Ethereum Fraud Detection

Kavish Narula

Dept. of Computer Science

Lakehead University Thunder bay, Canada

knarula@lakeheadu.ca

**Abstract—** Blockchain is a technology that serves as a foundation for cryptocurrency applications such as Bitcoin and Ethereum. The blockchain's goal is to eliminate the need for third-party trusted intermediaries like banks. In recent years, due to the qualities of this technology, such as immutability and transparency, it has been used in a variety of areas, including education, healthcare, banking, energy, government, and IoT, to provide better privacy, speedier transactions, and security. We analyzed fraudulent accounts on the Ethereum blockchain in this study and suggested a fraud detection strategy based on three machine learning algorithms: Logistic Regression, Random Forest, and XGB Classifier. These algorithms were applied on a data set obtained from Kaggle.com containing 51 features. We have used the correlation coefficient to select the most effective features and built a new data set using 17 features only. Experimented results shows the best approach is by Random Forest algorithm in the form of confusion matrix.

**Index Terms—**Blockchain, Ethereum, Fraudulent and Non-Fraud accounts, models, Logistic Regression(LR), Random Forest Classifier (RF), XGB, dataset, Ether, Ponzi scheme.

I. INTRODUCTION

The popularity of blockchain-based currencies, such as Bitcoin and Ethereum, has grown among enthusiasts since 2009. Satoshi Nakamoto described bitcoin for the first time in 2009[1]. Since then, digital currency has grown at a breakneck pace. Due to its characteristics such as decentration, openness, information unforgeability and non-tampering, and autonomy, blockchain technology is widely used in various industries such as financial services, medical treatment and health, intellectual property (IP) copyrights, Internet of Things, communications, social management, charity, and entertainment. Blockchain is a technology that allows us to create decentralized records. It is a distributed database that is shared among the nodes of a computer network[2]. As a database, a blockchain stores information electronically in digital format. Blockchains are best known for their crucial role in cryptocurrency systems, such as Bitcoin, Ethereum, LiteCoin, DodgeCoin, Cardano, etc, for maintaining a secure and decentralized record of transactions. As blockchain stores information digitally, it allows digital information to be recorded and distributed, but not edited. In this way, a blockchain is the foundation for immutable ledgers, or records of transactions that cannot be altered, deleted, or destroyed[3]. Because of the decentralized nature of Bitcoin’s blockchain, all transactions can be transparently viewed by either having a personal node or using blockchain explorers that allow anyone to see transactions occurring live. Each node has its own copy of the chain that gets updated as fresh blocks are confirmed and added. This means that if you wanted to, you could track Bitcoin wherever it goes.

Hustlers have adapted offline schemes to this new ecology, relying on the anonymity given by the blockchain. As a result, Ponzi schemes masquerading as secure investment schemes have proliferated on Ethereum[2]. Ethereum, as a currency, is utilized to exchange value worldwide in the absence of a third party to monitor or intervene. However, as online commerce grows, a slew of fraudulent activities, such as money laundering, bribery, and phishing, emerge as the primary threat to trade security. They are generating thousands of victims on Ethereum while stealing millions of dollars in ether, which is illegal in the real world.

According to a 2015 empirical investigation[1], fraudulent transactions account for 60% of total transaction volume in USD and are expected to rise even further. Successful Ponzi schemes have a small number of customers who make large amounts, making them particularly unfortunate victims. Smart Ponzi, the Ponzi scheme smart contract, takes advantage of the blockchain's transparency as a display of faith to attract naive customers who are unable to spot the deception and weaknesses in the implementation.

Ethereum frauds are not like regular cyber frauds in that they have a distinct goal. The Ethereum fraudsters' objective is no longer cash, but the Ethereum token, specifically the Ether. Fraudsters utilise Ethereum's public key to mask their true identities by transferring Ether. Secondly, Ethereum frauds differ from classic cyber frauds in a number of ways. The majority of classic cyber thefts include the use of sensitive data such as phone numbers, e-mail addresses, and bank accounts. As a result, typical methods for identifying cyber frauds detect lawbreakers' phone numbers, e-mail addresses, and other information, and then supply the lawbreakers' photographs to determine the lawbreakers' identities. Extreme data imbalance is one of the biggest obstacles for phishing detection on Ethereum[4]. It is very important to identify these type of accounts so that the sender can know that the account they are sending the crypto to might be a fraud.

II. LITERATURE REVIEW

In [1], they worked on classification to detect Ponzi scheme smart contracts on Ethereum. They used a web scrapper to collect 3,203 addresses of smart contracts with source codes verified by Etherscan, involved in at least 10 transactions, while trying to match the time frame of Ponzi scheme. Three different learning strategize were implemented : J48, RF, Stochastic Gradient Descent (SGD). Models were evaluated through three metrics: precession, recall and F-score. J48 shows the best recall of 0.872, while SGD resulted in better precision of 1.00.

In [2], they proposed the method for detecting Ethereum frauds by mining Ethereum-based transaction records. Specifically, web crawlers are used to capture labeled fraudulent addresses, and then a transaction network is reconstructed based on the public transaction book. At last, the graph convolutional network model is used to classify addresses into legal addresses and fraudulent addresses. They managed to achieve the accuracy of 95% of detecting fraudulent transactions.

In [5], Bartoletti et al. analyzed Ponzi scheme smart contracts' behavior on Ethereum, using similarities between contracts bytecode to locate 191 of them. In total, the contracts collected almost half a million USD from more than 2000 distinct users. [5]highlights characteristics of Ponzi schemes behavior such as a high Gini coefficient, a measure of inequality in the distribution of money made by contracts to investors.

