RESEARCH TREATISE

A Dissertation submitted to
SVKM's NMIMS (Deemed-to-be-University)
In Partial Fulfilment for the Degree of

M.Sc. Statistics and Data Science 2024-2026

"Anomaly Detection in Bitcoin Blockchain Transactions with Advanced ML and XAI"

By: Group-3

Roll No.	Name	SAP ID
A011	Vaibhavi Deo	86062400050
A012	Jayesh Deore	86062400067
A014	Zeal Dmello	86062400051
A015	Serena Dmello	86062400059

Project Mentor
Dr. Pradnya Khandeparkar
Prof. Pratik Desai

ACKNOWLEDGMENT

We would like to take this opportunity to sincerely thank our mentor,
Prof. Pratik Desai, for his help during the project. He gave us suggestions for topic selection. He was very communicative, which made it easier for us to reach out for assistance when we needed. He gave us advice regarding ways to make our project better. It would have been challenging to proceed with this project without his direction.
Also, we are grateful to **Dr. Pradnya Khandeparkar**, our course coordinator, for providing necessary resources required for completion of the project.
We also want to express our appreciation to the entire Department of Statistics for helping and mentoring us throughout the course of our project.
Finally, with the help of our team members and expert direction of our mentors we are able to complete our project on time.

TABLE OF CONTENTS

Contents

ABSTRACT **3**

INTRODUCTION **4**

LITERATURE REVIEW **10**

OBJECTIVES **11**

METHODOLGY **12**

DATASET 12
DATA PREPROCESSING 12
HANDLING DATA IMBALANCE 14
Under - Sampling Techniques *15*
Over - Sampling Techniques *17*
Hybrid - Sampling Techniques *18*
MODELS 19

RESULTS **23**

EVALUATION METRICS 23
EFFECTS ON UNDER-SAMPLING 24
EFFECTS OF OVER-SAMPLING AND HYBRID-SAMPLING 27
COMPARATIVE ANALYSIS BETWEEN THE SAMPLING TECHNIQUES 29
EFFECTS OF ENSEMBLE CLASSIFIERS 31
SHAP-BASED EXPLAINABILITY ANALYSIS 32
ANOMALY RULE GENERATION AND INTERPRETABILITY ANALYSIS 38
MODEL DEPLOYMENT 40

DISCUSSION **43**

CONCLUSION **45**

LIMITATIONS **46**

FUTUTRE SCOPE **47**

REFERENCES **48**

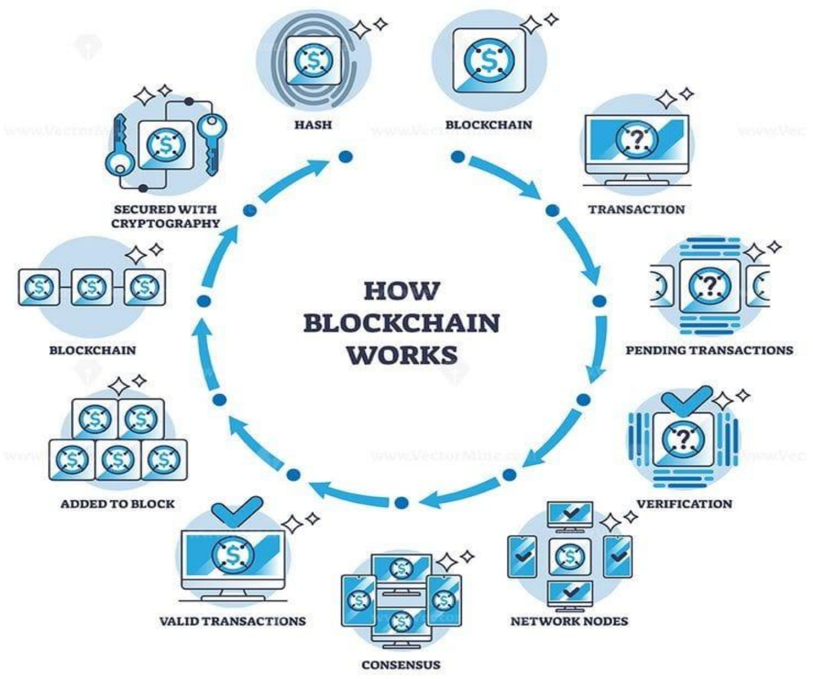
ABSTRACT

As blockchain technology continues to be used for digital transactions, it faces increasing threats from malicious activities. Detecting anomalies in blockchain transactions, vital for maintaining trust in such decentralized system, remains a challenge due to the highly imbalanced nature of illicit and legitimate transactions. Although existing research has made significant progress in this area, many approaches lack interpretability in their model predictions. This study reproduces a recent approach that addresses this gap by integrating explainable artificial intelligence (XAI) techniques and anomaly rules within tree-based ensemble classifiers for detecting anomalous Bitcoin transactions. The performance of a under-sampling algorithm, XGBCLUS with existing oversampling and under-sampling methods is assessed against other traditional sampling techniques. To enhance interpretability, the Shapley Additive Explanation (SHAP) method is utilized to evaluate the contribution of each feature. Experimental results confirm that our approach improves detection performance while making the decision-making process more transparent and interpretable. Additionally, anomaly rules for identifying anomalous transactions are derived from the tree-based classifiers to further understand and enhance the confidence in accurately identifying anomalies in the data. Furthermore, multiple tree-based classifiers are analysed and their performance is compared with stacking and voting classifiers. Experimental demonstrations confirm that the ensemble methods improve accuracy, True Positive Rate (TPR), and False Positive Rate (FPR) compared to single classifiers. Based on this comparative analysis, the best-performing model is deployed for effective anomaly detection in Bitcoin transactions.

Keywords: Anomaly detection, Blockchain, Bitcoin transactions, Data imbalance, Data sampling, Explainable AI, XAI, Machine Learning, Decision tree, Anomaly Rules

INTRODUCTION

Blockchain technology has emerged as a transformative force in digital transactions, offering a novel approach to recording and verifying transactions. At its core, a blockchain is a decentralized, distributed, and immutable ledger – a chain of blocks where each block contains a batch of transactions. This ledger is shared across a network of computers (nodes) in a peer-to-peer fashion. Each new block is cryptographically linked to the previous one using a hash, creating a chronological and tamper-evident chain. Key characteristics like decentralization (eliminating the need for central authorities), transparency (transactions are often publicly viewable), immutability (records are extremely difficult to alter once added), and enhanced security (through cryptographic hashing and consensus mechanisms) have fuelled its adoption across various sectors beyond finance.



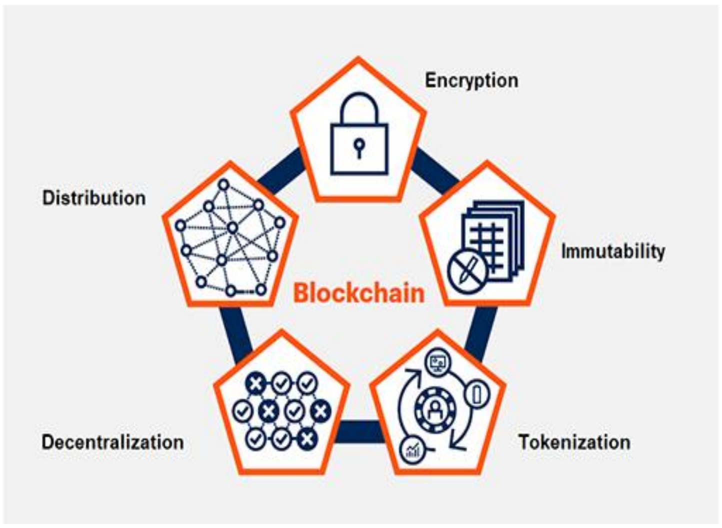
Blockchain transactions follow a systematic process designed to ensure transparency, security and immutability. The transaction process begins when a user initiates a transaction through a digital wallet, specifying details such as sender’s address, recipients address, and the amount of data to be transferred. Once initiated, this transaction enters a pool on unconfirmed transactions, awaiting validation by ‘miners’, the network participants (in Proof of Work systems) or validators, (in Proof of Stake systems). These participants verify the transaction by evaluating its adherence with the network rules and ensuring the sender has sufficient funds or permissions. After verification, the transaction is then broadcasted to all nodes in the blockchain network, where it undergoes further analysis. The nodes achieve an agreement on its legitimacy through processes such as PoW or PoS, guaranteeing that only legal transactions are allowed. Once consensus is achieved, the transaction is marked as genuine and grouped with other confirmed transactions to form a new block. This block is then appended to the current blockchain in chronological sequence, resulting in an immutable record. To secure the block, cryptographic hashing techniques are used, which generate a unique hash for the block and it is linked to the preceding block's hash. This assures continuity and tamper-proof integrity in the blockchain. Blockchain transactions gain unprecedented security and reliability across a wide range of applications thanks to this precisely designed approach.

Bitcoin, introduced by pseudonymous Satoshi Nakamoto in 2008, was the pioneering application of the blockchain technology. This step helped establish the idea of a distributed digital currency that would allow peer-to-peer electronic payments sans any intermediaries. These bitcoin transactions are digitally signed messages using cryptography that are sent to the bitcoin network and collected into blocks. Bitcoin ledger participants, known as miners, verify blocks using a consensus mechanism, typically Proof-of-Work, to ensure transaction integrity and chronological sequence.

Blockchain technology is a decentralized, distributed digital ledger that records transactions across a network of computers in a secure and transparent manner. It is most commonly associated with cryptocurrencies like Bitcoin, but its applications extend far beyond digital currencies, including supply chain management, healthcare, and finance.

Key Features of Blockchain Technology:

1) Decentralization: Blockchain operates on a decentralized network, meaning that no single entity controls the data. Instead, multiple nodes across the network maintain and update the ledger, ensuring that control is distributed among participants.
2) Distribution: Blockchain uses a distributed ledger, which is a shared database that stores transactions across a network of computers. Each node in the network has a copy of the entire blockchain, ensuring that all participants have access to the same information.
3) Immutability: Once data is recorded on the blockchain, it cannot be altered or deleted without consensus from the network. This is achieved through cryptographic hashing, which links blocks together, making tampering computationally infeasible.
4) Encryption: While encryption is a critical security feature of blockchain, it is not typically listed as a primary characteristic. However, blockchain uses cryptographic algorithms to secure transactions and ensure the integrity of the data stored on the blockchain.
5) Tokenization is not a primary characteristic of blockchain itself but is a concept often associated with blockchain technology. Tokenization refers to the process of converting assets into digital tokens that can be traded on blockchain platforms. However, blockchain's core characteristics focus more on its decentralized, distributed, immutable, and secure nature.



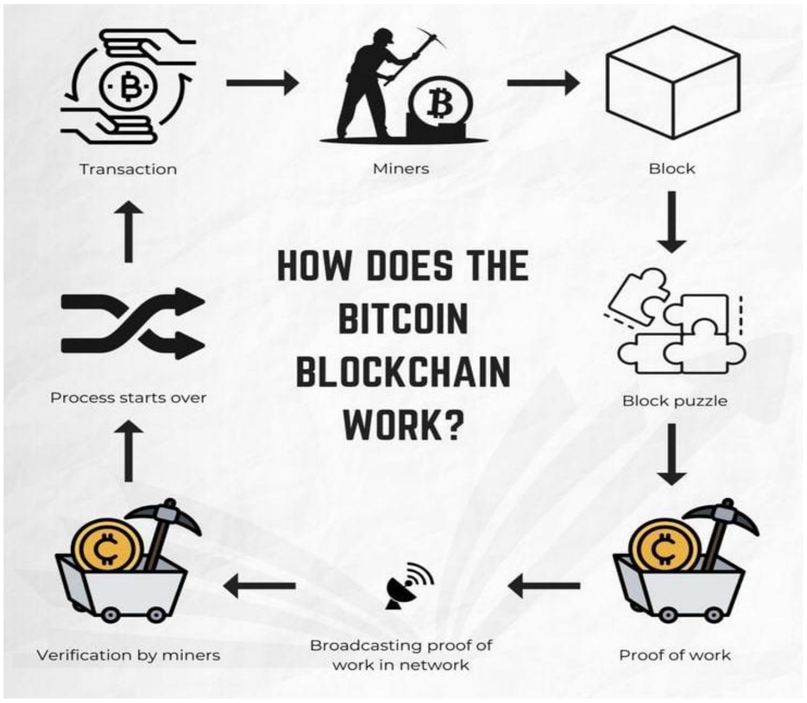
Blockchain is the underlying technology that enables cryptocurrencies to operate securely and transparently. Cryptocurrencies are digital or virtual currencies, secured by cryptography for security, meaning it is nearly impossible to counterfeit or double-spend and are decentralized, i.e. they are not controlled by any third-party organisation, for example – government or any institution. Cryptocurrencies are digital tokens that allow people to make payments directly to each other through an online system. Blockchain transactions capture details of activities such as the transfer of cryptocurrency from one account to another, the deployment of smart contracts, and interactions with smart contracts. Some cryptocurrencies in the Blockchain Ecosystem are Bitcoin (BTC), Ethereum (ETH), Stablecoins like Tether (USDT), USD Coin (USDC), Exchange Tokens like Binance Coin (BNB) and other platforms like Solana (SOL), etc.

Bitcoin, known as 'Digital Gold" remains the most well-known and the largest cryptocurrency by market capitalization. It is the widely recognized for its decentralized nature and limited supply. Bitcoin transactions follow a structured process to ensure security and compliance.

The process of Bitcoin transaction on the Blockchain is as follows:

1) Initialization – A user initiates a transaction via a Bitcoin wallet, which holds private keys for authorization.
2) Signing – The transaction is signed cryptographically using the sender's private key to verify the ownership and authorize the transfer.
3) Broadcasting – The signed transaction is then transmitted to the Bitcoin network through participating nodes.
4) Mempool – The nodes now validate the transaction's authenticity and availability of funds before placing it in the memory pool.
5) Mining and Validation - Miners select transactions from the mempool and attempt to solve a cryptographic puzzle via Proof-of-Work to include them in a block.
6) Block Addition – The first miner to find a valid hash broadcasts the block to the network. Once verified it is appended to the blockchain.

7) Confirmation – Transactions are deemed as confirmed once they are included in a block, with subsequent blocks reinforcing security and immutability.



Users typically pay transaction fees to incentivize miners. This process ensures transparency, decentralization, and resistance to tampering.

Initially designed to support Bitcoin, blockchain has expanded its applications beyond cryptocurrency, finding utility in domains such as healthcare, supply chain management, and the Internet of Things (IoT). Despite its advantages in transparency and security, blockchain transactions remain susceptible to various malicious activities or anomalies. Anomalies refer to patterns in data that deviate significantly from expected or normal behaviour. In the context of Bitcoin transactions, anomalies can represent fraudulent activities, security threats, or system vulnerabilities. These may include fraudulent transactions, money laundering, double-spending attacks and illicit trading. Successful detection of anomalous transactions within blockchain networks is therefore critical for maintaining integrity, security, compliance and trustworthiness in digital payments like the Bitcoin network.

Anomaly detection in blockchain transactions presents unique challenges, primarily due to the inherent imbalance in transaction data. Various Machine Learning techniques have shown great promise in identifying complex patterns and anomalies within large datasets, making them suitable for analysing bitcoin transactions.

However, a major challenge in applying ML to Bitcoin anomaly detection is the severe data imbalance. The vast majority of blockchain transactions are legitimate, while illicit or anomalous transactions occur infrequently, making them difficult to identify. Traditional anomaly detection techniques often struggle with such imbalance, as most machine learning models are trained on majority-class samples, leading to biases that overlook rare fraudulent transactions and hence may perform poorly in identifying the minority classes (anomalous transactions). This can lead to a high number of false negatives, where actual anomalies go undetected. To tackle the problem of imbalanced data, various data sampling techniques including under-sampling, over-sampling and combined sampling are explored. In particular, this study also the XGBCLUS algorithm, an under-sampling technique based on a new eXtreme Gradient Boosting as introduced in (Hasan, 2024) . This novel technique selects a subset of majority-class transactions that contribute the most to model learning, improving the detection of fraudulent transactions without compromising training data quality. The study evaluates XGBCLUS alongside other state-of-the-art sampling techniques to assess their effectiveness in creating a balanced dataset while also minimizing information loss. We compare the performance of XGBCLUS with other commonly used under-sampling and oversampling techniques, including SMOTE, ADASYN, and NearMiss, evaluating their impact on classification metrics such as accuracy, true positive rate (TPR), false positive rate (FPR), and ROC-AUC score.

Additionally, the effectiveness of several machine learning models is assessed, comparing single tree-based classifiers (Decision Tree, Random Forest, Gradient Boosting, and AdaBoost) with ensemble classifiers such as stacking and voting-based models. 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. Ensemble methods combine multiple individual models to improve prediction accuracy and robustness, often outperforming single classifiers, especially on complex tasks like anomaly detection.

Furthermore, existing studies have focused primarily on classification accuracy but lack transparency in explaining how and why a transaction is classified as anomalous. The black-box nature of machine learning models raises concerns regarding the interpretability of decisions, which is crucial for regulatory compliance and fraud investigations. Despite their strength, many sophisticated machine learning models—especially ensembles—frequently operate as "black box" models. This indicates that although they are capable of making precise predictions, it is challenging to comprehend how or why they reach a particular conclusion (such as identifying a transaction as unusual). This lack of transparency can be a significant drawback, especially in critical applications like financial security, where understanding the reasoning behind an anomaly flag is essential for investigation and trust.

