IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2022.DOI

A New Framework for Fraud Detection in Bitcoin Transactions through Ensemble Stacking Model in Smart Cities

NOOR NAYYER[1], NADEEM JAVAID[1,*], MARIAM AKBAR[1], ABDULAZIZ ALDEGHEISHEM[2], NABIL ALRAJEH[3], MOHSIN JAMIL[4]

[1]Department of Computer Science, COMSATS University Islamabad, Islamabad 44000, Pakistan
[2]Department of Urban Planning, College of Architecture and Planning, King Saud University, Riyadh 11574, Saudi Arabia
[3]Department of Biomedical Technology, College of Applied Medical Sciences, King Saud University, Riyadh, 11633, Saudi Arabia
[4]Department of Electrical and Computer Engineering, Memorial University of Newfoundland, 230 Elizabeth Ave, St. John's, NL A1B 3X5, Canada

*Correspondence: Nadeem Javaid (nadeemjavaidqau@gmail.com)

This project is funded by Researchers Supporting Project number (RSPD2023R295), King Saud University, Riyadh, Saudi Arabia.

ABSTRACT Bitcoin has a reputation of being used for unlawful activities, such as money laundering, dark web transactions, and payments for ransomware in the context of smart cities. Blockchain technology prevents illegal transactions, but cannot detect these transactions. Anomaly detection is a fundamental technique for recognizing potential fraud. The heuristic and signature-based approaches were the foundation of earlier detection techniques, but tragically, these methods were insufficient to explore the entire complexity of anomaly detection. Machine Learning (ML) is a promising approach to anomaly detection, as it can be trained on large datasets of known malware samples to identify patterns and features of the transactions. Researchers are focusing on determining an efficient fraud and security threat detection model that overcomes the drawbacks of the existing methods. Therefore, ensemble learning can be applied to anomaly detection in Bitcoin by combining multiple ML classifiers. In the proposed model, the ADASYN-TL (Adaptive Synthetic + Tomek Link) balancing technique is used for data balancing. Random search, grid search and Bayesian optimization are used for hyperparameter tuning. The hyperparameters have a great impact on the performance of the model. For classification, we used the stacking model by combining Decision Tree, Naive Bayes, K-Nearest Neighbors, and Random Forest. We used SHapley Additive exPlanation (SHAP) to interpret the predictions of the stacking model. The model also explores the performance of different classifiers using accuracy, F1-score, Area Under Curve-Receiver Operating Characteristic (AUC-ROC), precision, recall, False Positive Rate (FPR) and execution time, and ultimately selects the ideal model. The proposed model contributes to the development of effective fraud detection models that address the limitations of the existing algorithms. Our stacking model, which combines the prediction of multiple classifiers, achieved the highest F1-score of 97%, precision of 96%, recall of 98%, accuracy of 97%, AUC-ROC of 99% and FPR of 3%.

INDEX TERMS Bitcoin Transaction, Hyperparameter Tuning, Machine Learning, Ransomware Attack, Stacking Model, Smart Cities

I. INTRODUCTION

A decentralized and a distributed ledger that records transactions safely and publicly is called a blockchain. Each block in the chain contains a series of transactions that have been confirmed and accepted by the network. Without network consensus, a block cannot be modified or deleted once it has been added to the chain [1]. In order to control or validate transactions, Bitcoin relies on a decentralized network rather than a centralized organization like a government or financial institution. This enables transactions that are safe, quick, and affordable without the use of middlemen like banks or payment processors. Despite these benefits, blockchain technology is not entirely secure and is still susceptible to some threats and vulnerabilities. Bitcoin has a reputation for being used for unlawful activities, due

to its anonymity and lack of regulation, which attracts criminals trying to elude the authorities. The following are some unlawful activities connected to Bitcoin and other cryptocurrencies taking place in the smart cities [2].

- Money laundering: criminals can move illegal funds undetectably across borders using Bitcoin.
- Dark web transactions: Bitcoin is used to pay for criminal operations including selling of guns or drugs on the dark web because of its anonymity.
- Payments for ransomware: hackers and online criminals utilize Bitcoin to pay for ransomware attacks, in which they demand money in return for access to the victim's computer or data.

Bitcoin users are susceptible to hacking, which could result in financial losses and credit issues for commercial websites. Blockchain technology stops illegal activity, however, it is still vulnerable to different attacks. Thus, different strategies and procedures are required to identify attacks [3].

Figure 1 shows how the Bitcoin ransomware attacks are performed. Firstly, the victim receives a malicious email, then the victim downloads the email attachments. As a result, all the files in the victim's computer get encrypted. Afterward, the victim receives a message that says to send a ransom amount in the form of Bitcoin. The victim sends a ransom amount in terms of Bitcoin and the transaction is stored in a wallet.

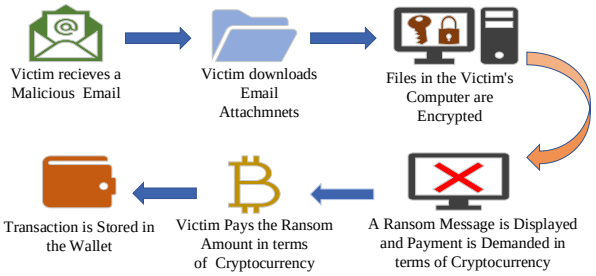


FIGURE 1: Anatomy of a Bitcoin Ransomware Attack

Previous detection methods depended on heuristic and signature-based approaches [4]. However, these approaches were insufficient to investigate the full complexity of anomaly detection, as shown in Figure 2. Therefore, we needed such an efficient fraud and security threat detection model that overcomes the drawbacks of the existing models. Researchers from all around the world have been drawn to Machine Learning (ML), because heuristic approaches are used for approximate solution while ML techniques are used for accurate solution [5]. ML techniques train to learn in the same way that people do with the aim of continuous improvement. ML techniques can be trained on large datasets of known malware samples to recognize patterns and attributes that set them apart from benign files. ML offers a potential method for anomaly detection. This can help to detect previously unseen variants of attacks and provide real-time protection against new attacks to comprehend the advantages of ML techniques. Using ML models, we can achieve the maximum accuracy [6]. For anomaly detection, a model comprising data collection, preparation, model development, validation, and deployment is required [7].

Both fraud detection and security threat detection in cryptocurrency transaction mechanisms require effective algorithms that can accurately identify fraudulent and malicious activities. However, the choice of algorithm depends on the amount of data in the dataset, and using a single algorithm may not provide sufficient accuracy. Therefore, the challenge is to develop such an effective fraud and security threat detection model that addresses the limitations of the existing algorithms. It can also handle datasets of varied sizes, and incorporates different ML techniques while exploring the influence of hyperparameters on the performance of the model.

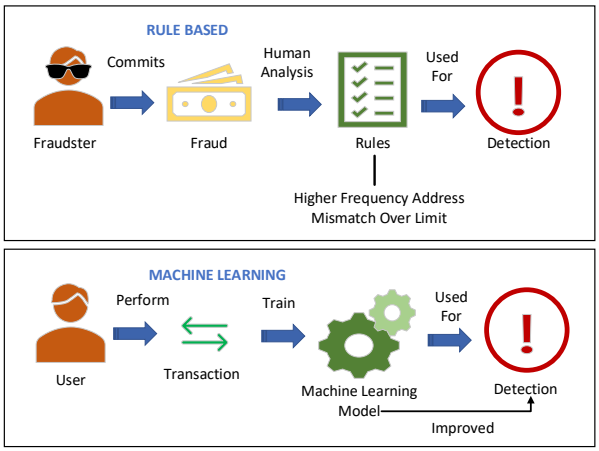


FIGURE 2: Rule-based vs. Machine Learning

A. MOTIVATION AND CONTRIBUTIONS

The motivation for this work comes from the fact that the existing methods were insufficient to explore the entire complexity of anomaly detection. ML can play an essential role in anomaly detection, as it can learn from historical data and detect new unseen attacks. Therefore, we need an efficient fraud and security threat detection model that overcomes the drawbacks of the existing methods. In this paper, we address the limitations of the existing techniques and present a stacking model by combining multiple ML techniques: Random Forest (RF), Decision Tree (DT), Naive Bayes (NB), and K-Nearest Neighbors (KNN).

The following are the main contributions made in the proposed work.

TABLE 1: List of Abbreviations

Notation	Description
DL	Deep Learning
GAT	Graph Attention neTwork
GAN	Generative Adversarial Networks
GRU	Gated Recurrent Unit
KNN	K Nearest Neighbors
LGBM	Light Gradient Boosting Model
LSTM	Long Short Term Memory
LR	Logistic Regression
ML	Machine Learning
MLP	Multilayer Perception
NB	Naive Bayes
NN	Neural Network
RF	Random Forest
SNN	Shallow Neural Network
SVM	Support Vector Machine
TDA	Topological Data Analysis
TPR	True Positive Rate
VPN	Virtual Private Network
XGBoost	eXtreme Gradient Boosting

- Data Balancing: On the Bitcoin Heist Ransomware dataset, hybrid balancing techniques are used to increase the model's accuracy.
- Hyperparameter Tuning: To determine the precise value of the classifier's parameters, random search, grid search and Bayesian optimization are used.
- SHAP is used to reveal how the stacking model makes predictions and to tell the significance of features.
- Classification: Proposed the stacking model using RF, DT, NB, and KNN for detecting anomalies in Bitcoin transactions.
- A comparison of the proposed model with ML techniques is performed using different balancing techniques and then choosing the ideal one in terms of performance.

II. RELATED WORK

This section provides a literature review of different papers.

Researchers have been focusing on determining an efficient fraud and security threat detection model that overcomes the drawbacks of existing methods. Light Gradient Boosting Model (LGBM) algorithm is proposed for detecting fraudulent transactions performed in the Ethereum blockchain [8], [9]. The authors first preprocessed the Ethereum transaction data and extracted important features. The authors then used these features to train the LGBM to detect and classify transactions as either legal or illegal. It turned out that LGBM achieved 98.06% classification accuracy.

In [10], the authors collected 19 variables from the Bitcoin network and then suggested a Graph-based Neural Network (GNN) model. The suggested model was compared with two cutting-edge methods for classifying illicit Bitcoin transactions, namely a Graph Attention neTwork (GAT) and eXtreme Gradient Boosting (XGBoost), trained on complex data. The ultimate goal was to improve the integrity of the cryptocurrency ecosystem and prevent it from being used for criminal activities.

Chen *et al.* in [11] proposed the use of supervised ML algorithms for detecting security threats in the Bitcoin blockchain network. Five different ML algorithms, Support Vector Machine (SVM), Multilayer Perceptron (MLP), Adaptive Boosting (Adaboost), RF, and KNN, were evaluated. The authors preprocessed the transaction data and extracted relevant features such as transaction amount, transaction fee, and transaction size. The results showed that KNN, RF, and Adaboost algorithms outperformed other algorithms in terms of accuracy.

In [12], the authors proposed a novel approach of fraud detection using a combination of ML and blockchain technologies. The paper aimed to address the limitations of traditional fraud detection mechanisms, which often rely on manual analysis and rule-based systems. The suggested ML algorithms predicted how the incoming transactions would behave. The simulation results demonstrated that the proposed model effectively detected transaction fraud.

The authors addressed the limitations of traditional approaches used for detecting fraud in the Bitcoin network in [13], which often rely on manual analysis and heuristics-based systems. The proposed approach used trimmed K-mean as a collective anomaly detection technique that leveraged the behavior of multiple entities in the network to identify suspicious activity using ML approaches.

In [14], the authors gave an extensive review of the approaches used to identify Bitcoin transactions associated with ransomware attacks. This study examined how different supervised ML techniques can be used to identify Bitcoin payments made to ransomware developers. According to the findings, RF with a k-fold cross-validation method may correctly recognize new attack categories.

The authors proposed a novel approach using ML for detecting malicious activities and adversarial behavior in permission-less blockchains in [15]. neural Network (NN) can acquire large recall value and detect adversarial feature vectors. ML models are inclined towards harmful activity with the greatest number of connected accounts. RF has the highest balanced accuracy of 96.5% among supervised ML models.

Singh *et al.* in [16] presented a comprehensive overview of the existing approaches used for fraud detection and analysis in blockchain systems. The authors discussed the different types of anomaly detection techniques that have been proposed for blockchain systems, including rule-based systems, ML-based systems, and graph-based systems. SVM was found to have the highest level of accuracy when employed to identify abnormalities

present in the transactions taking place in the Bitcoin network.

The authors described the architecture of the proposed system in [17] which included several modules such as data collection, Generative Adversarial Network (GAN) based data generation, and ML-based entity classification for improving Bitcoin entity classification by attacking the anonymity of the Bitcoin blockchain. GAN improved Bitcoin entity classification. It also protected user privacy, and ensured transparency and fairness.

A novel approach for detecting ransomware attacks on the Bitcoin blockchain using Topological Data Analysis (TDA) was proposed by authors in [18]. The proposed approach used TDA to analyze the topological structure of the Bitcoin blockchain and identify patterns that are indicative of ransomware attacks.

Talabani *et al.* in [19] outlined the design and implementation of a rule-based detection system that could analyze network traffic, file metadata, and other indicators to identify known ransomware strains and prevent their execution. The proposed approach leveraged existing knowledge about ransomware behavior and Bitcoin payment transactions to develop rules and conditions that could be used to identify and block ransomware attacks.

The authors proposed a DL-based approach to detect Nontechnical Losses (NTL) in smart meters in [20]. The proposed approach leverages a DL model consisting of an MLP and a Gated Recurrent Unit (GRU) to analyze smart meter data and identify anomalies that are indicative of NTLs. The proposed model gave ROC-AUC and PR-AUC values of 93% and 96%, respectively.

