Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.98549823

EFFICIENT FRAUD DETECTION IN ETHEREUM BLOCKCHAIN THROUGH MACHINE LEARNING AND DEEP LEARNING APPROACHES

Swapna Siddamsetti [1]*, Dr Muktevi Srivenkatesh [2]
[*1]Research Scholar, Dept. of Computer Science, GITAM School of Science, GITAM Deemed to be University, Vishakapatnam, and Assistant Professor, Dept. of Computer Science, Neil Gogte Institute of Technology, Hyderabad, Telangana, India.
[2]Associate Professor, Department of Computer Science, GITAM School of Science, GITAM Deemed to be University, Vishakapatnam, India.
*Corresponding author: email: swapnangit2021@gmail.com

Abstract:
Background: In this paper, we address the paramount issue of detecting fraudulent transactions within the Ethereum blockchain by applying machine learning algorithms. The significance of fraud detection lies in its contribution to preserving the security and integrity of the blockchain, which is imperative for preventing monetary losses attributable to fraudulent activities. The dataset employed in this study is a publicly accessible compendium of Ethereum transactions, encompassing 9,841 transactions. Each transaction in the dataset is characterized by attributes, including gas price, transaction fee, and timestamp. Methods: The research paper is methodically divided into two primary sections: data preprocessing and predictive modeling. The dataset is accurately processed in the data preprocessing phase, and salient features are extracted from the transactions. This step is crucial in refining the data for optimal performance in the subsequent phase. Findings: In the predictive modeling phase, machine learning algorithms are deployed to classify transactions as fraudulent or legitimate. The algorithms evaluated include decision trees, logistic regression, gradient boosting, XGBoost, and a hybrid model integrating random forests with deep neural networks (DNN). Novelty: The findings suggest that the proposed model achieves a precision rate of 97.16%, indicating a significant enhancement in detecting fraudulent Ethereum transactions compared to the current leading techniques. This research contributes to the ongoing efforts to enhance the security and reliability of blockchain transactions through advanced analytical techniques..

Keywords: Machine Learning algorithms, Blockchain, Deep learning, Fraud Detection, Ethereum

Introduction

The decentralized nature of the blockchain concept, which operates as a publicly accessible ledger, has garnered significant interest from various industries and scholars. The concept of blockchain was initially presented in a scholarly document by Nakamoto in 2008 [1]. A blockchain is an electronic ledger that operates decentralized and enables the recording, propagation, and synchronization of transactions across multiple ledgers. The decentralized structure of blockchain technology allows transactions to be executed without intermediaries, rendering it a suitable option for various financial services such as online payments, remittances, and digital assets [2; 3]. Notwithstanding its potential benefits, blockchain technology has demonstrated susceptibility to security risks and assaults, as exemplified by

previous incursions on cryptocurrencies that rely on blockchain [4]. The Ethereum platform, which operates on a decentralized blockchain network and is renowned for its ability to execute smart contracts, has experienced two separate attacks resulting in considerable disruptions to its network operations [5]. The assaults mentioned above were carried out without the express permission of the platform administrators. The decentralized architecture of Ethereum enables individuals to participate in digital transactions with minimal transaction costs and robust security measures. The extensive user base of the platform catalyzes for developers to introduce their network applications, thereby consolidating Ethereum's position as a powerful platform for decentralized applications, encompassing DeFi and NFTs.

The integration of automation with blockchain technology has facilitated the rapid adoption of this innovation in numerous sectors, such as online finance, the Internet of Things (IoT), supply chain management, healthcare, and insurance [6,7]. Blockchain technology is widely recognized as a universal and immutable ledger that streamlines documenting transactions between senders and recipients while monitoring the assets within a business network. Ethereum is a decentralized network that has gained popularity as a reliable platform for executing and validating smart contract accounts, essentially application codes. Smart contracts allow parties to participate in transactions without requiring a centralized authority to act as a custodian of trust. The field of literature offers numerous examples of attempts to replicate fraudulent transactional patterns. These models are then employed to identify previously unidentified transactions. The unwavering dedication to scholarly inquiry has been focused on enhancing models and assessing methodologies for detecting fraudulent behaviors. The current paper improves the fraudulent behavior detection in the blockchain. Our analysis focuses on the anomalies present in Ethereum, which deviate from established patterns and are identified as abnormal or problematic.