In addition to anomaly detection, this study emphasizes model interpretability. To address this, the field of explainable Artificial Intelligence (XAI) is integrated along with the ensemble classifiers for anomaly detection in Bitcoin transactions. The research paper utilizes a popular XAI technique called SHAP (SHapley Additive exPlanations). SHAP (SHapley Additive exPlanations) is a technique used to analyse feature importance and explain model predictions. It is based on game theory and assigns a value to each feature in a model, indicating its contribution to the prediction. SHAP Analysis helps to understand the predictions of complex models by quantifying the contribution of each input feature to the to the final prediction for both individual transactions and the model as a whole, hence enhancing the transparency of the machine learning models. This helps in recognizing the main elements influencing anomaly detection.

The study also investigates the creation of explicit anomaly rules using more straightforward, interpretable models, such as decision trees. These rules offer human-readable explanations for classifying transactions as anomalous or non-anomalous, and supplement the insights from SHAP analysis and make it easier to apply and validate the detection system in practice by providing concise, understandable reasoning for why certain transactions might be flagged.

LITERATURE REVIEW

The detection of anomalies in blockchain transactions has been widely studied using machine learning (ML) techniques, including both supervised and unsupervised learning approaches. However, existing studies still face several challenges, including class imbalance, lack of interpretability, and inefficiencies in classification models. This section explores prior research in blockchain anomaly detection, data balancing techniques, and explainable AI (XAI) methods.

Despite its growing popularity in digital payments, Bitcoin remains susceptible to a range of attacks, including temporal attacks, spatial attacks, and logical-partitioning attacks (A. A. Monrat, 2019). 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 (Signorini, 2018). 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) (Azar, 2022).

Tree-based ML classifiers have been used in many studies to classify malicious activities since faster training can be performed on tree-based classifiers (Sarker, 2023) ,(M. Rashid, 2022). 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. 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 (Y. Xia, 2021). Several supervised techniques have been used for anomaly detection in Bitcoin. (Michal Ostapowicz, 2019). Comparative study on various individual classifiers is done for fraud detection in Blockchain based on the nodes in (Madhuparna Bhowmik, 2021). A new framework of ensemble stacking models are evaluated for Fraud detection in Bitcoin Transactions (al, 2023). For handling, highly imbalanced datasets, Ahmed et al. explored several under-sampling techniques for early product back-order prediction (F. Ahmed, 2022).

OBJECTIVES

The primary objectives of this project are as follows:

1) **To detect anomalies in blockchain transactions** using machine learning classifiers to enhance security and reliability in Bitcoin trading data.
2) **To address the challenge of imbalanced datasets** by evaluating various sampling techniques, including under-sampling, oversampling and hybrid (combined) sampling techniques for improved classification accuracy.
3) **To implement and evaluate the XGBCLUS algorithm**, a new under-sampling approach designed to enhance true positive rates and improve anomaly detection performance comparing its effectiveness against existing methods.
4) **To compare the effectiveness of tree-based classifiers** such as Decision Tree, Random Forest, Gradient Boosting, and ensemble learning methods (stacking and voting classifiers) in anomaly detection.
5) **To improve model explainability** by implementing SHAP analysis, providing insights into feature importance and decision-making within machine learning models.
6) **To develop a set of interpretability rules** derived from tree-based models to enhance understanding and transparency in classifying blockchain transactions.
7) **To deploy the best performing model** for real-world fraud detection from the comparative analysis, ensuring optimal performance in practical applicaions.

METHODOLGY

DATASET

The dataset used in this study consists of Bitcoin transaction data collected from the IEEE Data Portal. It contains 30,248,134 transactions, out of which only 108 are labelled as anomalous (fraudulent), making the dataset highly imbalanced.

Features in the Dataset:
The dataset includes 12 attributes, with the target variable indicating whether a transaction is anomalous (1) or normal (0).

DATA PREPROCESSING

To perform feature selection, we conducted a hypothesis testing using the T-test to analyse the statistical significance of attributes in distinguishing between anomalous (positive) and non-anomalous (negative) Bitcoin transactions. This test evaluates whether there exists a significant difference between the mean values of the positive and negative samples across each attribute.

The correlation heatmap for all features, Figure 1, displays features such as in_btc, out_btc, total_btc, mean_in_btc, and mean_out_btc exhibiting 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.

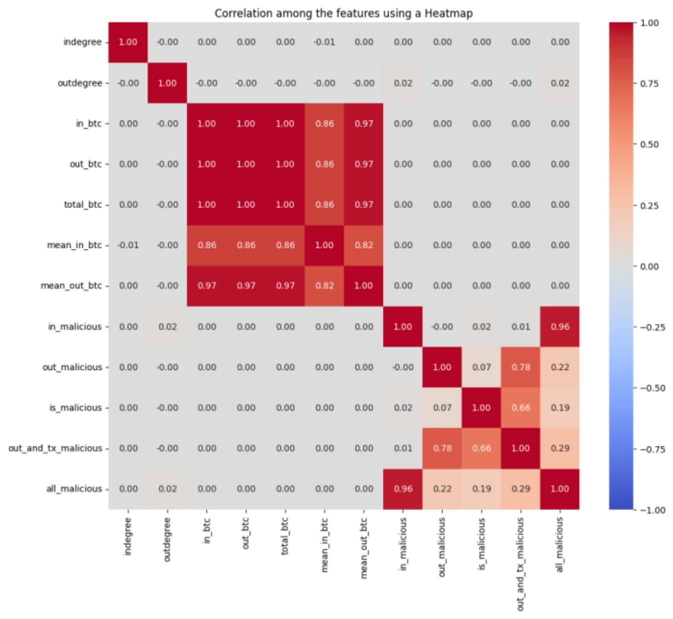


Figure 1: Correlation Heatmap

For each attribute, the T-statistic and the corresponding p-value is computed to assess the strength of the correlation. A T-test assists to check whether the mean values of a feature are significantly different between the two classes (anomalous v/s non-anomalous). Attributes with p-values below 0.01 were considered statistically significant in differentiating fraudulent transactions from legitimate ones. The results, as presented in Table 1, indicate that all attributes demonstrate statistical significance except for 'outdegree', which does not exhibit a strong correlation with transaction anomalies.

Attribute	**t value**	**p value**
indegree	-17.93	0.00
outdegree	**0.71**	**0.48**
in_btc	-21.56	0.00
out_btc	-20.62	0.00
total_btc	-21.11	0.00
mean_in_btc	-11.03	0.00
mean_out_btc	-20.01	0.00
in_malicious	-60.49	0.00
out_malicious	-6761.93	0.00
is_malicious	-4827.61	0.00
all_malicious	-1637.67	0.00

Table 1: The t values and p values for all features

Features with p-values greater than 0.01, such as outdegree along with all the features (in_malicious, out_malicious, is_malicious, and all_malicious) sharing the same value range as the target feature out_and_tx_malicious. As 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 is selected, including the target feature, for classification. The final set of selected features includes: indegree, in_btc, out_btc, total_btc, mean_in_btc, mean_out_btc, and the label out_and_tx_malicious.
By applying this feature selection approach, we ensure that only the most relevant attributes are retained for training machine learning models, thereby enhancing model interpretability and reducing computational complexity. Ultimately, summary of the selected features is given in Table 2.

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

Table 2: Summary of the selected features

Data Imbalance Issue

- Normal Transactions: 30,248,026 (~99.9996%)
- Anomalous Transactions: 108 (~0.0004%)

After feature selection using the T-test, the negative and positive samples were segregated, and duplicate entries were eliminated exclusively from the negative samples. We then reduced the data to address computational complexity, retaining only 200,000 negative (non-anomalous) samples along with all 108 positive (anomalous) samples. i.e. a total of 200,108 samples were retained. The class imbalance ratio still remains high as shown in the Figure 2: Class RatioFigure 2

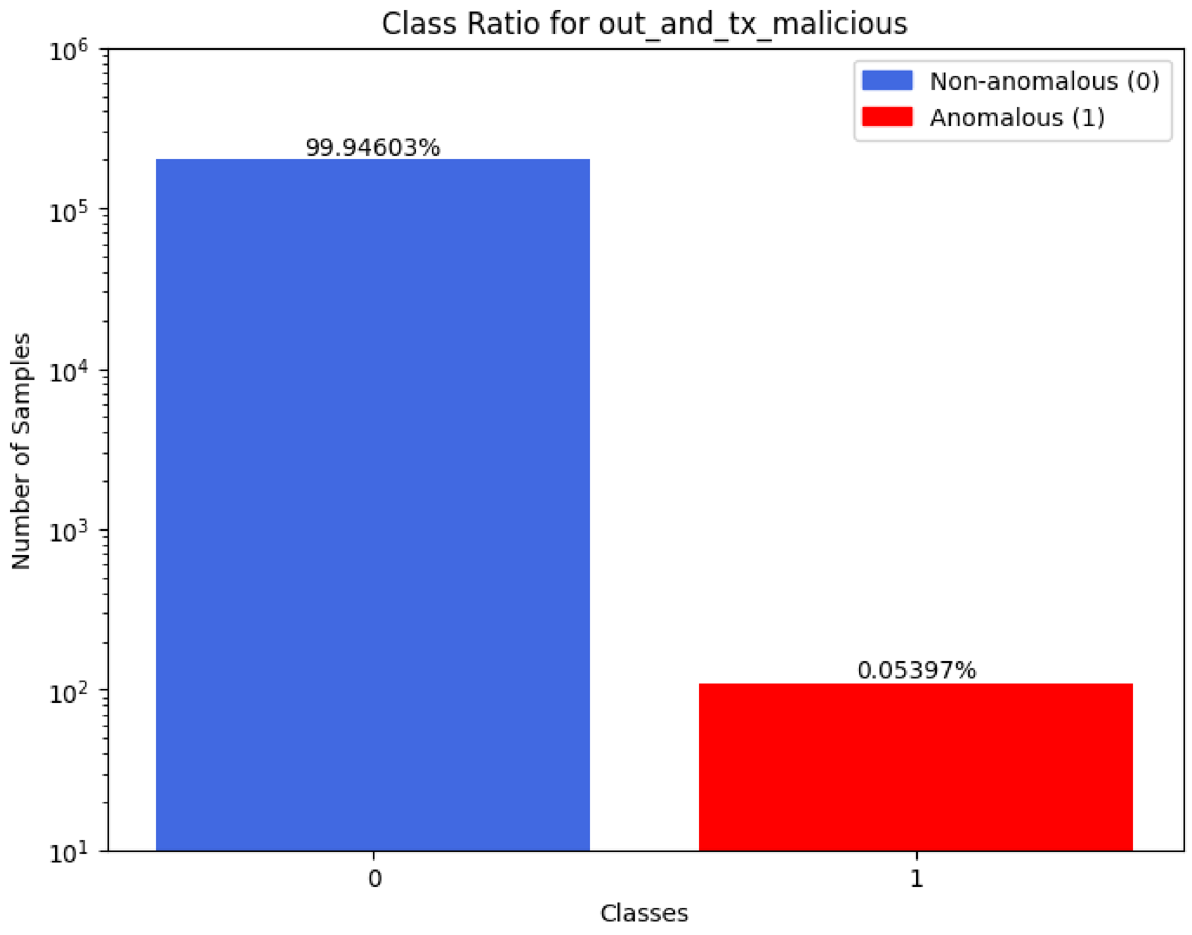


Figure 2: Class Ratio

The extreme class imbalance makes it challenging for machine learning models to detect fraudulent transactions accurately. To address this, various data balancing techniques (under-sampling, over-sampling) were applied.

HANDLING DATA IMBALANCE

A significantly lower number of illicit transactions in Bitcoin transactions compared to legitimate transactions leads to imbalanced data. As a result, ML classifiers frequently show bias in favour of the majority class. Although classification accuracy may seem adequate in many situations, there is frequently a noticeable difference between TPR and FPR values, suggesting that the models are incorrectly classifying anomalies. Prior to training the classification models, it is essential to balance the dataset using pre-existing or specially designed sampling strategies. Positive examples are usually rare in situations involving anomaly detection, fraud identification, or money laundering.

Under-sampling strategies can be useful for rebalancing the dataset while giving precise positive case identification top priority. However, ML classifiers may be trained on a limited dataset produced by the under-sampling strategy in situations when the minority class has an abnormally low number of positive events. Conversely, the goal of over-sampling techniques is to raise the minority class's instance count to equal that of the majority class. Even if these techniques create fictitious data using a mix of majority and minority samples, they can still be useful. Given the skewed nature of the dataset, both under-sampling and over-sampling methods along with some hybrid methods were employed to balance the classes prior to model training. Our research includes the introduction of the XGBCLUS under-sampling algorithm as well as an examination of well-known under-sampling methods like NearMiss and random under sampling (RUS). We also look at common over-sampling strategies like SMOTE, ADASYN, and coupled strategies like SMOTEENN and SMOTETOMEK.

Under - Sampling Techniques

Three under-sampling techniques used to handle imbalance in the data are: Random Under Sampling (RUS), NearMiss (NM) and a newly proposed technique called XGBCLUS
A newly proposed under under-sampling technique ‘XGBCLUS’ is evaluated along with other commonly used under-sampling algorithms Random Under Sampling (RUS), NearMiss (NM).

1) **Random Under Sampling**
   RUS is a straightforward method for dealing with dataset class imbalance. Chooses a portion of the majority class at random to equal the size of the minority class. A subset of instances from the majority class is chosen at random. The intended balancing ratio between the majority and minority classes is used to calculate the size of this subgroup. The ratio of balance α_{u_s} is defined by

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

   Where, N_m is the number of minority class samples and N_{r_m} is the number of majority class samples after resampling. A balanced dataset is then created by combining the instances from the minority class and the randomly chosen subset of the majority class instances.

2) NearMiss

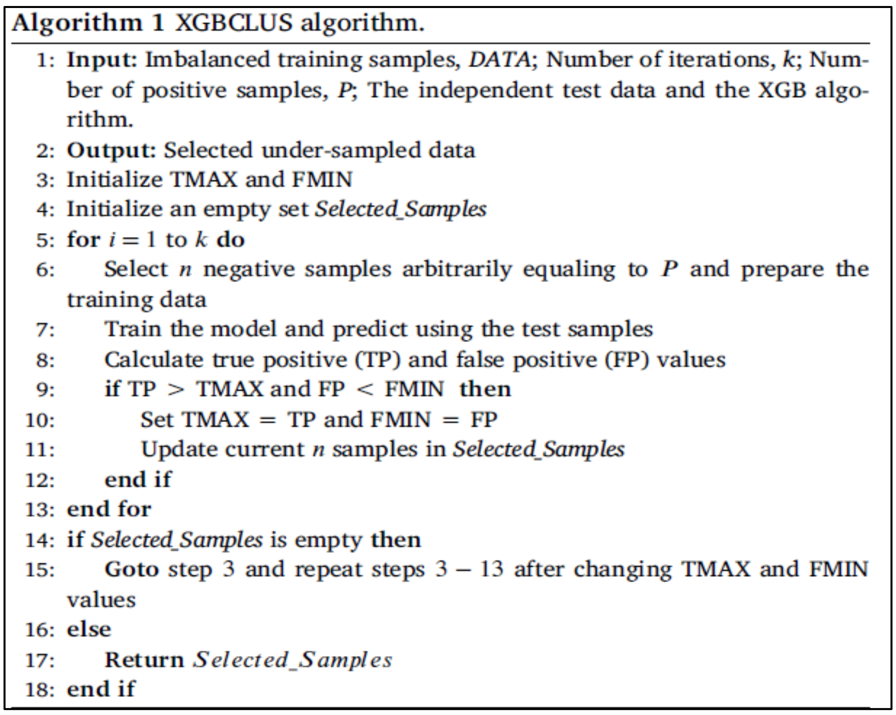
The dataset is also balanced using NearMiss another undersampling technique. By keeping a portion of samples from the majority class that are near instances from the minority class, reducing the imbalance. This down-sampling technique selects instances based on their proximity to the minority class, preserving important samples. NearMiss uses a variety of distance metrics, including Manhattan distance and Euclidean distance, to determine the distances between each instance in the minority class and every instance in the majority class. Each instance in the minority class is thus closest to the k instances from the majority class identified by this approach. The degree of under-sampling is determined by the value of k, which is usually defined as a hyperparameter. Since we set the value to 1, we refer to it as NearMiss-1. Ultimately, it creates an under-sampled dataset by combining the instances from the majority class and the minority class.

3) XGBCLUS

The XGBCLUS algorithm functions by combining the XGB algorithm with clusters, which are achieved by choosing random instances from the majority class in the training data that are equal in number to the positive instances from the minority class. The entire dataset is first divided into training and test data by the algorithm. The number of positive samples, P, from the training set is counted while maintaining the test data's independence. The total number of negative samples in the training data is then divided by the number of positive samples, P, to determine the number of iterations, k.

The algorithm randomly chooses ‘n’ negative samples equal to P for each iteration, and a fresh training set is created to train the XGBoost model. The model forecasts the true positive (TP) and false positive (FP) values using the independent test set. TMAX and FMIN are used to compare the TP and FP values, respectively. Both TMAX and FMIN are initialized with arbitrary values; FMIN defines the minimum false positive value, while TMAX represents the highest true positive value. Both TMAX and FMIN are updated with the TP and FP values, respectively, if the TP value is higher than the TMAX value and the FP value is lower than the FMIN value. Current n samples are updated in the Selected_Samples set simultaneously. If not, the current iteration remains unchanged.

Following the completion of the k iterations, the Selected_Samples set is examined. Rerunning the algorithm with fresh TMAX and FMIN values is necessary if the set is empty. If not, the algorithm's under-sampled data is represented by the samples in the Selected_Samples set.