The potential of ML techniques such as clustering, and classification to identify fraudulent transactions in the blockchain were explored in [21]. The authors suggested that these techniques could be used to analyze the large and complex datasets that were inherent in blockchain systems, and did not provide accurate and real-time fraud detection. The findings demonstrated that the system could identify fraudulent activity with 97% accuracy.

The authors proposed a high-performance Bitcoin transaction prediction system in [22] for heterogeneous Bitcoin networks that examined Bitcoin payment transactions to find and classify ransomware payments. The two supervised ML techniques used in this study are Shallow Neural Networks (SNN) and DT. The ML-based predictive models outperformed cutting-edge models.

To recognize the distinctive characteristics of Bitcoin payment transaction patterns, the strategy proposed in [23] used three supervised ML techniques, Logistic Regression (LR), RF, and XGBoost. It turned out that the XGBoost model performed better than the existing techniques.

In [24], the proposed approach, called Spline Interpolation envisioned NN based Ransomware Detection (SINN-RD), effectively detected the ransomware attacks by analyzing the entropy of the encrypted data and using spline interpolation techniques to extract relevant features from the data. The authors proposed a NN-based classification model that used these features to classify data as either ransomware or non-ransomware. Moreover, it was shown that the proposed model performed better than the other existing techniques.

Li *et al.* in [25] aimed to develop an efficient risk assessment technique for households investing in cryptocurrencies. The authors proposed an explainable ML approach that combined different ML algorithms with transparent and interpretable decision-making. The LR algorithm outperformed other ML algorithms.

In [26], the proposed method used a multiperspective approach that considered various perspectives of the cryptocurrency transaction, including the transaction amount, payment method, shipping method, and user behavior. The authors used ML algorithms, including DT and SVM, to identify potentially fraudulent transactions based on the features extracted from the various transaction perspectives. The authors proposed a supervised learning approach in [27] that used various ML algorithms, including LR, DT, and SVM, to classify transactional addresses based on their transactional behavior. The authors used various features extracted from the transactional behavior, including the number of transactions, the transaction frequency, and the transaction value, to train the ML models. Al Badawi *et al.* in [28] proposed a framework for identifying money laundering in Bitcoin transactions. The proposed framework presented a promising approach for detecting potential money laundering activities in Bitcoin transactions using ML techniques, which could have important implications for improving security and transparency of the Bitcoin network and preventing illegal activities. The suggested framework achieved 95% accuracy. The authors in [29] discussed difficulties associated with anomaly detection in blockchain networks. Different ensemble strategies, including bagging, boosting, stacking, and a combination of experts, were thoroughly explained. The authors also discussed different types of classifiers that could be used for anomaly detection, such as DT, SVM, and NN. In [30], the authors highlighted the need for effective methods for detecting and preventing illegal activities to ensure the integrity and stability of the financial system. The authors suggested that an ensemble DT-based model might be trained using nine distinct features. The classification accuracy was 91%, which was comparable to the RF model.

The authors aimed to shed light on the growing threat of fraudulent transactions involving cryptocurrencies, including money laundering, terrorist financing, and other illicit activities in [31]. However, the authors noted that these measures alone might not be sufficient. Greater

TABLE 2: Literature Review Summary

Limitations	Methodology/Contributions	Evaluation Metrics
Inadequacy of existing ML models for analyzing Bitcoin transaction graphs	Used a Graph-based NN model [10]	Accuracy, Precision and Recall
The existing models are not able to provide accurate result Data imbalance problem	Used RF and XGBoost Generated synthetic malicious data points through SMOTE [12]	Precision, Accuracy and F1-score
The existing techniques for detecting frauds on the Bitcoin network have drawbacks (i.e., misclassification)	Proposed a collective anomaly detection technique [13]	F1-score, Accuracy, Precision, and Recall
Existing techniques are not ideal for fraud detection	Used RF using K fold method [14]	Accuracy, F1-score
Scalability issue in existing techniques	Used K mean clustering and NN [15]	F1-score and Accuracy
Rule-based techniques do not detect new and unseen fraud	Used KNN, SVM and Isolation Forest [16]	Accuracy, Precision and Recall
Data imbalance issue and ML techniques are not a good choice for big data as it leads to overfitting	Used Random Oversampling Used GAN to improve classification accuracy [17]	Accuracy, Precision, Recall, F1-score and AUC
Existing techniques do not employ deep data analysis	TDA is used [18]	Accuracy, Precision and Recall
Data imbalance issue and existing techniques do not detect fraud accurately	SMOTE and GRU are used [20]	ROC-AUC and PR-AUC
Existing techniques are not good choice for big data as they lead to overfitting	Used SMOTE and ML techniques (DT, RF, KNN and SVM) [21]	Accuracy, Precision, Recall and F1-score
Existing techniques are bit challenging and complex for handling big data	Used SNN and optimizable DT [22]	Accuracy, Precision, Recall, AUC, and F1-score
Data imbalance problem	Used random undersampling and co-relation for feature extraction [23]	Accuracy and F1-score
Poor performance of existing techniques on big data	Proposed SINN-RD [24]	Accuracy, Recall and Precision
Existing techniques do not tackle Blackbox issue	Used LR, KNN, RF, and NN [25]	ROC, F1score, Specificity and Recall
Existing techniques are not efficient for handling big data	Used SVM [26]	Precision, Recall, F1-score and AUC
Binary classification issue in cryptocurrency	Used ML techniques (LR, KNN, NB, RF and XGBoost) [27]	Accuracy, F1-score, Precision and Recall
Limited accuracy and performance of individual classifiers	Used SNN and optimizeable DT [29]	Precision, Recall, F1-score, and AUC.
Existing research do not concentrate on fraud detection	Used RF and XGBoost [30]	Accuracy, F1-score and AUC-ROC curve
Issue in Virtual Private Network (VPN) tunneling	Used SVM, KNN, NB and RF [32]	MSE, TP and FP
Existing techniques are not efficient in terms of big data	Used LR, DT, RF and NN [33]	Error rate
Unsupervised techniques are not efficient for anomaly detection because they learn without previous knowledge	Used DT, LR and Gradient Boosting (GB) [34]	Sensitivity, Specificity and Accuracy
The existing techniques do not focus on account-level detection	Used XGBoost for fraud detection [35]	TP, FP, TN and FN
Data imbalance issue Existing techniques are less efficient as it do not suitable for big data	SMOTE is used to balance the dataset Used GB, KNN and RF [36]	F1-score, Accuracy and AUC score
Existing techniques are not good choice for anomaly detection as they do not handle big data	Used XGBoost model with blockchain [37]	Accuracy
All existing techniques are biased in relation to the available ground truth	Used deep NN for fraud detection [38]	Inter event Time, In-Degree and Out-Degree

awareness and education among the public were needed to prevent fraudulent transactions involving cryptocurrencies.

Caprolu *et al.* in [32] proposed an ML-based approach for detecting crypto-jacking attacks, which are a type of cyber-attack where a victim's computer or mobile device is taken over by an attacker to mine Bitcoin without the victim's knowledge or consent. The proposed solution achieved a stunning F1-score of 96% and AUC of 99%. In [33], ML methods like RF, DT, and KNN were used to categorize ransomware samples into several families. The RF algorithm performed the best among the DT and KNN algorithms, achieving an accuracy of 98.6%. The study also compared the performance of the ML algorithms with traditional signature-based detection techniques and showed that ML algorithms outperformed the traditional techniques.

Nerurkar *et al.* in [34] proposed an ML model to recognize illegal actors operating within the Bitcoin network. They used an ensemble of DT as their classification model and trained it on subsets of the data to improve accuracy and prevent overfitting. The model gave a precision of 92% and a recall of 85% for correctly identifying illegal organizations.

The authors in [35] proposed a model for the Ethereum network in which the XGBoost classifier managed to obtain an accuracy of 96% and an average AUC of 99.4%. The findings indicated that the proposed method was quite successful in locating fraudulent accounts on the Ethereum network. A comparative analysis of different supervised ML algorithms for money laundering in Bitcoin was performed in [36]. The proposed methodology helped financial institutions identify and prevent money laundering activities in the Bitcoin network. Thus, improving the security and trust of the system. The study found that ensemble classifiers were effective in detecting and preventing money laundering activities in Bitcoin. Ayushi Maurya *et al.* in [37] suggested a ML-based method to identify fraudulent credit card transactions. The outcomes demonstrated that the system had a high level of success in identifying fraudulent transactions. The authors found that XGBoost obtained the highest classification accuracy.

In [38], the authors proposed a method for identifying fraudulent addresses in Bitcoin using ML techniques. The suggested model was designed for detecting and preventing fraud. The classifier was able to accurately identify malicious addresses with a precision of 96% and recall of 87%. Due to the availability of label data, supervised learning methods have been found to be the most effective, however, a single algorithm is insufficient to produce an accurate result because a single ML model may overfit the training data and not generalize well to new or unseen data, leading to poor anomaly detection performance. Table 2 provides a summary of the related work.

III. PROBLEM STATEMENT

To accurately identify fraudulent and illegal transactions in cryptocurrency, an effective fraud detection model is needed. The choice of algorithm depends on the dataset size, and using just one technique may not yield reliable results. The authors have proposed various algorithms for detecting security threats on the Bitcoin blockchain, including the LGBM algorithm for Ethereum, the Adaboost algorithm for Bitcoin, and the supervised ML algorithms for Bitcoin ransomware attacks in [8], [11]. However, a single algorithm may not provide accurate results and may not generalize well to new or unseen data. Additionally, the authors in [19], [28] developed an efficient system for detecting money laundering in cryptocurrency transactions using ML techniques like SNN and DT, but their classification accuracy decreases as input features increase and are prone to overfitting.

IV. PROPOSED SYSTEM MODEL

In this section, we discuss dataset, data balancing techniques, model optimization through hyperparameter tuning, and the construction of a stacking model.

A. DATASET

We used the Bitcoin Heist Ransomware dataset for the detection of ransomware transactions. The original multiclass dataset is converted into a binary class dataset by assigning label 0 to a normal transaction and 1 to a fraudulent transaction. The dataset is publicly available on the Kaggle and UCI websites [39], [40]. It has 2916697 instances and 10 features. To deal with such a big dataset, a large number of computational resources are needed. As we do not have enough resources, so we filter out records that transfer amounts in Bitcoin less than 0.3 threshold, and the dataset is left with 1387721 instances. However, the dataset is still big. So, we extracted the data of 3 years (2016-2018) from the dataset and are left with 381464 instances. The dataset is then split into an 80:20 ratio. Moreover, to remove outliers, the Z-score technique is used. Data preprocessing is performed on the dataset, as the dataset is highly imbalanced.

B. DATA BALANCING TECHNIQUE

Data balancing is the process of reducing imbalance ratio between classes. Data imbalance occurs when one class has more instances than the other class. This problem can lead to the poor performance of the model [41]. Several methods are used to balance the classes but one thing to keep in mind is that balancing should be done very carefully, as data balancing can lead to loss of information or data redundancy. The performance of classifiers is evaluated on the balanced data to ensure that they perform well. We used SMOTE-ENN [42] and ADASYN-TL [43] which are hybrid balancing techniques. Synthetic Minority Oversampling

Technique (SMOTE) and ADASYN are used for oversampling, TL and Edited Nearest Neighbors (ENN) are used for undersampling. ADASYN-TL is a good choice for reducing the biasness by creating synthetic samples. It solves the overfitting issue by removing the duplicate samples. However, it is important to note that ADASYN-TL is a hybrid technique and may require extra computational resources. The mathematical representation and algorithm of ADASYN is given below [44].

In Equation 1, λ represents the random number between 0 and 1. X_{zi}, X_i represent the two minority samples within the same neighborhood and s_i represents the synthetic samples.

$$s_i = X_i + (X_{zi} - X_i)\lambda \tag{1}$$

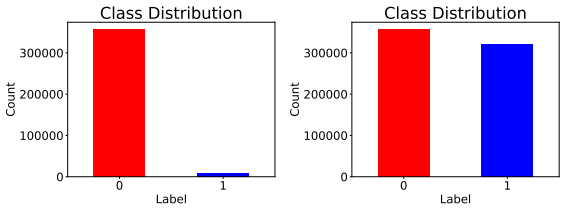
Mathematically, TL can be expressed as given in Equation 2.

Let $d(z_i, z_j)$ is the Euclidean distance between z_i and z_j, where z_i represents the minority samples and z_j represents the majority samples. If there is no sample z_k, then the following conditions must be fulfilled.

$$d(z_i, z_k) < d(z_i, z_j) \tag{2}$$
$$d(z_j, z_k) < d(z_i, z_j) \tag{3}$$

Then, the pair of (z_i, z_j) is a TL pair.

Figure 3(a) is the visualization of imbalanced data and Figure 3(b) is the representation of ADASYN-TL balanced data.



(a) Imbalance Classes (b) Balanced Classes

FIGURE 3: Data Representation of Before and After Balancing

In Algorithm 1, m_s and m_i represent the majority and minority sample of data. Whereas, K represents the number of the nearest neighbors, and C represents the synthetic samples to generate. This algorithm balances the dataset by iterating over each minority sample and finding its k nearest neighbors using the KNN. For each synthetic sample, the algorithm uses ADASYN to generate a new sample. After generating all the synthetic samples, the algorithm uses TL to remove overlapping samples, improving the performance of the classifier.