It is imperative to underscore the differentiation between lawful and malicious transactions. These phenomena warrant additional scrutiny. A comprehensive methodology involving a series of systematic experiments is employed to evaluate and contrast diverse ML techniques [8, 9]. Within the domain of Ponzi schemes, an investment scheme is characterized by a pyramid-like hierarchical structure, wherein the orchestrator of the scheme occupies the apex, and the participants at level 1 are situated below. The role of the level 1 participant is to stabilize the investors' investments at the highest level. The architectural design under consideration could be more sustainable as it is subject to a gradual decrease in acquiring new investors. This leads to a situation where the upper hierarchy benefits from profits while the lower levels experience financial losses [10]. The significance of Artificial Intelligence (AI) in accurately categorizing fraudulent accounts based on their transactional records, mainly through the framework of Deep Learning (DL), cannot be overstated. Deep Learning (DL) is considered a highly advanced form of Artificial Intelligence (AI) in comparison to Machine Learning (ML), as it requires less human intervention. Our present investigation aims to determine our model's efficacy in identifying fraudulent activities in the blockchain. Our study involves an analysis of the Ethereum Blockchain's transactional data, aiming to identify deviations from established patterns that may be considered anomalous or questionable. Acknowledging that these transactions require scrutiny regardless of their authenticity or shady character is essential.

The paper presents three main contributions:

It proposes a model for effectively identifying fraudulent actions.

It introduces machine learning and Deep Learning techniques designed to detect Ethereum fraud.

The paper thoroughly evaluates different classifiers using various performance metrics in conjunction with the proposed model, which can operate independently with minimal human intervention.

2. Related works

As demonstrated in scholarly literature, machine learning (ML) algorithms have been effectively utilized to detect anomalies. The standard methodology involves constructing records of security breaches by relying on documented instances of intrusion. The anomaly detection scheme proposed by the authors cited in reference [6] is founded on network behavioral patterns. According to the authors, the suggested approach is applicable universally and can be implemented on all blockchain frameworks.

The study aimed to create a tailored system for detecting anomalies in the Ethereum blockchain during the eleventh symphony [11]. This was achieved by applying Machine Learning methodologies, which were implemented to address security-related obstacles. The mechanism proposed employs a dynamic methodology by utilizing the conventional operational patterns of the Ethereum blockchain as a platform for training machine learning algorithms. Any departure from this established norm has been recognized as an aberration and recorded by the system. The study employed four discrete machine learning algorithms, namely Gaussian Naive Bayes (GaussianNB), K-Nearest Neighbours (KNN), Stochastic Gradient Descent (SDG), and Random Forest to train the model and evaluate the precision of the proposed methodology. According to the experimental results, it can be inferred that the Random Forest algorithm outperformed other algorithms, demonstrating a significantly high degree of accuracy of 99.84%.

The model presented by the author in reference [12] is focused on leveraging deep learning methodologies to identify plausible security risks within the Ethereum blockchain. The proposed methodology employs a deep neural network framework to identify attacks. It also incorporates unsupervised and supervised machine learning techniques for categorizing attacks into discrete taxonomies. Based on the performance metrics, the model demonstrates a 97.72% accuracy rate in detecting Ethereum attacks and an exceptional 99.4% accuracy rate in attack classification.

In [13], the authors classified the intrinsic susceptibilities of Ethereum intelligent contracts. The accomplishment was attained by employing machine learning techniques to perform feature extraction. The dataset's production was achieved using pixel values obtained from images and trigram feature extraction. The dataset was subjected to training using a range of algorithms. The Naive Bayes methodology exhibited exceptional performance in the F1-score, respectively. The Random Forest algorithm exhibited significant efficiency, corresponding to F1-scores of 96.71% and 96.61% through utilizing Binary Relevance and Classifier Chain.

However, it is essential to note that the MLkNN and BRkNN algorithms exhibited relatively unsatisfactory performance.

In [13], the authors proposed a novel approach to identifying and analyzing energy-efficient intelligent contracts. During the initial stage, parsing smart contracts involves transforming code elements into vectors encompassing each term's semantic and syntactic features. The proposed effort creates an annotated text that balances and informs energy resources contract terms. Annotating smart energy contracts using a domain-specific corpus builder as an embedding layer improves classification accuracy to 98.34%. Next, an architecture examines the source code to verify and find patterns in coding sections, operations, and transactions throughout the Ethereum platform.

In [15], the authors automated for identifying blockchain accounts involved in malicious activities. This is done to reduce the unintentional support that permissionless blockchains based on cryptocurrency may provide to malicious entities. The study employs cosine similarity (CS) metrics to investigate similarities among the feature vectors of accounts linked to various malicious activities. The findings indicate that Ethereum blockchain-related malicious activities demonstrate similar behavior. The K-Means clustering algorithm is subsequently utilized to determine whether accounts engaged in similar malicious activities exhibit clustering tendencies. The presented study also examined the influence of bias on the efficacy of a machine learning algorithm, which is dependent on the number of accounts associated with hateful conduct. A comparative examination of contemporary models has revealed that Neural Networks (NNs) resist biases arising from malicious activities and can withstand adversarial attacks. In contrast, machine learning algorithms previously employed to identify malicious accounts tended to exhibit bias towards over-representing malicious activities.