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
```

Over - Sampling Techniques

Two popular over-sampling techniques for handling the class imbalance in Bitcoin transactions data are evaluated along with the under-sampling techniques.

1) **SMOTE (Synthetic Minority Oversampling Technique)**
SMOTE is a popular method for addressing the issue of class imbalance, particularly in Bitcoin transactions for anomaly detection. By creating artificial representations of the minority class, it seeks to balance the distribution of classes, reducing bias and enhancing model generalization. It randomly chooses the k nearest neighbors from the same class for every minority sample X_i. One of the k nearest neighbors X_{zi} from k is used to create a new sample X_n, which is then created using an equation given below,
$$X_n = X_i + \lambda * (X_{z_i} - X_i)$$

where λ is a random number in the range of 0 and 1. In the feature space, a fresh synthetic instance is produced as a result. Until the samples in the majority and minority classes are identical, this procedure is repeated. Finally, a balanced dataset is created by combining the freshly created synthetic examples with the original minority cases.

2) **ADASYN (Adaptive Synthetic Sampling)**
ADASYN generates the new sample using the same formula, with the exception of choosing X_i. In areas that are more difficult to categorize, it raises the density of synthetic instances, offering a more sophisticated solution to class imbalance. The minority sample's proximity to the majority class is determined by first calculating the

number of its k nearest neighbors who are members of the majority class. The imbalance ratio α_{o_s} is then calculated using equation below for each minority instance,

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

where Nm is the number of samples in the majority class and Nrm is the number of samples in the minority class after resampling. The number of synthetic instances that should be created for the present minority instance is determined using this imbalance ratio. In order to determine the hard level, a difficulty ratio is also computed. It takes into account more neighbors while creating synthetic instances if the ratio is large. A new synthetic instance is created by interpolating feature values between the minority instance and its chosen neighbors. Until the samples in the majority and minority classes are identical, this procedure is repeated for every minority case.
In order to create a balanced dataset, the newly created synthetic instances are then joined with the original minority instances.

Hybrid - Sampling Techniques

These techniques combine both over-sampling (increasing minority class samples) and under-sampling (reducing majority class samples) to address class imbalance in datasets.

1) **SMOTENN (SMOTE + ENN)**
SMOTEENN combines SMOTE AND ENN (Edited Nearest Neighbors). In order to address class imbalance, it uses SMOTE to create synthetic samples and ENN to refine the dataset and remove any cases that might be noisy or incorrectly classified. SMOTEENN seeks to improve the quality of the final dataset by removing potential noise and offering a more sophisticated method of balancing unbalanced datasets. ENN first determines the k nearest neighbors for each instance in the dataset after SMOTE generates the synthetic examples. Instances are eliminated from the dataset if their class differs from that of the majority of their neighbors. The noisy or incorrectly categorized instances are thus removed by ENN.

2) **SMOTETOMEK (SMOTE + TOMEK)**
SMOTETOMEK, another combined resampling technique, creates synthetic instances using both SMOTE and the under-sampling technique. The TOMEK connection determines the closest pairs of instances from various classes. The instance from the majority class is eliminated for every pair of instances that are recognized as TOMEK linkages. This under-sampling technique aids in eliminating cases that may be incorrectly classified and are near the decision boundary. A more impartial, discriminative, and useful dataset for ML model training may result from this.

MODELS

It is crucial to use the correct ML classifier for precise predictions, particularly when identifying anomalies in imbalanced data. Tree-based ML classifiers have been shown to be very useful in classifying malicious activities, as these models can be trained faster. Tree-based classifiers are a family of machine learning algorithms that use decision trees as their core model. These models split data based on feature values to make predictions and are favored for their interpretability, speed of training, and ability to handle both numerical and categorical data.

Common single tree-based models include Decision Trees (DT), which are prone to overfitting but serve as a strong foundation for more advanced techniques. To overcome the limitations of single decision trees and enhance performance, ensemble methods such as Random Forest (RF), Gradient Boosting (GB), and AdaBoost (AB) combine multiple decision trees to improve accuracy, robustness, and generalization.

Each of these algorithm functions as an independent classifier:

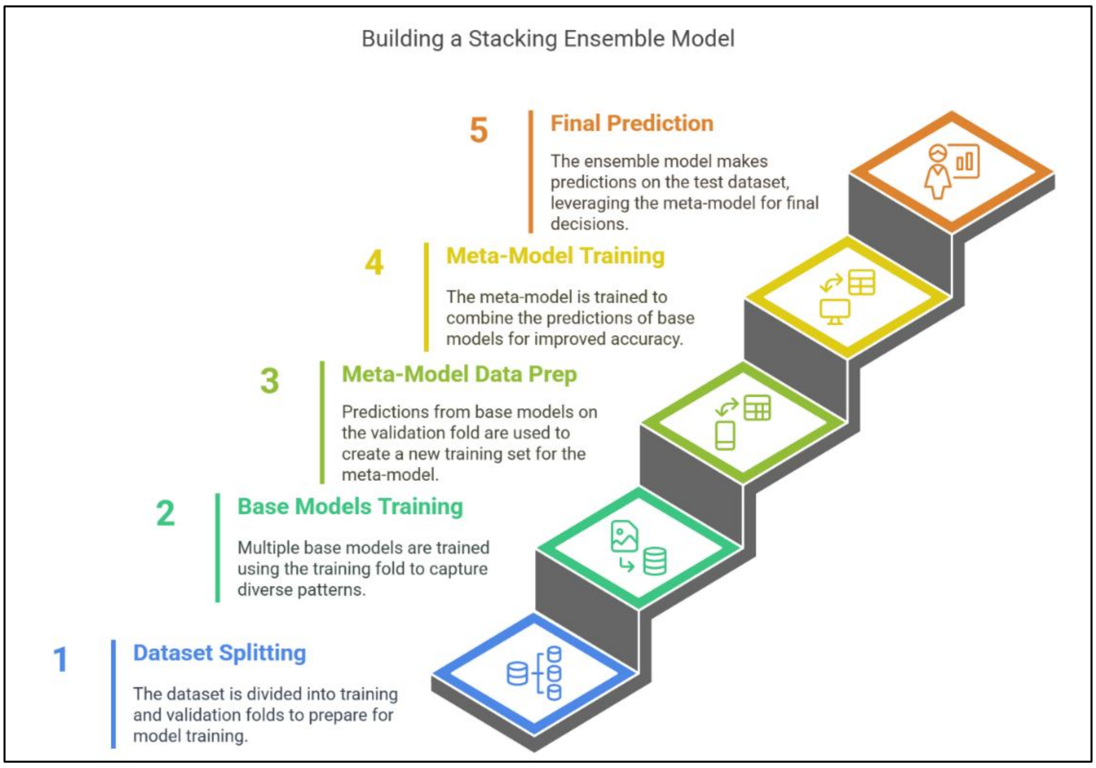
1) Decision Tree is a simple and interpretable model based on recursive partitioning.
2) Random Forest is an ensemble of multiple decision trees trained via bagging, which reduces variance and overfitting.
3) AdaBoost and Gradient Boosting are boosting-based ensemble methods that combine several weak learners (typically shallow trees) trained sequentially to improve prediction accuracy.

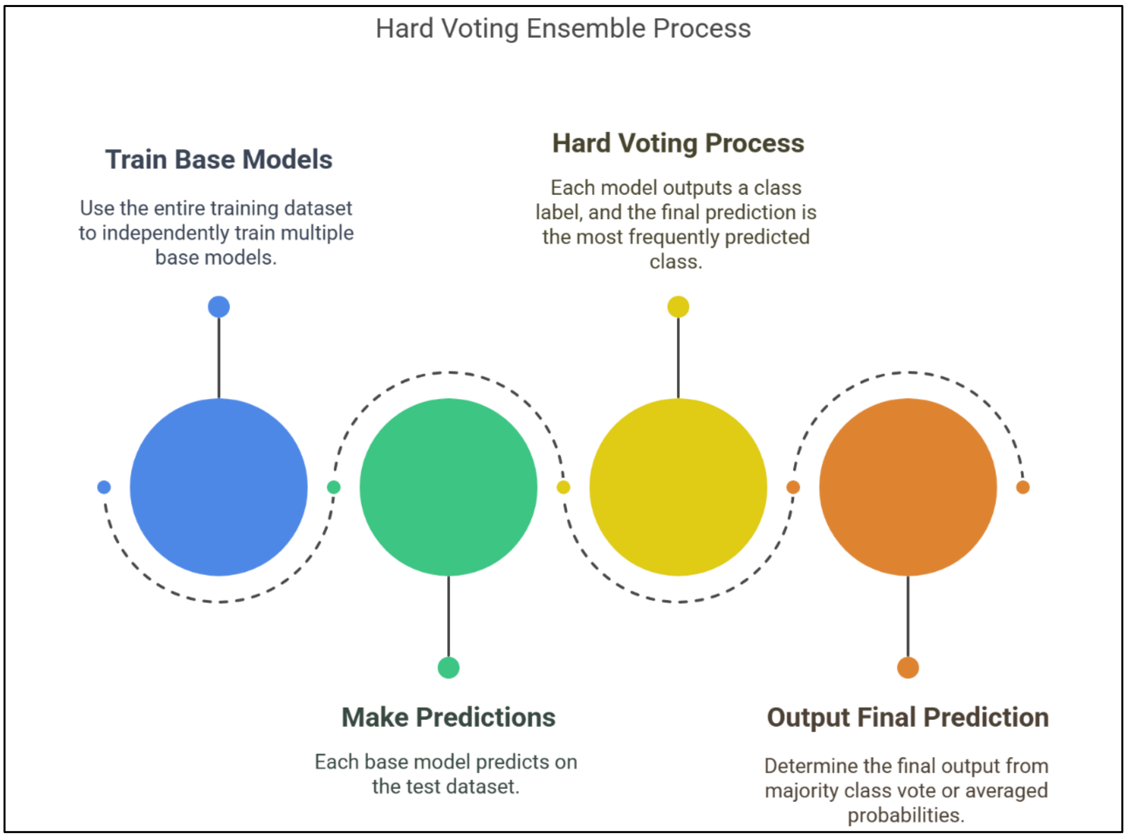
A further ensemble of these individual tree-based classifiers is used to increase performance by reducing errors given by the individual models. Ensemble methods such as stacking and voting are explored using exclusively tree-based models. Unlike bagging and boosting—which typically replicate a single algorithm with different subsets of data or sequential learning—stacked ensembles integrate diverse models with varying algorithms or hyperparameters. This diversity enhances generalization and often improves predictive performance. Voting ensembles, on the other hand, combine predictions from multiple models either by majority vote (hard voting) or by averaging prediction probabilities (soft voting). While these methods involve training several base models, which may increase computational overhead, tree-based classifiers are particularly efficient, as they require minimal data preprocessing and train quickly compared to other models like SVMs or KNNs.

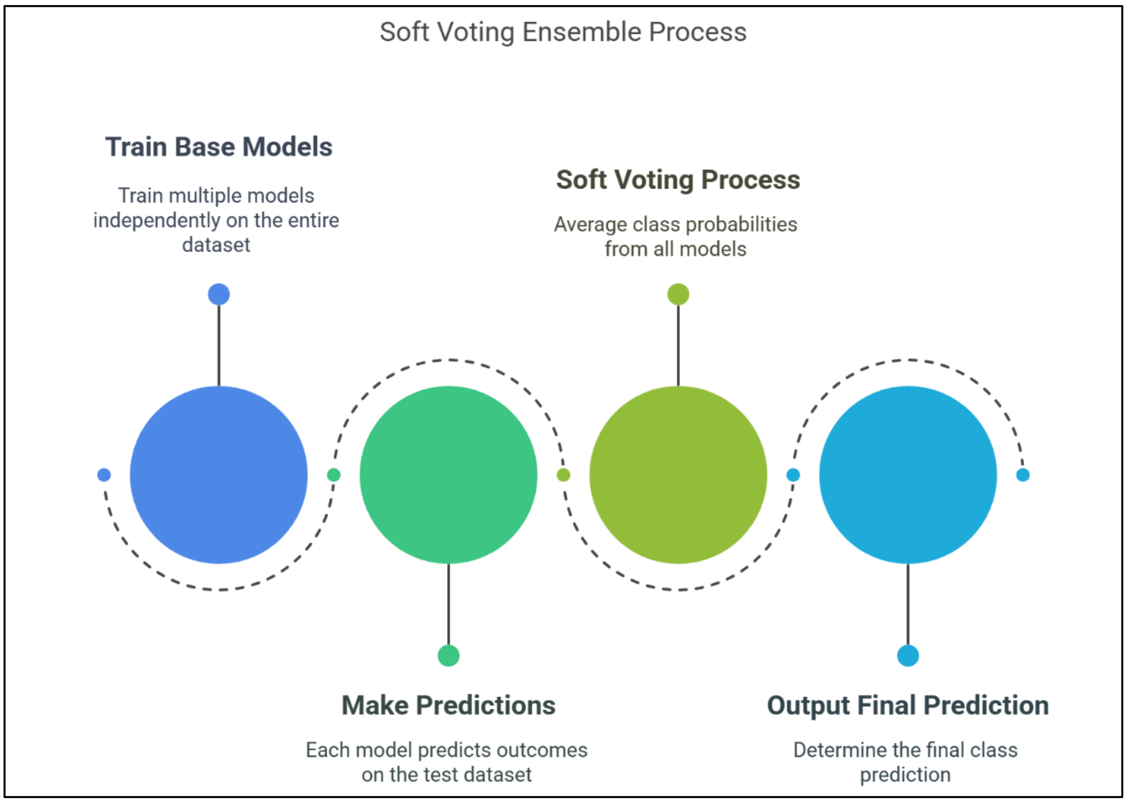
By merging several models, the meta-classification ensemble method based on stacked generation is an ML technique that increases prediction accuracy. Two classifiers make up the stacking-based ensemble model. The basic classifier and the meta classifier are the two types. Training a set of base models with several classifiers is the first step. The base-level classifier finds a new dataset, which is then used to train the meta-classifier, sometimes referred to as a mixer or combiner. The independent test dataset is then predicted using the learnt meta-classifier.

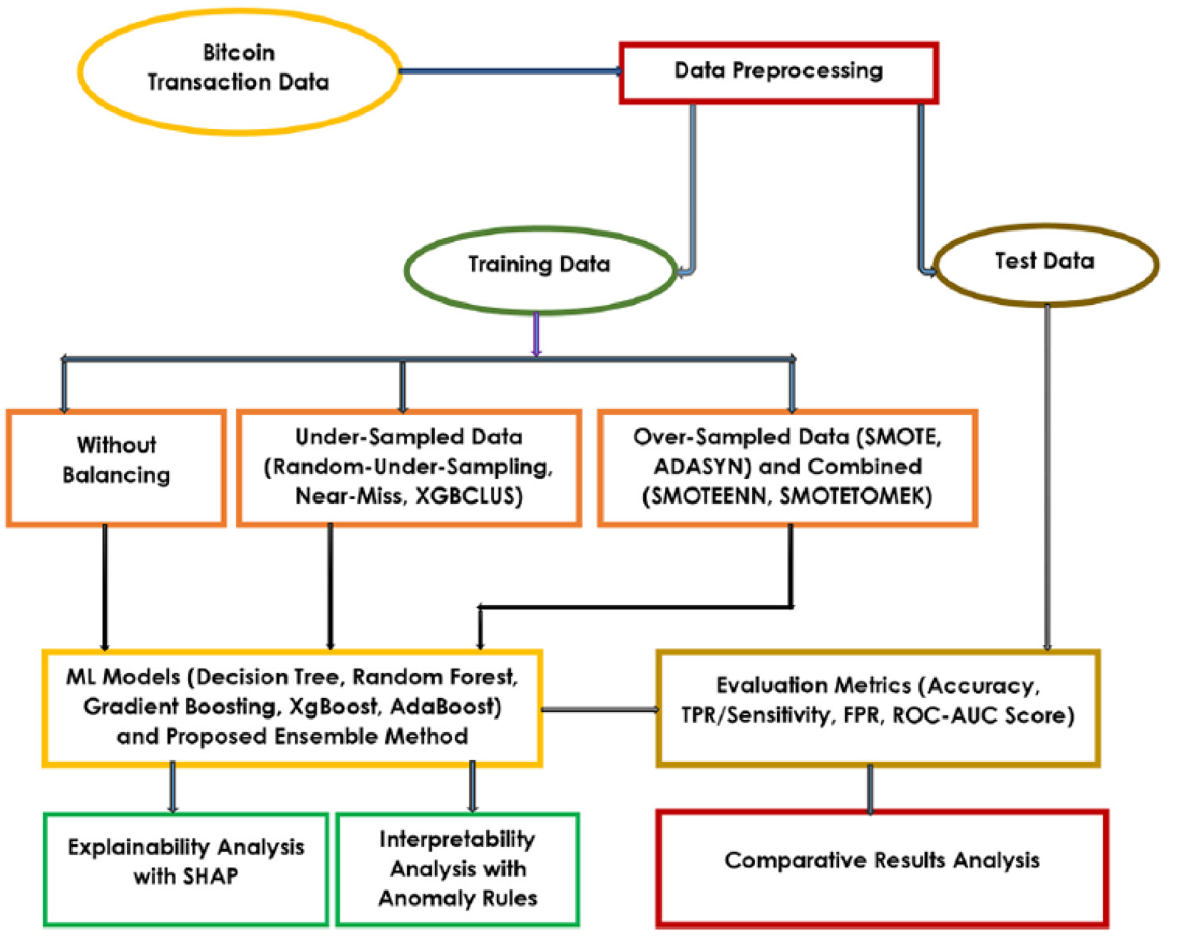
The base models for our suggested stacking-based ensemble model include Random Forest (RF), Decision tree (DT), Gradient boosting (GB), and Adaptive boosting (AdB). The base models are trained using the training dataset, and a new dataset is created by combining the outputs of the four base models with the validation fold. The meta-classifier in our suggested model, logistic regression (LR), then learns from the newly created dataset by using the base classifier's predictions as input. Finally, the test dataset is utilized to forecast both non-anomalous and anomalous transactions.
However, the voting classifier is built solely with tree-based models, a family of machine learning techniques renowned for their interpretability and resilience. The ensemble is created by merging the predictions of multiple separate tree-based models, and the voting classifier uses the training set as input. In this sense, each model denotes a distinct implementation of the tree-based algorithm with a particular set of configurations or hyperparameters. The voting classifier is used to provide predictions on test data following training and evaluation. Using a voting mechanism, the voting classifier combines the predictions of every single tree-based model.
There are two types of voting: ‘Hard’ and ‘Soft’. Hard voting assigns the final prediction to the majority class after each model in the ensemble votes once for the projected class label. The class with the highest average probability is selected as the final prediction in soft voting, which averages the probabilities (confidence ratings) of each model's projected classes.

In this study, we perform a comparative analysis of various tree-based models – Decision Tree, Random Forest, Gradient Boost, Adaboost. Additionally, we evaluate ensemble methods combining these four models using both hard and soft voting strategies. Furthermore, a stacked ensemble model is implemented, employing logistic regression as the meta-classifier.