Algorithm 1 Adaptive Synthetic-Tomek Link

Require: Imbalanced dataset
m_i majority samples
m_s minority samples
C synthetic samples
K k nearest neighbors
Ensure: Balanced dataset
1: **Begin**
2: Determine the minority sample to majority sample ratio.
d= m_s/m_i
3: Determine the total number of synthetic minority samples that will be produced.
C=$(m_i - m_s)\beta$
4: For each minority sample, determine the K nearest neighbours.
5: Determine the number of synthetic samples needed to generate the neighbors.
6: Generate data for each neighborhood
7: Start Tomek Link, choose random data from the majority class
8: Find the nearest neighbors, keeping K = 3
9: If the data from the minority class is the random data's nearest neighbor, then build the Tomek Link, otherwise, remove the Tomek Link
10: **End**

Algorithm 2 Synthetic Minority Oversampling Technique -Edited Nearest Neighbors

Require: Imbalanced Dataset
Training Data X_{train}
K number of nearest neighbors for removing samples
n number of nearest neighbors for smote
Ensure: Balanced Dataset
1: **Begin**
2: Divide X_{train} into P and N $X_{train} = P \cup N$
3: Find the K nearest neighbors
4: Oversample the minority class
$New_P \leftarrow SMOTE(P, n)$
and $|New_P| = |N|$;
5: $New_{X_{train}} = New_P \cup N$
6: for x $\in New_{X_{train}}$ do
7: Compute distance according to K
8: end for
9: Remove samples using ENN on the basis of distance
$New_{X_{train}} \leftarrow ENN(New_{X_{train}}, D, K)$
10: **End**

The combined capabilities of SMOTE and ENN are presented by the Algorithm 2. In the minority class, synthetic samples are generated using SMOTE, and the distinct data points from both classes are removed using

ENN. In Algorithm 2, K represents nearest neighbors and D represents the distance. P are the positive samples, N are the negative samples and New_P is the new positive sample. X_{train} is the training set and n number of nearest neighbors for smote.

C. HYPERPARAMETER TUNING

To improve the performance of the model, hyperparameter tuning is performed to choose the best value for the classifier hyperparameters. Hyperparameters reduce the overall number of iterations, which can be helpful in increasing the efficiency of the model. There are several methods for hyperparameter tuning.

1) Random Search

The method we used is random search. It is a good choice for big data as it randomly selects the hyperparameters. Whereas, grid search searches through all the possible combinations and increases the computational cost. The mathematical representation of random search is given in Equation 4 [45].
In Equation 4, x represents vector, S is an n-dimensional feasible region and f is a real-valued function defined over S. The objective is to locate an x value in S that minimizes f. x' and y' represent the global optimal solutions.

$$x' = argmin_{x \in S} f(x) \tag{4}$$

$$y' = f(x') = min_{x \in S f(x)}. \tag{5}$$

The basic random search algorithm is written as follows [46]:

Algorithm 3 Random Search

Require: x is a random position
y is the new position of sample

1: **Begin**
2: Let x be a random position in the search-space.
3: Number of iterations performed to find different random position, until a termination criterion is met
4: The hypersphere with a specific radius surrounding the current point x is sampled to produce a new position y
5: if $(f(y) < f(x))$
6: Move to the new position by setting $x = y$
7: **End**

In Algorithm 3, the current best solution is initialized by selecting a random solution from the search-space. It then enters a loop that generates random solutions and evaluates the objective function for each. If any of the solutions has a better objective function value, it updates the current best solution. The *if* statement determines weather to move x to the new position y or not.
The range of values for the hyperparameters used by the ML classifiers is provided in Table 3. With the random search method, the hyperparameters of all classifiers are adjusted.

2) Bayesian Optimization

Bayesian optimization is a frequently used ML technique for hyperparameter optimization. The Sequential Model-Based Optimization (SMBO) algorithm known as Bayesian optimization enhances experimental sampling techniques. In order to select the best hyperparameter values, the model computes a posterior expectation of the hyperparameter space. After that, it repeatedly iterates till convergence. A Gaussian process approximates a continuous score function and identifies candidate hyperparameter values with the biggest expected improvement by modeling the prior probability of model scores across the hyperparameter space. The Bayes' theorem is used to calculate the conditional probability. The working flow of Bayesian optimization is given in Algorithm 4. a are the samples from data S, b_{n+1} is the objective function and S_{n+1} is the augmented data.

Algorithm 4 Bayesian Optimization

Require: Maximize unknown function $f(x)$ on data S
S is the complete dataset $f(x)$ is the acquisition function

1: **Begin**
2: for n loop do
3: Pick a sample a_{n+1}.
4: Optimize given S by using acquisition function $a_{n+1} = max\alpha(a; Dn)$
5: Obtain new observation b_{n+1}.
6: $S_{n+1} = \{S_n(a_{n+1}, b_{n+1})\}$ augment data
7: Update Model
8: **End**

3) Grid Search

Grid search is used for hyperparameter tuning. It divides the hyperparameter domain into discrete grid and tries all possible combinations of every value of the grid. The grid's ideal set of values for the hyperparameters is the point where cross-validation's average value is maximized. It is a comprehensive technique that checks all the combinations to find the best point from the domain. Its significant flaw is that it moves very slowly. Checking every possible configuration of the space would take a lot of time, which is occasionally not available. Remember that k training steps are needed for k-fold cross-validation at each location in the grid. Therefore, fine-tuning a model's hyperparameters in this manner can be both time-consuming and costly. However, grid search is a really smart notion if we're looking for the ideal mix of hyperparameter values. The mathematical

TABLE 3: The Range of Hyperparameters of all Utilized Machine Learning Techniques.

Classifiers	Hyperparameters	Range	Selected Values
Random Forest	max_depth	[1, 10]	10
	min_samples_split	randint(2, 20)	18
	min_samples_leaf	randint(1, 10)	5
	n_estimators	25, 50, 100, 150	25
	max_features	[sqrt, log2, None]	log2
Decision Tree	max_depth	[1, 10]	10
	min_samples_split	randint(2, 20)	18
	min_samples_leaf	randint(1, 10)	5
	max_features	[sqrt, log2, None]	log2
K-Nearest Neighbors	n_neighbor	randint(1, 50)	15
	weights	[uniform, distance]	distance
	p	[1, 2]	1
Naive Bayes	var_smoothing	[1e-9, 1e-6, 1e-12]	1e-12
Stacking Model	n_neighbor	randint(1, 50)	15
	min_samples_split	randint(2, 20)	18
	max_depth	1, 10	10
	n_estimators	[25, 50, 100, 150]	25
	min_samples_leaf	randint(1, 10)	5

expression for grid serach is given in Equation 6.

$$argmin_{a,b \in S} f(a, b) \tag{6}$$

In Equation 6, a and b represents parameters [0,1], S represents the dataset and f is the function.

D. SHAPLEY ADDITIVE EXPLANATION

SHAP is used to provide the information about how the model makes predictions. SHAP breaks down the model's output into the sums of each feature's inputs. Each feature's contribution to the model's result is calculated by SHAP. These values can be utilized to describe the outcome of the model to a person and to comprehend the significance of each aspect. This is particularly helpful for organizations and teams who answer the clients or management.

The remarkable characteristics of SHAP are its fairness towards models. This enables it to manage complex model behaviors (such as when features interact with one another), witnessed when dealing with a learning model or generating consistent explanations.

E. STACKING MODEL

An ensemble learning technique called stacking involves merging the results of various base models to get a more reliable and accurate final prediction. The base layer and the meta layer are the two layers of the stacking model. The stacking model's first layer consists of multiple base learners, each of which generates its own predictions based on the input features. The outputs of these models is then used as input for the second layer of the model, which is a meta-model that combines the predictions of the base models. Therefore, there are some things that should be kept in mind regarding the stacking.

1) The stacking model may take more time to train the data. It is costly in terms of computation.
2) The stacking model needs a large dataset to train the base layer or meta layer, so that model cannot overfit.

Besides all these limitations, the stacking model can be a good choice to improve accuracy and reduce the overfitting issue by combining multiple classifiers [47]. In Algorithm 5, B represents the base model, R represents the prediction of base model and P represents the final prediction [48]. The first step is to split the dataset into training and testing sets. After splitting, the base models are trained on the dataset. The predictions of the base models are given to the meta model as an input. Finally, the model is trained on the entire dataset without any test fold and uses it to make predictions on new unseen data points.

Algorithm 5 Stacking Model

Require: Dataset split into Training X_{train} and Testing y_{train} sets

1: **Begin**
2: B[i] $\leftarrow$ fitted with Training set Xi_{train} where i represents the number of base learners
3: R is added into Xi_{train}
4: Repeat steps 2 and 3, so it will give an array of X_{train}
5: $B[i]$.fit (X_{train}, y_{train})
6: Now the base models are trained on the dataset
7: The meta-level model is trained on the output of the base models
8: Prediction P is made using the final output
9: **End**

The ensemble technique creates a stacking model by incorporating different classifiers. These single classifiers are termed as weak learners and their ensemble technique is termed as a strong learner. The ensemble models are designed to improve the accuracy [29]. The

ensemble method is used to reduce the bias and variance of the model by combining multiple classifiers for better performance. However, it has some limitations such as, it is costly in terms of computations, as shown in Figure 4.

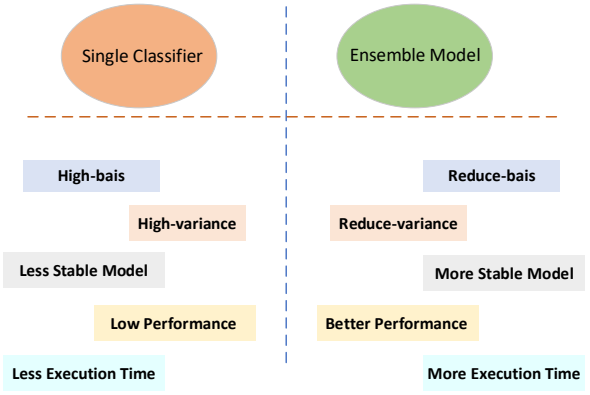


FIGURE 4: Single Learner vs. Ensemble Model

F. CONSTRUCTION OF A NEW ENSEMBLE BITCOIN DETECTOR

Fraud detection in Bitcoin transactions is a crucial task in maintaining the integrity of the cryptocurrency market. So, researchers focused on developing a robust model for fraud detection. The Ensemble Bitcoin Detector (EBD) model is a great contribution to anomaly detection. The base layer of the EBD model consists of three base models. Each model generates its own predictions based on the input features. The base learners that we used in the stacking model are DT, NB, and KNN. While RF is used as a meta-model. The detector model is trained using a labeled Bitcoin transaction dataset, where the labels indicate whether a transaction is fraudulent or not. The features used for training the model include various characteristics of the transactions, such as the transaction amount, sender and receiver addresses, and transaction time.

Algorithm 6 discusses the proposed model where M represents the best model, R represents the base model prediction and $Pred$ represents the final prediction.

The algorithm discusses the steps involved in the proposed model: acquiring the dataset, preprocessing, hyperparameter tuning and classification via the stacking model. The dataset is imported and filtered, and each feature is standardized by subtracting its mean and dividing by its standard deviation. The dataset is split into training and testing sets, and hyperparameters of ML models (RF, DT, NB, and KNN) are tuned using a randomized search approach. The best hyperparameters for each model are used to train the model on the training set. Afterwards, the models are stacked together to create a level-1 model. Finally, the level-1 model is trained on the entire training set and is used to make predictions on new unseen data points.

Algorithm 6 EBD: Proposed Ensemble Bitcoin Detector Model Framework

Require: Import Dataset

Begin
if (income > 0.3 B)
    Keep record that transfers amount more than 0.3
Else
    Delete record that transfers amount less than 0.3
End if
for each observed year = 2016
    Keep record, year + 1
Else
    Otherwise delete record
end for
If ($label == white$)
    Covert white label into 0
Else
    Convert ransom attack record into 1
end if
Compute the mean (mu) and standard deviation (sigma) of each feature in X: mu_j = mean($X[:, j]$) $sigma_j$ = std($X[:, j]$)
Standardize each feature by subtracting its mean and dividing by its standard deviation:
    $X[:, j] = (X[:, j]$ - $mu_j)$ / $sigma_j$
Split dataset into training and testing sets
$ADASYN - TL$.fit(X_{train}, y_{train})
Hyperparameter Tuning
M(RF, DT, NB, and KNN)
for i in range(len(M))
    RandomSearch($M[i]$)
Stacking Model
    $M[i]$.fit(X_{train}, y_{train})
    R is added into $X1_{train}$
    Repeat steps 2 and 3, so it will give array of X_{train}
    The model is trained
    Train the M on level 1
Make prediction $Pred$
End

Figure 5 describes the techniques that we used in the proposed model. We used four different supervised learning techniques RF, DT, NB, and KNN (RDNK). The flowchart of EBD is a pictorial representation of the proposed model's algorithm, as shown in Figure 6. First of all, the dataset is loaded and is checked for the outliers. If outliers are found, they are removed. After that, the dataset is balanced using the balanc-

ing technique and is splitted into two sets: training and testing. Hyperparameter tuning is performed before classification. In the end, the model is evaluated using different performance indicators.

Figure 7 shows the internal working of the proposed model. This figure shows how the stacking model works. Each classifier is trained on the dataset and makes prediction. The predictions of the base classifiers are added to the dataset and given to the meta-classifier as input. The meta-classifier makes the final prediction and provides accurate results. The framework of the EBD is described in Figure 8. A comparison is performed between different balancing techniques and the performance is validated at step 1. In step 2, we compared different classifiers on the same data and evaluated their performance using different performance metrics. At the last step, we proposed a model trained on the same dataset and different balancing techniques and classifiers, and evaluated using different performance metrics. The proposed model is compared with their baseline classifiers in terms of accuracy, precision, recall, F1-score, and AUC-ROC.