3. Materials and methods

The operational methodology presented in this paper is visually explained in Figure 1. The Ethereum dataset is initially partitioned into two discrete subsets, one designated for training and the other for testing. The training set instructs the Deep Learning (DL) model. The model undergoes a rigorous evaluation process, requiring it to accurately classify transactions as fraudulent or legitimate solely based on the testing dataset. The above dataset is obtained from reference [16] and encompasses legitimate and fraudulent transactions on the Ethereum platform. The dataset comprises 9,841 transactions executed on the Ethereum network, quantitatively speaking. Upon more detailed analysis, it has been determined that out of the total number of transactions, 7,662 are classified as legitimate, whereas 2,179 exhibit features indicative of fraudulent activity. The dataset exhibits a several range of 51 features, encompassing various transactional details such as the average transaction duration and value. Upon obtaining this dataset, we proceed to the preprocessing stage, where the data is cleaned and improved to achieve optimal efficacy. Following this, ,machine learning (ML) or Deep Learning models classify accounts into fraudulent or non-fraudulent classes.

Splitting the dataset

The present research endeavor involves the division of the dataset into two distinct subsets, namely the training and testing datasets. Utilizing the training dataset facilitates the acquisition and adjustment of the model's knowledge to the distinctive attributes of the data. Conversely, the dataset designated for testing functions as a mechanism for verifying and assessing the model's efficacy that has undergone training. Partitioning is crucial in determining the model's capacity to generalize to novel data and prevent it from overfitting the training data. In this particular investigation, the dataset is partitioned into a ratio of 4:1, whereby 80% of the data is assigned for training objectives, while the remaining 20% is reserved for testing purposes. Dividing data into an 80-20 split is widely used in machine learning. This approach ensures that the model has access to a substantial amount of data for learning purposes while preserving a significant subset for evaluating its performance on unseen data during the training phase.

Handling imbalances in the data

The Synthetic Minority Oversampling Technique, commonly known as SMOTE, is a general approach in ML utilized to tackle class imbalance in datasets. Imbalanced datasets, wherein one class has a significantly higher number of instances than the others, are frequently encountered in real-world scenarios.

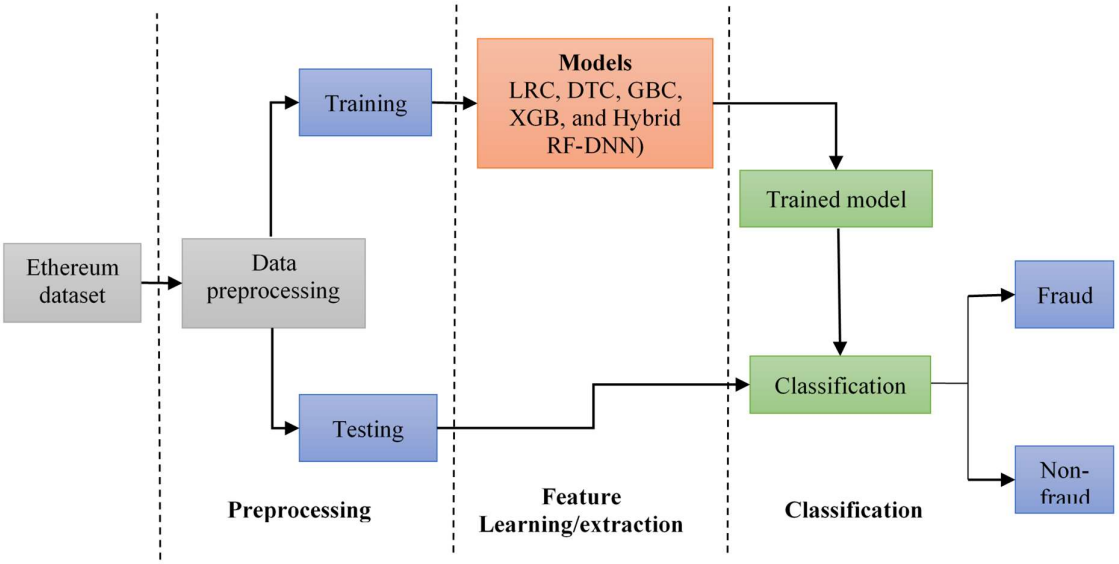

Figure 1: Operational methodology of proposed method

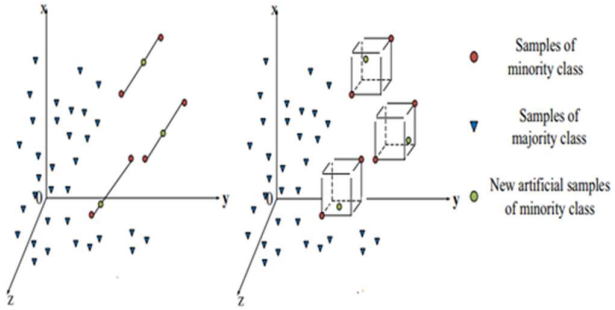

Figure 2: Diagrams of synthetic generation using (a) SMOTE and (b) ISMOTE algorithms with k=3