Building a Stacking Ensemble Model
5 Final Prediction
The ensemble model makes predictions on the test dataset, leveraging the meta-model for final decisions.
4 Meta-Model Training
The meta-model is trained to combine the predictions of base models for improved accuracy.
3 Meta-Model Data Prep
Predictions from base models on the validation fold are used to create a new training set for the meta-model.
2 Base Models Training
Multiple base models are trained using the training fold to capture diverse patterns.
1 Dataset Splitting
The dataset is divided into training and validation folds to prepare for model training.


Hard Voting Ensemble Process
Train Base Models
Use the entire training dataset to independently train multiple base models.
Hard Voting Process
Each model outputs a class label, and the final prediction is the most frequently predicted class.
Make Predictions
Each base model predicts on the test dataset.
Output Final Prediction
Determine the final output from majority class vote or averaged probabilities.


Soft Voting Ensemble Process
Train Base Models
Train multiple models independently on the entire dataset
Soft Voting Process
Average class probabilities from all models
Make Predictions
Each model predicts outcomes on the test dataset
Output Final Prediction
Determine the final class prediction


Bitcoin Transaction Data
Data Preprocessing
Training Data
Test Data
Without Balancing
Under-Sampled Data (Random-Under-Sampling, Near-Miss, XGBCLUS)
Over-Sampled Data (SMOTE, ADASYN) and Combined (SMOTEENN, SMOTETOMEK)
ML Models (Decision Tree, Random Forest, Gradient Boosting, XgBoost, AdaBoost) and Proposed Ensemble Method
Evaluation Metrics (Accuracy, TPR/Sensitivity, FPR, ROC-AUC Score)
Explainability Analysis with SHAP
Interpretability Analysis with Anomaly Rules
Comparative Results Analysis

Methodology

RESULTS

The performance of the individual as well as ensemble tree-based classifiers is assessed through a comparative analysis in this section. We first set up the experimental setup before presenting results that examine different aspects of the model's performance with regard to Bitcoin anomaly identification. Extensive performance analyses have been carried out on a dataset that includes both typical and unusual Bitcoin transaction activity. The use of many Python modules, including scikit-learn, Pandas, and NumPy, made performance evaluation easier. We executed the python script on colab.

Evaluation Metrics

It becomes essential to use additional metrics, such as TPR to evaluate the accurate identification of anomalous transactions and FPR to evaluate the correct identification of non-anomalous transactions, since accuracy alone is insufficient to assess the performance of an anomaly detection system. This is consistent with our study's main goal. The performance of the individual models with and without data balancing has been compared to the suggested ensemble models using assessment criteria like accuracy, TPR, and FPR. Furthermore, we display the features' hierarchy using the feature importance score. The ROC score, which contrasts the TPR and FPR, has also been taken into account.
The performance metrics are defined below:

Accuracy = TP+TN / TP+TN+FP+FN

TPR = Sensitivity = TP / TP+FN

TNR = Specificity = TN / TN+FP

FPR = FP / TN+FP

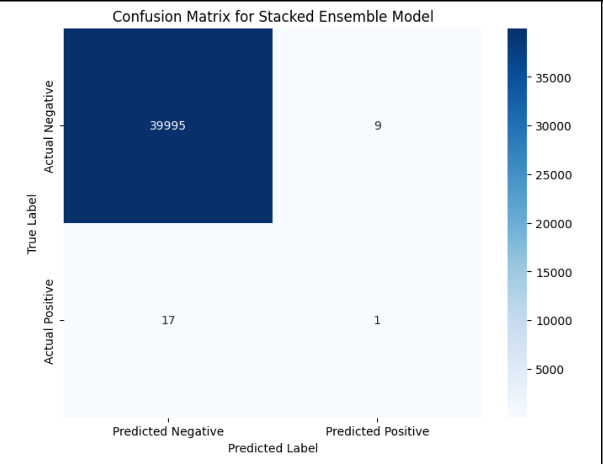
AUC is calculated as the area under the sensitivity (TPR) - (FPR) curve.
where,
TP = True Positive: an anomalous transaction is correctly identified as anomalous.
TN = True Negative: a non-anomalous or normal transaction is correctly identified as non-anomalous.
FP = False Positive: a non-anomalous transaction is incorrectly identified as anomalous.
FN = False Negative: an anomalous transaction is incorrectly identified as non-anomalous.

Effects on Under-Sampling

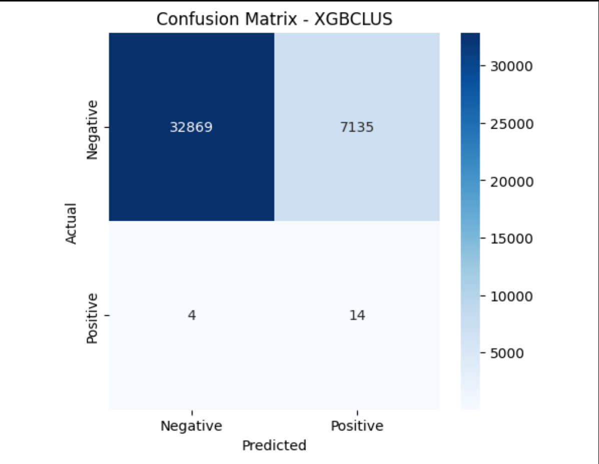
The models were trained using on 80% of the data, with 20% set aside for the independent test set. In order to demonstrate the issue of data imbalance, the classifiers were trained without balancing the train set. A high true negative value is produced by the ML classifiers becoming biased toward the majority of data. But because the ML classifiers are unable to accurately identify the positive samples, the true positive rate is extremely low—in certain circumstances, it is zero. Table 3 compares the DT, GBoost, RF, and AdaBoost classifiers' Accuracy, TPR, and ROC-AUC score. For the AB classifier, the TP values are zero, meaning that all transactions are categorized as regular and no anomalous transactions are accurately identified. Even though the classifiers appear to have adequate accuracy, the TP score tends to zero, meaning that anomalous transactions are missed due to model biases toward the bulk of transactions. Since our study's main goal is to identify anomalous transactions, classifiers might have trouble spotting those transactions in the absence of balanced data. In order to raise the TPR and lower the FPR values, we investigated a number of balancing strategies.

Classifiers	**Accuracy**	**True Positive Rate**	**False Positive Rate**	**AUC-ROC**
Decision Tree	0.999076	0.055556	0.000500	0.527528
Gradient Boosting	0.999026	0.222222	0.000625	0.610799
Random Forest	0.999325	0.055556	0.000250	0.527653
Adaptive Boosting	0.999550	0	0	0.500000

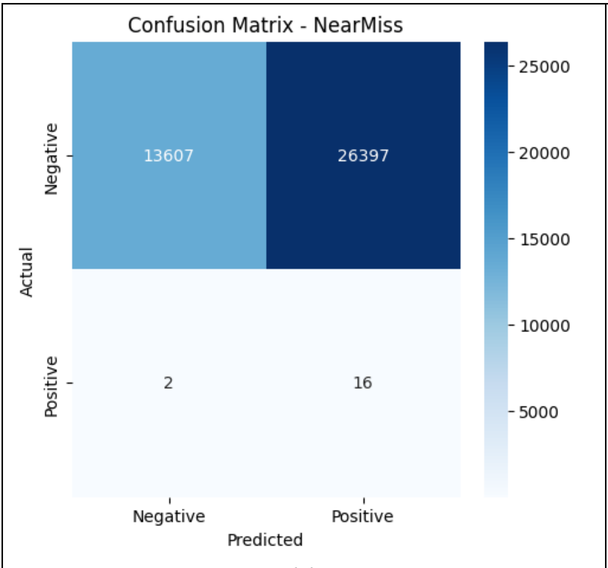
Table 3: Comparison among the classifiers without balancing the data



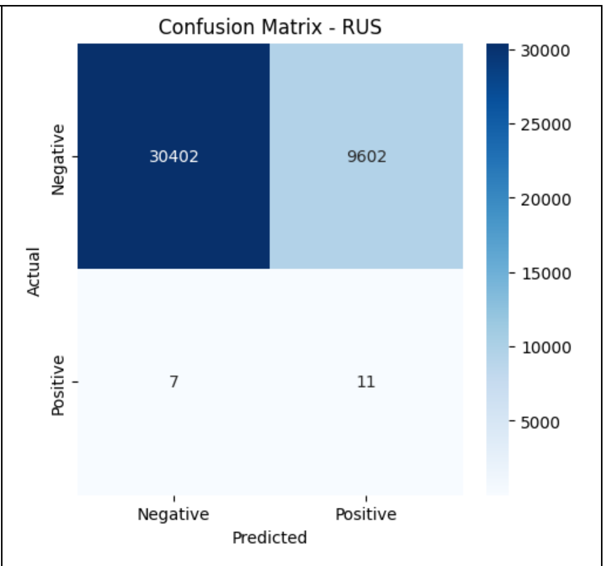
(a)



(b)



(c)



(d)

Figure 3: Confusion Matrix for Ensembles from Under-Sampling

The confusion matrices Figure 3 show how well the ensemble classifier performs when using the RUS, NearMiss-1, and XGBCLUS under-sampling techniques over the unbalanced data. Interestingly, the TP values show an increase when contrasted to situations where there is no balancing, where the TP values are always zero. It is crucial to recognize that, in spite of the TP gains, the FP values exhibit differences between the under-sampling techniques. In particular, even when there is no balancing and the FP value is zero, NearMiss-1 exhibits a comparatively high FP count in comparison to RUS and XGBCLUS.

In our study, we suggested the XGBCLUS under-sampling method, which achieves the maximum TP value and while also maintaining relatively low FP values, outperforming the existing methods. **The reason for this superiority is that, unlike existing methods, which choose examples at random and hence miss significant cases, our technique takes into account every instance while under-sampling.** Under-sampling strategies were investigated in order to balance the numbers of normal and anomalous transactions, since the TPR or sensitivity represents the count of correctly classified positive transactions.

	Under-sampling		
	RUS	XGBCLUS	Near Miss
DT	0.312369	0.266548	0.686381
GBoost	0.242276	0.181282	0.666258
RF	0.210154	0.165708	0.624638
AdaBoost	0.167858	0.123863	0.699130
ES	0.240026	0.178357	0.659859
EHV	0.196305	0.131637	0.656134
ESV	0.267398	0.199555	0.675482

Table 4: FPR of ML classifiers after under-sampling

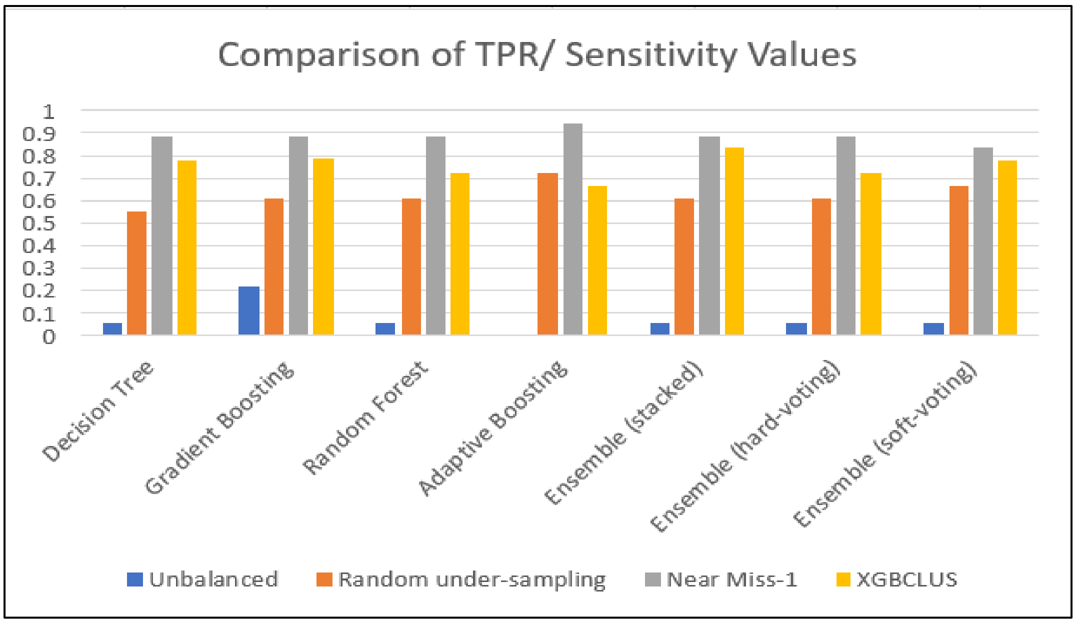


Figure 4: Comparison of TPR/ sensitivity values using under-sampling techniques

We combined our suggested under-sampling method, XGBCLUS, with other well-known down-sampling strategies. With the exception of the NearMiss-1 under-sampling approach, both single and ensemble ML classifiers showed an increase in sensitivity values, as seen in Figure 4. Notably, the sensitivity values for the classifiers using the XGBCLUS algorithm are 0.77, 0.83, 0.72, 0.66, 0.88, 0.88, and 0.83 for DT, GBoost, RF, AdaBoost, Ensemble-stacked, Ensemble (hard-voting), and Ensemble (soft-voting), respectively. These values are higher than the sensitivity values derived using random under-sampling techniques and without balancing. Although the NearMiss-1 under-sampling strategy produces a greater sensitivity value than the XGBCLUS method, Table 4**Error! Reference source not found.**Table 1 shows that the related FPR value is noticeably high. Indicating the accurate identification of non-anomalous transactions, the TNR rises as the FPR falls. The greater FPR of the NearMiss-1 under-sampling technique indicates that it may not be able to detect non-anomalous transactions with sufficient accuracy. On the other hand, the average FPR values obtained by the random under-sampling approach are higher than those of XGBCLUS. When it comes to TPR and FPR values, XGBCLUS performs better than alternative under-sampling methods.

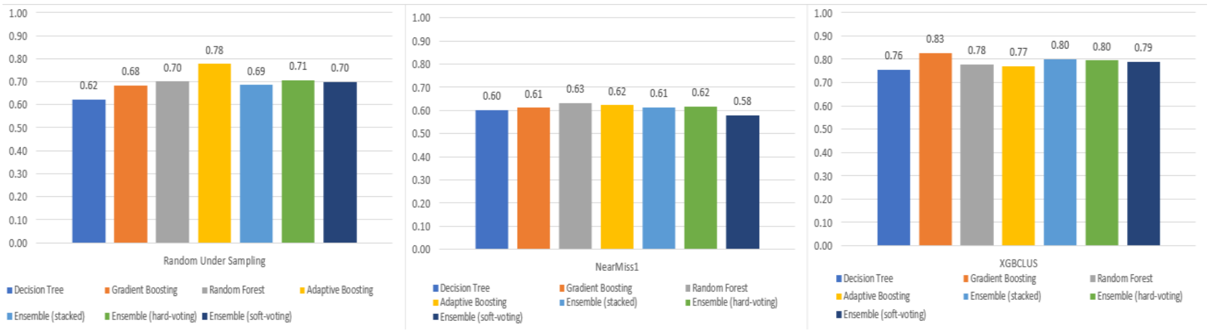
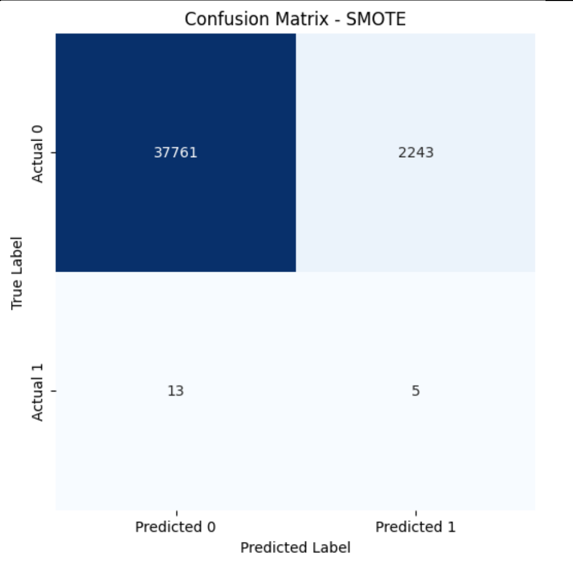


Figure 5: Improved ROC-AUC values XGBCLUS

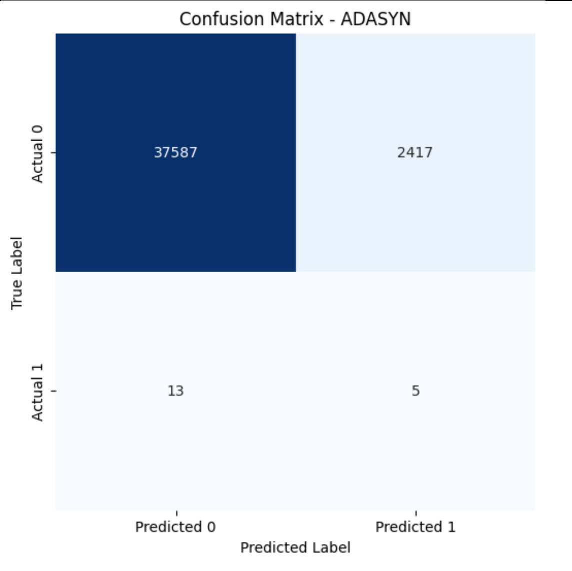
The improved ROC-AUC scores attained by using under-sampled data are shown in Figure 5Figure 5. Notably, in terms of ROC-AUC scores, the suggested XGBCLUS under-sampling strategy performs better than RUS and NearMiss-1. Out of all single and ensemble classifiers, the GBoost classifier attains the greatest ROC-AUC score of 0.83, which is the peak. Additionally, the ROC-AUC scores of the remaining classifiers fall between 0.76 and 0.83. This range indicates that the true positive and false positive rates are which is consistent with expectations.