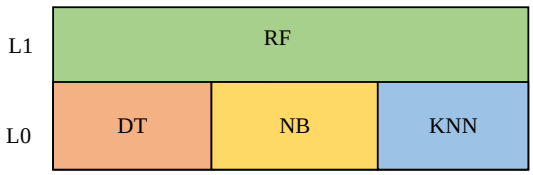


FIGURE 5: RDNK: Proposed Stacking Model

1) K-Nearest Neighbors

KNN is an ML approach that is used both for classification and regression problems. It is a non-parametric and supervised apporach. It makes predictions by locating the k-nearest data points and figuring out which group or average value these neighbors belong to [49], [50]. The mathematical representation and algorithm of KNN are given as follows.

In Equation 7, X and Y represent the number of features and data points, respectively. The similarity between two data points is determined using this equation.

$$d(\mathbf{X}, \mathbf{Y}) = \sqrt{(X_1 - Y_1)^2 + \cdots + (X_n - Y_n)^2} \quad (7)$$

In Algorithm 7, the number of nearest neighbors is represented by K. X_i and X_j represent training and testing parts. Whereas, the similarity between data points is calculated using distance formula. KNN algorithm takes three inputs: the training dataset X_i, the testing dataset X_j, and K for considering number of neighbors. For each data point X_j in the testing set, the algorithm calculates the distance between X_j and each data point in the training set using a distance metric.

Algorithm 7 K-Nearest Neighbor

Require: Training dataset X_i, Testing dataset X_j, Number of neighbors to consider K
Ensure: Predicted classes or values for each data point in X_j
1: **Begin**
2: for $j = 1$ to n do
3: Determine distance D(X_j , x)
4: end for
5: Indices for the K smallest distances $D(Xi, x)$ are contained in the computed set.
6: for X_i where $i \in 1$
7: **Return** majority label
8: **End**

2) Decision Tree

DT is a supervised ML technique, which is used for classification. It divides the training data into subsets depending on features and then constructs a model that resembles a tree of decisions. The mathematical representation and algorithm of DT are given below [51].

In Equation 8, S represents the class of data and p represents the data point. Entropy is used to find the gain. The feature with the highest gain is placed as a root node.

$$\text{Entropy formula:-}H(S) = -\sum_{i=1}^{c} p_i log_2(p_i) \quad (8)$$

In Equation 9, T represents the target and X represents the feature that is to be split.

$$Gain(T, X) = Entropy(T) - Entropy(T, X) \quad (9)$$

In Algorithm 8, x represents the datapoint and S represents the dataset. The algorithm traverses the DT recursively based on splitting rules until it reaches a leaf node, where it returns the predicted label for the input data point x.

Algorithm 8 Decision Tree

Require: Take original dataset S
1: **Begin**
2: Predict($tree, S$)
If $tree$ is a leaf node
Return the prediction of $tree$
3: for each $n \in S$
4: Choose the attribute with the lowest entropy and the highest gain
5: Consider the attribute with the highest gain as a root node A
6: while A not leaf node do
7: Get output on A using n
8: Identify from n the correct output
A = node at the end
9: Make prediction on n on the based of labeling of A
10: **End**

3) Naive Bayes

NB is a probabilistic classification technique that determines the probability of a given class using Bayes' theorem. The Bayes' theorem and its algorithm are provided below [52].

In Equation 10, A and B are events. The probability of event B depends upon the probability that event A has happened in the past.

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)} \tag{10}$$

In Algorithm 9, x represents the new sample, y represents the class label of new sample while n represents the training set and P represents the probability. $N(x_j, y_i) \leftarrow$ count of training samples with class label y_i and feature value x_j.

Algorithm 9 Naive Bayes

Require: A training dataset of labeled examples $\{(x_1, y_1)$,..., $(x_n, y_n)\}$ and a new example to classify, x

Ensure: A predicted class label for the new example, y

1: **Begin**
2: Calculate the prior probabilities for each class For each unique class label y_i in the training set
$P(y_i) \leftarrow \frac{\text{count of examples with class label } y_i}{\text{total number of training examples}}$
3: Calculate the likelihood of each feature given each class
4: For each unique feature value x_j and class label y_i in the training set
$P(x_j|y_i) \leftarrow \frac{N(x_j, y_i)}{\text{count of training examples with class label } y_i}$
5: Use Bayes' theorem to calculate the posterior probabilities for each class
6: For each unique class label y_i in the training set
$P(x|y_i) \leftarrow \prod_{x_j \in x} P(x_j|y_i)$
$P(y_i|x) \leftarrow \frac{P(y_i) \cdot P(x|y_i)}{P(x)}$
7: Choose the class with the highest posterior probability as the predicted class label for the new example
8: $y \leftarrow \text{argmax}_{y_i} P(y_i|x)$ for each unique class label y_i in the training set
9: **Output** y
10: **End**

4) Random Forest

RF is an ML supervised technique which is built by combining hundreds of DTs. All the DTs work in a parallel manner. RF is a bagging ensemble method. The mathematical representation and algorithm of the RF are given below [53], [54].

In Equation 11, i represents the number of important features calculated for all trees j. While T represents the total number of trees.

$$RFfi_i = \sum_J normfi_{ij}/sumT \tag{11}$$

Algorithm 10 Random Forest

Require: X_{train}: Training set with n instances
F: number of features
A: number of classes in a target class B: number of tress

1: **Begin**
2: for $i = 1$ to B do
3: Generate the bootstrap samples $X_{train}[i]$ from the training set X_{train}
4: Using random sample from $X_{train}[i]$ and create a tree
5: For a selected node t
- Randomly select $m \approx \sqrt{F}$
- Find the best point from the subset
- Pass down the data using the best points
Repeat these step until the termination condition are met

6: Construct the trained classifier
7: **End**

In Algorithm 10, S represents the training set, F is the result of final prediction and B represents the number of DT.

G. CRITICAL ANALYSIS

In this study, we use different classifiers like AdaBoost, XGBoost, and LR with different balancing techniques and evaluate the performance using different performance indicators, e.g., accuracy, F1-score, precision, recall, etc. These selected classifiers show excellent performance. Moreover, we use RF because it reduces the overfitting issue. NB is used because it supposes that all features are independent, KNN handles the non-linearity in the dataset and DT is the rule-based algorithm, which makes decision based on the specific rules.

H. EVALUATION METRICS

The validation of the model is the primary concern in a fraud detection system. The model is validated in this research using a variety of performance parameters, including accuracy, precision, recall, F1 score, execution time, and AUC-ROC curve. The calculations for each of these measures use False Positive (FP), False Negative (FN), True Negative (TN), and True Positive (TP) [55].

- FP: normal transaction is classified as fraudulent.
- FN: fraudulent transaction is classified as normal.
- TP: fraudulent transaction is classified as fraudulent.
- TN: normal transaction is classified as normal.

Accuracy: it is used to measure the percentage of correctly classified instances out of all instances.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

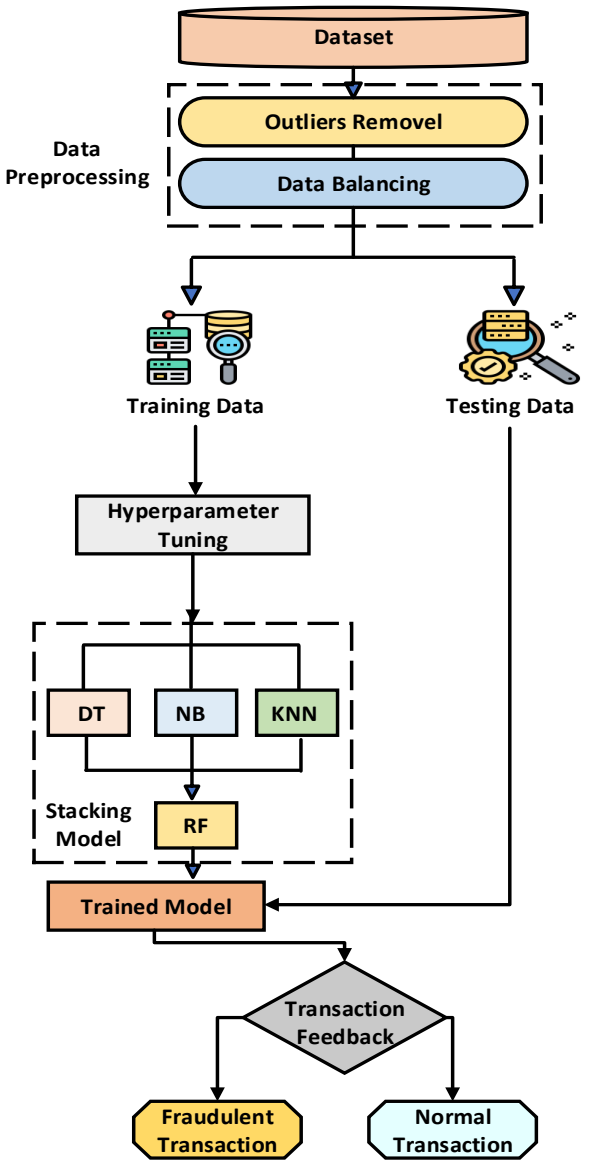


FIGURE 6: Working Flow of Ensemble Bitcoin Detector.

Recall: it is used to measure TP out of all positive instances from the dataset.

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

Precision: it is used to measure TP from all the positive predicted instances.

$$Precision = \frac{TP}{TP + FP} \tag{14}$$

F1-score: it is the harmonic mean of precision and recall.

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{15}$$

AUC-ROC curve: it is used for the evaluation of the model's performance. In Equation 16, t represents the decision rate, FPR is the False Positive Rate and TPR is the True Positive Rate.

$$AUC_{ROC} = \frac{1}{2} \int_{-\infty}^{\infty} (TPR(FPR^{-1}(t)))dt \tag{16}$$

V. RESULTS AND DISCUSSION

The results that are provided in this section are obtained by taking an average of every simulation.

A. EVALUATION FINDINGS

The results demonstrate that the ADASYN-TL with the stacking model performs well among all individual classifiers in terms of accuracy and F1-score. The results show that using the stacking model it is possible to improve performance by incorporating the predictions of ML classifiers. Although the stacking model requires a lot of processing, it might be helpful for handling complex datasets. However, the stacking model can be costly in terms of computation. The stacking model defeats all the benchmark classifiers in terms of accuracy, which is 97% and F1-score, which is 97%. The performance of KNN classifier is the best. Because, it works effectively for datasets with non-linear decision boundaries since it makes no assumptions about the distribution of the data. RF and DT did relatively well in terms of accuracy, precision, recall and F1-score than NB. However, the time required by RF for execution is significantly longer than it is for other classifiers. Moreover, a small change in the data may affect the performance of DT because it is a rule-based algorithm. NB poorly performs because it supposes that all features are independent of one another. It is not an ideal classifier for the real detection scenario. However, by combining these classifiers in the stacking mode, a powerful model is built.

Figure 9 and Table 4 validate that the suggested model achieve the maximum accuracy of 97%, F1-score of 97%, precision of 96%, recall of 98%, AUC-ROC of 99% and FPR of 3%.

Figure 10 illustrates that the stacking model needs more time for training. The RF is also costly in terms of time complexity. However, the selection of the classifiers depends upon priority. If reduction in the training time is the priority, time-efficient classifiers are selected. If the increase in accuracy is the priority, classifiers that give high accuracy are selected.

In Figure 11, the AUC-ROC curves show that the proposed model has high accuracy in identifying the TP while minimizing the FP rate. AUC-ROC value of 99% indicates that the proposed model performs well. The performance of NB is not promising in comparison with other techniques. Because if two features contain the same data, NB double counts their effects and reaches a wrong assumption. In Figure 12, the confusion matrix

