both majority and minority classes are the same. Finally, the original minority instances are combined with the newly generated synthetic instances to form a balanced dataset.

SMOTEENN is a combination of two resampling techniques, SMOTE and ENN. It tackles class imbalance by creating synthetic samples using SMOTE and subsequently refining the dataset using ENN to eliminate potentially noisy or misclassified instances. SMOTEENN aims to provide a more refined approach to balancing imbalanced datasets while also improving the quality of the final dataset by eliminating potential noise. After generating the synthetic instances by SMOTE, for each instance in the dataset, ENN first finds the k nearest neighbors. If the instance's class is different from the majority class of its neighbors, it removes the instance from the dataset. Thus, ENN eliminates noisy or misclassified instances. Another combined resampling technique is SMOTETOMEK where synthetic instances are generated by SMOTE and the under-sampling technique TOMEK link identifies pairs of instances from different classes that are closest to each other. For each pair of instances identified as TOMEK links, it removes the instance from the majority class. This under-sampling method helps in removing instances that are close to the decision boundary and might be misclassified. This can lead to a more balanced, discriminative, and effective dataset for training ML models.

3.3. Proposed ensemble model

The meta-classification ensemble method based on stacked generalization is an ML approach used to improve the accuracy of predictions by combining multiple models [52]. The stacking-based ensemble model is formed by two classifiers. One is the base classifier and the other is the meta classifier. It starts with training a set of base models using several classifiers. A new dataset is found from the base-level classifier and then the meta-classifier, also known as a combiner or a blender, is trained using the new dataset. After that, the learned meta-classifier is used to predict the independent test dataset. The architecture of the proposed stacked-ensemble model is shown in Fig. 4.

In our proposed stacking-based ensemble model, RF, DT, GBoost, and AdaBoost have been used as the base models. The training dataset is used to train the base models and the outputs of the four base models along with the validation fold are combined to create a new dataset. Then logistic regression (LR) [53], which is the meta-classifier in our proposed model, receives the newly formed dataset by combining the predictions of the base classifier as input and learns on that input set. Finally, the test dataset has been used to predict anomalous and non-anomalous transactions.

On the other side, the voting classifier is constructed using only tree-based models, which are a family of ML algorithms known for their robustness and interpretability. The architecture of a voting classifier that incorporates tree-based models like DT, XGB, GBoost, RF, and AdaBoost is shown in Fig. 5. The voting classifier takes the training set as input and the ensemble is constructed by combining the predictions of several individual tree-based models. Each model in this context refers to a unique instantiation of the tree-based algorithm with a specific set of hyperparameters or configurations. After the voting classifier has been trained and evaluated, it is used to make predictions on test data. The voting classifier aggregates the predictions of each individual tree-based model using a voting mechanism. The voting can be either "hard" or "soft". In hard voting, each model in the ensemble casts a single vote for the predicted class label, and the majority class receives the final prediction. For soft voting, the probabilities (confidence scores) of each model's predicted classes are averaged, and the class with the highest average probability is chosen as the final prediction.

4. Results analysis

In this section, we assess the performance of the proposed ensemble models using both under-sampled and over-sampled data. Additionally, a comprehensive comparative analysis between single classifiers and

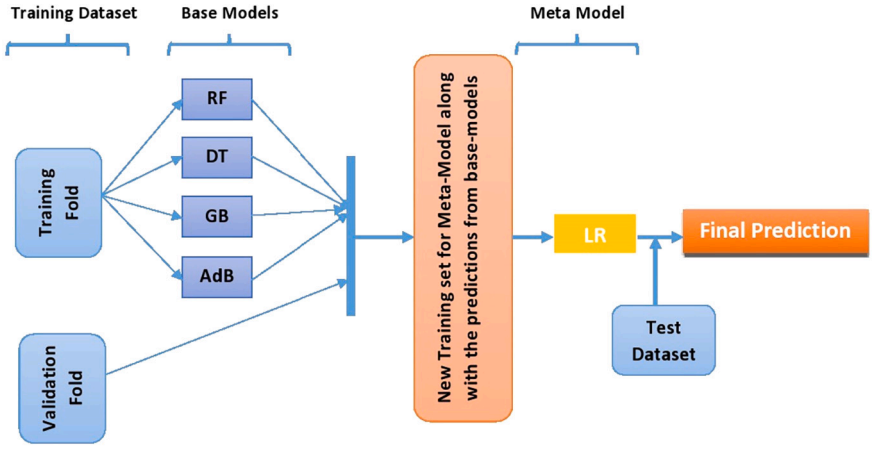


Fig. 4. Stacked-ensemble model architecture.

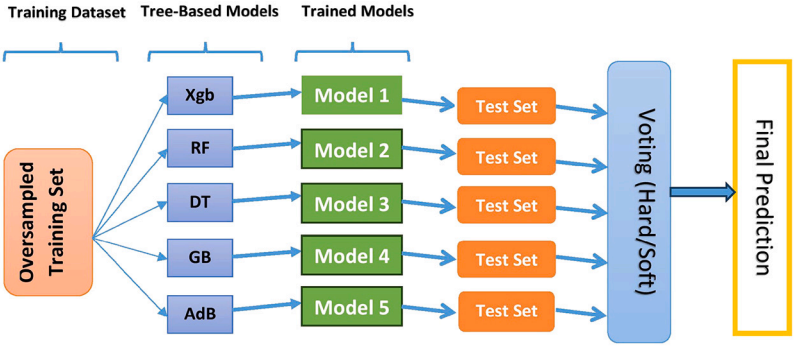


Fig. 5. Voting-ensemble model architecture.

ensemble classifiers is presented herein. We commence by configuring the experimental environment and subsequently present outputs that scrutinize various facets of the model's performance concerning Bitcoin anomaly detection. Thorough performance analyses have been conducted on a dataset encompassing both normal and abnormal transactions within the realm of Bitcoin transactions. An overview of the dataset is provided in Section 3. To facilitate performance evaluation, several Python libraries such as Pandas, NumPy, and scikit-learn were employed. The Python script was executed on Colab Pro, utilizing 12.68 GB of RAM and 225.83 GB of disk space.

4.1. Evaluation metrics

Given that accuracy alone is insufficient to gauge the performance of an anomaly detection system, it becomes crucial to employ additional metrics such as TPR to assess the accurate identification of anomalous transactions and FPR to evaluate the correct identification of non-anomalous transactions. This aligns with the primary objective of our study. The evaluation metrics, e.g., accuracy, TPR, and FPR, have been used to compare the performance of the single models without and with balancing the data against the proposed ensemble models. Additionally, we use the feature importance score to show the hierarchy of the features. We have also considered the ROC score, which compares the TPR against the FPR. The performance metrics are defined below:

TP = True Positive: an anomalous transaction is correctly identified as anomalous

TN = True Negative: a non-anomalous or normal transaction is correctly identified as non-anomalous

FP = False Positive: a non-anomalous transaction is incorrectly identified as anomalous

FN = False Negative: an anomalous transaction is incorrectly identified as non-anomalous

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$TPR = \text{Sensitivity} = \frac{TP}{TP + FN} \tag{5}$$

$$TNR = \text{Specificity} = \frac{TN}{TN + FP} \tag{6}$$

$$FPR = \frac{FP}{TN + FP} \tag{7}$$

AUC is calculated as the area under the Sensitivity (TPR)-(1−Specificity) (FPR) curve.

4.2. Effects of under-sampling in classification

We have kept 20% of data for the independent test set and the remaining 80% has been used for training the models. To prove the data imbalanced problem, the classifiers have been trained without balancing the train set. The ML classifiers become biased to the majority samples and result in a high true negative value. However, the ML classifiers cannot identify the positive samples correctly that's why the true positive rate is very low and in some cases, it is zero. Table 3 shows the comparison of accuracy, TPR, and ROC-AUC score of the DT, GBoost, RF, and AdaBoost classifiers. The TP values are zero for DT and RF classifiers which indicates that no anomalous transactions are correctly identified and all transactions are classified as normal transactions. Although the accuracy seems to be good enough for the classifiers, the TP score tends to zero i.e., anomalous transactions are not identified because of the biases of models to the majority of transactions. Given that detecting anomalous transactions is the primary objective of our study, classifiers may struggle to identify those transactions without balanced data. Therefore, we explored several balancing techniques to enhance the TPR and decrease the FPR values.