Effects of Over-Sampling and Hybrid-Sampling

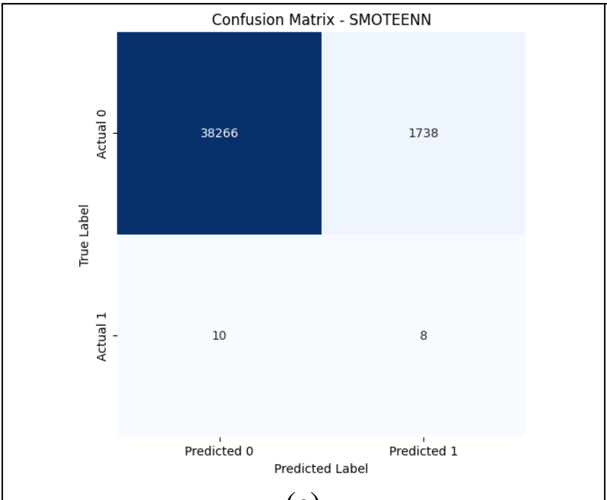
The FPR values are still unsatisfactory even if the TPR scores have improved for all ML classifiers after the data was under-sampled. Raising the TPR while lowering the FPR is the goal. Several over-sampling and combination techniques have been used to balance the training data in order to accomplish this balance. The confusion matrices in Figure 6 show how the ensemble classifier performs when using a variety of over-sampling techniques, such as SMOTE, ADASYN, SMOTEENN, and SMOTETOMEK. Interestingly, when compared to the outcomes of under-sampling strategies, the TP values show a decline, as compared to those of the under-sampling strategies. It is important to note that, in spite of this decline in TP values, FP values have decreased in comparison to the numbers obtained using under-sampling techniques, implying that the ensembles are classifying non-anomalous transactions more correctly using these techniques as compared to under-sampling techniques.



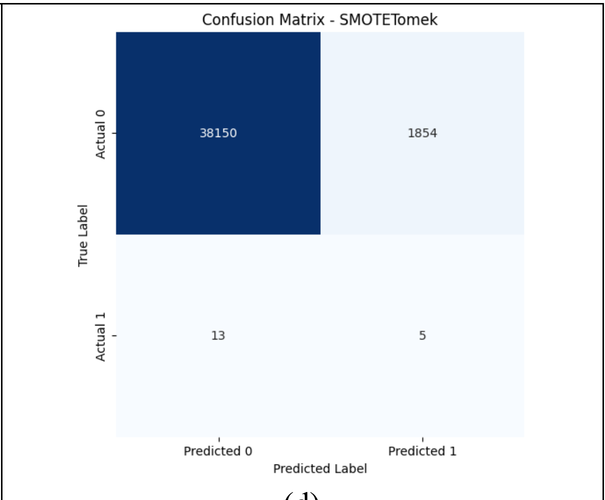
(a)



(b)



(c)



(d)

Figure 6: Confusion Matrix for Ensembles from Over-Sampling and Hybrid-Sampling

Table 5 demonstrates that various over-sampling and hybrid-sampling strategies have reduced false positive rates (FPRs) for both solo and ensemble machine learning classifiers. True positive rates (TPRs) are lowered as a result of this FPR decrease. AdaBoost has the greatest TPR of any classifier, with 0.722222 for all over-sampling techniques. Using the SMOTE approach, however, the Ensemble Hard Voting (EHV) classifier achieves the lowest FPR of 0.033547. Classifiers like Decision Tree (DT), Gradient Boosting (GBoost), and AdaBoost have comparatively higher FPRs ranging between 0.042771 and 0.133412, than ensemble classifiers like Ensemble Stacked (ES) and Ensemble Soft Voting (ESV) which have FPR values between 0.034997 and 0.060419. When it comes to average TPR, the GradientBoost (GB) classifier consistently displays the TPR over 50%, whereas the majority of classifiers fall around 0.31.

	SMOTE		**ADASYN**		**SMOTEENN**		**SMOTETOMEK**	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
DT	0.277778	0.054195	0.277778	0.058694	0.444444	0.045170	0.277778	0.049020
GBoost	0.500000	0.073943	0.500000	0.075817	0.611111	0.076367	0.555556	0.066718
RF	0.277778	0.050520	0.277778	0.053920	0.444444	0.042771	0.277778	0.046145
AdaBoost	0.722222	0.129937	0.722222	0.129812	0.722222	0.133412	0.722222	0.130712
ES	0.277778	0.056069	0.277778	0.060419	0.444444	0.043446	0.277778	0.046345
EHV	0.277778	0.033547	0.277778	0.034997	0.444444	0.036971	0.277778	0.033772
ESV	0.277778	0.052170	0.277778	0.055719	0.444444	0.044321	0.277778	0.047295

Table 5: Comparison between TPR and FPR values of ML classifiers after over-sampling and hybrid-sampling

Moving on to ROC-AUC performance, Table 6 shows that AdaBoost performs better than all other classifiers, with the greatest ROC-AUC scores across all over-sampling techniques, ranging from 0.794405 to 0.796205. The DT classifier, on the other hand, produces the lowest ROC-AUC scores, which range from 0.609542 to 0.699637. The most successful sampling method is SMOTEENN, which produces better ROC-AUC scores for the majority of classifiers, such as 0.767372 for GBoost and 0.703737 for EHV. When compared to other over-sampling and hybrid-sampling techniques, the SMOTEENN over-sampling methodology performs better overall.

	SMOTE	ADASYN	SMOTEENN	SMOTETOMEK
DT	0.611792	0.609542	0.699637	0.614379
GBoost	0.713029	0.712091	0.767372	0.744419
RF	0.613629	0.611929	0.700837	0.615816
AdaBoost	0.796143	0.796205	0.794405	0.795755
ES	0.610854	0.608679	0.700499	0.615716
EHV	0.622116	0.621391	0.703737	0.622003
ESV	0.612804	0.611029	0.700062	0.615241

Table 6: ROC-AUC scores of ML classifiers after over-sampling and hybrid-sampling

Comparative analysis between the Sampling Techniques

Techniques for over-sampling and under-sampling have different impacts on blockchain anomaly detection. Using a variety of distance-based strategies, under-sampling algorithms choose an equivalent number of examples from the majority class to address the minority class instances. On the other hand, over-sampling strategies use various distance algorithms to create an equal number of synthetic instances from the minority class, starting with the majority class instances. The two sample strategies have different effects on how Bitcoin transaction data is categorized. Tables 7 to Table 10 provide a detailed comparison study. The confusion matrices in Figure 3 and Figure 6 highlight the significance of a thorough assessment of the model's performance under various sampling strategies by indicating a subtle trade-off between true positives and false positives.

	Under-Sampling		Over-Sampling		Hybrid-Sampling	
	RUS	XGBCLUS	SMOTE	ADASYN	SMOTEENN	SMOTETOMEK
DT	0.555556	0.777778	0.277778	0.277778	0.444444	0.277778
GBoost	0.611111	0.787778	0.500000	0.500000	0.611111	0.555556
RF	0.611111	0.722222	0.277778	0.277778	0.444444	0.277778
AdaBoost	0.722222	0.666667	0.722222	0.722222	0.722222	0.722222
ES	0.611111	0.833333	0.277778	0.277778	0.444444	0.277778
EHV	0.611111	0.722222	0.277778	0.277778	0.444444	0.277778
ESV	0.666667	0.777778	0.277778	0.277778	0.444444	0.277778

Table 7: TPR of ML classifiers after different sampling techniques

Table 7 demonstrates that, in contrast to over-sampling and mixed sampling techniques, TPR scores are comparatively high for under-sampling techniques like RUS and the suggested XGBCLUS method. On the other hand, in terms of FPR scores, over-sampling and hybrid strategies outperform under-sampling techniques. Among under-sampling approaches, XG-BCLUS achieves the maximum TPR value of 0.83, which is also the highest over all under-sampling, over-sampling, and combination methodology. SMOTEENN achieves the greatest TPR value of 0.72, and shows higher TPR on an average for all ML classifiers among over-sampling and hybrid-sampling techniques. When it comes to raising TPRs, under-sampling techniques work better than over-sampling techniques. A higher percentage of abnormal transactions can be correctly identified by ML classifiers using under-sampling techniques.

	Under-Sampling		**Over-Sampling**		**Hybrid-sampling**	
	RUS	XGBCLUS	SMOTE	ADASYN	SMOTEENN	SMOTETOMEK
DT	0.312369	0.266548	0.054195	0.058694	0.045170	0.049020
GBoost	0.242276	0.181282	0.073943	0.075817	0.076367	0.066718
RF	0.210154	0.165708	0.050520	0.053920	0.042771	0.046145
AdaBoost	0.167858	0.123863	0.129937	0.129812	0.133412	0.130712
ES	0.240026	0.178357	0.056069	0.060419	0.043446	0.046345
EHV	0.196305	0.131637	0.033547	0.034997	0.036971	0.033772
ESV	0.267398	0.199555	0.052170	0.055719	0.044321	0.047295

Table 8: FPR of ML classifiers after different sampling techniques

All over-sampling and combination approaches yield the best value of 0.03 in terms of FPR scores, as indicated in Table 8. Under-sampling techniques, however, produce a modest FPR score of 0.13 using XGBCLUS. Methods of oversampling are more successful than those of undersampling in lowering FPRs. ML classifiers may correctly detect a higher percentage of non-anomalous transactions when over-sampling techniques are used.

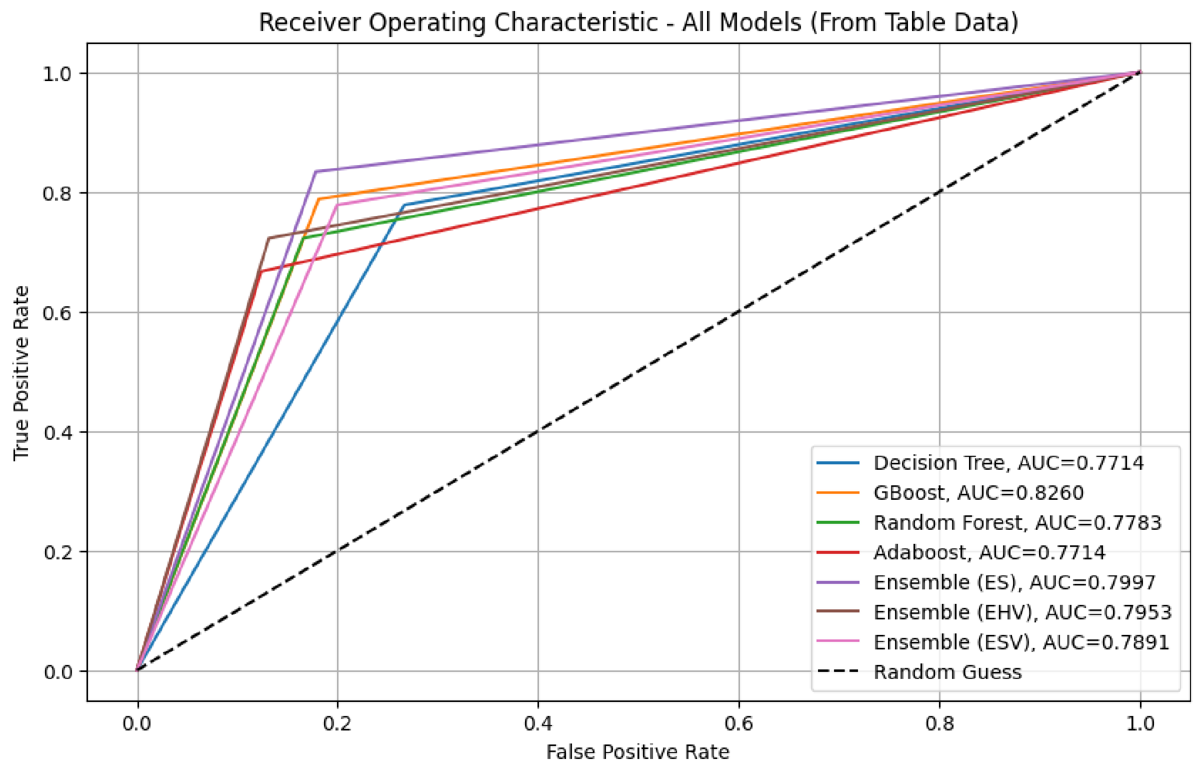


Figure 7: ROC-AUC curves for XGBCLUS

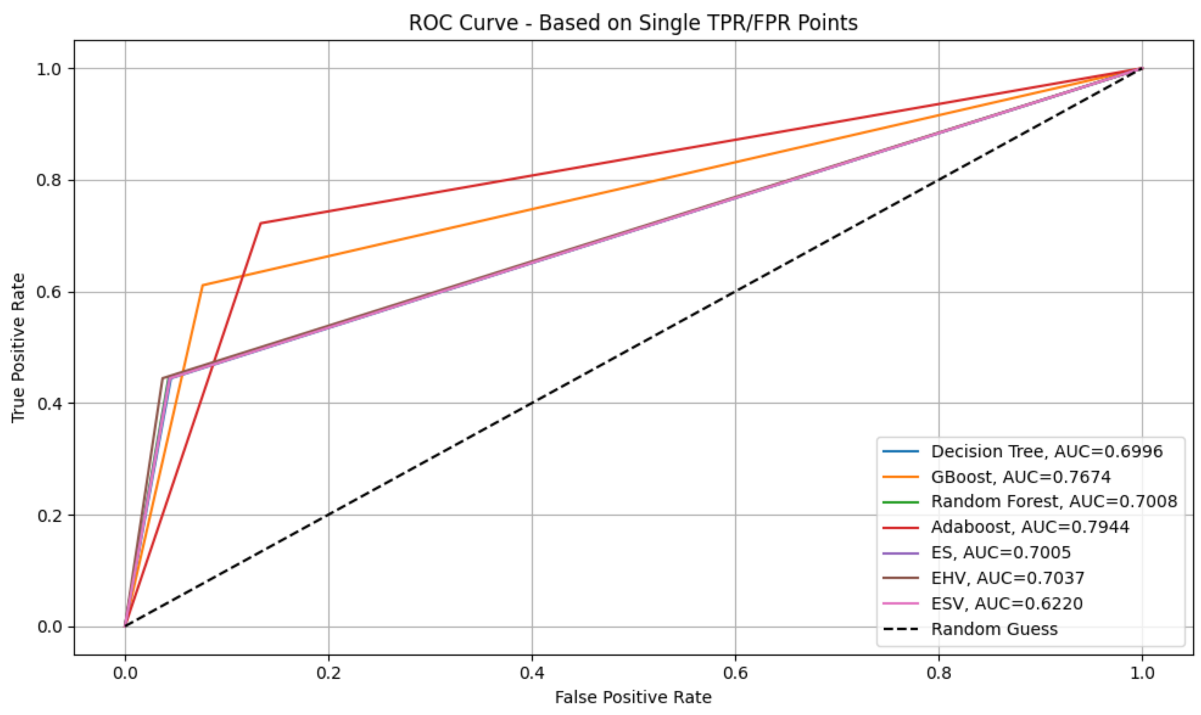


Figure 8: ROC-AUC Curves for SMOTEENN

Comparison between Figure 7 and Figure 8 shows that all ML classifiers exhibit improved ROC-AUC scores using XGBCLUS under-sampling method, outperforming the SMOTEENN over-sampling technique. This implies that, in terms of ROC-AUC scores, under-sampling techniques outperform over-sampling techniques.

Effects of Ensemble Classifiers

Although both anomalous and non-anomalous transactions can be identified by a single tree-based machine learning classifier, stacking and voting classifiers have been used to reduce mistakes and overfitting. Ensemble Stacked model with XGBCLUS under-sampling performs better than individual tree-based models, especially in increasing TPR, and ROC-AUC score, as shown by the experimental results in Table 9.

	Under-sampling (XGBCLUS)				**Over-sampling (SMOTEENN)**			
	Accuracy	TPR	FPR	ROC-AUC	Accuracy	TPR	FPR	ROC-AUC
DT	0.733	0.778	0.267	0.756	0.955	0.444	0.045	0.700
GBoost	0.819	0.788	0.181	0.826	0.923	0.611	0.076	0.767
RF	0.834	0.722	0.166	0.778	0.956	0.444	0.043	0.701
AdaBoost	0.866	0.667	0.124	0.771	0.867	0.722	0.133	0.794
ES	0.821	0.833	0.178	0.800	0.957	0.444	0.043	0.700

Table 9: Evaluation metrics for XGBCLUS and SMOTEENN

Table 10 illustrates how the three ensemble approaches outperform single classifiers in terms of test accuracy. Among all single classifiers, the voting (hard) classifier achieves the best accuracy of 96.6%, which is the peak value. When using under-sampled data from the XGBCLUS technique, the voting (hard) classifier achieves the greatest TPR or sensitivity value of 0.83 among the ensemble classifiers.

	Under-Sampling		**Over-Sampling**		**Hybrid-Sampling**	
	RUS	XGBCLUS	SMOTE	ADASYN	SMOTEENN	SMOTETOMEK
DT	0.687572	0.733472	0.945505	0.941007	0.954600	0.950677
GBoost	0.757658	0.818725	0.925866	0.923992	0.923492	0.933112
RF	0.789766	0.834241	0.949178	0.945780	0.956999	0.953551