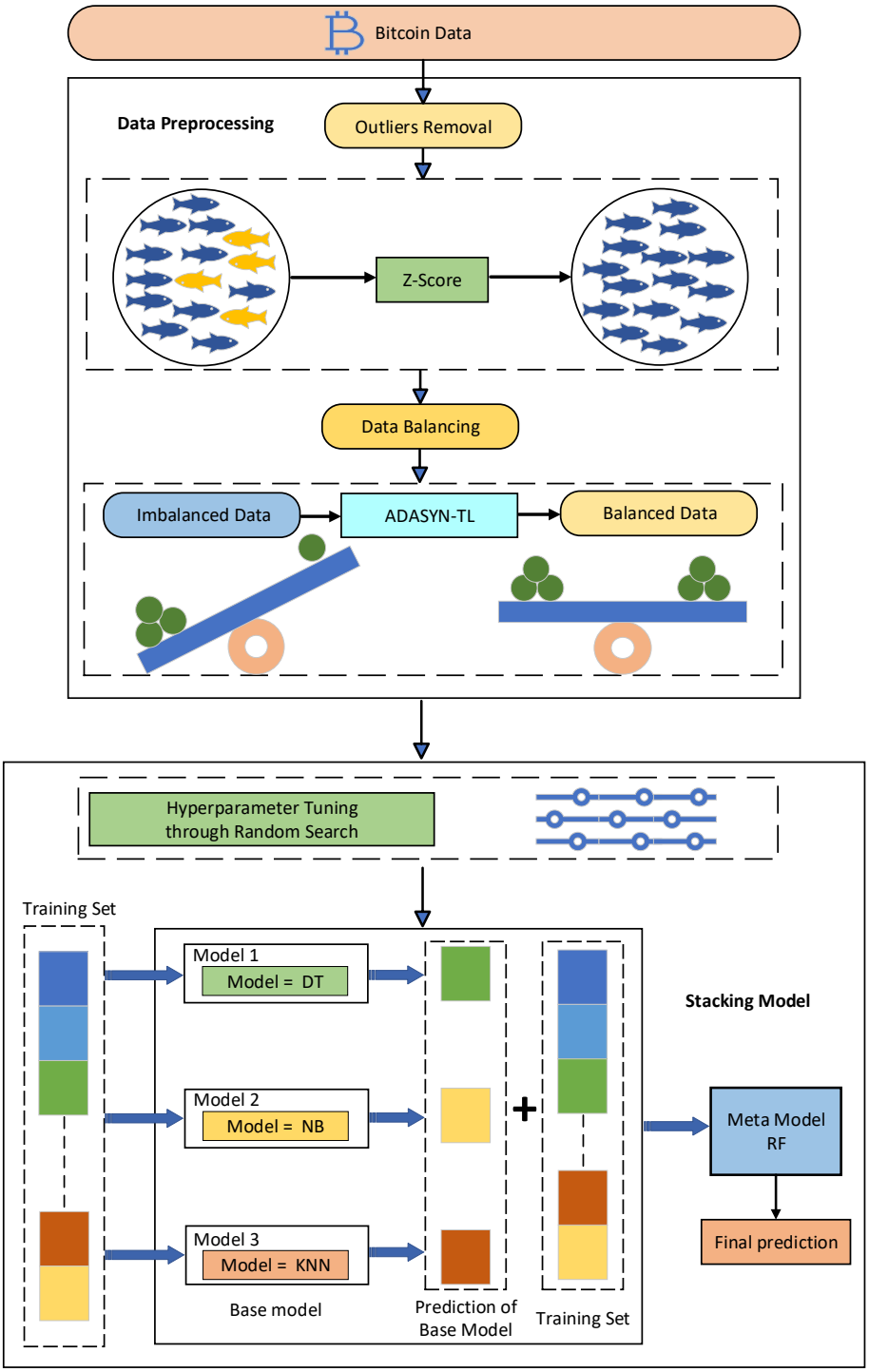


FIGURE 7: Proposed Model: Ensemble Bitcoin Detector

shows the TP of 96.9, FP of 3, FN of 1.8 and TN of 98.2 the unit is percentage. The first quadrant represents TP, second quadrant represents FP, third quadrant represents FN and fourth quadrant represents TN. The percentages of predicted instances are given in these categories. The percentage of each category shows the distribution of predictions. So, their sum should be equal to 100. Both TP and TN rates are high which indicates that the model performs well and correctly indicates the positive and negative classes.

Figure 13 shows the heatmap of the confusion matrix where the value of TP, FP, FN and TN are shown in each

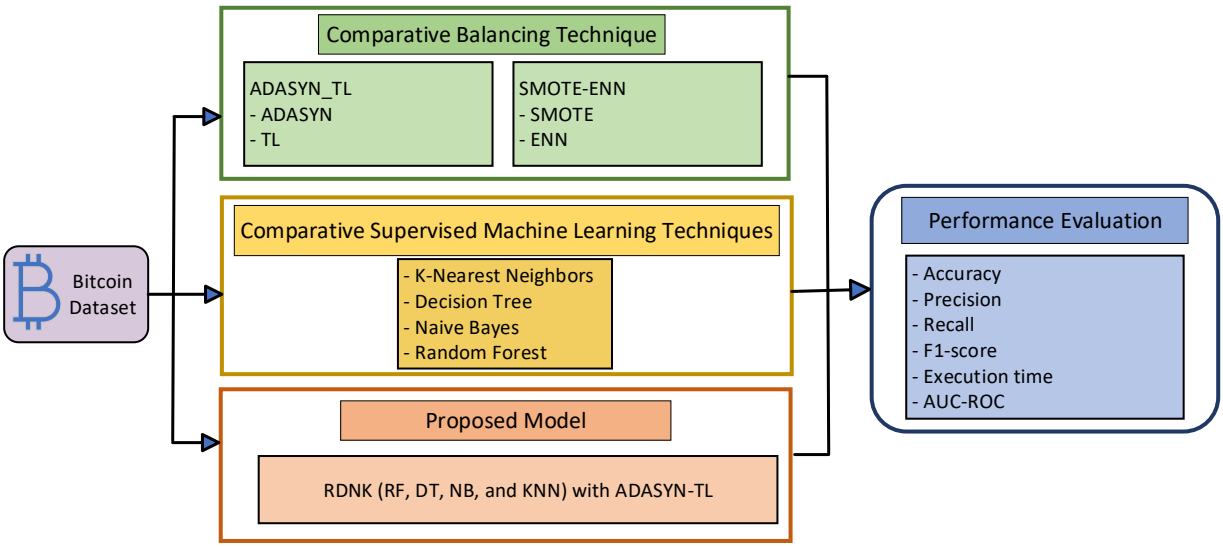


FIGURE 8: Comparative Study of the Proposed Model and Baseline Techniques

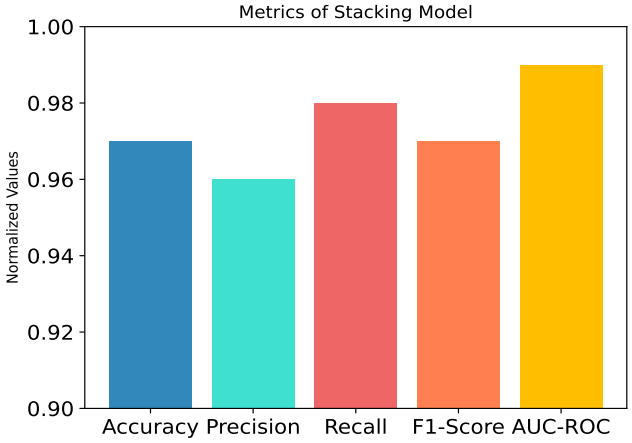


FIGURE 9: Simulation Results of the Proposed Model

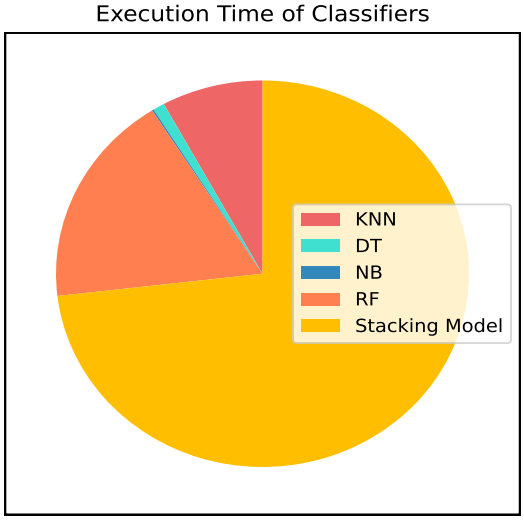


FIGURE 10: Execution Time (sec) of the Proposed Model and Baseline Techniques

cell. We can see the effectiveness of our model based on the confusion matrix by comparing the color density of each cell to the number of data instances that fit into the given category.

Table 5 shows the mapping of the proposed solution to the validation results. L1-L4 are the labels of identified limitations, similarly, S1-S4 are the labels of the proposed solutions and V1-V4 are the labels of validation results. Firstly, the model is validated in terms of reducing the overfitting issue, as shown in Figure 12. Secondly, the model achieves maximum accuracy. Thirdly, by incorporating DT with other classifiers, accuracy is increased by 10 points. Fourthly, a stacking model incorporates different classifiers and is able to handle big datasets.

Table 6 and Figure 14 show the comparison of different optimization techniques namely random search, grid search and Bayesian optimization. Random search gives better result than the Bayesian optimization and grid search and also takes less time to train. The stacking model achieves the highest AUC-ROC score of 99%, accuracy of 97%, and F1-score of 97%. For large datasets, random search is found to be more efficient than Bayesian optimization and grid search. It is because it selects random samples and finds the best parameter values.

Table 7 compares two balancing methods ADASYN-TL and SMOTE-ENN. These techniques are employed on four different classifiers and on the stacking model.

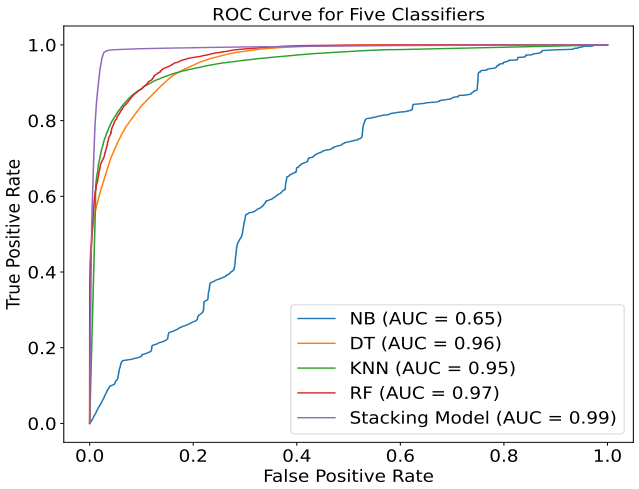


FIGURE 11: AUC-ROC Curve of Baseline Techniques and Proposed Model

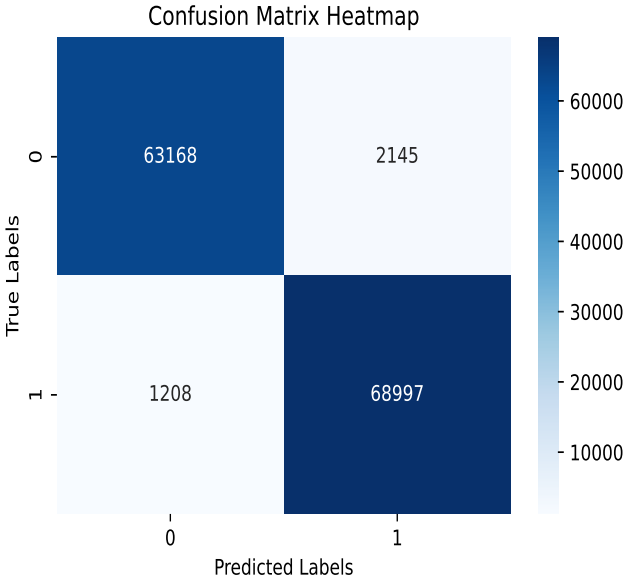


FIGURE 13: Heatmap of Confusion Matrix

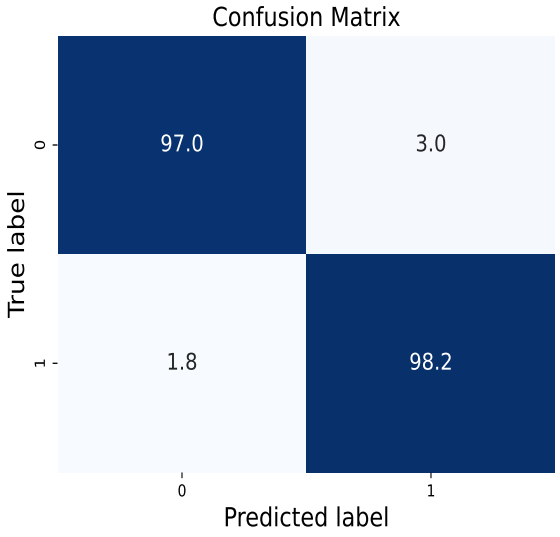


FIGURE 12: Confusion Matrix (%) of the Proposed Model

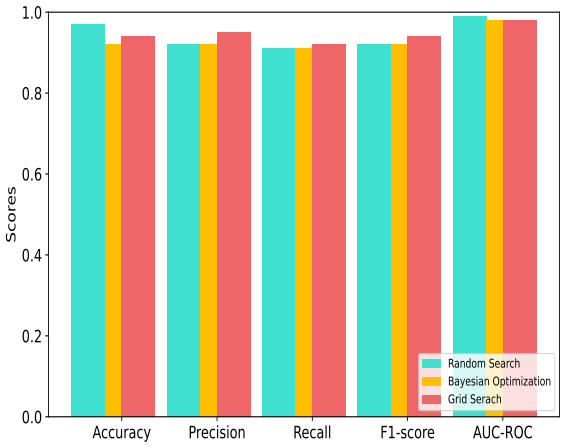


FIGURE 14: Performance Metrics Comparison: Random Search, Grid Search and Bayesian Optimization

SMOTE-ENN outperforms ADASYN-TL in terms of accuracy and F1-score for KNN, RF, and DT. On the other hand, ADASYN-TL defeats SMOTE-ENN in terms of accuracy and F1-score for NB. Regardless of the balancing method employed, it is important to note that the performance of NB is somewhat poor as compared to other classifiers. A stacking model using ADASYN-TL performs well as compared to other classifiers but at the expense of more execution time.

B. BALANCING TECHNIQUES IN COMBINATION WITH HYPERPARAMETER TUNING

Figure 11 compiles the AUC-ROC curve in one graph and validates that the suggested model performs well among all the baseline techniques. Table 8 and Figure 15a show the comparison of the execution time of two balancing techniques. ADASYN-TL takes more time than SMOTE-ENN. The selection of the technique depends upon the priority. If the priority is time, then we choose a balancing technique that takes less time. However, our top priority is the performance of the model, so we choose ADASYN-TL for balancing because it performs well for the stacking model.