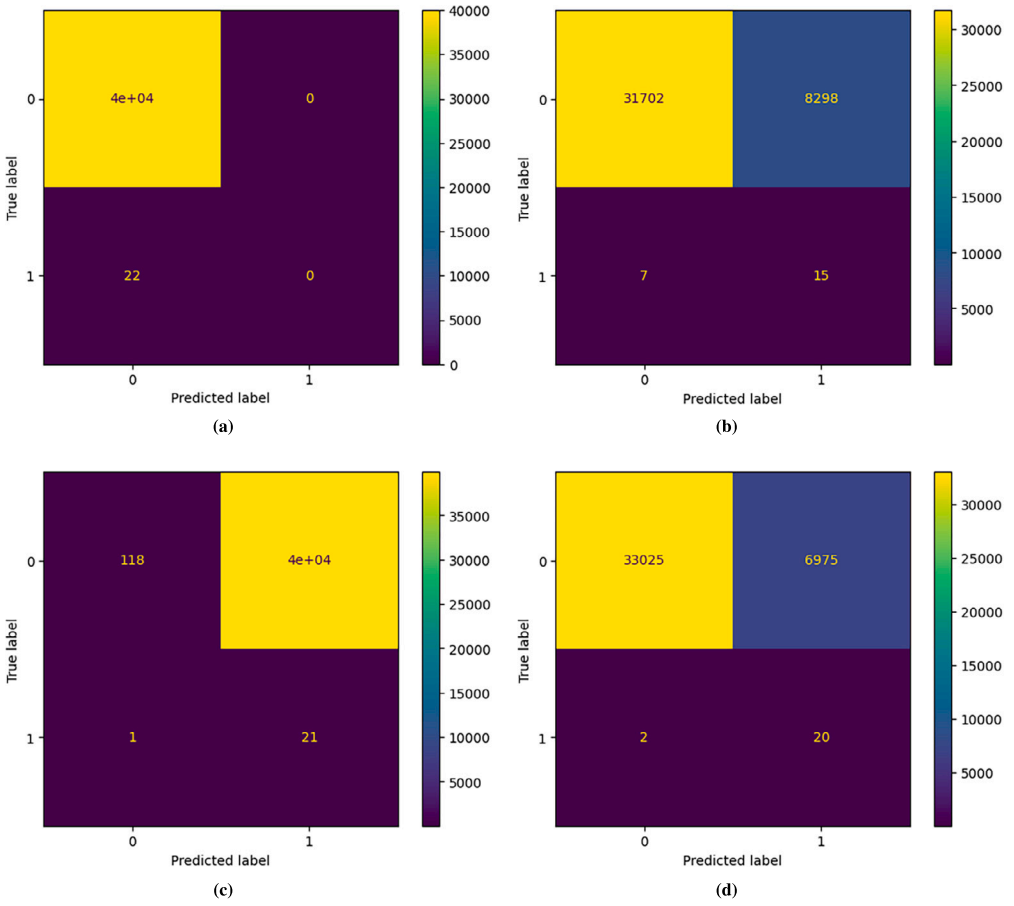


Fig. 6. Confusion matrix for (a) without balancing, (b) ensemble classifier with RUS, (c) ensemble classifier with NearMiss-1, (d) ensemble classifier with XGBCLUS.

Table 3
Comparison among the classifiers without balancing the data.

Classifiers	Accuracy	True positive rate	ROC-AUC score
Decision tree	0.99	0.0	0.55
Gradient boosting	0.99	0.09	0.62
Random forest	0.99	0.0	0.72
Adaptive boosting	0.99	0.05	0.82

Table 4
FPR of machine learning classifiers after under-sampling.

Classifier	Random under-sampling	NearMiss-1	XGBCLUS
Decision tree	0.26	0.99	0.18
Gradient boosting	0.26	0.99	0.19
Random forest	0.20	0.99	0.16
Adaptive boosting	0.26	0.99	0.21
Ensemble (stacked)	0.22	0.99	0.15
Ensemble (hard-voting)	0.21	0.99	0.14
Ensemble (soft-voting)	0.21	0.99	0.17

In Fig. 6, the confusion matrices illustrate the performance of the ensemble classifier under different under-sampling methods, namely RUS, NearMiss-1, and XGBCLUS. Notably, the TP values exhibit an increase compared to scenarios without balancing, where TP values are consistently zero. However, it is essential to discern that, despite the improvements in TP, the FP values show variations among the under-sampling methods. Specifically, NearMiss-1 displays a relatively high FP count compared to RUS and XGBCLUS, even though the FP value is zero in the absence of balancing. The XGBCLUS under-sampling method proposed in our study outperforms the existing methods by achieving the highest TP value along with relatively low FP values. This superiority stems from our algorithm's approach of considering all instances in down-sampling, whereas existing algorithms randomly select instances, leading to the omission of important cases.

Given that the TPR or sensitivity signifies the count of correctly classified positive transactions, down-sampling techniques were explored to equalize the numbers of normal and anomalous transactions. We employed XGBCLUS, our proposed under-sampling method, in conjunction with other established techniques for down-sampling. Fig. 7 illustrates that the sensitivity values for all single and ensemble ML classifiers witnessed an increase, except for the NearMiss-1 under-sampling algorithm. Notably, the sensitivity values for classifiers DT, GBoost, RF, AdaBoost, Ensemble (hard-voting), and Ensemble (soft-voting) utilizing the XGBCLUS algorithm stand at 0.82, 0.86, 0.86, 0.81, 0.86, 0.81, and 0.91, respectively. These values exceed the sensitivity values obtained without balancing and those from random under-sampling techniques.

While the NearMiss-1 under-sampling technique yields a higher sensitivity value compared to the XGBCLUS method, the corresponding FPR value is markedly high, as demonstrated in Table 4. As the FPR decreases, the TNR increases, indicating the correct identification of non-anomalous transactions. The NearMiss-1 under-sampling technique exhibits a higher FPR, suggesting its limitation in accurately identifying non-anomalous transactions. In contrast, the random under-sampling method produces average FPR values, although they are higher than the FPR values of XGBCLUS. In terms of TPR and FPR values, XGBCLUS outperforms other under-sampling techniques.

Fig. 8 illustrates the enhanced ROC-AUC scores obtained through the utilization of under-sampled data. Notably, the proposed XGBCLUS under-sampling technique outperforms RUS and NearMiss-1 in terms of

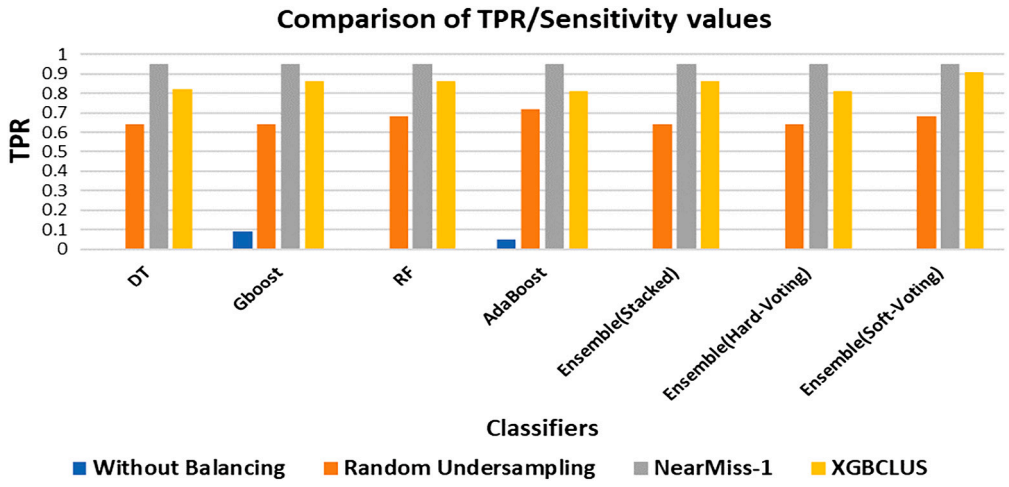


Fig. 7. Comparison of TPR or sensitivity values using under-sampling methods.

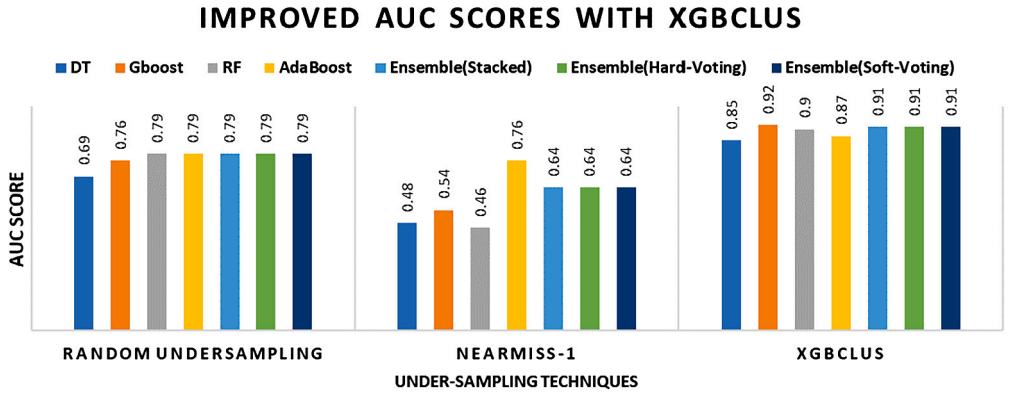


Fig. 8. Improved ROC-AUC value using XGBCLUS.