AdaBoost	0.832092	0.876043	0.869997	0.870121	0.866523	0.869222
ES	0.759907	0.821623	0.943631	0.939283	0.956324	0.956324
EHV	0.803608	0.868297	0.966144	0.964694	0.962795	0.965919
ESV	0.732572	0.800435	0.947529	0.943981	0.955450	0.952401

Table 10: Accuracy of ML classifiers after different sampling techniques

Even though the TPR values for the three ensemble classifiers show a slight decrease with over-sampled data, the voting (hard) classifier achieves the best FPR value of 0.43. Additionally, ensemble classifiers work well to lower the FPR, which improves the accuracy of non-anomalous transaction identification. The voting (hard) classifier has the highest accuracy of 0.96 and a respectable ROC-AUC score of 0.70, outperforming the other two ensemble approaches.

Although identifying harmful transactions is the main objective, a strong emphasis on accurately detecting non-malicious transactions is maintained. In this sense, our research aims to reduce false positive rates in order to increase the true positive rate. In order to locate unusual transactions, we conclude that the ensemble approach performed better than the other classifiers that were investigated. Our ensemble approach performs exceptionally well in terms of accuracy and FPR value. Two key differences between the studies are that our research includes explainability and decision rules, which were not included in their analysis.

SHAP-based Explainability Analysis

Explainable AI (XAI) is a set of approaches, methodologies and processes that aim to make the decision-making processes of AI systems discrenable by people. It seeks to address the 'black box' character of many AI models, by offering insights into how these systems make certain predictions.
SHAP (SHapley Additive exPlanations) is an advanced method in Explainable AI used to interpret ML model predictions. It is based on Shapley value concept from cooperative game theory, which provides a fair way to allocate contributions among participants in a collaborative setting. SHAP calculates the contribution of each feature to a model's prediction using Shapley values. These values are derived by averaging the marginal contributions of a feature across all possible subsets of features.

One effective method for understanding our Machine Learning (ML) model predictions is SHAP. It helps us explain why a model made a particular choice by giving us a means to assign each feature's contribution to a particular prediction. SHAP values are calculated in this study using the SHAP KernelExplainer method to explain the behaviour of the ensemble model for both individual instances and the complete dataset.

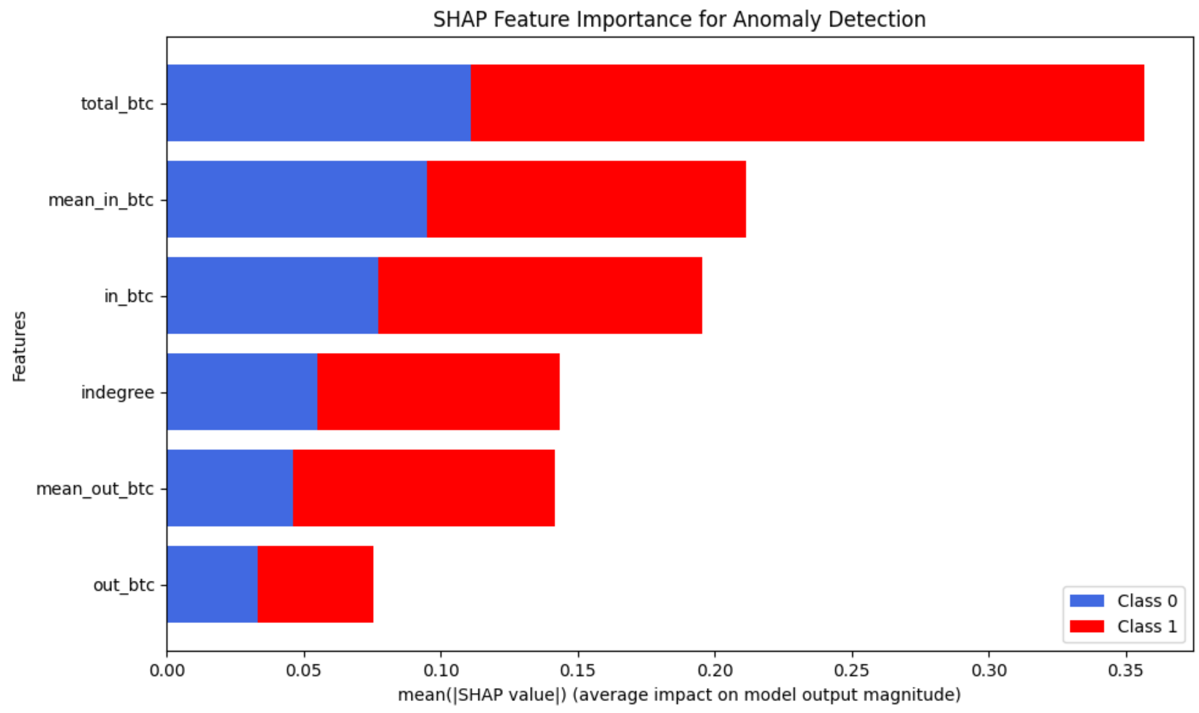


Figure 9: Hierarchy of the features contributing to the classification

The SHAP feature importance bar plot provides a visual representation of how each feature contributes, on average, to the model’s predictions for both normal (Class 0) and anomalous (Class 1) Bitcoin transactions. In this plot, the x-axis represents the mean absolute SHAP value, indicating the average impact of each feature on the prediction magnitude. The red and blue bars show the average contribution of each feature toward Class 1 and Class 0 predictions, respectively. Normal transactions are represented by blue colour SHAP values, whereas red colour SHAP values show their average influence on detecting anomalous transactions.

A hierarchical overview of features, arranged according to how they contribute to the model's output, is shown in Figure 9. The strength of the feature's influence is shown by the magnitude of the SHAP values A feature's influence on the model's anomaly prediction increases as its absolute SHAP value becomes larger, indicating a greater impact on the final classification decision.

This implies that regardless of whether the feature pushes the prediction toward Class 0 (normal) or Class 1 (anomalous), the magnitude of the SHAP value reflects how important that feature is in shaping the model’s output.

total_btc is the most influential feature, with a dominant contribution toward classifying transactions as anomalous (Class 1). This suggests that total amount of Bitcoin exchanged has the biggest impact on the model's judgment and transactions with higher total BTC values are more likely to be classified as anomalous. This aligns with force plot results, where high transaction totals strongly pushed predictions toward anomaly. mean_in_btc and in_btc also have strong positive impacts for Class 1, indicating that larger average or total incoming values are key drivers of anomalous classifications. indegree exhibits influence for both classes but contributes more toward Class 1, especially when extremely high (as seen in the second force plot with indegree = 478.0).

Conversely, the feature "out_btc" contributes the smallest average SHAP value, making it the least significant trait. Accordingly, "mean_out_btc" is the second least significant feature, suggesting that the amount of Bitcoin leaving the system is not very significant in relation to other aspects. Hence mean_out_btc and out_btc exhibit lower overall impact but still lean toward supporting Class 1 prediction. It is noteworthy that although "mean_out_btc" has a moderate impact, it is not as strong as "mean_in_btc" or "total_btc." This implies that, in contrast to outgoing transactions, entering Bitcoin transactions might offer more potent signs of irregularities.

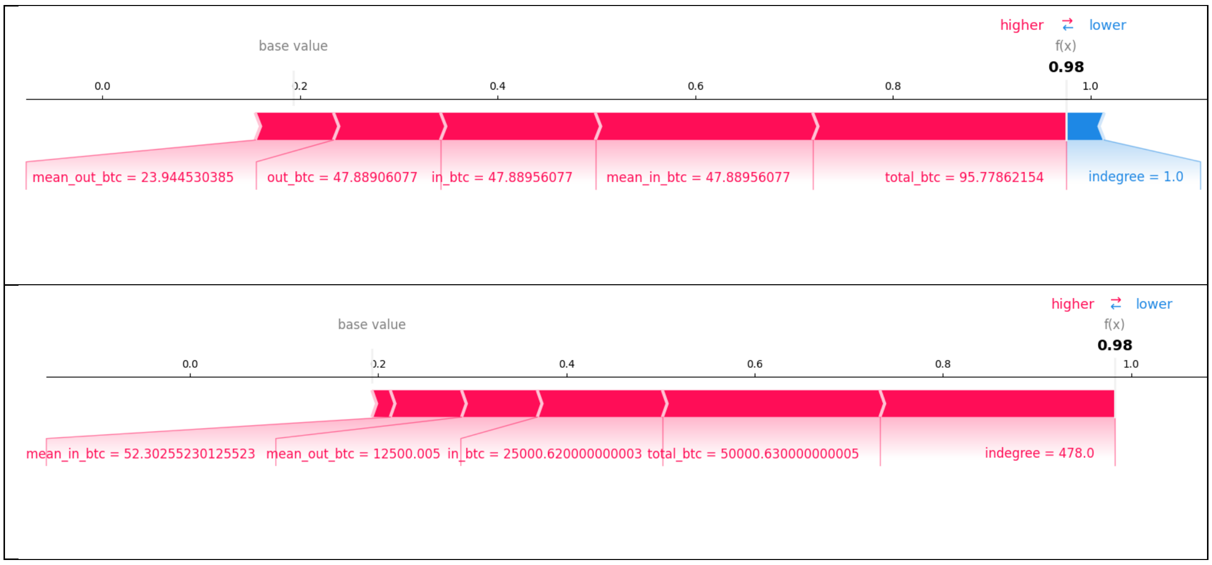


Table 11: SHAP Force Plots for Anomalous Transactions

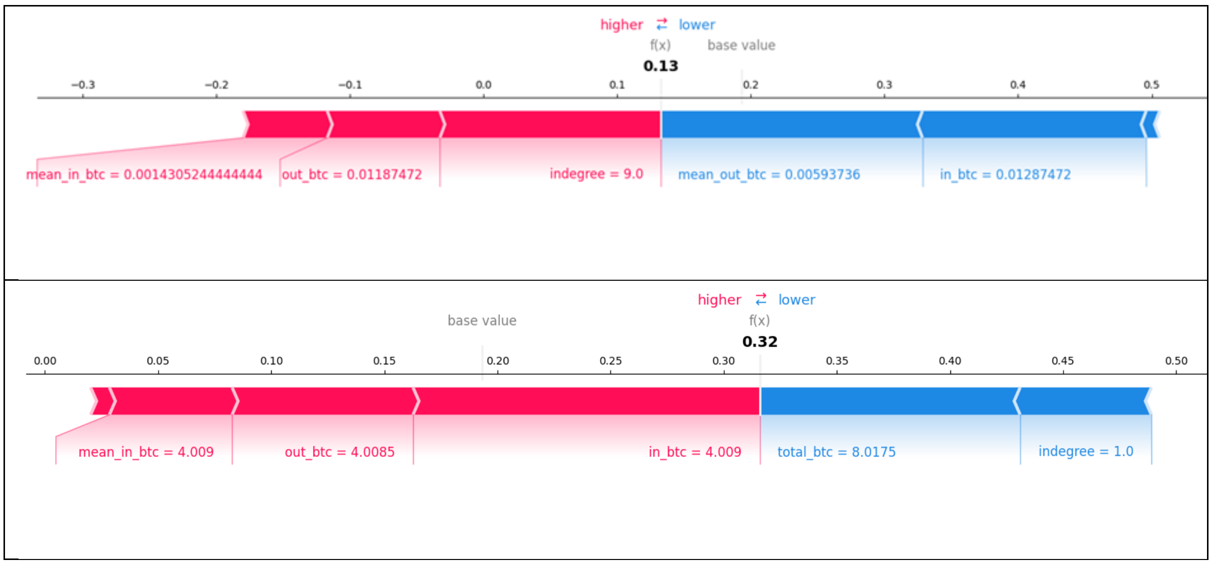


Table 12: SHAP Force Plots for Non-Anomalous Transactions

The ensemble model's ability to differentiate between typical and unusual Bitcoin transactions is demonstrated by the SHAP Force Plots. Force plots were utilized to visualize the contribution of individual features toward the model’s decision on whether a Bitcoin transaction is anomalous or not. The color-coded SHAP values demonstrate the many ways in which features influence the model's predictions by contributing to Class 0 (blue) and Class 1 (red). The visualizations aid in understanding how specific features drive classification outcomes, especially in distinguishing between anomalous and normal transactions. The base value for transactions is approximately 0.2. Notably, the scores for anomalous transactions are 0.98 for both the instances and for non-anomalous are 0.13, 0.32.

Anomalous Transactions:

The first two plots correspond to transactions classified as anomalous, both with high predicted probabilities (approximately 0.98). These plots reveal the strong influence of features related to transaction value on the anomaly classification. High values in total_btc, mean_in_btc, in_btc, out_btc, and mean_out_btc significantly contribute to pushing the prediction towards anomaly. These features reflect unusually large transaction volumes, which are typical indicators of potentially suspicious behavior. total_btc, representing the cumulative transaction amount, also exerts a strong positive force on the anomaly prediction. Interestingly, in the first plot, indegree = 1.0 (i.e., the number of incoming edges or connections) has a minor negative influence, but this is insufficient to outweigh the overwhelming contribution of the transaction value-related features. In the second plot, a much higher indegree (478.0) actually contributes positively toward the anomaly classification, suggesting that transactions with a vast number of connections may also be suspicious. These insights indicate that both high monetary values and large network interactions can serve as strong indicators of anomalous behavior in Bitcoin transactions.

Normal Transactions:
The third and fourth plots represent transactions classified as normal, with predicted probabilities of 0.32 and 0.13, respectively. In these instances, feature values contribute minimally or negatively toward the anomaly prediction: In both plots, the values of mean_in_btc, in_btc, and out_btc are relatively low, contributing modestly to a higher anomaly score but ultimately being outweighed by other features. total_btc and indegree have a strong negative impact on the anomaly score, especially in the fourth plot where the indegree is 9.0 and feature values such as mean_out_btc and in_btc are extremely small. These transactions show balanced or low-volume activities, which align with typical normal behavior in the network.

The force plots showed how these features act on individual predictions:

(a) For anomalous transactions, high total_btc, mean_in_btc, and in_btc values collectively push predictions toward Class 1. These are transactions with either very large amounts or extensive connectivity.
(b) For normal transactions, the same features tend to have lower values and contribute negatively or minimally, keeping the prediction closer to Class 0.

This consistency across global and local SHAP interpretations highlights the model's heavy reliance on financial magnitude and structural transaction context to make accurate predictions.

Overall, the SHAP force plots illustrate that the model assigns higher anomaly scores to transactions with significantly large values and/or extensive network interaction, while normal transactions are characterized by moderate or low feature values across the same variables