Figure 15b shows that the proposed model has a lower FPR of 3%, which indicates that the negative classes are correctly predicted as negative classes. NB has a high FPR, which means NB classifies a negative class as a positive class. The main reason behind high FPR is that NB treats all the features in the same way as each feature

TABLE 4: Performance Metrics of Different Classifiers and Proposed Model using Random Search

Classifier	Accuracy	Precision	Recall	F1-score	AUC-ROC	Time (sec)	FPR
KNN	0.89	0.88	0.88	0.88	0.95	347.97	0.10
DT	0.87	0.84	0.90	0.87	0.96	39.83	0.14
NB	0.52	0.48	0.99	0.65	0.65	6.17	0.88
RF	0.88	0.85	0.91	0.88	0.97	782.81	0.13
Stacking Model	0.97	0.96	0.98	0.97	0.99	3208.09	0.03

TABLE 5: Mapping of Limitations, Proposed Solution and Validation Results

Identified Limitations	**Proposed Solution**	**Performed Validation**
L1: LGBM algorithm may not be the best choice for small datasets, as it may lead to overfitting [8]. **L2:** As the number of input features increases, the classification accuracy of SNN and DT tends to decrease [28]. **L3:** The authors noted that a single algorithm may not be enough to provide accurate results. Moreover, a single ML model may overfit the training data and not generalize well to new or unseen data, resulting in poor anomaly detection performance [11]. **L4:** The proposed algorithm is relying heavily on the accuracy and completeness of the rule set. It may not be effective in detecting new or previously unknown ransomware [19].	**S1, S2, S3:** The stacking ensemble model is used instead of boosting. **S4:** The stacking model increases the accuracy of the proposed algorithm by incorporating ML techniques with rule-based approach	**V1:** The stacking model solves the issue of overfitting, as shown in Figure 11. **V2:** The stacking model achieves maximum accuracy of 97%, as shown in Table 4. **V3:** The size of the dataset does not affect the performance of the stacking model. **V4:** The stacking model incorporates DT with other classifiers, increasing the accuracy by 10 percent, as shown in Table 4.

TABLE 6: Comparison of Different Optimization Techniques

	Random Search	Grid Search	Bayesian Optimization
Performance Metrics	Stacking model	Stacking Model	Stacking Model
Accuracy	0.97	0.94	0.92
Precision	0.96	0.95	0.92
Recall	0.98	0.92	0.91
F1-score	0.97	0.94	0.92
AUC-ROC	0.99	0.98	0.97
Time(sec)	3208.09	5089.67	3337.97

has the same impact on the result. So, sometimes this leads to falsification.

Figure 15c shows the comparison of different classifiers and the proposed model using different metrics, i.e., accuracy, precision, recall, F1-score, and AUC-ROC. The figure validates that the suggested model performs well among all the benchmarks by achieving an accuracy of 97%, recall of 98%, AUC-ROC of 99%, F1-score of 97%, and precision of 96%.

Using ADASYN-TL in the stacking model a higher F1-score value than SMOTE-ENN is achieved in combination with hyperparameters tuning. ADASYN is the extension of SMOTE and can generate synthetic samples in more adaptive way which results in better per-

formance. The value of the F1-score and accuracy of the stacking model using ADASYN-TL is 97% and 97%, and with SMOTE-ENN, the value is 96% and 97%. The comparison of the balancing techniques using a hyperparameter is shown in Figure 16a.

Table 8 shows the performance of four classifiers (KNN, DT, NB, and RF) and a stacking model employing two balancing methods ADASYN-TL and SMOTE-ENN in combination with random search hyperparameter tuning. NB required very low time. SMOTE-ENN also takes less time to train and test than ADASYN-TL for all classifiers. The stacking model using ADASYN-TL performs the best using both SMOTE-ENN and ADASYN-TL in terms of the F1-score.

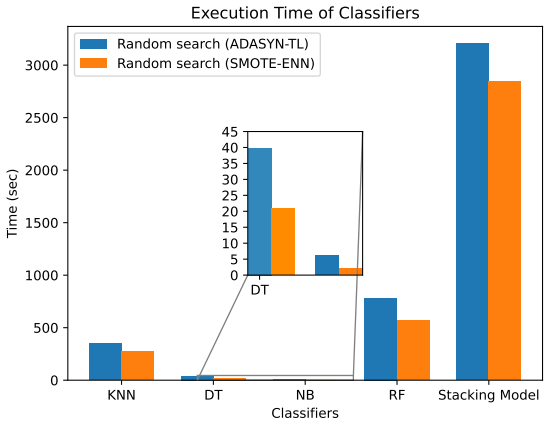
The results of random search hyperparameter tuning and without hyperparameter tuning for four classifiers (RF, DT, NB, and KNN) and the stacking model using different balancing techniques are shown in Tables 9, 10 and Figures 16b, 16c. Results show that using random search hyperparameters tuning, the overall performance of the classifiers can be improved. However, the time required to train the classifiers may increase because hyperparameter tuning needs more computations. Both balancing strategies perform poorly for NB. For the stacking model, ADASYN-TL defeats SMOTE-ENN and the value of the F1-score of ADASYN-TL is 1 percent greater than SMOTE-ENN.

From Figure 9 it is inferred that the stacking model achieves the highest accuracy. We applied random search hyperparameter tuning on our model and got optimal results. Figure 16b shows the results of SMOTE-ENN with hyperparameter tuning and without hyperparameter tuning.

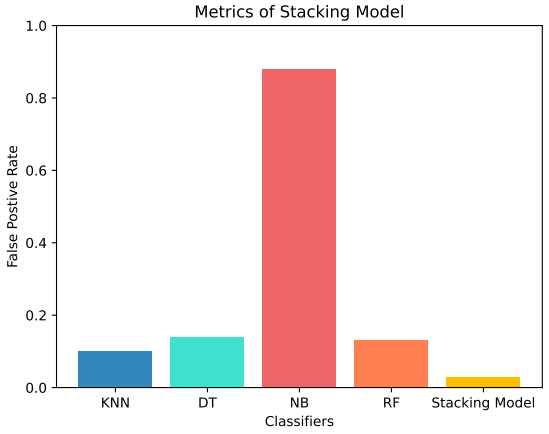
Hyperparameter tuning in combination with the stacking model using SMOTE-ENN for balancing has a great impact in increasing the model's performance. From Table 10 it is inferred that the stacking model's accuracy increased by 1 point and F1-score increased by 2 percent.

Figure 16c also shows the comparison of using hyperparameter tuning with and without the ADASYN-TL balancing technique. ADASYN-TL in combination with the stacking model increases the accuracy by 2 points and F1-Score by 2 points. Hyperparameter tuning plays an important role in optimizing the performance of the model on unseen data, enhancing model complexity, and reducing overfitting. Both balancing techniques perform well but ADASYN-TL achieved the maximum F1-score value of 97%. However, F1-score is better than accuracy because it takes both precision and recall into account.

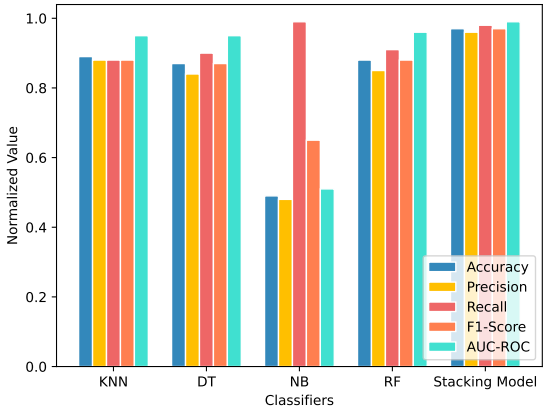
Figures 17a, 17b, and 17c show the dependence plot of day, weight, and income features. These plots provide information about the influence of the day, weight, and income features on the predicted value. The vertical location displays the impact a feature has on the prediction, while the horizontal location represents the actual value from the dataset. In figure 17a you can see that the feature day is mostly high with the megative value



(a) Execution Time (sec) comparison of Different Classifiers and Stacking Model



(b) False Positive Rate of Proposed Model and Benchmark Techniques



(c) Comparison of Proposed Model with Baseline Classifiers using Different Metrics

FIGURE 15: Comparison of the Proposed Model with Baseline Classifiers in terms of Different performance Metrics

TABLE 7: Comparison of Classifiers using Different Balancing Techniques

	SMOTE-ENN			ADASYN-TL		
Classifier	Accuracy	F1-Score	Time (sec)	Accuracy	F1-Score	Time (sec)
KNN	0.91	0.88	1.31	0.82	0.82	2.19
DT	0.84	0.79	1.04	0.82	0.81	1.95
NB	0.43	0.55	0.36	0.52	0.66	0.27
RF	0.85	0.78	31.55	0.83	0.82	50.09
Stacking Model	0.96	0.94	130.82	0.95	0.95	197.90

TABLE 8: Random Search with Balancing Techniques

	Random search (SMOTE ENN)			Random search (ADASYN-TL)		
Classifier	Accuracy	F1-Score	Time (sec)	Accuracy	F1-Score	Time (sec)
KNN	0.93	0.90	275.75	0.89	0.88	347.97
DT	0.90	0.87	20.83	0.87	0.87	39.83
NB	0.43	0.55	2.11	0.52	0.65	6.17
RF	0.91	0.87	567.07	0.88	0.88	782.81
Stacking Model	0.97	0.96	2845.40	0.97	0.97	3208.09

TABLE 9: Comparison of ADASYN-TL with and without using Hyperparameter Tuning

	Without Hyperparameter Tuning			Random Search		
Classifier	Accuracy	F1-Score	Time (sec)	Accuracy	F1-Score	Time (sec)
KNN	0.82	0.82	2.19	0.89	0.88	347.97
DT	0.82	0.81	1.95	0.87	0.87	39.83
NB	0.54	0.68	0.27	0.52	0.65	6.17
RF	0.82	0.83	50.09	0.88	0.88	782.81
Stacking Model	0.95	0.95	197.90	0.97	0.97	3208.09

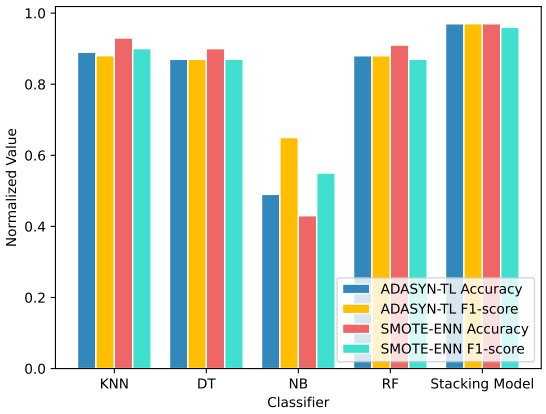
TABLE 10: Comparison of SMOTE-ENN with and without using Hyperparameter Tuning

	Without Hyperparameter Tuning			Random Search		
Classifier	Accuracy	F1-Score	Time (sec)	Accuracy	F1-Score	Time (sec)
KNN	0.91	0.88	1.31	0.93	0.90	275.75
DT	0.84	0.79	1.04	0.90	0.87	20.83
NB	0.43	0.55	0.36	0.43	0.55	2.11
RF	0.85	0.78	31.55	0.91	0.87	567.07
Stacking Model	0.96	0.94	130.82	0.97	0.96	2845.40

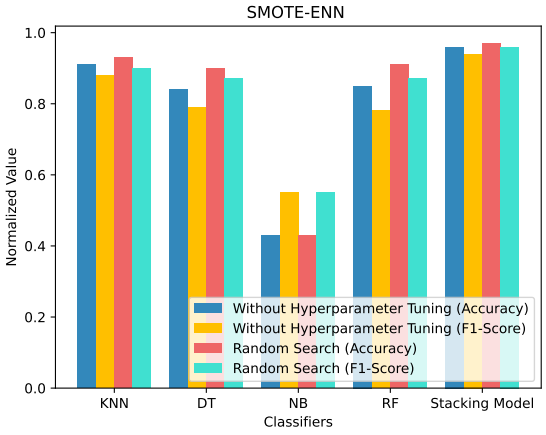
SHAP value. It means higher day counts tend negative affect on the output.

In Figure 18, a summary plot is shown, which is used to reveal information about the significance of each feature as well as the effect of the Shapley value. Each feature value is represented by a specific color. Red color represents a low shapley value and blue color represents a high shapley value. If the color is red, it will push transaction towards class 0 and if the color is blue, it will push transaction towards class 1.

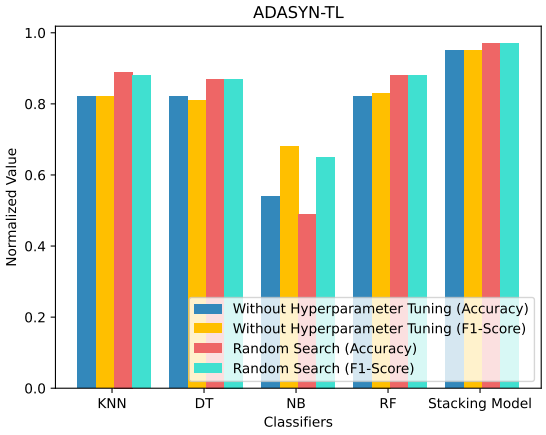
In Figure 19, we observe how features influence the model's prediction for a single observation using a force plot. The bold **1.0** is the model's value for the specific observation.