This section presents a summary of the Improved Synthetic Minority Oversampling Technique (ISMOTE) algorithm, devised to overcome the constraints associated with the conventional SMOTE in rectifying datasets with uneven class distributions. As depicted in Figure 2, the conventional SMOTE algorithm utilizes linear interpolation belonging to the minority class. Nevertheless, this methodology limits the scope in which the specimens can be produced. The ISMOTE technique broadens the scope of synthetic sample generation by prioritizing the proximity of minority class instances. ISMOTE effectively manages the generation process of new synthetic samples by implementing two predetermined constraints. Implementing this enhancement improves classifiers' generalization ability compared to the conventional SMOTE technique. The ISMOTE technique has been developed to achieve a more equitable and appropriate distribution of minority class instances following the balancing process. It is worth noting that the ISMOTE technique can produce synthetic samples that are on par with those generated by Random Oversampling (ROS) and conventional SMOTE algorithms. However, ISMOTE offers additional advantages in terms of distribution and generalization. The ISMOTE methodology is a sophisticated oversampling approach that effectively mitigates the limitations of the conventional SMOTE technique. It enhances the efficacy of classifiers that handle imbalanced datasets.

Classifiers

To verify different classifiers performance related to proposed method.

Logistic regression

This method predicts binary results using a set of independent variables within a given dataset. The design pertains to the probability of a discrete binary occurrence, where the potential outcomes are limited to 0 or 1. Logistic regression is a statistical technique used in sales to forecast the probability of a customer's purchase intention. Logistic regression is a statistical technique interpreted as modifying the linear regression model specifically designed to address classification tasks. Linear regression is concerned with continuous outcomes, whereas logistic regression is primarily concerned with binary classification. The dependent variable is categorized as 0 or 1 class in logistic regression. Linear regression considers the values of 0 and 1 as numerical quantities, while logistic regression treats them as separate categories. The utilization of the logistic function is a fundamental characteristic of logistic regression, whereby the output of the linear model is compressed into a bounded range of values between 0 and 1. Figure 3 depicts the visual representation of the concept mentioned above. In contrast to linear regression, the logistic function guarantees a bounded output between 0 and 1, rendering it appropriate for predicting probabilities [17, 18]. Logistic regression is a crucial technique in machine learning, particularly for tasks involving binary classification. It enables the estimation of class membership probabilities by converting linear combinations of features into a bounded range.

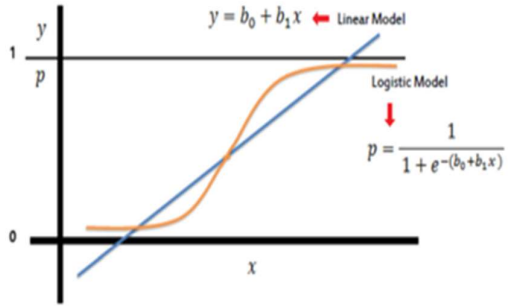

Figure 3: Overview of regression model

Random forest

This is used in various tasks, including classification and regression. The process entails constructing multiple decision trees during the training phase. In predictive modeling, the mode of the predicted classes generated by individual decision trees is utilized for classification tasks. In contrast, the mean prediction of the trees is considered for regression tasks. The Gini impurity is a commonly utilized metric for assessing the efficacy of the splits at every decision node when constructing decision trees. The Gini impurity formula considers a given dataset's probability and quantity of classes. Specifically, pi denotes the relative frequency of class I within the dataset, while c represents the total number of classes. The objective is to optimize the homogeneity of the group distribution in the derived subgroups following each division. Utilizing this methodology empowers Random Forests to generate models of high precision through the amalgamation of results derived from diverse decision trees.

$$Gini = 1 - \sum_{i=1}^{C} (pi)^2 \qquad (1)$$

Entropy determines a node's feasibility to bifurcate, contingent upon the probability of a specific outcome. The measure's mathematical complexity is higher than the Gini index [19, 20].

$$Entropy = \sum_{i=1}^{C} -pi \times log_2(pi) \qquad (2)$$

Gradient Boosting Classifier

This iterative functional gradient algorithm aims to minimize a loss function by iteratively choosing a weak hypothesis pointing toward the negative gradient.

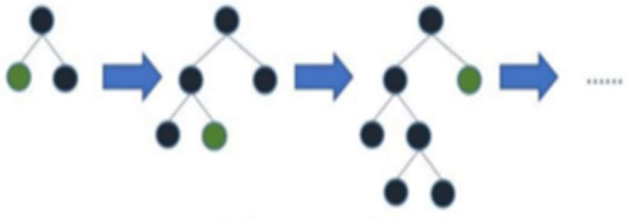

Figure 4: LGBM Classifier leaf-wise tree growth

Gradient Boosting is an ensemble learning technique that comprises three primary constituents: The loss function is a metric that measures the degree of alignment between the model's predictions and the data. The evaluation of a model's predictive performance is a crucial aspect

of its application, and the specific methodology employed for this purpose may be contingent upon the nature of the problem under consideration. The weak learner is a classification model with limited efficacy in accurately classifying data. The performance of the system may exhibit a marginal improvement over random selection while generally presenting a considerable margin of error. The Gradient Boosting technique employs an additive model approach, whereby decision trees commonly regarded as weak learners are incorporated sequentially and iteratively to build models. The objective is to minimize the loss function in each successive iteration, gradually approaching the ultimate model.