ROC-AUC scores. The GBoost classifier achieves the highest ROC-AUC score of 0.92, which stands as the peak among all single and ensemble classifiers. Furthermore, the remaining classifiers achieve ROC-AUC scores ranging from 0.85 to 0.91. This range indicates that the true positive and false positive rates adhere to their expected levels.

4.3. Effects of over-sampling in classification

Although the TPR scores have increased for all ML classifiers following the under-sampling of data, the FPR values remain unsatisfactory. The objective is to elevate the TPR while reducing the FPR. To achieve this balance, various over-sampling and combined methods have been applied to balance the training data. In Fig. 9, the confusion matrices depict the performance of the ensemble classifier when employing various over-sampling methods, including SMOTE, ADASYN, SMOTEENN, and SMOTETOMEK. Notably, the TP values exhibit a decrease compared to the results obtained with under-sampling techniques. Despite this reduction in TP values, it is crucial to observe that the FP values have also shown a decrease when contrasted with the figures achieved through under-sampling methods.

Table 5 demonstrates that FPRs have decreased for both individual and ensemble ML classifiers. However, the attained TPRs have reduced. Among all classifiers, AdaBoost attains the highest TPR score of 0.59 with the SMOTETOMEK sampling technique, while the ensemble hard-voting (EHV) classifier secures the lowest FPR value of 0.03, i.e., only 3% across all over-sampling methods. The remaining ML classifiers, including ensemble-stacked (ES) and ensemble soft-voting (ESV), achieve FPR scores between 0.04 and 0.05, except for DT, GBoost, and AdaBoost, which record higher FPR values between 0.06 and 0.08. However, DT records an FPR of 0.05 when using SMOTETOMEK. Moreover, All the ML classifiers have achieved an average TPR value of about 50% except the RF which scores the lowest TPR of 0.23.

Table 5
Comparison between true positive rate and false positive rate values of machine learning classifiers after over-sampling.

	SMOTE		ADASYN		SMOTEENN		SMOTETOMEK	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
DT	0.41	0.06	0.41	0.06	0.36	0.06	0.41	0.05
GBoost	0.45	0.07	0.55	0.06	0.59	0.07	0.50	0.07
RF	0.23	0.04	0.32	0.04	0.36	0.04	0.23	0.04
AdaBoost	0.59	0.08	0.59	0.08	0.55	0.08	0.59	0.08
ES	0.32	0.04	0.41	0.05	0.36	0.04	0.23	0.04
EHV	0.36	0.03	0.36	0.03	0.36	0.04	0.36	0.03
ESV	0.27	0.04	0.36	0.04	0.36	0.04	0.32	0.04

Turning to ROC-AUC scores with over-sampling techniques, Table 6 displays the performance of all ML classifiers. Among all classifiers, AdaBoost secures the highest ROC-AUC values, ranging from 0.80 to 0.85. Conversely, the DT classifier exhibits the lowest ROC-AUC scores with oversampled data. The ADASYN over-sampling technique demonstrates superior performance compared to other sampling methods, yielding favorable ROC-AUC values for all ML classifiers within the range of 0.71 to 0.83, except for DT, which scores 0.58. Overall, the ADASYN over-sampling technique shows a better performance compared to other over- and combined-sampling methods.

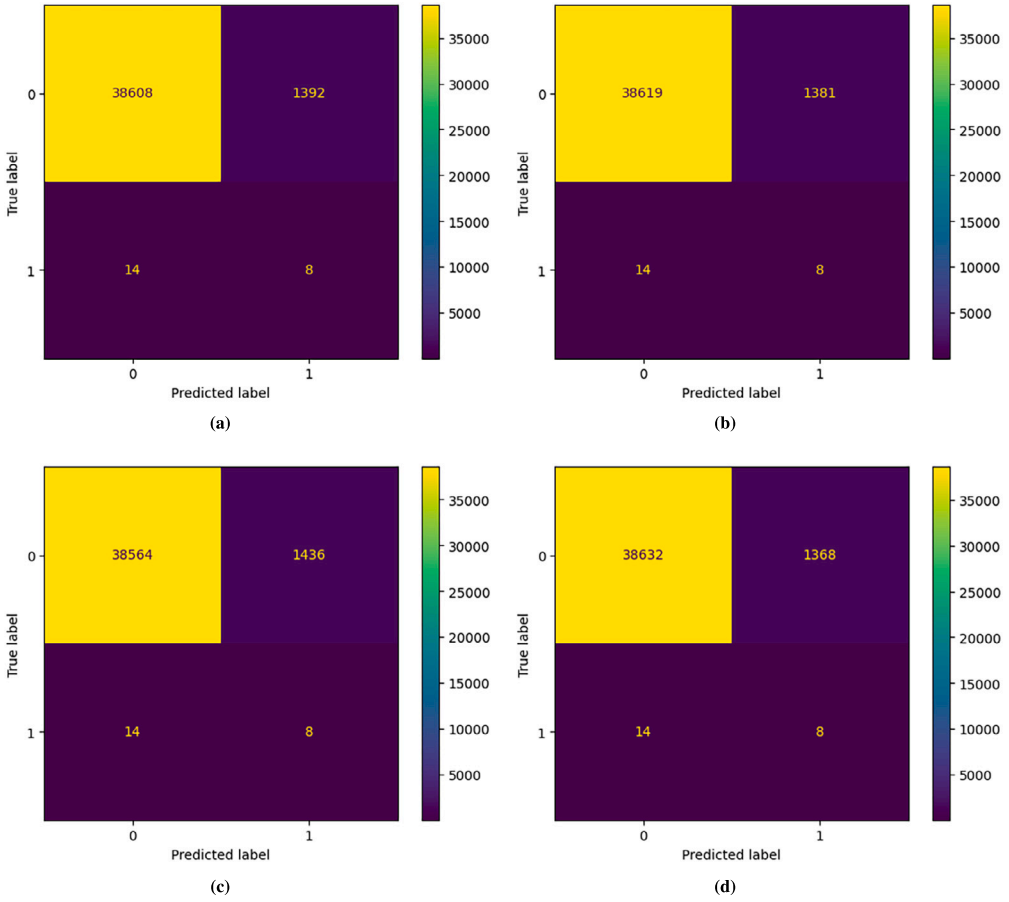


Fig. 9. Confusion matrix for (a) ensemble classifier with SMOTE, (b) ensemble classifier with ADASYN, (c) ensemble classifier with SMOTEENN, (d) ensemble classifier with SMOTETOMEK.

Table 6
ROC-AUC scores after over-sampling the data.

Classifiers	SMOTE	ADASYN	SMOTEENN	SMOTETOMEK
DT	0.59	0.58	0.56	0.59
GBoost	0.77	0.79	0.78	0.84
RF	0.71	0.71	0.71	0.71
AdaBoost	0.83	0.83	0.80	0.85
Ensemble (Stacked)	0.63	0.75	0.75	0.65
Ensemble (Hard-Voting)	0.63	0.75	0.75	0.6
Ensemble (Soft-Voting)	0.63	0.75	0.75	0.6

4.4. Comparative analysis between under-sampling and over-sampling methods

Both under-sampling and over-sampling techniques exert distinct effects on blockchain anomaly detection. Under-sampling methods address the minority class instances by selecting an equivalent number of instances from the majority class through various distance-based techniques [54]. Conversely, over-sampling methods originate from the majority class instances and craft an equal number of synthetic instances from the minority class using different distance techniques [55]. Both sampling techniques yield varying impacts on the classification of Bitcoin transaction data. A comprehensive comparative analysis is detailed from Table 7 to Table 10. The confusion matrices in Figs. 6 and 9 suggest a nuanced trade-off between true positives and false positives, emphasizing the importance of a comprehensive evaluation of the model's performance under different sampling strategies.