(a) Comparison of Balancing Techniques using Hyperparameter Tuning in terms of Accuracy and F1 Score

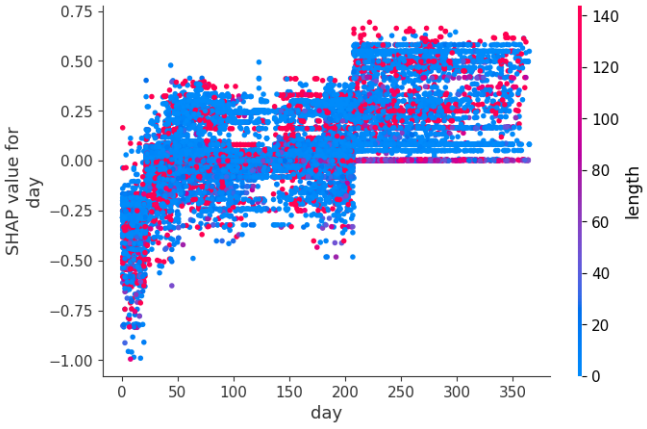


(b) Comparison of SMOTE-ENN without Hyperparameter Tuning and Random Search in terms of Accuracy and F1-score

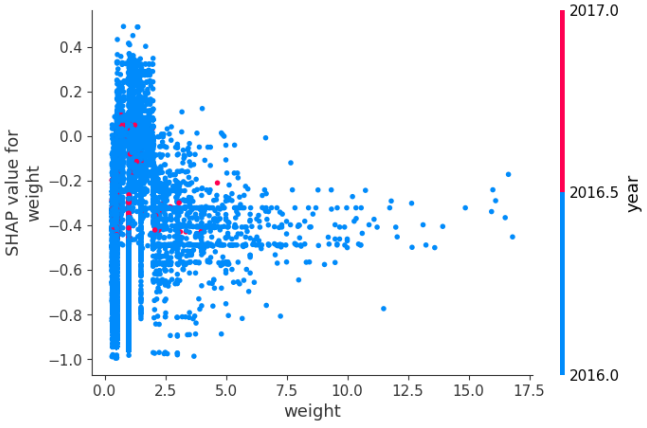


(c) Comparison of ADASYN-TL without Hyperparameter Tuning and Random Search in terms of Accuracy and F1-score

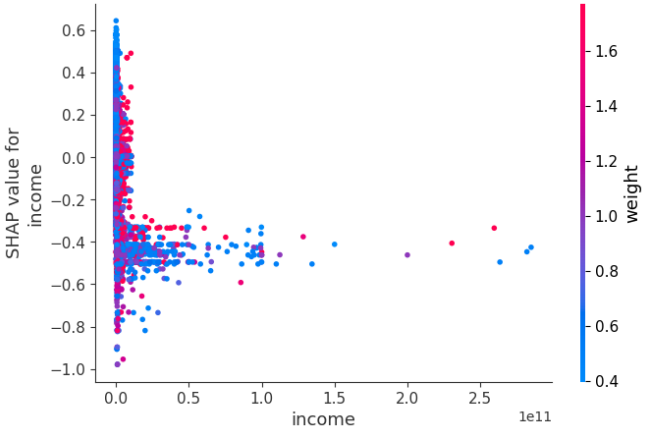
FIGURE 16: Comparison of the Proposed Model with Baseline Classifiers using different Balancing Techniques



(a) Dependence Plot of Day Feature



(b) Dependence Plot of Weight Feature



(c) Dependence Plot of Income Feature

FIGURE 17: Dependence Plots of Features

Figure 20 shows the expected value of the model's output, displayed at the bottom of a waterfall plot, followed by each row's depiction of the positive (red) or negative (blue) contribution made by each feature in transforming the value from the expected model output across the background dataset to the model output for this prediction.

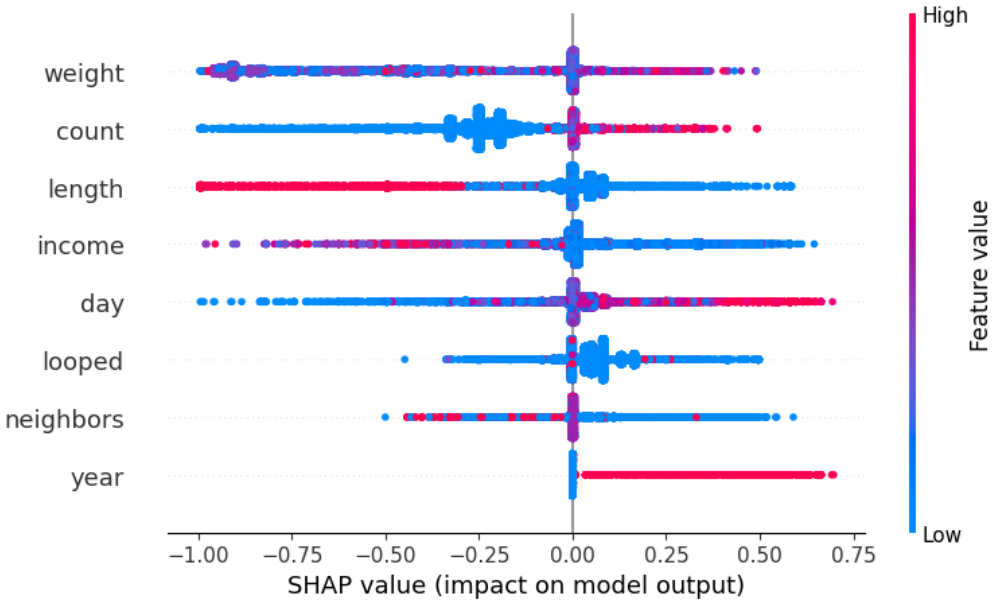


FIGURE 18: Summary Plot of Bitcoin Heist Dataset

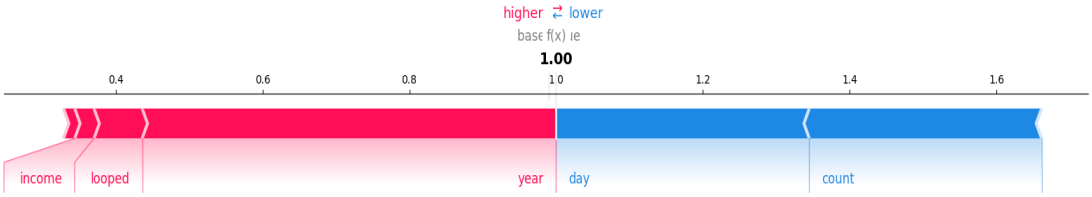


FIGURE 19: Force Plot

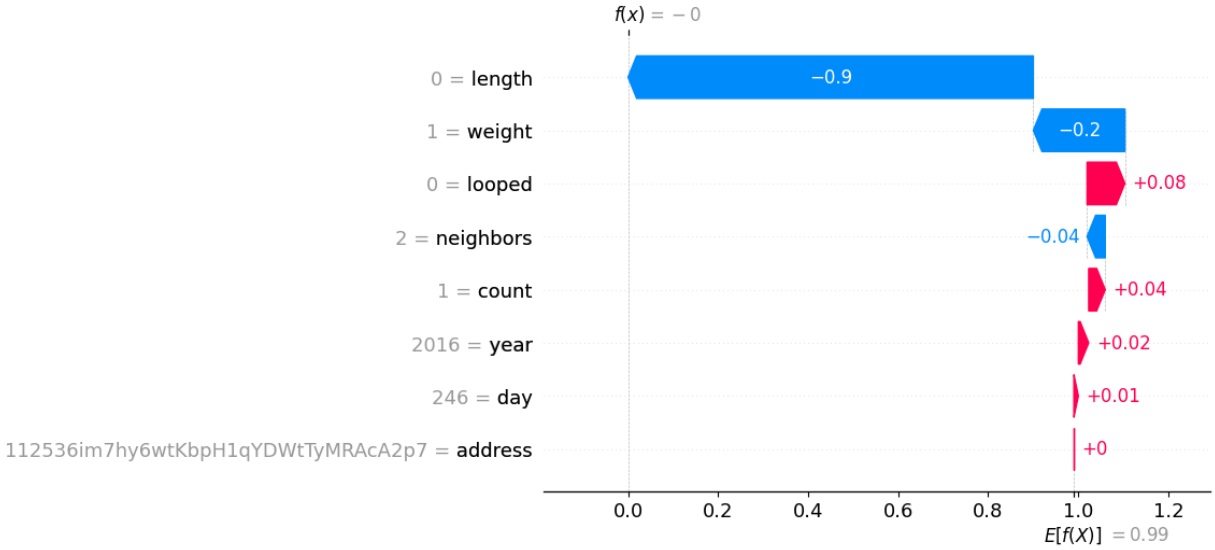


FIGURE 20: Waterfall Plot

VI. CONCLUSION

This paper presents the model for detecting fraud in Bitcoin transactions taking place in the smart cities. Firstly, 381464 instances are extracted out of 2916697 instances, by setting a threshold value on the year and the transfer amount. The amount filter is used to exclude the instances that are outside the range and the year filter is used to exclude the data beyond the 2016 year. After the data is gathered, we remove outliers from the data and then balance the dataset using ADASYN-TL, as the dataset is highly imbalanced. Random search, grid search, and Bayesian optimization hyperparameter tuning techniques are used to find the specific value for the parameters of the classifier. For classification, the stacking model is formed by combining DT, KNN, and NB on the base layer and using RF on the meta layer. The performance of the proposed model is validated by comparing it with different classifiers. SHAP is used to provide the information about the impact of features on the model's prediction. The simulation results show that the proposed stacking model using ADASYN-TL performs well among all algorithms by achieving an accuracy of 97%, AUC of 99%, precision of 96%, recall of 98%, FPR of 3% and an F1-score of 97%. In addition, the balancing techniques, ADASYN-TL and SMOTE-ENN, are also compared. ADASYN-TL outperforms SMOTE-ENN by achieving an F1-score of 97%. The stacking model achieved an accuracy of 95% without hyperparameter tuning. Whereas, the stacking model's accuracy increased by 2 percent when hyperparameter tuning is performed.

ACKNOWLEDGMENT

The authors extend their appreciation to the Researchers Supporting Program for Project number (RSPD2023R295), at King Saud University for supporting this research project.

REFERENCES

[1] M. U. Hassan, M. H. Rehmani, and J. Chen, "Anomaly Detection in Blockchain Networks: A Comprehensive Survey," IEEE Commun. Surv. Tutorials, vol. 25, no. 1, pp. 289–318, 2022, doi: 10.1109/COMST.2022.3205643.

[2] K. G. Al-Hashedi and P. Magalingam, "Financial fraud detection applying data mining techniques: A comprehensive review from 2009 to 2019," Comput. Sci. Rev., vol. 40, 2021, doi: 10.1016/j.cosrev.2021.100402.

[3] L. Pahuja and A. Kamal, "Enlfade: Ensemble Learning Based Fake Account Detection on Ethereum Blockchain," SSRN Electron. J., 2022, doi: 10.2139/ssrn.4180768.

[4] https://www.cylynx.io/blog/machine-learning-for-fraud-detection/ [Accessed on 17-04-2023]

[5] J. Nicholls, A. Kuppa, and N.-A. Le-Khac, "SoK: The Next Phase of Identifying Illicit Activity in Bitcoin," 2023 IEEE Int. Conf. Blockchain Cryptocurrency, pp. 1–10, 2023, doi: 10.1109/ICBC56567.2023.10174963.

[6] N. Kumar, A. Hashmi, M. Gupta, and A. Kundu, "Automatic Diagnosis of Covid-19 Related Pneumonia from CXR and CT-Scan Images," Eng. Technol. Appl. Sci. Res., vol. 12, no. 1, pp. 7993–7997, 2022, doi: 10.48084/etasr.4613.

[7] M. Horduna, "A note on machine learning applied in ransomware detection," no. May 2021, p. 17, [Online]. Available: https://eprint.iacr.org/2023/045.pdf

[8] R. M. Aziz, M. F. Baluch, S. Patel, and A. H. Ganie, "LGBM: a machine learning approach for Ethereum fraud detection," Int. J. Inf. Technol., vol. 14, no. 7, pp. 3321–3331, 2022, doi: 10.1007/s41870-022-00864-6.

[9] L. Pahuja and A. Kamal, "EnLEFD-DM: Ensemble Learning based Ethereum Fraud Detection using CRISP-DM framework," Expert Syst., pp. 1–18, 2023, doi: 10.1111/exsy.13379.

[10] P. Nerurkar, "Illegal activity detection on Bitcoin transaction using deep learning," Soft Comput., 2023, doi: 10.1007/s00500-022-07779-1.

[11] B. Chen, F. Wei, and C. Gu, "Bitcoin Theft Detection Based on Supervised Machine Learning Algorithms," Secur. Commun. Networks, vol. 2021, doi: 10.1155/2021/6643763.

[12] Ashfaq, Tehreem, Rabiya Khalid, Adamu Sani Yahaya, Sheraz Aslam, Ahmad Taher Azar, Safa Alsafari, and Ibrahim A. Hameed. "A Machine Learning and Blockchain Based Efficient Fraud Detection Mechanism." Sensors 22, no. 19 (2022): 7162, doi: 10.3390/s22197162

[13] M. J. Shayegan, H. R. Sabor, M. Uddin, and C. L. Chen, "A Collective Anomaly Detection Technique to Detect Crypto Wallet Frauds on Bitcoin Network," Symmetry (Basel)., vol. 14, no. 2, pp. 1–13, 2022, doi: 10.3390/sym14020328.

[14] S. Karim and S. Pa, "A Survey on Detection and Classification of Ransomware Bitcoin Transactions," no. 3, pp. 2987–2994, 2021, doi: 16.0415/IJARIIE-14703.

[15] R. Agarwal, T. Thapliyal, and S. Shukla, "Analyzing Malicious Activities and Detecting Adversarial Behavior in Cryptocurrency based Permissionless Blockchains: An Ethereum Usecase," Distrib. Ledger Technol. Res. Pract., vol. 1, no. 2, pp. 1–21, 2022, doi: 10.1145/3549527.

[16] Singh, Priyanshi, Deepika Agrawal, and Sudhakar Pandey. "Anomaly detection and analysis in blockchain systems." (2023), doi: 10.21203/rs.3.rs-2414745/v1.