The "Light gradient boosting machine" (LGBM) is a tree-based approach that enhances the pace of conventional Gradient Boosting techniques through vertical tree growth [21-23]. The algorithm exhibits a leaf-wise growth pattern, which reduces loss along the vertical axis. This is in contrast to other algorithms that adopt a horizontal growth pattern. The diagrams illustrate that Light Gradient Boosting Machine (LGBM) initiates from the leftmost leaf and expands leaf-wise, substantially reducing the loss in the vertical direction. The LGBM algorithm has been observed to exhibit high efficiency when processing large datasets. However, it has been noted that in some instances, the algorithm may exhibit overfitting tendencies when processing smaller datasets. The literature acknowledges its memory efficiency and ability to process substantial volumes of data while utilizing minimal memory resources [24].

Decision Tree Classifier

Decision Trees are a prime example of supervised machine learning, as they clarify the connections between input characteristics and their corresponding outputs in the training dataset. The Decision Tree's architecture is characterized by a hierarchical structure, where the data undergoes a series of bifurcations based on specific attributes. The aforementioned hierarchical arrangement can be deconstructed into two fundamental constituents: decision nodes and leaves. The leaves of a tree represent the final determinations or results obtained by descending from the root to a leaf, considering the input data's characteristics. Decision nodes serve as points of divergence where the data is partitioned into subsets according to a specific attribute or criterion. Developing a Decision Tree generally comprises two consecutive stages: tree expansion and trimming. The process of growing a tree is intrinsically recursive, involving the identification of a splitting attribute, implementing the split, and subsequent recursive application of the same procedure to the offspring nodes. The iterative procedure persists until the data associated with a specific pathway attains homogeneity or the subset is considered too insignificant to justify additional division. The following pruning stage is similar to that of a discerning horticulturist who prunes a tree, as it entails the removal of segments located at the lower ends of the Decision Tree that may have overfitted to the noise or peculiarities within the training data. Pruning enhances the model's ability to generalize to previously encountered data effectively.

Extreme Gradient Boosting (XGB) Classifier

This is a powerful ensemble ML technique that utilizes trees. It is widely recognized for its effectiveness in performing classification and regression tasks. XGBoost is an ensemble technique that combines several models' predictive capabilities to create a more resilient and precise final model. The subject matter comprises three gradient-boosting iterations [25, 26].

XGBoost is distinguished by its rapidity in comparison to alternative gradient-boosting methodologies. The acceleration observed can be partly attributed to the clever optimizations implemented in the underlying code and the parallelization of the tree-building process. Furthermore, XGBoost possesses an automatic pruning capability that is a self-imposed constraint on the expansion of decision trees. The feature in question prevents the excessive growth of trees beyond a pre-established limit, thereby reducing the risks associated with overfitting. Overfitting occurs when the model becomes excessively sensitive to the training data, decreasing the ability to generalize to new data. Furthermore, XGBoost achieves a favorable equilibrium between precision and memory usage, frequently outperforming alternative boosting algorithms while avoiding excessive demands on memory resources. Due to its various benefits, XGBoost has established itself as a prominent and extensively utilized gradient-boosting algorithm in machine learning.

In the RF-DNN model, a forest is a collection of decision trees.

$$\mathcal{F}(\Theta) = \{\ell_m(\Theta_m)\}, m = 1, \dots, M \quad (3)$$

The parameters in F can be represented by Θ = {Θ1, …, ΘM}, where M denotes the total number of trees in the forest. The parameters in random forests encompass the splitting variables and their corresponding splitting values. During the feature detection stage, the training data X and y are utilized to fit F. X ∈ R n×p represents the input data matrix with n samples and p features, while y ∈Rn denotes the outcome vector containing classification labels. The prediction for each observation xi, where I ranges from 1 to n, can be obtained from every tree in the fitted forest F.

$$f(x_i: \Theta) = \big(T_1(x_i: \Theta_1)\big), \dots, T_M\big((x_i: \Theta_M)\big)^T \quad (4)$$

The present study determines the location of an observation xi, where Tm i (; x Θ = m) yˆim represents the binary prediction. The binary prediction is derived from the forest's signal detection and is denoted by □m. The forest signal's binary vector is represented as fi:=f(xi;Θ) for simplicity. Then it becomes DNN input features. The DNN with l hidden layers uses the forest's new feature representations to follow a conventional design.