Table 7 showcases that TPR scores are relatively high for under-sampling methods such as RUS and the proposed XGBCLUS method, compared to over-sampling and combined sampling methods. Conversely, over-sampling and combined-techniques surpass under-sampling methods in terms of FPR scores. Among under-sampling methods, XGBCLUS secures the highest TPR value of 0.91, which also stands as the pinnacle across all under-sampling, over-sampling, and combined techniques. Among over and combined balancing methods, the highest TPR value is 0.64 achieved through SMOTETOMEK. Under-sampling methods prove more effective than over-sampling methods in increasing TPRs. With under-sampling methods, ML classifiers can accurately identify a larger portion of anomalous transactions.

In terms of FPR scores as shown in Table 8, the best value of 0.03 is achieved across all over-sampling and combined methods. However, under-sampling methods yield a modest FPR score of 0.14 with XGBCLUS. Over-sampling methods prove more effective than under-sampling methods in reducing FPRs. With over-sampling methods, ML classifiers can accurately identify a larger portion of non-anomalous transactions.

Table 7
TPR or sensitivity of ML classifiers after under-sampling and over-sampling.

	Under-sampling		Over-sampling		Combined-sampling	
	RUS	XGBCLUS	SMOTE	ADASYN	SMOTEENN	SMOTETOMEK
DT	0.64	0.82	0.41	0.41	0.36	0.41
GBoost	0.64	0.86	0.45	0.55	0.59	0.55
RF	0.68	0.86	0.23	0.32	0.36	0.23
AdaBoost	0.72	0.81	0.59	0.59	0.55	0.64
ES	0.64	0.86	0.32	0.41	0.36	0.23
EHV	0.64	0.81	0.36	0.36	0.36	0.36
ESV	0.68	0.91	0.27	0.36	0.36	0.32

Table 8
FPR of ML classifiers after under-sampling and over-sampling.

	Under-sampling		Over-sampling		Combined-sampling	
	RUS	XGBCLUS	SMOTE	ADASYN	SMOTEENN	SMOTETOMEK
DT	0.26	0.18	0.06	0.06	0.06	0.05
GBoost	0.26	0.19	0.07	0.06	0.07	0.07
RF	0.20	0.16	0.04	0.04	0.04	0.04
AdaBoost	0.26	0.21	0.08	0.08	0.08	0.08
ES	0.22	0.15	0.04	0.05	0.04	0.04
EHV	0.21	0.14	0.03	0.03	0.04	0.03
ESV	0.21	0.17	0.04	0.04	0.04	0.04

Table 9
Accuracy of single and ensemble classifiers after under-sampling and over-sampling.

	Under-sampling		Over-sampling		Combined-sampling	
	RUS	XGBCLUS	SMOTE	ADASYN	SMOTEENN	SMOTETOMEK
DT	0.74	0.82	0.94	0.94	0.94	0.94
GBoost	0.74	0.81	0.93	0.94	0.93	0.93
RF	0.80	0.83	0.96	0.96	0.96	0.96
AdaBoost	0.74	0.79	0.92	0.92	0.92	0.92
ES	0.78	0.85	0.96	0.95	0.96	0.96
EHV	0.79	0.86	0.96	0.97	0.96	0.97
ESV	0.79	0.83	0.96	0.96	0.96	0.96

Table 10
Evaluation metrics after under-sampling and over-sampling with test data.

	Under-sampling				Over-sampling			
Classifier	Acc	TPR/Sensitivity	FPR	ROC-AUC	Acc	TPR/Sensitivity	FPR	ROC-AUC
DT	0.82	0.82	0.18	0.85	0.94	0.41	0.05	0.58
GBoost	0.81	0.86	0.19	**0.92**	0.94	0.55	0.06	0.79
RF	0.83	0.86	0.16	0.90	0.96	0.36	0.04	0.71
AdaBoost	0.79	0.81	0.21	0.87	0.92	**0.59**	0.08	**0.83**
ES	**0.86**	**0.91**	**0.14**	0.91	**0.97**	0.41	**0.03**	0.75

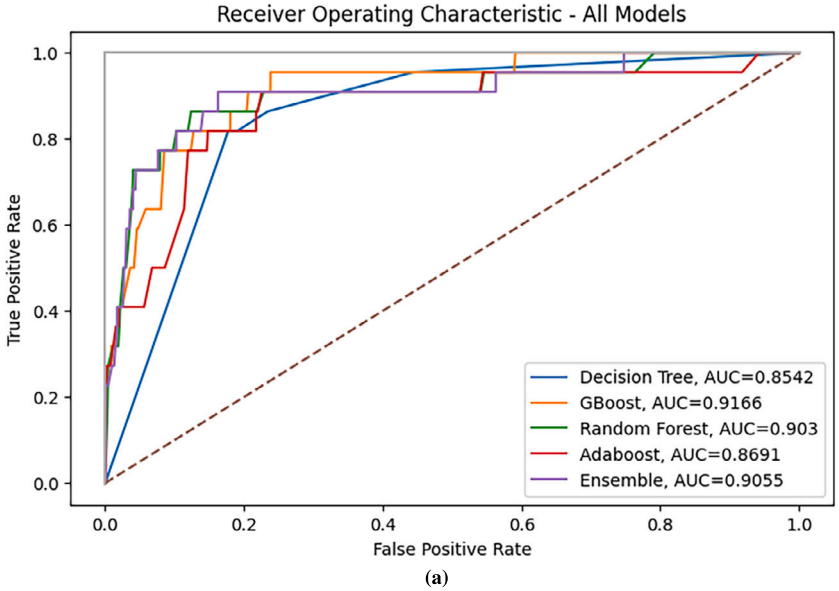
Note: the best values are shown in bold.

Fig. 10 depicts that all ML classifiers exhibit improved ROC-AUC scores using the XGBCLUS under-sampling method, outperforming the scores achieved through the ADASYN over-sampling technique. This suggests that under-sampling methods excel over over-sampling methods in terms of ROC-AUC scores.

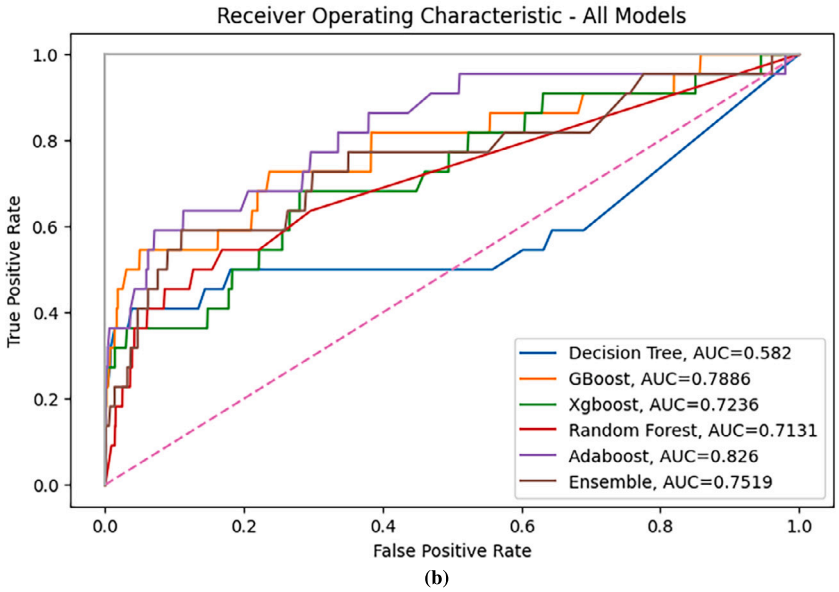
4.5. Effects of ensemble classifiers

While a single tree-based ML classifier can identify both anomalous and non-anomalous transactions, stacked and voting classifiers have been implemented to mitigate issues related to errors and overfitting. As demonstrated by the experimental results in Table 10, ensemble classifiers exhibit superior performance compared to individual tree-based models, particularly for accuracy and FPR values.

Table 9 showcases the enhanced test accuracy of the three ensemble methods in comparison to single classifiers. The voting (hard) classifier attains the highest accuracy of 97%, which stands as the peak value across all single classifiers. Among the ensemble classifiers, the voting (soft) classifier secures the highest TPR or sensitivity value of 0.91 when utilizing under-sampled data from the XGBCLUS method. Concerning



(a)



(b)

Fig. 10. ROC-AUC curves for all models after (a) under-sampling with XGBCLUS, (b) over-sampling with ADASYN.

the FPR, the voting (hard) classifier achieves the best value of 0.03, capitalizing on the XGBCLUS under-sampling technique.

With over-sampled data, although the TPR values for the three ensemble classifiers exhibit a minor decline, the voting (hard) classifier attains the best FPR value of 0.03. Furthermore, ensemble classifiers prove effective in reducing the FPR, thereby enhancing the correct identification of non-anomalous transactions. The voting (hard) classifier outperforms the other two ensemble methods by securing the highest accuracy of 0.97, alongside a commendable ROC-AUC score of 0.75.