When comparing all four cases, it is clear that anomalous transactions can be easily identified because they have greater feature values than normal transactions. Furthermore, the SHAP values in these figures verify that "total_btc," mean_out_btc" " and "indegree" are important indicators of normal and unusual Bitcoin transactions. As a result, transaction anomalies are easier for humans to understand because to the SHAP architecture, which clearly describes how the model classifies data.

If these stacked plots are rotated by 90^o, we get a stacked force plot Figure 10. This plot helps visualize SHAP values across multiple instances, helping to compare how features contribute to predictions across samples.

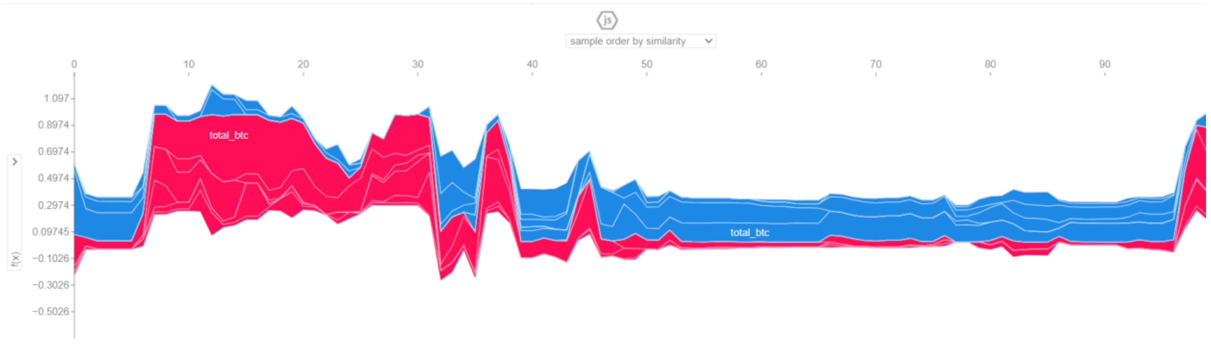


Figure 10: Stacked Force Plot

The samples are ordered on the X-axis according to their similarity. The sample prediction is plotted on the Y-axis. The left side can be seen with high peak than left, confirming that the samples being predicted as Anomalous (1) are grouped together on the left side.

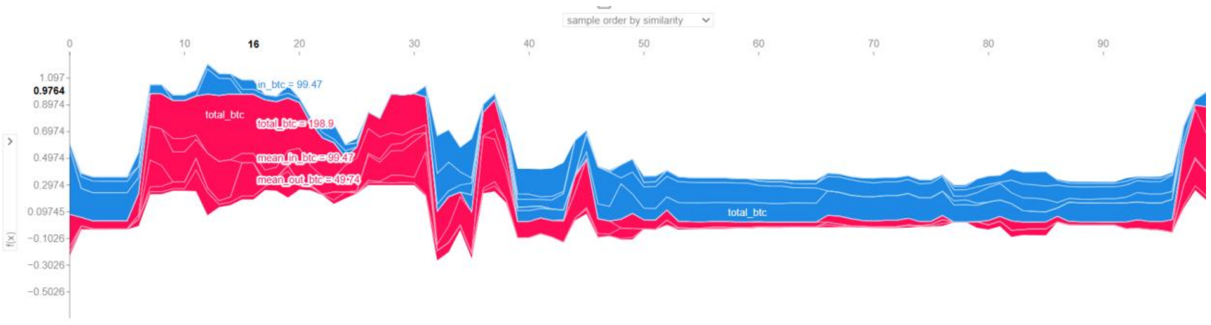


Figure 11: Stacked Force Plot (Sample No. 16)

When hovered over this plot, Figure 11, on Python, we can see that for Sample no. 16 (X-axis), the prediction is 0.9764 (Y-axis). That is, this sample is classified as Anomalous.

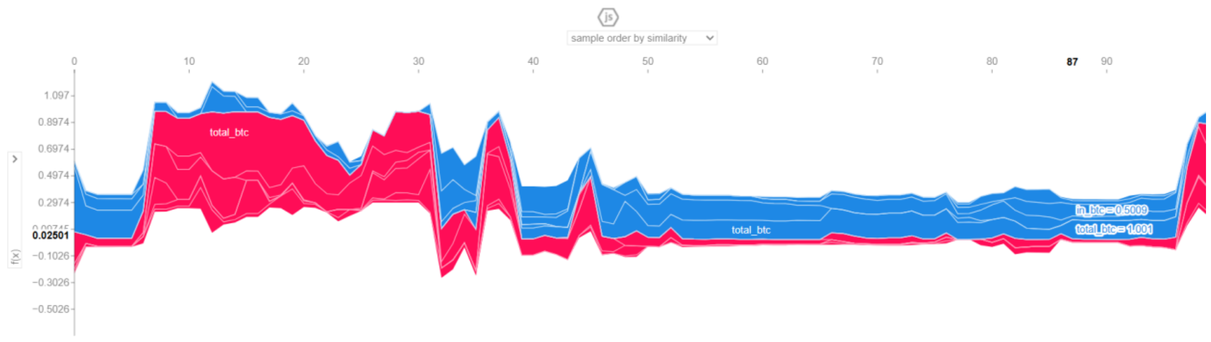


Figure 12: Stacked Force Plot (Sample No. 87)

Here, Figure 12, Sample no. 87 has been predicted as Non-Anomalous since the prediction is 0.02501.

From here, it can be concluded that transactions with greater number of bitcoins involved tend to be strongly linked to be classified as suspicious or anomalous

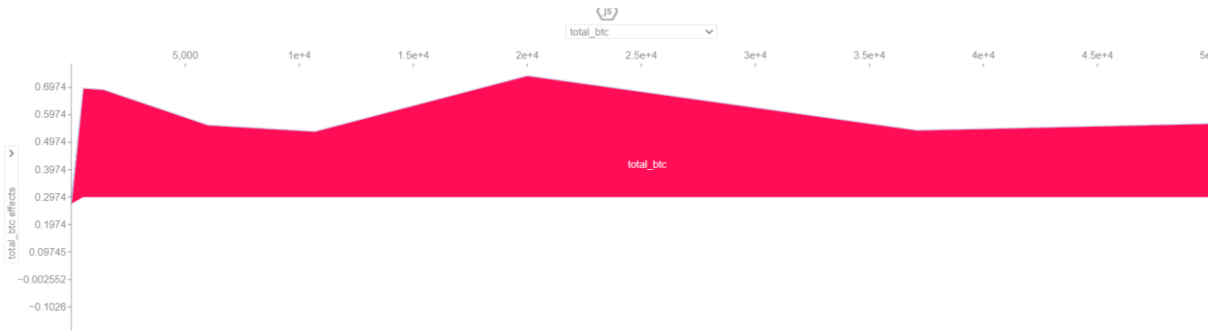


Figure 13: Stacked Force Plot (total_btc v/s SHAP values of total_btc)

In Figure 13, SHAP values of total_btc plotted against its value range show that total_btc has consistently high positive SHAP values, indicating that it is strongly influential in classifying the transactions as Anomalous. As the value of total_btc increases, it can be implied that the transactions will more likely be Anomalous.

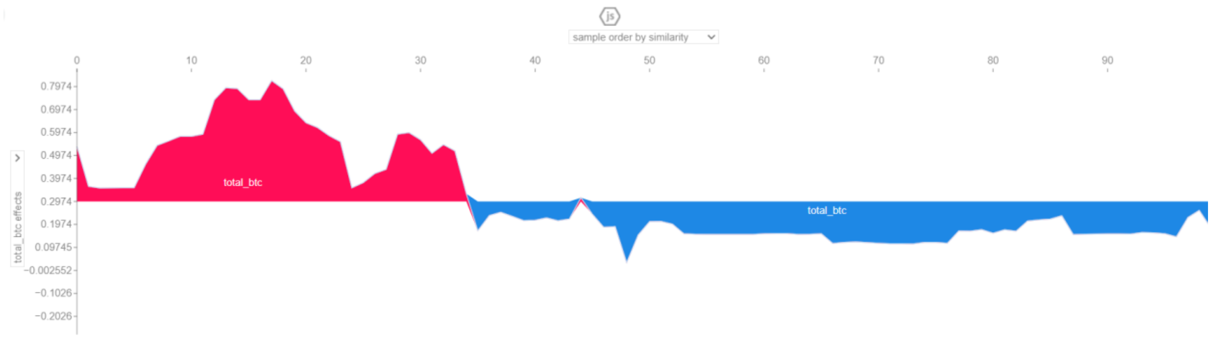


Figure 14: Stacked Force Plot (Samples ordered by similarity v/s SHAP values of total_btc)

In Figure 14, SHAP values of total_btc against the samples indicate that for smaller values of total_btc i.e. transactions with small number of Bitcoins involved have negative SHAP values and hence push the transaction towards being classified as Non-Anomalous. The values of total_btc in the right side of the graph (the blue region) range from around 0 to 25 bitcoins. Whereas, the transactions that have total_btc greater that 25 (approximately) are pushing towards classifying the transaction as Anomalous, since the SHAP values are higher. Therefore, a threshold of ~ 25 bitcoins can be identified, above the transactions will likely have greater likelihood to be classified as anomalous.

Anomaly Rule Generation and Interpretability Analysis

Using tree representations to interpret anomaly rules can aid in comprehending the reasons behind the classification of particular cases as anomalies. It gives a detailed explanation of the decision-making process and identifies the elements and thresholds that were most important in the process. Tree visualization provides an organized and comprehensible method of understanding how the model makes decisions based on input features. The tree-based machine learning classifiers are investigated for identifying anomalous Bitcoin transactions. In Figure 15, the decision tree with max-depth 10 is visualized. The root node, the top node in the tree, is the representation of the complete dataset. Every succeeding node denotes a decision point determined by a certain threshold and characteristic. A feature and its associated threshold are represented by each node as it moves down the sub-trees. Whether or not an instance's feature values meet the specified threshold determines which branch it is sent to. In this scenario, anomalous or non-anomalous is the prediction class that each leaf node corresponds to.

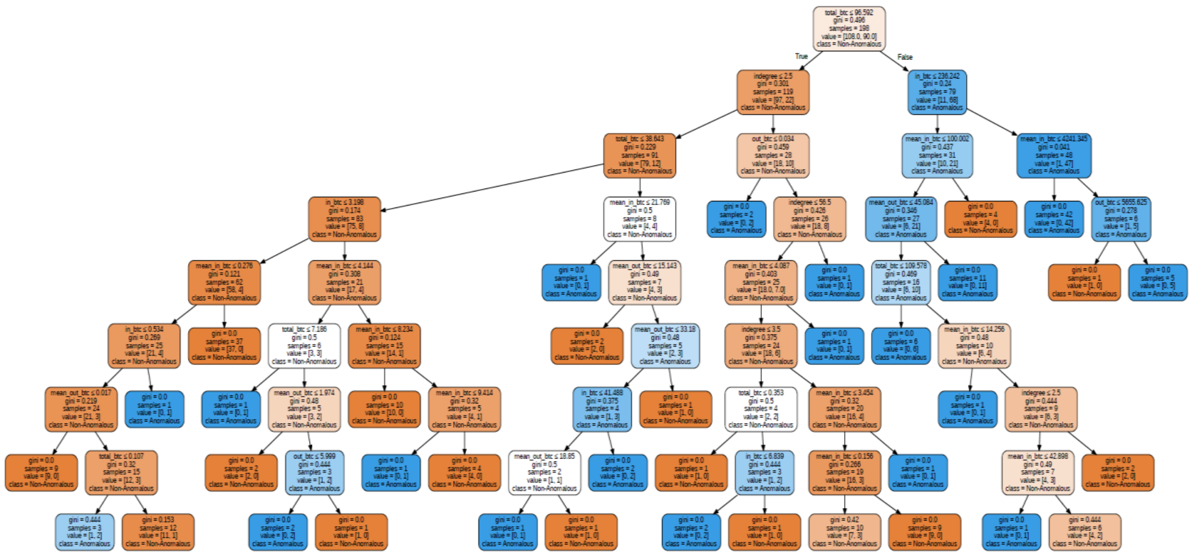


Figure 15: Visualisation of Decision Tree to generate anomalous rules

Feature Name	**Importance**
total_btc	0.582618
mean_in_btc	0.160639
in_btc	0.082033
mean_out_btc	0.071227
out_btc	0.054229
indegree	0.049253

Table 13: Feature Importance values based on reduction in Gini Impurity

The declining feature importance is shown in Table 13, where the total normalized reduction in Gini impurity attained by dividing the corresponding feature along the tree is represented by numerical values. Not unexpectedly, with a value of 0.582618, the feature used to divide at the root node, "total_btc," has the most relevance. This suggests that "total_btc" and "mean_in_btc" (0.160639) are the most important characteristics in identifying whether a Bitcoin transaction is anomalous. A series of feature-threshold rules that result in the anomaly classification can be obtained by moving through the decision tree from the root to an anomalous leaf node. These "anomaly rules" offer clear and logical explanations of the model's reasoning behind decisions.

Table 14 provides a set of criteria and confidence scores that, in relation to the tree in Figure 15, results in an anomalous node. A feature importance metric is also provided by decision trees. A feature's total impact on the decision-making process is greater when it is at the tree's root. Thus, total_btc is the most crucial factor that has the biggest impact on determining whether a transaction is unusual or not. Additionally, mean_in_btc and mean_out_btc

(0.071227) both produce useful results when it comes to categorizing unusual transactions. In certain instances, though, the indegree (0.049253) might also help identify anomalous transactions.

Anomaly Rule	**Class**	**Total Samples**	**Correctly Identified**	**Confidence (%)**
1. If (total_btc >96.592, in_btc>236.242, mean_in_btc<=4241.345) then	Anomalous	42	42	100
2. If (total_btc >96.592, in_btc>236.242) then	Anomalous	48	47	98
3. If (total_btc >96.592, in_btc>236.242, mean_in_btc>4241.34) then	Anomalous	6	5	83
4. If (total_btc>96.592, in_btc<=236.242, mean_in_btc<= 100.002 & mean_out_btc>45.084) then	Anomalous	11	11	100
5. If (total_btc<=96.592, indegree>2.5 & out_btc>0.034) then	Anomalous	2	2	100
6. If (total_btc<=96.592, indegree>2.5, out_btc>0.034 & indegree>56.5	Anomalous	1	1	100

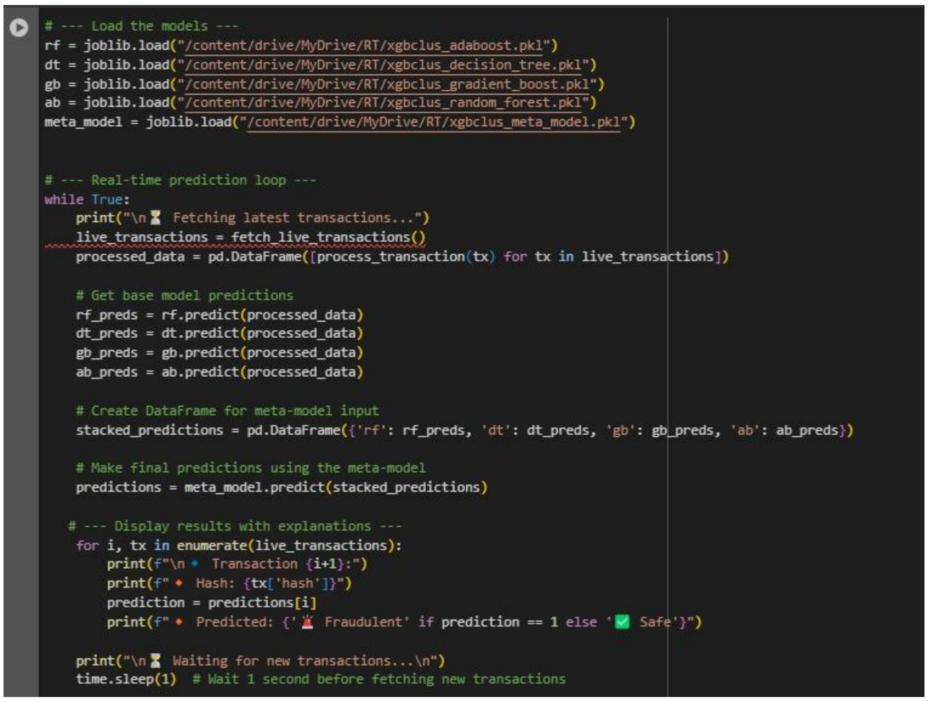
Table 14: Significant rules for being an anomalous transaction

The confidence score indicates the certainty of the model in predicting that the transaction is anomalous given the conditions outlined in the rules. The highest possibility of being anomalous is indicated by the highest confidence scores of 100%, which are provided by Rules 1, 4, 5, and 6. Rules 2 and 3, on the other hand, have somewhat lower confidence scores—98% and 83%, respectively. According to Table 14, the more carefully decision criteria are examined, the more confidently anomalous transactions can be identified. Even though we have specially designed anomaly rules for our Bitcoin transaction data, this approach has the potential to produce useful decision rules in a variety of fields, including blockchain.

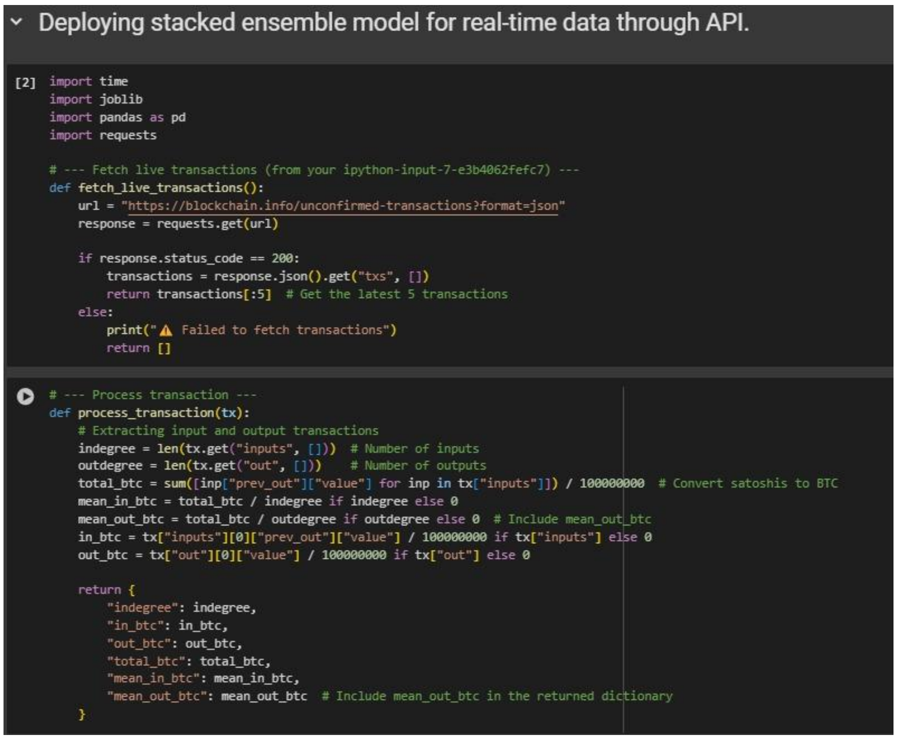
Model Deployment

Following an extensive comparative analysis of multiple tree-based classifiers and ensemble techniques, the stacked ensemble model, trained on a sample balanced using the XGCLUS algorithm, emerged as the best-performing approach for detecting anomalous Bitcoin transactions. This model integrates the strengths of four base classifiers — Decision Tree, Random Forest, Gradient Boosting, and AdaBoost — with logistic regression serving as the meta-classifier. The stacking technique was selected due to its superior performance across key evaluation metrics, including accuracy, true positive rate (TPR), and ROC-AUC score, outperforming both individual classifiers and other ensemble methods such as hard and soft voting. The complexity and class imbalance included in blockchain transaction data were effectively addressed by its capacity to capture a variety of decision boundaries through model heterogeneity.

The final stacked ensemble model has been deployed on Google Colab to detect anomalies in blockchain transaction data in real time. The platform offers a flexible and interactive environment for running the model on live or batch transaction datasets, enabling efficient identification of suspicious patterns. This setup provides a practical demonstration of the model's capabilities, showcasing how advanced ensemble techniques can be leveraged for real-time anomaly detection in digital payment systems.



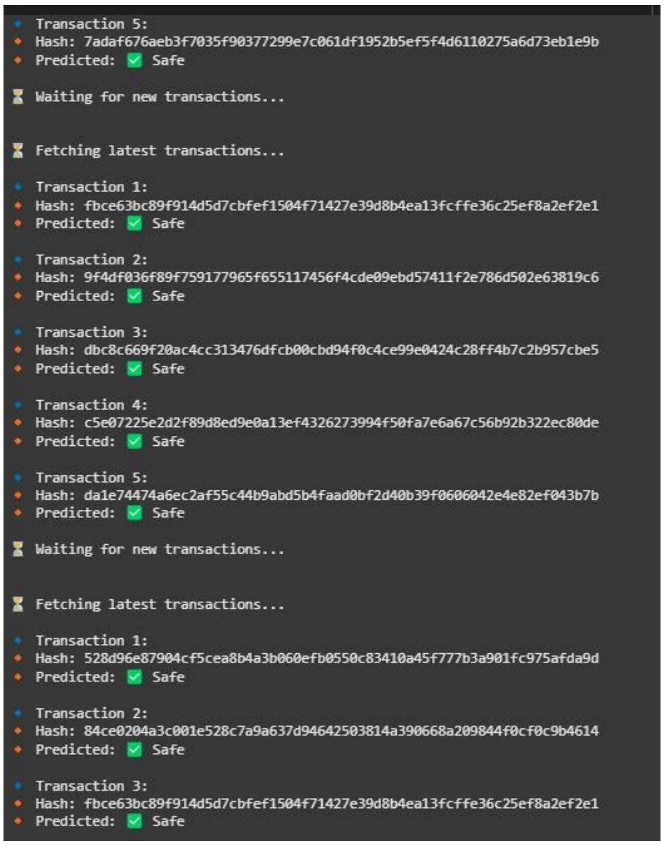
```
# --- Load the models ---
rf = joblib.load("/content/drive/MyDrive/RT/xgbclus_adaboost.pkl")
dt = joblib.load("/content/drive/MyDrive/RT/xgbclus_decision_tree.pkl")
gb = joblib.load("/content/drive/MyDrive/RT/xgbclus_gradient_boost.pkl")
ab = joblib.load("/content/drive/MyDrive/RT/xgbclus_random_forest.pkl")
meta_model = joblib.load("/content/drive/MyDrive/RT/xgbclus_meta_model.pkl")