$$Pr(y|F, \Psi) = g(Z_{out}W_{out} + b_{out}) \quad (5)$$

$$Z_{out} = \sigma(Z_l W_l + b_l) \quad (6)$$

$$Z_{k+1} = \sigma(Z_k W_k + b_k) \quad (7)$$

$$Z_l = \sigma(FW_{in} + b_{in}) \quad (8)$$

The forest matrix F = (f_i, …, f_M)T contains n samples and M tree predictions. The classification output vector is y, and the DNN model parameters are. Hidden neurons Z_{out} and Z_k, where k = 1,..., l-1, have weight matrices W_{out}, W_k and bias vectors b_{out}, b_k. The input

dimension M, number of classes h_{out}, and number of hidden neurons h_{in} and h_k, where k is an integer from 1 to l, determine Z and W. In the context of binary classification tasks, it is commonly observed that $h_{out} \equiv 2$, owing to the binary nature of the elements in $\boldsymbol{y}$. Typically, the quantity of concealed neurons diminishes descending from the input layer, precisely $h_{in} = M > h_1 > h_2 \ldots > h_{out}$. The activation function $\sigma(\cdot)$ is commonly utilized in neural networks and can take on various forms such as sigmoid, hyperbolic tangent, or rectifiers. The function $g(\cdot)$ is a softmax function that transforms output layer values into probability predictions.

$$P_i = g(\mu_{i1}) = \frac{e^{\mu_{i1}}}{e^{\mu_{i0}} + e^{\mu_{i1}}} \tag{9}$$

where

$$P_i := P_r(y_i = 1 | f_i) \tag{10}$$

$$\mu_{i0} := \left[x_i^{(out)}\right]^T w_0^{(out)} + b_i^{(out)} \tag{11}$$

$$\mu_{i1} := \left[x_i^{(out)}\right]^T w_1^{(out)} + b_i^{(out)} \tag{12}$$

where $i = 1, \ldots, n$.

The weights and biases are the parameters that require estimation in the DNN. The model's training can be accomplished by utilizing a stochastic gradient descent (SGD) algorithm, which operates by minimizing the cross-entropy loss function.

$$\mathcal{L}(\Psi) = -\frac{1}{n}\sum_{i=1}^{n}\{y_i log(\hat{p}_i) + (1 - y_i) log(1 - \hat{p}_i)\} \tag{13}$$

where again Ψ denotes all the model parameters, and p^i is the fitted value of p_i. The entire architecture of the RF-DNN model is visualized in Figure 5.

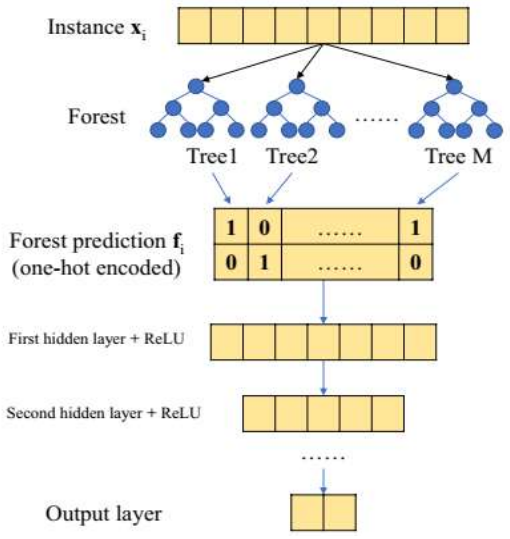

Figure 5: Representation of the hybrid RF-DNN model's architecture

In Figure 5, the forest prediction feature *fi* is one-hot encoded. Since the DNN output dimension is two, this encoding is similar to label vectors *yi*. This implementation's DNN input is a $n \times M \times 2$ tensor substituted for a matrix F. Rectified linear unit (ReLU) is employed as the activation function in the DNN model.

$$\sigma_{ReLU}(x) = max(x, 0) \tag{14}$$

This particular activation function confers a notable benefit compared to the sigmoid and hyperbolic tangent functions, as it effectively circumvents the issue of the vanishing gradient problem that may arise during the optimization process regarding the optimization algorithm.

4. Results and discussion

In this experiment is done on Intel™ Core i5 processor, with a CPU clock speed of 3.2GHz, attached with 8GB of RAM. The feature extraction techniques are compatible with the Windows operating system and employ a programming language such as Python. To ascertain the accuracy and efficacy of the proposed framework, a machine learning and deep learning algorithms were used for the training and testing processes. Python played an instrumental role as the primary language for implementing the experiments. The dataset was strategically assigned, with 70% kept for training, and 30% given to testing. This dataset represents the model of expected behavior, serving as the benchmark against which the anomaly detection system compares new data to separate anomalies.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{15}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{16}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{17}$$

$$\text{F1} - \text{measure} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{18}$$

Table 1 presents the results of the precision metrics, recall, accuracy, and F1 score. The data mentioned above is derived from the confusion matrix of a model that incorporates a classifier. The random forest and logistic regression models exhibited the least accuracy among all the observed models. The Hybrid (RF-DNN) classifier is cultivated in a two-pronged approach. During the inaugural stage, the training data, replete with labels, is employed to groom the forest. Subsequently, predictions emanating from each arboreal constituent within the forest for every instance are channeled into a fully-connected Deep Neural Network (DNN) for the second stage of training. Upon the culmination of this dual-stage training regimen, when faced with a testing instance, the prediction is computed via an end-to-end traverse through the model, harnessing the synergies of both the meticulously fitted forest and the DNN. This

holistic approach ensures that the classifier benefits from the combined strengths of random forests and deep neural networks.