In a comparative analysis with two prior studies of [56] and [37] that focused on blockchain anomaly detection using the Bitcoin dataset as ours, Table 11 illustrates a detailed juxtaposition. Notably, our proposed ensemble method showcases superior performance across all metrics. While the studies share the primary objective of identifying malicious transactions, our emphasis extends to the accurate detection of non-malicious transactions as well. In this regard, our study endeavors to mitigate false positive rates, resulting in an enhanced true positive rate. Shafiq [56] delved into various unsupervised Machine Learning approaches to identify anomalous transactions, ultimately finding that the ensemble method outperformed other explored classifiers. Conversely, Sayadi et al. [37] investigated the use of K-means clustering and OCSVM for classifying anomalous Bitcoin transactions. However, both studies encountered challenges, as they exhibited low accuracy and a high FPR. The proposed ensemble method exhibits substantial enhancements in both true positive and false positive values, leveraging preprocessed sampled data obtained through various preprocessing steps, including feature selections. Our ensemble method excels in accuracy and notably the FPR value. A significant distinction among the studies is that our research incorporates explainability, along with the inclusion of decision rules, a component absent in their investigation.

4.6. SHAP-based explainability analysis

SHAP is a powerful technique used to explain the predictions of ML models. It provides a way to attribute the contribution of each feature to a specific prediction, helping us understand why a model made a certain decision. In this study, the SHAP KernelExplainer method is employed to compute SHAP values for explaining the ensemble model's behavior, both for individual instances and the entire dataset. SHAP values are indicated by the blue color, representing their average impact on detect-

Table 11
Comparison between proposed and existing work for Bitcoin anomaly detection.

References	Model	Accuracy	FPR	Explainability	Anomaly rules
[56]	Ensemble classifiers	0.94	0.05	No	No
[37]	OCSVM	0.90	0.09	No	No
Our study	Ensemble classifiers	0.97	0.03	Yes	Yes

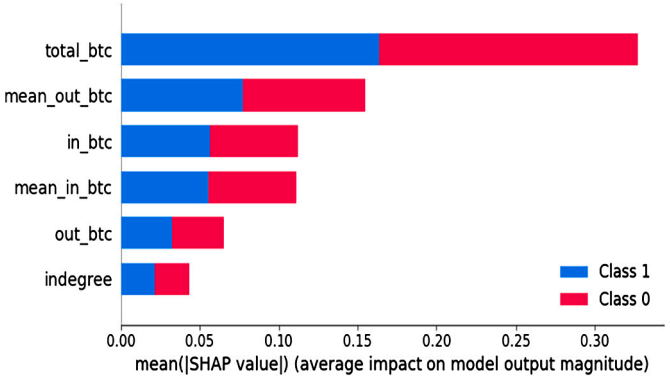


Fig. 11. Hierarchy of the features contributing to classification.

ing anomalous transactions, while normal transactions are represented by red color SHAP values. Fig. 11 provides a hierarchical summary of features, ordered by their contributions to the model's output. The magnitude of the SHAP values indicates the strength of the feature's influence. Features with larger absolute SHAP values have a more substantial impact on the anomaly prediction. Notably, the feature "total_btc" exerts the most substantial average impact on the model's classification, whereas "indegree" contributes the least. Furthermore, the second-highest mean SHAP value is attributed to "mean_out_btc," playing a significant role in anomaly detection. Conversely, "out_btc" contributes the second least to the classification. "in_btc" and "mean_in_btc" exhibit nearly equivalent contributions to the classification of Bitcoin transactions.

Fig. 12 illustrates a comparison of SHAP values for four randomly selected anomalous and normal transactions, as predicted by the ensemble model post both under-sampling and over-sampling techniques. As depicted in Fig. 12, the baseline value for the expected anomalous transaction is 0.49 under under-sampling and 0.50 under over-sampling. Notably, the scores for anomalous Bitcoin transactions are 0.06 and 0.40 with under-sampled and over-sampled data, respectively. These scores are notably lower than the baseline.

The SHAP value representations in Figs. 12a and 12b elucidate the features contributing the most to lowering the score. Fig. 12a underscores that "total_btc" holds the highest importance, followed by "in_btc," "mean_out_btc," and "indegree." For normal Bitcoin transaction data, the anomalous transaction scores are 0.81 and 0.86 with under-sampled and over-sampled data, respectively. These scores exceed the baseline. SHAP value representations for normal transactions are displayed in Figs. 12c and 12d. As evident from the figures, the SHAP values are generally small, except for "mean_out_btc" and "indegree" under under-sampling and over-sampling, respectively.

Comparing all four cases in Fig. 12, it is evident that the feature values for anomalous instances are higher than those for normal instances, making them discernible to human observation. To facilitate a better human understanding, a comparison between actual anomalous and normal data instances is presented in Table 12. This table indicates that the features identified by SHAP values in Fig. 12 exhibit significant disparities in actual transaction data. Hence, it can be concluded that SHAP values are effective in explaining both normal and anomalous Bitcoin transactions.

Table 12
Comparison of actual values of an anomalous and normal Bitcoin transaction.

Feature name	Anomalous	Normal
indegree	7	2
in_btc	2902	15.96
out_btc	2902	15.96
total_btc	5804	31.92
mean_in_btc	414.6	7.98
mean_out_btc	1451	5.32

Table 13
Feature importance values.

Feature name	Importance
total_btc	0.666268
mean_in_btc	0.124338
mean_out_btc	0.092236
in_btc	0.046185
out_btc	0.036751
indegree	0.034222

4.7. Anomaly rule generation and interpretability analysis

Interpreting anomaly rules using tree representations can help in understanding why certain instances are classified as anomalies. It provides a step-by-step breakdown of the decision process and highlights which features and thresholds played a crucial role in making the decision. In this study, having explored tree-based ML classifiers for detecting anomalous Bitcoin transactions, tree visualization offers a structured and interpretable approach to comprehending how the model arrives at decisions using input features. A visualization of the decision tree with max-depth 10 is shown in Fig. 13. The top node of the tree is called the root node, and it represents the entire dataset. Each subsequent node represents a decision point based on a specific feature and threshold. Moving down the sub-trees, each node represents a feature and a corresponding threshold. Instances are directed to different branches based on whether their feature values satisfy the given threshold. Each leaf node corresponds to a specific prediction class, in this case, anomalous or non-anomalous.

In Table 13, the descending order of feature importance is presented, with numerical values representing the total normalized reduction in Gini impurity achieved by splitting the respective feature throughout the tree. Unsurprisingly, 'total_btc,' the feature utilized to split at the root node, holds the highest importance. This implies that 'total_btc,' followed by 'mean_in_btc', are the most crucial features in determining the anomaly status of a Bitcoin transaction.

By traversing the decision tree from the root node to a specific leaf node, we can extract the sequence of decisions (feature and threshold combinations) that lead to an anomaly prediction. These decisions essentially form the "anomaly rules" for that instance. In Table 14, we have a set of rules along with the confidence scores that lead to an anomalous node regarding the tree in Fig. 13. Decision trees also provide a measure of feature importance. Features closer to the root of the

Fig. 12. Features contributing to (a) detect a positive case after under-sampling, (b) detect a positive case after over-sampling, (c) detect a negative case after under-sampling, (d) detect a negative case after over-sampling.