[17] F. Zola, L. Segurola-Gil, J. L. Bruse, M. Galar, and R. Orduna-Urrutia, "Attacking Bitcoin anonymity: generative adversarial networks for improving Bitcoin entity classification," Appl. Intell., vol. 52, no. 15, pp. 17289–17314, 2022, doi: 10.1007/s10489-022-03378-7.

[18] C. G. Akcora, Y. Li, Y. R. Gel, and M. Kantarcioglu, "Bitcoin heist: Topological data analysis for ransomware prediction on the Bitcoin blockchain," IJCAI Int. Jt. Conf. Artif. Intell., vol. 2021-Janua, pp. 4439–4445, 2020, doi: 10.24963/ijcai.2020/612.

[19] H. S. Talabani and H. M. T. Abdulhadi, "Bitcoin Ransomware Detection Employing Rule-Based Algorithms," Sci. J. Univ. Zakho, vol. 10, no. 1, pp. 5–10, 2022, doi: 10.25271/sjuoz.2022.10.1.865.

[20] B. Kabir, U. Qasim, N. Javaid, A. Aldegheishem, N. Alrajeh, and E. A. Mohammed, "Detecting Nontechnical Losses in Smart Meters Using a MLP-GRU Deep Model and Augmenting Data via Theft Attacks," Sustain., vol. 14, no. 22, pp. 1–19, 2022, doi: 10.3390/su142215001.

[21] B. Kilic, A. Sen, and C. Ozturan, "Fraud Detection in Blockchains using Machine Learning," 2022 4th Int. Conf. Blockchain Comput. Appl. BCCA 2022, pp. 214–218, 2022, doi: 10.1109/BCCA55292.2022.9922045.

[22] Q. A. Al-Haija and A. A. Alsulami, "High performance classification model to identify ransomware payments for heterogeneous Bitcoin networks," Electron., vol. 10, no. 17, 2021, doi: 10.3390/electronics10172113.

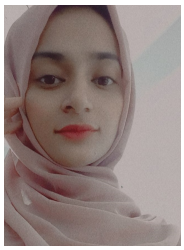
[23] S. A. Alsaif, "Machine Learning-Based Ransomware Classification of Bitcoin Transactions," Appl. Comput. Intell. Soft Comput., vol. 2023, no. Ml, pp. 1–10, 2023, doi: 10.1155/2023/6274260.

[24] J. Singh, K. Sharma, M. Wazid, and A. K. Das, "SINN-RD: Spline interpolation-envisioned neural network-based ransomware detection scheme," Comput. Electr. Eng., vol. 106, no. December 2022, p. 108601, 2023, doi: 10.1016/j.compeleceng.2023.108601.

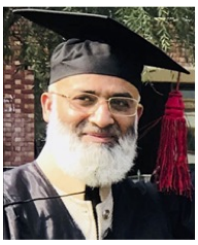
[25] Li, Lin. "Investigating risk assessment in post-pandemic household cryptocurrency investments: an explainable machine learning approach." Journal of Asset Management (2023): 1-13, doi: 10.1057/s41260-022-00302-z.

[26] W. Yu, Y. Wang, L. Liu, Y. An, B. Yuan, and J. Panneerselvam, "A Multiperspective Fraud Detection Method for Multiparticipant

E-Commerce Transactions," IEEE Trans. Comput. Soc. Syst., pp. 1–13, 2023, doi: 10.1109/tcss.2022.3232619.
[27] R. Saxena, D. Arora, and V. Nagar, "ScienceDirect ScienceDirect Classifying Transactional Addresses using Supervised Learning Classifying Transactional Addresses using Supervised Learning Approaches over Ethereum Blockchain Approaches over Ethereum Blockchain," Procedia Comput. Sci., vol. 218, no. 2022, pp. 2018–2025, 2023, doi: 10.1016/j.procs.2023.01.178.
[28] A. A. Badawi and Q. A. Al-Haija, "Detection of money laundering in bitcoin transactions," pp. 458–464, 2022, doi: 10.1049/icp.2022.0387.
[29] S. Hisham, M. Makhtar, and A. A. Aziz, "Combining Multiple Classifiers using Ensemble Method for Anomaly Detection in Blockchain Networks: A Comprehensive Review," Int. J. Adv. Comput. Sci. Appl., vol. 13, no. 8, pp. 404–422, 2022, doi: 10.14569/IJACSA.2022.0130848.
[30] P. Nerurkar, S. Bhirud, D. Patel, R. Ludinard, Y. Busnel, and S. Kumari, "Supervised learning model for identifying illegal activities in Bitcoin," Appl. Intell., vol. 51, no. 6, pp. 3824–3843, 2021, doi: 10.1007/s10489-020-02048-w.
[31] D. Sanz-Bas, C. del Rosal, S. L. Náñez Alonso, and M. Á. Echarte Fernández, "Cryptocurrencies and Fraudulent Transactions: Risks, Practices, and Legislation for Their Prevention in Europe and Spain," Laws, vol. 10, no. 3, p. 57, 2021, doi: 10.3390/laws10030057.
[32] M. Caprolu, S. Raponi, G. Oligeri, and R. Di Pietro, "Cryptomining makes noise: Detecting cryptojacking via Machine Learning," Comput. Commun., vol. 171, no. February, pp. 126–139, 2021, doi: 10.1016/j.comcom.2021.02.016.
[33] S. Xu, "The Application of Machine Learning in Bitcoin Ransomware Family Prediction," ACM Int. Conf. Proceeding Ser., pp. 21–27, 2021, doi: 10.1145/3471287.3471300.
[34] P. Nerurkar, Y. Busnel, R. Ludinard, K. Shah, S. Bhirud, and D. Patel, "Detecting Illicit Entities in Bitcoin using Supervised Learning of Ensemble Decision Trees," ACM Int. Conf. Proceeding Ser., pp. 25–30, 2020, doi: 10.1145/3418981.3418984.
[35] S. Farrugia, J. Ellul, and G. Azzopardi, "Detection of illicit accounts over the Ethereum blockchain," Expert Syst. Appl., vol. 150, p. 113318, 2020, doi: 10.1016/j.eswa.2020.113318.
[36] I. Alarab, S. Prakoonwit, and M. I. Nacer, "Comparative Analysis Using Supervised Learning Methods for Anti-Money Laundering in Bitcoin," ACM Int. Conf. Proceeding Ser., pp. 11–17, 2020, doi: 10.1145/3409073.3409078.
[37] A. Maurya and A. Kumar, "Credit card fraud detection system using machine learning technique," Proc. - 2022 IEEE Int. Conf. Cybern. Comput. Intell. Cybern. 2022, pp. 500–504, 2022, doi: 10.1109/CyberneticsCom55287.2022.9865466.
[38] D. Chaudhari, R. Agarwal, and S. K. Shukla, "Towards Malicious address identification in Bitcoin," Proc. - 2021 IEEE Int. Conf. Blockchain, Blockchain 2021, pp. 425–432, 2021, doi: 10.1109/Blockchain53845.2021.00066.
[39] https://medium.com/analytics-vidhya/using-ai-to-detect-bitcoin-addresses-involved-in-ransomware-transactions-3beaeccba176 [Accessed on: 14-04-2023]
[40] https://www.kaggle.com/datasets/gopalmahadevan/bitcoin-heist-ransomware-address-dataset [Accessed on: 14-04-2023]
[41] T. A. Alghamdi and N. Javaid, "A Survey of Preprocessing Methods Used for Analysis of Big Data Originated from Smart Grids," IEEE Access, vol. 10, pp. 29149–29171, 2022, doi: 10.1109/ACCESS.2022.3157941.
[42] https://towardsdatascience.com/imbalanced-classification-in-python-smote-enn-method-db5db06b8d50. [Accessed on 03-05-2023]
[43] A. Ullah, N. Javaid, M. U. Javed, Pamir, B. S. Kim, and S. A. Bahaj, "Adaptive Data Balancing Method Using Stacking Ensemble Model and Its Application to Non-Technical Loss Detection in Smart Grids," IEEE Access, vol. 10, no. November, pp. 133244–133255, 2022, doi: 10.1109/ACCESS.2022.3230952.
[44] https://medium.com/@ruinian/an-introduction-to-adasyn-with-code-1383a5ece7aa [Accessed on 024-04-2023]
[45] Zabinsky, Zelda B. "Random Search Algorithms, Department of Industrial and Engineering, University of Washington." Seattle. April 5 (2009).
[46] https://en.wikipedia.org/wiki/Random_search [Accesed on 27-04-2023]
[47] I. U. Khan, N. Javeid, C. J. Taylor, K. A. A. Gamage, and X. Ma, "A Stacked Machine and Deep Learning-Based Approach for Analysing Electricity Theft in Smart Grids," IEEE Trans. Smart Grid, vol. 13, no. 2, pp. 1633–1644, 2022, doi: 10.1109/TSG.2021.3134018.
[48] https://www.javatpoint.com/stacking-in-machine-learning [Accessed on 17-04-2023]
[49] https://openi.nlm.nih.gov/detailedresult?img=PMC3918356_CMMM2014-276589.alg.001&req=4 [Accessed on 15-04-2023]
[50] N. Kumar, N. Narayan Das, D. Gupta, K. Gupta, and J. Bindra, "Efficient Automated Disease Diagnosis Using Machine Learning Models," J. Healthc. Eng., vol. 2021, 2021, doi: 10.1155/2021/9983652.
[51] https://scikit-learn.org/stable/modules/tree.html [Accessed on 15-04-2023]
[52] https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html [Accessed on 11-04-2023]
[53] https://www.datacamp.com/tutorial/random-forests-classifier-python [Accessed on 11-04-2023]
[54] https://towardsdatascience.com/random-forests-algorithm-explained-with-a-real-life-example-and-some-python-code-affbfa5a942c
[55] https://www.javatpoint.com/performance-metrics-in-machine-learning [Accessed on 09-04-2023]



NOOR NAYYER received her bachelor's degree in Information Technology from University of Gujrat, in 2019. Her bachelor's research is in Code Smell Detector and their Refactoring Solution under the supervision of Ikram ul Haq. She continued her MS degree in Information Security in Bitcoin Fraud Detection under the supervision of Prof. Dr. Nadeem Javaid. Her thesis entitled Fraud Detection in Bitcoin Transaction through Ensemble Stacking Model shows her research interests.



NADEEM JAVAID (Senior Member, IEEE) received the bachelor's degree in computer science and physics from Gomal University, Dera Ismail Khan, Pakistan, in 1995, the master's degree in electronics from Quaid-i-Azam University, Islamabad, Pakistan, in 1999, and the Ph.D. degree from the University of Paris-Est, France, in 2010. He has Teaching and Research Experience of 25 years. He has worked as a Visiting Professor at the University of Technology Sydney, Australia. He is currently a Tenured Professor and the Founding Director of the Communications Over Sensors (ComSens) Research Laboratory, Department of Computer Science, COMSATS University Islamabad, Islamabad Campus. He has supervised 187 master's and 30 Ph.D. theses. He has authored over 950 papers in technical journals and international conferences. His research interests include energy optimization in smart/microgrids and wireless sensor networks using data analytics and blockchain. He was a recipient of the Best University Teacher Award (BUTA'16) from the Higher Education Commission (HEC) of Pakistan, in 2016, and the Research Productivity Award (RPA'17) from the Pakistan Council for Science and Technology (PCST), in 2017. He is an Editor of Sustainable Cities and Society journal.

MARIAM AKBAR received the M.Sc. degree from the Department of Physics and the M.Phil. degree from the Department of Electronics, Quaid-i-Azam University, Islamabad, Pakistan, in 2001 and 2004, respectively, under the supervision of Prof. Nadeem Javaid, and the Ph.D. degree from COMSATS University Islamabad, Pakistan, in 2016, with the thesis titled "On Network Lifetime Maximization In Wireless Sensor Networks with Sink Mobility." She is currently working as an Assistant Professor with the Department of Computer Science, COMSATS University Islamabad. Her research interests include underwater wireless sensor networks and energy management and blockchain.

ABDULAZIZ ALDEGHEISHEM received the Ph.D. degree in urban planning and spatial information from the University of Illinois at Urbana–Champaign, USA. He worked as the Head of the Department of Urban Planning, in 2012. He worked as an Adviser in a number of government agencies and supervised many projects and specialized studies. He is currently the Dean of the College of Architecture and Planning, King Saud University, and a Professor with the Department of Urban Planning. He also works as an Adviser with the Vision Realization Office (VRO) at the University, and the Supervisor of the Traffic Safety Technologies Chair. His research interests include spatial information in urban planning and management, also he focuses on areas related to city planning, spatial management, and smart city technologies.

NABIL ALRAJEH obtained his Ph.D. in biomedical Informatics engineering from Vanderbilt University, USA. Currently, he is a professor of Health Informatics at King Saud University. Dr. Alrajeh worked as a senior advisor for the Ministry of Higher Education, his role was implementing development programs including educational affairs, strategic planning, and research and innovation. He served as a member of the boards of trustees for five private universities in Saudi Arabia. His research interests include E-health Applications, Hospital Information Systems, Telemedicine, Healthcare applications of smart cities, and Wireless Sensor Networks.

MOHSIN JAMIL is an Associate Professor in the Department of Electrical and Computer Engineering at the Memorial University of Newfoundland, Canada. He serves as Associate Editor of IEEE Access, IEEE Canadian Journal of Electrical and Computer Engineering and Energies. His main interests include control system techniques for different applications, primarily focusing on power electronic converters for smart grids, hybrid renewable energy systems and mechatronic systems. He likes to work in other multidisciplinary areas, such as system identification, biomedical engineering, and communication systems using artificial intelligence-based control techniques.