# --- Real-time prediction loop ---
while True:
    print("\n⏳ Fetching latest transactions...")
    live_transactions = fetch_live_transactions()
    processed_data = pd.DataFrame([process_transaction(tx) for tx in live_transactions])

    # Get base model predictions
    rf_preds = rf.predict(processed_data)
    dt_preds = dt.predict(processed_data)
    gb_preds = gb.predict(processed_data)
    ab_preds = ab.predict(processed_data)

    # Create DataFrame for meta-model input
    stacked_predictions = pd.DataFrame({'rf': rf_preds, 'dt': dt_preds, 'gb': gb_preds, 'ab': ab_preds})

    # Make final predictions using the meta-model
    predictions = meta_model.predict(stacked_predictions)

   # --- Display results with explanations ---
    for i, tx in enumerate(live_transactions):
        print(f"\n🔹 Transaction {i+1}:")
        print(f"🔸 Hash: {tx['hash']}")
        prediction = predictions[i]
        print(f"🔸 Predicted: {'🚨 Fraudulent' if prediction == 1 else '✅ Safe'}")

    print("\n⏳ Waiting for new transactions...\n")
    time.sleep(1)  # Wait 1 second before fetching new transactions
```



Deploying stacked ensemble model for real-time data through API.

```
[2] import time
    import joblib
    import pandas as pd
    import requests

    # --- Fetch live transactions (from your ipython-input-7-e3b4062fefc7) ---
    def fetch_live_transactions():
        url = "https://blockchain.info/unconfirmed-transactions?format=json"
        response = requests.get(url)

        if response.status_code == 200:
            transactions = response.json().get("txs", [])
            return transactions[:5]  # Get the latest 5 transactions
        else:
            print("⚠ Failed to fetch transactions")
            return []
```

```
# --- Process transaction ---
def process_transaction(tx):
    # Extracting input and output transactions
    indegree = len(tx.get("inputs", []))  # Number of inputs
    outdegree = len(tx.get("out", []))    # Number of outputs
    total_btc = sum([inp["prev_out"]["value"] for inp in tx["inputs"]]) / 100000000  # Convert satoshis to BTC
    mean_in_btc = total_btc / indegree if indegree else 0
    mean_out_btc = total_btc / outdegree if outdegree else 0  # Include mean_out_btc
    in_btc = tx["inputs"][0]["prev_out"]["value"] / 100000000 if tx["inputs"] else 0
    out_btc = tx["out"][0]["value"] / 100000000 if tx["out"] else 0

    return {
        "indegree": indegree,
        "in_btc": in_btc,
        "out_btc": out_btc,
        "total_btc": total_btc,
        "mean_in_btc": mean_in_btc,
        "mean_out_btc": mean_out_btc  # Include mean_out_btc in the returned dictionary
    }
```



```
🔹 Transaction 5:
🔸 Hash: 7adaf676aeb3f7035f90377299e7c061df1952b5ef5f4d6110275a6d73eb1e9b
🔸 Predicted: ✅ Safe

⏳ Waiting for new transactions...


⏳ Fetching latest transactions...

🔹 Transaction 1:
🔸 Hash: fbce63bc89f914d5d7cbfef1504f71427e39d8b4ea13fcffe36c25ef8a2ef2e1
🔸 Predicted: ✅ Safe

🔹 Transaction 2:
🔸 Hash: 9f4df036f89f759177965f655117456f4cde09ebd57411f2e786d502e63819c6
🔸 Predicted: ✅ Safe

🔹 Transaction 3:
🔸 Hash: dbc8c669f20ac4cc313476dfcb00cbd94f0c4ce99e0424c28ff4b7c2b957cbe5
🔸 Predicted: ✅ Safe

🔹 Transaction 4:
🔸 Hash: c5e07225e2d2f89d8ed9e0a13ef4326273994f50fa7e6a67c56b92b322ec80de
🔸 Predicted: ✅ Safe

🔹 Transaction 5:
🔸 Hash: da1e74474a6ec2af55c44b9abd5b4faad0bf2d40b39f0606042e4e82ef043b7b
🔸 Predicted: ✅ Safe

⏳ Waiting for new transactions...


⏳ Fetching latest transactions...

🔹 Transaction 1:
🔸 Hash: 528d96e87904cf5cea8b4a3b060efb0550c83410a45f777b3a901fc975afda9d
🔸 Predicted: ✅ Safe

🔹 Transaction 2:
🔸 Hash: 84ce0204a3c001e528c7a9a637d94642503814a390668a209844f0cf0c9b4614
🔸 Predicted: ✅ Safe

🔹 Transaction 3:
🔸 Hash: fbce63bc89f914d5d7cbfef1504f71427e39d8b4ea13fcffe36c25ef8a2ef2e1
🔸 Predicted: ✅ Safe
```

DISCUSSION

It can be difficult to identify anomalies in data that is extremely unbalanced. The substantial data imbalance tends to bias tree-based machine learning classifiers towards non-anomalous transactions, despite the fact that these classifiers show effectiveness in anomaly detection. For single tree-based ML classifiers, this imbalance frequently leads to low TPRs, as seen in Table 3. Addressing data imbalance prior to model training becomes essential as the study's main goal is the accurate detection of abnormal transactions.

There are two primary methods for balancing data: under-sampling and over-sampling. It is crucial for researchers to choose the right approach, especially when working with extremely unbalanced data. To balance the majority and minority classes, over-sampling techniques create synthetic data, but they might not be able to identify the majority of unusual transactions. However, by concentrating on minority samples, under-sampling approaches seek to balance classes, which results in low FPR and high TPR values. Hybrid-sampling techniques are generated by combining both over-sampling and under-sampling techniques.

Under-sampling may be appropriate for anomaly classification situations if it can increase the TP value while decreasing the FP value. As seen in the results analysis section, the widely used under-sampling techniques in use today frequently fail to achieve this balance. There are disadvantages to the RUS approach, even though it can help achieve greater class balance. By reducing the amount of training data available, this method may cause important samples to be lost, which would then result in less-than-ideal model performance. The NearMiss-1 technique shows a large FPR even though it improves model performance with a TPR score of 0.88. Understanding the shortcomings of NearMiss-1 and RUS, we have developed the XGBCLUS under-sampling technique to address these problems. Its ability to identify anomalous Bitcoin transactions allows XGBCLUS to outperforms the other techniques. When trained on under-sampled data using XGBCLUS, both individual and ensemble classifiers exhibit superior TPR, FPR, and ROC-AUC values.

In order to emphasize the differences between under-sampling and over-sampling strategies, our research also looks into over-sampling and hybrid-sampling balancing techniques. The comparative findings show that while under-sampling techniques increase the FPR, they are effective in raising the TPR value. In contrast to imbalanced data, over-sampling techniques help to achieve a higher TPR and a lower FPR. ADASYN performs better than SMOTE among the various over- and combined-sampling approaches, but SMOTEENN surpasses SMOTETOMEK.

We present individual tree-based ML classifiers as well as voting (hard and soft) and stacking ensemble classifiers in the classification space. When applied to both under- and over-sampled data, ensemble classifiers clearly outperform individual machine learning classifiers. With the highest TPR and lowest FPR values for both under- and over-sampled data, the ensemble classifier outperforms the other two voting classifiers. The FPR becomes significant as a crucial evaluation indicator in cases with unbalanced data. When compared to single ML classifiers, the ensemble classifiers consistently produce higher FPR values, highlighting how well ensemble techniques handle unbalanced data.

However, machine learning models frequently function as opaque "black boxes". We explore the field of explainable AI utilizing SHAP, to validate human assumptions and model predictions. Finding the most important feature—in this case, ‘total_btc’—is made possible by averaging the SHAP values over all dataset instances. The significance of this prominent characteristic is clearly discernible to human observers. We also elaborate on anomalous rules that are derived from DTs. ‘total_btc’ appears as the root node, indicating its critical function in categorizing unusual transactions. Together, the tree representation and SHAP's analysis identify ‘total_btc’ as the key characteristic assisting in the detection of unusual Bitcoin transactions. All things considered, SHAP offers a clear and intelligible method to comprehend how every attribute assists in the anomaly identification procedure. This can assist domain experts and data analysts in finding trends, correlations, and possible abnormalities in the data that the model uses to make predictions. Furthermore, utilizing tree representations to analyze anomalous rules enables in comprehending the rationale behind the classification of some events as anomalies. It offers a detailed explanation of the decision-making process and identifies the characteristics and cutoff points that were most important in reaching the conclusion. This openness is particularly useful in fields where explainability is crucial since it enables stakeholders to verify the model's conclusions and spot possible problems with the data or model.

Our study's conclusions have important results for improving anomaly detection and blockchain security. The effectiveness of the suggested ensemble model in identifying unusual Bitcoin transactions points to a viable strategy for bolstering blockchain system security. Furthermore, the anomaly identification process is made more transparent and understandable by the use of XAI techniques like SHAP analysis in conjunction with anomaly rules for interpretability analysis. By putting such explainability methods into practice, blockchain technology's overall security can be strengthened and strong anomaly detection systems can be created. Furthermore, the investigation of diverse sampling approaches, such as the suggested under-sampling and combined-sampling procedures, aids in the creation of strategies for managing extremely unbalanced datasets in the blockchain space. This has further ramifications for detecting anomalies in other situations where unbalanced data is a frequent problem.

CONCLUSION

We have carried out a thorough comparative analysis in this study with the goal of identifying anomalies in blockchain transaction data. Although a lot of research has been done in this area, a common drawback has been the lack of justifications for model projections. Our study aims to address this limitation by utilizing Bitcoin transactions to integrate tree-based ensemble classifiers with XAI approaches and anomaly rules. Interestingly, the SHAP approach is essential for quantifying each feature's contribution to the model's output prediction. Additionally, the anomaly criteria aid in determining whether or not a Bitcoin transaction is an anomaly. As a result, it is easy to determine which properties are essential for anomaly identification. In order to improve our approach even more, we have studied an under-sampling algorithm called XGBCLUS. This algorithm's performance has been evaluated against other well-known under-sampling and over-sampling strategies, and it helps balance anomalous and non-anomalous transaction data. Following that, the outcomes of voting and stacking ensemble classifiers are compared to those of several single tree-based classifiers. Our results clearly show that TPR and ROC-AUC scores have improved with our suggested under-sampling technique, XGBCLUS. Additionally, when compared to well-known single ML classifiers, ensemble classifiers have demonstrated better performance. Our study concludes by showing that ensemble stacking classifier using the XGBCLUS under-sampling algorithm has achieved the best success in identifying unusual transactions on the Blockchain when paired with appropriate balancing techniques.

LIMITATIONS

Although the suggested stacked ensemble model is highly effective in identifying anomalous Bitcoin transactions, several limitations must be acknowledged. A key challenge lies in accessing high-quality, labeled blockchain transaction data. Publicly accessible datasets are often limited in size or lack comprehensive annotations for anomalous behavior, which hinders the scope of training and assessment. Also, whereas SHAP increases model explainability, it is computationally expensive—particularly for ensemble models—because it considers all possible feature combinations in computing their contributions, leading to increased processing time.

In addition, the training of machine learning models using massive blockchain transaction datasets requires considerable computational power, especially when applying intricate ensemble approaches. This requirement is further intensified by applying more sophisticated methods like deep learning, which need powerful hardware and highly optimized environments. Equally, containerizing models within Docker environments for real-time execution and third-party consumption brings about further infrastructure, configuration, and maintenance overhead. These factors cumulatively impose practical restrictions for real-world applicability and scalability.

Additionally, limitations of blockchain infrastructure, including its decentralized nature, can lead to slower transaction rates and higher computational overhead, thus impacting the responsiveness of the system in real-time anomaly detection. The model's accuracy also largely relies on the quality of transaction data; noisy or tampered inputs could jeopardize detection results. Finally, the changing nature of fraud and anomaly behavior requires frequent updates and retraining of the models, and it proves to be difficult to maintain pace with changing strategies without retraining pipelines.

FUTUTRE SCOPE

While the data used within this research is comprised of Bitcoin transactions between 2011 and 2013, it does not present in a sequential chronological manner, therefore acting more as a representative sample of transactional activity and not temporal development. Model deployment has been performed on Google Colab in order to replicate real-time identification of suspicious blockchain transactions. Along with anomaly detection, the deployment architecture can be augmented with SHAP analysis, allowing for interpretability of the predictions of the model. The SHAP force plots, for instance, enable users to see whether Bitcoin-specific characteristics or network-based metrics are most driving a specific classification choice, thereby building confidence and transparency into the prediction mechanism.

In the future, a number of directions are available for widening the applicability of this research. The present stacked ensemble model can be wrapped as a Docker container for ease of reuse by external developers and incorporation in advanced blockchain security frameworks. Ensemble learning methods could be further tuned for improved prediction accuracy and resilience. Moreover, a study comparing Machine Learning models versus Deep Learning architectures can be done to determine the best method to use when detecting anomalies in blockchain transactions. Lastly, applying these approaches to other cryptocurrencies and blockchain platforms will serve to test their generalizability and applicability throughout the decentralized finance ecosystem at large.

REFERENCES

1. A. A. Monrat, O. S. (2019). A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities. *IEEE Access*.
2. Azar, A. T. (2022). A machine learning and blockchain based efficient fraud detection mechanism. *Sensors*.
3. Hasan, M. R. (2024). Detecting Anomalies in Blockchain Transactions using Machine Learning Classifiers and Explainability Analysis. *Blockchain: Research and Applications*, 17. doi:https://doi.org/10.1016/j.bcra.2024.100207
4. M. Rashid, J. K. (2022). A tree-based stacking ensemble technique with feature selection for network intrusion detection. *Applied Intelligence*.
5. Madhuparna Bhowmik, T. S. (2021). Comparative Study of Machine Learning Algorithms for Fraud Detection in Blockchain. *IEEE*.
6. Sarker, I. H. (2023). Machine Learning for Intelligent Data Analysis and Automation in Cybersecurity: Current and Future Prospects. *Ann. Data Sci*.
7. Signorini, M. P. (2018). Anomaly Detection tool for blockchaIn SystEms. *IEEE Access*.
8. Y. Xia, K. C. (2021). Multi-label classification with weighted classifier selection and stacked ensemble. *Information Sciences*.