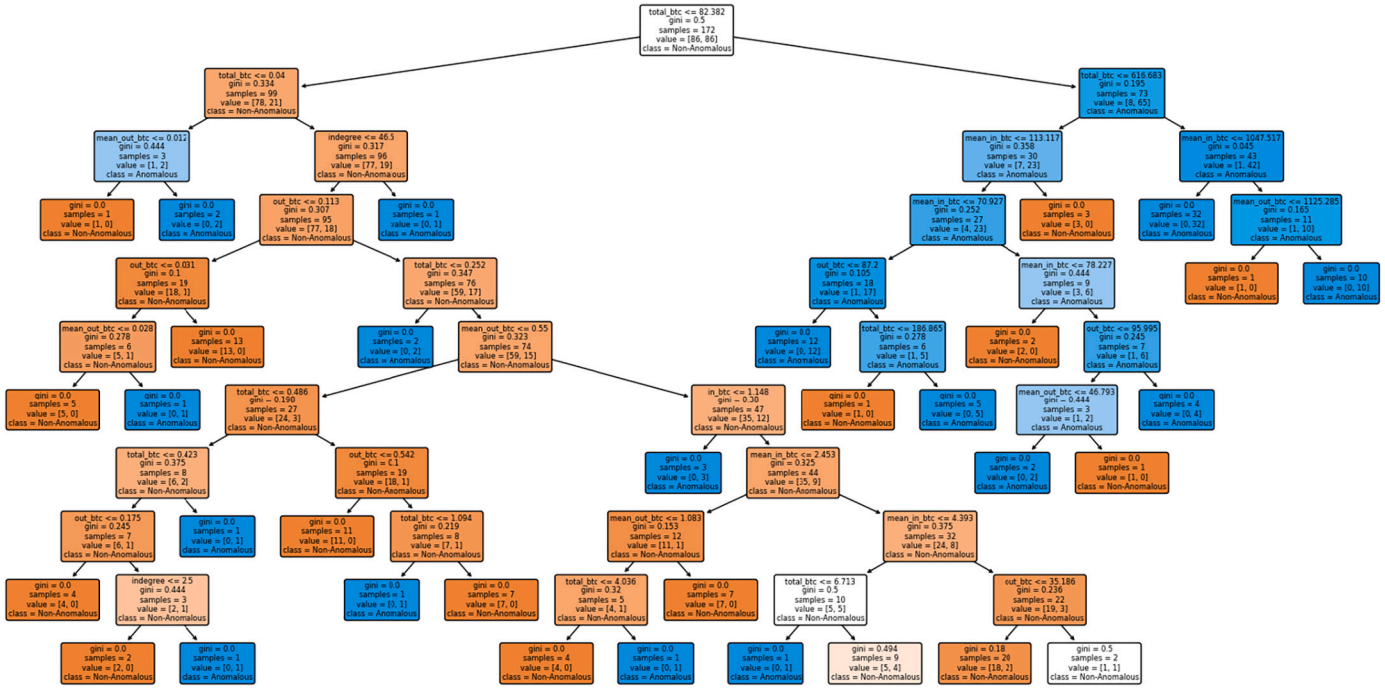


Fig. 13. Visualization of DT of depth 10 for generating anomalous rules.

Table 14
Significant rules for being an anomalous Bitcoin transaction.

Anomaly rule	Class	Total Samples	Correctly identified	Confidence (%)
1. If (total_btc is greater than 616.683 and mean_in_btc is less than or equal to 1047.517) then	Anomalous	32	32	100
2. If (total_btc is greater than 616.683, mean_in_btc is greater than 1047.517, and mean_out_btc is greater than 1125.285) then	Anomalous	10	10	100
3. If (total_btc is greater than 82.38 and total_btc is less than or equal to 616.683, mean_in_btc is less than or equal to 70.927, and out_btc is less or equal to 87.2) then	Anomalous	12	12	100
4. If (total_btc is greater than 82.38 and total_btc is less than or equal to 616.683 and mean_in_btc is less than or equal to 70.927) then	Anomalous	18	17	94
5. If (total_btc is greater than 616.683 and mean_in_btc is greater than 1047.517) then	Anomalous	11	10	91

tree have a more significant influence on the overall decision-making process. So total_btc is the most important feature that contributes the highest to deciding whether a transaction is anomalous or not. Moreover, both mean_in_btc and mean_out_btc have valuable effects on classifying anomalous transactions. However, in some cases, the indegree may also contribute to detecting anomalous transactions. The confidence score reflects the model's confidence or certainty in its prediction that the transaction is anomalous based on the specified conditions in the rules. Rules 1 to 3 give the highest confidence scores, i.e., the highest probability for being anomalous. However, rules 4 and 5 have an average probability for the Bitcoin transactions to be anomalous. From Table 14, the more thorough the examination of decision rules, the greater the confidence in accurately identifying anomalous transactions. Although we've specifically crafted anomaly rules for our Bitcoin transaction data, this methodology holds promise for creating effective decision rules in diverse domains, particularly within the realm of blockchain.

5. Discussion

Detecting anomalies from highly imbalanced data presents a formidable challenge. Although tree-based ML classifiers demonstrate effectiveness in anomaly detection, the significant data imbalance tends to bias these classifiers towards non-anomalous transactions. As demonstrated in Table 3, this imbalance often results in low TPRs for single tree-based ML classifiers. Given that accurate detection of anomalous transactions is the primary objective of this study, addressing data imbalance before model training becomes crucial.

Two main approaches for balancing data, over-sampling and under-sampling techniques, are available. The selection of the appropriate technique is a critical decision for researchers, particularly when dealing with highly imbalanced data. Over-sampling methods generate synthetic data to balance both majority and minority classes, but they may not effectively detect most anomalous transactions. On the other hand, under-sampling techniques aim to balance classes by focusing on minority samples, leading to high TPR and low FPR values.

For anomaly classification problems, an under-sampling technique could be suitable if it can maximize the TP value while minimizing the FP value. Regrettably, the prevalent down-sampling algorithms in existence often fall short of achieving this balance, as illustrated in the results analysis section. While the RUS technique can assist in achieving better class balance, it does come with drawbacks. This technique decreases the volume of training data available, potentially resulting in the loss of crucial samples and subsequently leading to suboptimal model performance. The NearMiss-1 technique, although contributing to enhanced model performance with a TPR score of 0.95, exhibits a significant FPR. Recognizing the limitations of RUS and NearMiss-1, we have introduced the XGBCLUS under-sampling technique to address these issues. XGBCLUS surpasses its predecessors in detecting anomalous Bitcoin transactions. Both individual and ensemble classifiers demonstrate superior TPR, FPR, and ROC-AUC values when trained on under-sampled data using XGBCLUS, as demonstrated in Section 4.2.

Furthermore, our investigation extends to over-sampling and combined data balancing techniques, aiming to highlight the distinctions between under-sampling and over-sampling approaches. Comparative results in Section 4.4 reveal that under-sampling methods are effective in improving the TP value but lead to a higher FPR. Conversely, over-sampling methods contribute to achieving an increase in TPR and a decrease in FPR compared to unbalanced data. Among the various over- and combined-sampling techniques, ADASYN outperforms SMOTE, while SMOTEENN proves superior to SMOTETOMEK.

In the realm of classification, we introduce both stacked and voting (hard and soft) ensemble classifiers in addition to individual tree-based ML classifiers. It's evident that ensemble classifiers excel over individual ML classifiers when applied to both under- and over-sampled data. Among the three ensemble classifiers, the hard-voting classifier demonstrates superior performance, achieving the highest TPR and lowest FPR values for both under- and over-sampled data. In situations involving imbalanced data, the FPR assumes significance as an essential evaluation metric. The ensemble classifiers consistently yield improved FPR values compared to single ML classifiers, emphasizing the effectiveness of ensemble approaches in handling imbalanced data.

Nonetheless, ML models often act as opaque "black boxes". In order to substantiate human assumptions and model predictions, we delve into the realm of explainable AI using SHAP, a facet elaborated in Section 4.6. Aggregating the SHAP values across the entirety of dataset instances enables the identification of the most pivotal feature, which, in this context, is "total_btc". This salient feature's importance is also discernible to human observers. Furthermore, we expound upon anomaly rules derived from DTs. As illustrated in Fig. 13, "total_btc" emerges as the root node, signifying its paramount role in classifying anomalous transactions. Both SHAP's analysis and the tree representation collectively pinpoint "total_btc" as the quintessential feature contributing to the identification of Bitcoin anomalous transactions. Overall, SHAP provides a transparent and interpretable way to understand how each feature contributes to the anomaly detection process. This can help data analysts and domain experts identify patterns, correlations, and potential anomalies in the data that the model is leveraging for its predictions. In addition, interpreting anomaly rules using tree representations can

help in understanding why certain instances are classified as anomalies. It provides a step-by-step breakdown of the decision process and highlights which features and thresholds played a crucial role in making the decision. This transparency is especially valuable in domains where explainability is important, allowing stakeholders to validate the model's decisions and identify potential issues in the data or model.

The findings of our study bear significant implications for enhancing both blockchain security and anomaly detection. The efficacy of the proposed ensemble model in detecting anomalous transactions in Bitcoin signifies a promising approach to enhance the security of blockchain systems. Additionally, the incorporation of XAI techniques, including SHAP analysis, along with anomaly rules for interpretability analysis, contributes transparency and clarity to the anomaly detection process. Implementing such explainability measures can contribute to robust anomaly detection systems and, consequently, fortify the overall security of blockchain technologies. Additionally, the exploration of various sampling techniques, including the proposed under-sampling and combined-sampling methods, contributes to the development of strategies for handling highly imbalanced datasets in the blockchain domain. This has broader implications for anomaly detection in other contexts where imbalanced data is a common challenge.