Table 1: Performance comparison of the algorithms (with LRC, DTC, GBC, XGB, and Hybrid (RF-DNN).

Method	Accuracy	Precision	Recall	F1-score
LRC	0.64	0.64	0.70	0.59
DTC	0..95	0.93	0.93	0.93
GBC	0.96	0.92	0.96	0.94
XGB	0.90	0.88	0.87	0.87
Hybrid (RF-DNN)	**0.97**	**0.95**	**0.96**	**0.96**

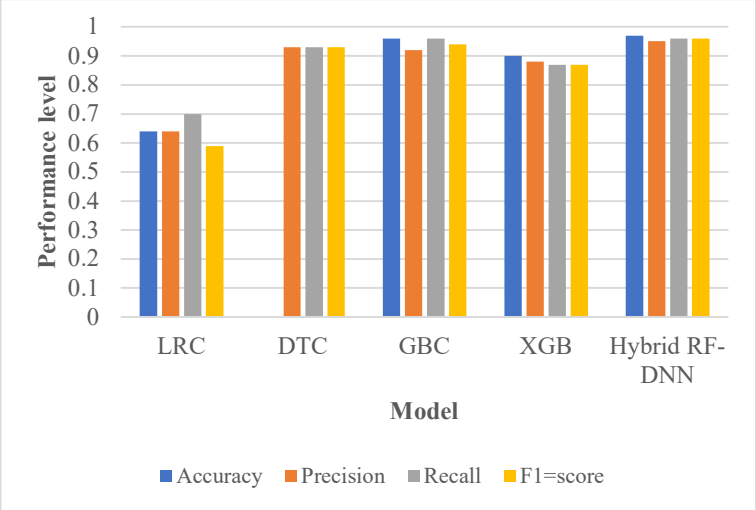

Figure 6: Performance of the proposed algorithm with traditional algorithms

ROC of different models by different classifiers

During the training phase, model induction was carried out, and significant performance metrics such as accuracy, recall, precision, and the F1 score were evaluated. The efficacy of each model was assessed through the utilization of the Receiver Operating Characteristic (ROC) curve, a graphical representation employed to evaluate the performance of a binary classification model. The Receiver Operating Characteristic (ROC) curve is a visual depiction that illustrates the correlation between the True Positive Rate (TPR) and the False Positive Rate (FPR) concerning the fluctuation of the classification threshold. A threshold is a demarcation point for discerning whether a specimen should be categorized as affirmative or negative. Adjusting the threshold makes it feasible to balance the true positive rate (TPR) and the false positive rate (FPR). The ROC curve provides a clear depiction of the relationship between the performance of the model and the modification of the threshold. The suggested model exhibited exceptional classification capabilities, surpassing the Decision Tree (DT) algorithm with a documented accuracy rate of 93.46%. The analyzed model exhibited the inherent trade-offs between the true positive rate (TPR) and the false positive rate (FPR). The findings suggest

that the model exhibited superior performance compared to the DT algorithm concerning TPR, attaining a more significant percentage of 93.46% while preserving an equal FPR. The results indicate that the model under consideration, exhibiting a precision rate of 96.21%, demonstrated superior efficacy in identifying instances of fraudulent transactions.

5. CONCLUSION

In this article, we trained and predicted using all the above models and selected a random forest as the final model. It performed well compared to other models, with an accuracy of 97%. According to this model, our predicted value matches the target values. We can see that in the confusion matrix. We have performed EDA, and preprocessing built different models, visualized feature importance, hyperparameter tuning, and made predictions. We also perform necessary operations to handle the imbalanced and skewed nature of data 'ERC20 uniq rec token name' and 'time diff between first and last (min).' These two are the essential feature. This paper demonstrates how machine learning algorithms can detect fraudulent transactions in the Ethereum blockchain. The work showcases how to preprocess the dataset and extract transaction features. Blockchain developers and financial institutions can use the results of this analysis to prevent fraudulent activity and improve the security of the blockchain.

REFERENCES

[1] Nakamoto, S., 2008. [online] Bitcoin.org. Available at https://bitcoin.org/bitcoin.pdf.

[2] Namecoin.org. 2014. Namecoin. [online] Available at https://www.namecoin.org.