In future research endeavors, there is potential to delve into the efficacy of our proposed under-sampling techniques, alongside over-sampling methods, for anomaly detection in different blockchain domains such as Ethereum transactions [57], credit card fraud detection [58], and money laundering [59]. The exploration of a data-driven approach [60] could lead to the development of real-time blockchain anomaly detection systems. In future research, we plan to gather additional data related to blockchain and perform further experimental analyses. Furthermore, a comparative analysis between ML and deep learning algorithms could be undertaken to determine the optimal model for anomaly detection in blockchain transactions. This pursuit holds promise for enhancing the precision and effectiveness of anomaly detection methodologies.

6. Conclusion

In this study, we have conducted an extensive comparative analysis aimed at detecting anomalies within blockchain transaction data. While numerous studies have been conducted in this field, a prevailing limitation has been the absence of explanations for model predictions. To address this shortcoming, our study endeavors to combine XAI techniques and anomaly rules with tree-based ensemble classifiers using Bitcoin transactions. Notably, the SHAP method plays a pivotal role in quantifying the contribution of each feature toward predicting the model's output. Moreover, the anomaly rules help to interpret whether a Bitcoin transaction is anomalous or not. Consequently, one can readily identify which features play a pivotal role in anomaly detection. To further enhance our methodology, we have developed an under-sampling algorithm termed XGBCLUS. This algorithm facilitates the balance between anomalous and non-anomalous transaction data, and its performance has been compared against other well-known under-sampling and over-sampling techniques. Subsequently, the results derived from various single tree-based classifiers are juxtaposed against those obtained from stacking and voting ensemble classifiers. Our findings unequivocally demonstrate that our proposed under-sampling method, XGBCLUS, has yielded improvements in TPR and ROC-AUC scores. Additionally, ensemble classifiers have exhibited superior performance in comparison to popular single ML classifiers. In conclusion, our study highlights that ensemble classifiers, when combined with suitable balancing techniques, prove effective in detecting Blockchain anomalous transactions.

CRediT authorship contribution statement

Mohammad Hasan: Writing – original draft, Methodology, Data curation, Implementation, Experimental analysis, Conceptualization. **Mohammad Shahriar Rahman:** Writing – review & editing. **Helge Janicke:** Writing – review & editing. **Iqbal H. Sarker:** Writing – review & editing, Conceptualization and Supervision.

Declaration of competing interest

The authors declare no conflict of interests.

References

[1] M. Nofer, P. Gomber, O. Hinz, et al., Blockchain, Bus. Inf. Syst. Eng. 59 (3) (2017) 183–187, https://doi.org/10.1007/s12599-017-0467-3.

[2] S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system, https://bitcoin.org/bitcoin.pdf, 2008.

[3] D. Yaga, P. Mell, N. Roby, et al., Blockchain technology overview, arXiv, 2019, preprint arXiv:1906.11078, 2019.

[4] A.A. Monrat, O. Schelén, K. Andersson, A survey of blockchain from the perspectives of applications, challenges, and opportunities, IEEE Access 7 (2019) 117134–117151, https://doi.org/10.1109/ACCESS.2019.2936094.

[5] M. Saad, V. Cook, L. Nguyen, et al., Partitioning attacks on bitcoin: colliding space, time, and logic, in: Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2019, pp. 1175–1187, https://doi.org/10.1109/ICDCS.2019.00119.

[6] M.U. Hassan, M.H. Rehmani, J. Chen, Anomaly detection in blockchain networks: a comprehensive survey, IEEE Commun. Surv. Tutor. 25 (1) (2022), https://doi.org/10.1109/COMST.2022.3205643.

[7] M. Signorini, M. Pontecorvi, W. Kanoun, et al., Advise: anomaly detection tool for blockchain systems, in: Proceedings of the 2018 IEEE World Congress on Services (SERVICES), IEEE, 2018, pp. 65–66, https://doi.org/10.1109/SERVICES.2018.00046.

[8] T. Ashfaq, R. Khalid, A.S. Yahaya, et al., A machine learning and blockchain based efficient fraud detection mechanism, Sensors 22 (19) (2022) 7162, https://doi.org/10.3390/s22197162.

[9] V. Ganganwar, An overview of classification algorithms for imbalanced datasets, Int. J. Emerg. Technol. Adv. Eng. 2 (4) (2012) 42–47.

[10] S. El Hajjami, J. Malki, M. Berrada, et al., Machine learning for anomaly detection. Performance study considering anomaly distribution in an imbalanced dataset, in: Proceedings of the 2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech), IEEE, 2020, pp. 1–8, https://doi.org/10.1109/CloudTech49835.2020.9365887.

[11] J. Han, J. Woo, J.W.-K. Hong, Oversampling techniques for detecting bitcoin illegal transactions, in: Proceedings of the 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS), IEEE, 2020, pp. 330–333, https://doi.org/10.23919/APNOMS50412.2020.9236780.

[12] R. Ahsan, W. Shi, X. Ma, et al., A comparative analysis of cgan-based oversampling for anomaly detection, IET Cyber Phys. Syst. Theory Appl. 7 (1) (2022) 40–50, https://doi.org/10.1049/cps2.12019.

[13] M. Saripuddin, A. Suliman, S.S. Sameon, et al., Random undersampling on imbalance time series data for anomaly detection, in: Proceedings of the 2021 4th International Conference on Machine Learning and Machine Intelligence, ACM, 2021, pp. 151–156, https://doi.org/10.1145/3490725.3490748.

[14] R.A. Alsowail, An insider threat detection model using one-hot encoding and NearMiss under-sampling techniques, in: M.S. Uddin, P.K. Jamwal, J.C. Bansal (Eds.), Algorithms for Intelligent Systems, Springer, Singapore, 2022, pp. 183–196, https://doi.org/10.1007/978-981-19-0332-8_13.

[15] I.H. Sarker, Machine learning for intelligent data analysis and automation in cybersecurity: current and future prospects, Ann. Data Sci. 10 (6) (2023) 1473–1498, https://doi.org/10.1007/s40745-022-00444-2.

[16] M. Rashid, J. Kamruzzaman, T. Imam, et al., A tree-based stacking ensemble technique with feature selection for network intrusion detection, Appl. Intell. 52 (9) (2022) 9768–9781, https://doi.org/10.1007/s10489-021-02968-1.

[17] Y. Zhou, G. Cheng, S. Jiang, et al., Building an efficient intrusion detection system based on feature selection and ensemble classifier, Comput. Netw. 174 (2020) 107247, https://doi.org/10.1016/j.comnet.2020.107247.

[18] Y. Xia, K. Chen, Y. Yang, Multi-label classification with weighted classifier selection and stacked ensemble, Inf. Sci. 557 (2021) 421–442, https://doi.org/10.1016/j.ins.2020.06.017.

[19] T.H. Yang, Y.T. Lin, C.L. Wu, et al., Voting-based ensemble model for network anomaly detection, in: Proceedings of the ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021, pp. 8543–8547, https://doi.org/10.1109/ICASSP39728.2021.9414532.

[20] N.T. Pham, E. Foo, S. Suriadi, et al., Improving performance of intrusion detection system using ensemble methods and feature selection, in: Proceedings of the Australasian Computer Science Week Multiconference, ACM, 2018, pp. 1–6, https://doi.org/10.1145/3167918.3167951.

[21] I.H. Sarker, AI-driven cybersecurity and threat intelligence: cyber automation, intelligent decision-making and explainability, 1st ed., Springer, Cham, 2024, https://doi.org/10.1007/978-3-031-54497-2.

[22] I.R. Ward, L. Wang, J. Lu, et al., Explainable artificial intelligence for pharmacovigilance: what features are important when predicting adverse outcomes?, Comput. Methods Programs Biomed. 212 (2021) 106415, https://doi.org/10.1016/j.cmpb.2021.106415.
[23] M.T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?" explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 1135–1144, https://doi.org/10.1145/2939672.2939778.
[24] S.M. Lundberg, S.I. Lee, A unified approach to interpreting model predictions, in: I. Guyon, U. Von Luxburg, S. Bengio, et al. (Eds.), Advances in Neural Information Processing Systems 30 (NIPS 2017), Curran Associates, Red Hook, NY, 2020, pp. 1–8, https://dl.acm.org/doi/10.5555/3295222.3295230.
[25] T. Chen, T. He, M. Benesty, et al., Xgboost: Extreme Gradient Boosting. R Package Version 0.4-2, 2015.
[26] G. Biau, E. Scornet, A random forest guided tour, Test 25 (2) (2016) 197–227, https://doi.org/10.1007/s11749-016-0481-7.
[27] H. Sharma, S. Kumar, et al., A survey on decision tree algorithms of classification in data mining, Int. J. Sci. Res. (IJSR) 5 (4) (2016) 2094–2097, https://doi.org/10.21275/v5i4.nov162954.
[28] A. Natekin, A. Knoll, Gradient boosting machines, a tutorial, Front. Neurorobot. 7 (2013) 21, https://doi.org/10.3389/fnbot.2013.00021.
[29] R. Rojas, Adaboost and the super bowl of classifiers a tutorial introduction to adaptive boosting, https://www.inf.fu-berlin.de/inst/ag-ki/adaboost4.pdf, 2009.
[30] Z. Zheng, H.N. Dai, J. Wu, Blockchain intelligence: when blockchain meets artificial intelligence, arXiv, 2019, preprint arXiv:1912.06485, 2019.
[31] B. Chen, F. Wei, C. Gu, Bitcoin theft detection based on supervised machine learning algorithms, Secur. Commun. Netw. 2021 (2021) 6643763, https://doi.org/10.1155/2021/6643763.
[32] H.S. Yin, R. Vatrapu, A first estimation of the proportion of cybercriminal entities in the bitcoin ecosystem using supervised machine learning, in: Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), IEEE, 2017, pp. 3690–3699, https://doi.org/10.1109/BigData.2017.8258365.
[33] A. Singh, Anomaly detection in the Ethereum network, MS Thesis, Indian Institute of Technology Kanpur, Kanpur, India, 2019.
[34] J. Lorenz, M.I. Silva, D. Aparício, et al., Machine learning methods to detect money laundering in the bitcoin blockchain in the presence of label scarcity, in: Proceedings of the First ACM International Conference on AI in Finance, ACM, 2020, pp. 1–8, https://doi.org/10.1145/3383455.3422549.
[35] I. Alarab, S. Prakoonwit, M.I. Nacer, Comparative analysis using supervised learning methods for anti-money laundering in bitcoin, in: Proceedings of the 2020 5th International Conference on Machine Learning Technologies, ACM, 2020, pp. 11–17, https://doi.org/10.1145/3409073.3409078.
[36] T. Pham, S. Lee, Anomaly detection in bitcoin network using unsupervised learning methods, arXiv, 2016, preprint arXiv:1611.03941, 2016.
[37] S. Sayadi, S.B. Rejeb, Z. Choukair, Anomaly detection model over blockchain electronic transactions, in: Proceedings of the 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), IEEE, 2019, pp. 895–900, https://doi.org/10.1109/IWCMC.2019.8766765.
[38] G.D. Arya, K.V.S. Harika, D.V. Rahul, et al., Analysis of unsupervised learning algorithms for anomaly mining with bitcoin, in: S. Agrawal, K. Kumar Gupta, J.H. Chan, et al. (Eds.), Machine Intelligence and Smart Systems. Algorithms for Intelligent Systems, Springer, Singapore, 2021, pp. 365–373, https://doi.org/10.1007/978-981-33-4893-6_32.
[39] P. Monamo, V. Marivate, B. Twala, Unsupervised learning for robust Bitcoin fraud detection, in: Proceedings of the 2016 Information Security for South Africa (ISSA), IEEE, 2016, pp. 129–134, https://doi.org/10.1109/ISSA.2016.7802939.
[40] F. Scicchitano, A. Liguori, M. Guarascio, et al., A deep learning approach for detecting security attacks on blockchain, in: CEUR Workshop Proceedings, CEUR-WS, 2020, pp. 212–222.
[41] J. Hirshman, Y. Huang, S. Macke, Unsupervised approaches to detecting anomalous behavior in the bitcoin transaction network, Technical report, Stanford University, 2013.
[42] Y. Li, Y. Cai, H. Tian, et al., Identifying illicit addresses in bitcoin network, in: Z. Zheng, H.N. Dai, X. Fu, et al. (Eds.), Blockchain and Trustworthy Systems, Springer, Singapore, 2020, pp. 99–111, https://doi.org/10.1007/978-981-15-9213-3_8.
[43] I. Alarab, S. Prakoonwit, Effect of data resampling on feature importance in imbalanced blockchain data: comparison studies of resampling techniques, Data Sci. Manag. 5 (2) (2022) 66–76, https://doi.org/10.1016/j.dsm.2022.04.003.
[44] S. Taneja, B. Suri, C. Kothari, Application of balancing techniques with ensemble approach for credit card fraud detection, in: Proceedings of the 2019 International Conference on Computing, Power and Communication Technologies (GUCON), IEEE, 2019, pp. 753–758.
[45] H. Ahmad, B. Kasasbeh, B. Aldabaybah, et al., Class balancing framework for credit card fraud detection based on clustering and similarity-based selection (SBS), Int. J. Inf. Technol. 15 (2023) 325–333, https://doi.org/10.1007/s41870-022-00987-w.
[46] B. Prasetiyo, M. Muslim, N. Baroroh, et al., Evaluation Performance Recall and F2 Score of Credit Card Fraud Detection Unbalanced Dataset Using Smote Oversampling Technique, J. Phys.: Conf. Ser. 1918 (4) (2021) 042002, https://doi.org/10.1088/1742-6596/1918/4/042002.
[47] W. Yang, Y. Zhang, K. Ye, et al., Ffd: a federated learning based method for credit card fraud detection, in: K. Chen, S. Seshadri, L.J. Zhang (Eds.), Big Data – BigData 2019, Springer, Cham, 2019, pp. 18–32, https://doi.org/10.1007/978-3-030-23551-2_2.
[48] F. Itoo, Meenakshi, S. Singh, Comparison and analysis of logistic regression, naïve Bayes and knn machine learning algorithms for credit card fraud detection, Int. J. Inf. Technol. 13 (2021) 1503–1511, https://doi.org/10.1007/s41870-020-00430-y.
[49] S. Xuan, G. Liu, Z. Li, et al., Random forest for credit card fraud detection, in: Proceedings of the 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC), IEEE, 2018, pp. 1–6, https://doi.org/10.1109/ICNSC.2018.8361343.
[50] F. Ahmed, M. Hasan, M.S. Hossain, et al., Comparative performance of tree based machine learning classifiers in product backorder prediction, in: P. Vasant, G.-W. Weber, J.A. Marmolejo-Saucedo, et al. (Eds.), Intelligent Computing & Optimization, Springer, Cham, 2022, pp. 572–584, https://doi.org/10.1007/978-3-031-19958-5_54.
[51] N. Rout, D. Mishra, M.K. Mallick, Handling imbalanced data: a survey, in: M. Reddy, K. Viswanath, S. K.M. (Eds.), Advances in Intelligent Systems and Computing, Springer, Singapore, 2018, pp. 431–443, https://doi.org/10.1007/978-981-10-5272-9_39.
[52] S. Rajagopal, P.P. Kundapur, K.S. Hareesha, A stacking ensemble for network intrusion detection using heterogeneous datasets, Secur. Commun. Netw. 2020 (2020) 4586875, https://doi.org/10.1155/2020/4586875.
[53] J.E. King, Binary logistic regression, in: J. Osborne (Eds.), Best Practices in Quantitative Methods, SAGE Publications, Thousand Oaks, CA, 2008, pp. 358–384, https://doi.org/10.4135/9781412995627.d29.
[54] X.Y. Liu, J. Wu, Z.H. Zhou, Exploratory undersampling for class-imbalance learning, IEEE Trans. Syst. Man Cybern., Part B, Cybern. 39 (2) (2008) 539–550, https://doi.org/10.1109/TSMCB.2008.2007853.
[55] A. Gosain, S. Sardana, Handling class imbalance problem using oversampling techniques: a review, in: Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, 2017, pp. 79–85, https://doi.org/10.1109/ICACCI.2017.8125820.
[56] O. Shafiq, Anomaly detection in blockchain, MS Thesis, Tampere University, Tampere, Finland, 2019.
[57] S. Tikhomirov, Ethereum: state of knowledge and research perspectives, in: A. Imine, J. Fernandez, J.Y. Marion, et al. (Eds.), Foundations and Practice of Security, Springer, Cham, 2018, pp. 206–221, https://doi.org/10.1007/978-3-319-75650-9_14.
[58] V.N. Dornadula, S. Geetha, Credit card fraud detection using machine learning algorithms, Proc. Comput. Sci. 165 (2019) 631–641, https://doi.org/10.1016/j.procs.2020.01.057.
[59] Z. Chen, L.D. Van Khoa, E.N. Teoh, et al., Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: a review, Knowl. Inf. Syst. 57 (2) (2018) 245–285, https://doi.org/10.1007/s10115-017-1144-z.
[60] I.H. Sarker, Data science and analytics: an overview from data-driven smart computing, decision-making and applications perspective, SN Comput. Sci. 2 (5) (2021) 377, https://doi.org/10.1007/s42979-021-00765-8.