[3] Peters, G., Panayi, E. and Chapelle, A. "Trends in Crypto-Currencies and Blockchain Technologies: A Monetary Theory and Regulation Perspective." Journal of Financial Perspectives, Vol. 3, No. 3, 2015.

[4] A. Bogner. "Seeing is understanding: Anomaly detection in blockchains with visualized features." In Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers, UbiComp '17, pages 5–8. ACM, 2017.

[5] Chen, Xuhui & Ji, Jinlong & Luo, Changqing & Liao, Weixian & Li, Pan. "When Machine Learning Meets Blockchain: A Decentralized, Privacy-preserving and Secure Design." In the 2018 IEEE International Conference on Big Data (Big Data), 2018.

[6] Aziz, R.M. Cuckoo search-based optimization for cancer classification: A new hybrid approach. J. Comput. Biol. 2022, 29, 565–584.

[7] Panarello, A.; Tapas, N.; Merlino, G.; Longo, F.; Puliafito, A. Blockchain and IoT integration: A systematic survey. Sensors 2018, 18, 2575.

[8] Aziz, R.M.; Baluch, M.F.; Patel, S.; Kumar, P. A Machine Learning based Approach to Detect the Ethereum Fraud Transactions with Limited Attributes. Karbala Int. J. Mod. Sci. 2022, 8, 139–151.

[9] Brauneis, A.; Mestel, R.; Theissen, E. What drives the liquidity of cryptocurrencies? A long-term analysis. Financ. Res. Lett. 2021, 39, 101537.

[10] Chow SS, Choo KKR, Han J (2021) Editorial for accountability and privacy issues in blockchain and cryptocurrency. Futur Gener Comput Syst 114:647–648

[11] Anthony, Njoku & Shafik, Mahmoud & Kurugollu, Fatih & Atlam, Hany. (2022). Anomaly Detection System for Ethereum Blockchain Using Machine Learning. 10.3233/ATDE220608.

[12] Rabieinejad, Elnaz & Yazdinejad, Abbas & Parizi, Reza. (2021). A Deep Learning Model for Threat Hunting in Ethereum Blockchain. 10.1109/TrustCom53373.2021.00160.

[13] J, Lohith & K, Anusree & P, Guru & Srinivasan, Pooja. (2023). TP-Detect: trigram-pixel based vulnerability detection for Ethereum smart contracts. Multimedia Tools and Applications. 1-15. 10.1007/s11042-023-15042-4.

[14] Lashkari, Bahareh & Musilek, Petr. (2023). Detection and Analysis of Ethereum Energy Smart Contracts. Applied Sciences. 13. 6027. 10.3390/app13106027.

[15] Agarwal, Rachit & Thapliyal, Tanmay & Shukla, Sandeep. (2022). Analyzing Malicious Activities and Detecting Adversarial Behavior in Cryptocurrency based Permissionless Blockchains: An Ethereum Usecase. Distributed Ledger Technologies: Research and Practice. 1. 1-21. 10.1145/3549527.

[16] V. Aliyev, "Ethereum fraud detection dataset," Jan 2021. Available: https://www.kaggle.com/datasets/vagifa/ethereumfrauddetection-dataset/

[17] Hilbe JM (2016) Practical guide to logistic regression. crc Press; 2016 Apr 5.

[18] Koc¸ak B, Durmaz ES¸, Ates¸ E, Kılıc¸kesmez O¨ (2019) Radiomics with artificial intelligence: a practical guide for beginners. Diagn Interv Radiol 25(6):485

[19] Sreejith S, Rahul S, Jisha R (2016) A real time patient monitoring system for heart disease prediction using random forest algorithm. In: Advances in signal processing and intelligent recognition systems: Springer, pp 485–500

[20] Aziz R, Verma C, Srivastava N (2016) A fuzzy based feature selection from independent component subspace for machine learning classification of microarray data. Genomics Data 8:4–15

[21] Musheer RA, Verma C, Srivastava NJSC (2019) Novel machine learning approach for classification of high-dimensional microarray data. Soft Comput 23(24):13409–13421

[22] Ahamed BS, Arya S (2021) LGBM classifier based technique for predicting Type-2 Diabetes. Eur J Mol Clin Med 8(3):454–467

[23] Aziz R, Verma CK, Srivastava N (2018) Artificial neural network classification of high dimensional data with novel optimization approach of dimension reduction. Ann Data Sci 5(4):615–635

[24] Ahamed BS (2021) Prediction of Type-2 Diabetes using the LGBM classifier methods and techniques. Turk J Comput Math Educ (TURCOMAT) 12(12):223–231

[25] Fan S, Shuhui SF, Haoran X, Xiaochun C (2021) Al-SPSD: antileakage smart Ponzi schemes detection in blockchain. Inf Process Manag 58(4):102587

Chen W, Cui J, Guo X, Chen Z, Lu Y (2021) Misbehavior Detection on Blockchain Data. In: Blockchain Intelligence: Springer, New York, 95–133.