In [6], they investigated Illicit accounts on Ethereum blockchain and proposed a Fraud detection model using three different machine learning algorithms: decision tree (j48), Random Forest and K-nearest neighbors (KNN). These algorithms were applied on a data set obtained from Kaggle.com containing 42 features. We have used the correlation coefficient to select the most effective features and built a new data set using 6 features only. Our research results show a significant improvement in time measurements using the three algorithms and an improvement in the F measure using the Random Forest algorithm.

III. DATASET

We will be using “Ethereum Fraud Detection Dataset” by Vagif Aliyev [7].

There are millions of accounts and billions of transactions in the Ethereum network, but there are only few accounts that are labeled (as fraud or not). This dataset contains rows of known fraud and valid transactions made over Ethereum, a type of cryptocurrency.

There are total of 9841 accounts in which 7662 accounts as not fraud and 2179 accounts as fraud, which is small and imbalanced.

Description of the rows of the dataset :

* Index: the index number of a row
* Address: the address of the ethereum account
* FLAG: whether the transaction is fraud or not
* Avg min between sent tnx: Average time between sent transactions for account in minutes
* Avgminbetweenreceivedtnx: Average time between received transactions for account in minutes
* TimeDiffbetweenfirstand\_last(Mins): Time difference between the first and last transaction
* Sent\_tnx: Total number of sent normal transactions
* Received\_tnx: Total number of received normal transactions
* NumberofCreated\_Contracts: Total Number of created contract transactions
* UniqueReceivedFrom\_Addresses: Total Unique addresses from which account received transactions
* UniqueSentTo\_Addresses20: Total Unique addresses from which account sent transactions
* MinValueReceived: Minimum value in Ether ever received
* MaxValueReceived: Maximum value in Ether ever received
* AvgValueReceived5Average value in Ether ever received
* MinValSent: Minimum value of Ether ever sent
* MaxValSent: Maximum value of Ether ever sent
* AvgValSent: Average value of Ether ever sent
* MinValueSentToContract: Minimum value of Ether sent to a contract
* MaxValueSentToContract: Maximum value of Ether sent to a contract
* AvgValueSentToContract: Average value of Ether sent to contracts
* TotalTransactions(IncludingTnxtoCreate\_Contract): Total number of transactions
* TotalEtherSent:Total Ether sent for account address
* TotalEtherReceived: Total Ether received for account address
* TotalEtherSent\_Contracts: Total Ether sent to Contract addresses
* TotalEtherBalance: Total Ether Balance following enacted transactions
* TotalERC20Tnxs: Total number of ERC20 token transfer transactions
* ERC20TotalEther\_Received: Total ERC20 token received transactions in Ether
* ERC20TotalEther\_Sent: Total ERC20token sent transactions in Ether
* ERC20TotalEtherSentContract: Total ERC20 token transfer to other contracts in Ether
* ERC20UniqSent\_Addr: Number of ERC20 token transactions sent to Unique account addresses
* ERC20UniqRec\_Addr: Number of ERC20 token transactions received from Unique addresses
* ERC20UniqRecContractAddr: Number of ERC20token transactions received from Unique contract addresses
* ERC20AvgTimeBetweenSent\_Tnx: Average time between ERC20 token sent transactions in minutes
* ERC20AvgTimeBetweenRec\_Tnx: Average time between ERC20 token received transactions in minutes
* ERC20AvgTimeBetweenContract\_Tnx: Average time ERC20 token between sent token transactions
* ERC20MinVal\_Rec: Minimum value in Ether received from ERC20 token transactions for account
* ERC20MaxVal\_Rec: Maximum value in Ether received from ERC20 token transactions for account
* ERC20AvgVal\_Rec: Average value in Ether received from ERC20 token transactions for account
* ERC20MinVal\_Sent: Minimum value in Ether sent from ERC20 token transactions for account
* ERC20MaxVal\_Sent: Maximum value in Ether sent from ERC20 token transactions for account
* ERC20AvgVal\_Sent: Average value in Ether sent from ERC20 token transactions for account
* ERC20UniqSentTokenName: Number of Unique ERC20 tokens transferred
* ERC20UniqRecTokenName: Number of Unique ERC20 tokens received
* ERC20MostSentTokenType: Most sent token for account via ERC20 transaction
* ERC20MostRecTokenType: Most received token for account via ERC20 transactions

IV. METHODOLOGY

In this section we represent an overview of the algorithms that we used in our proposed model and selected features in our experiment, as we will go through data collection, feature selection and validation.

We first inspected target distribution of the data differentiating between non-fraud accounts and fraudulent accounts in which we used Flag – 0 for non-fraud accounts and Flag – 1 for fraud accounts and found out that there were total of 8981 accounts in which 7631 accounts were non-fraud accounts and 1350 were fraudulent accounts as shown in Fig.1.

Chart, pie chart

Description automatically generated

Fig 1. Target Distribution differentiating between Non-Fraud and Fraud accounts

The dataset was imbalanced and has missing values, also dataset was highly skewed. So to overcome these problems :

*PRE-PROCESSING*

* + Given that the dataset lists all transactions that took place during the period under study, it is noted that some nodes were involved only in either sending or receiving Ethereum.
  + This led to the existence of missing values in the ﬁnal dataset and hence imputation was in this regard exercised.
  + All the missing values were imputed with zeroes based on the premise of equivalence to sending or receiving 0 ETH.
  + To have appropriate metrics between instances in our multivariate environment, we opted to transform our data by centring around mean zero and unit variance.

*FEATURE SELECTION*

* + After replacing missing values with median and assigned 0 variance, we Dropped features with variance 0. These features will not help in the performance of the model. After dropping features with 0 variance, we depicted corelation matrix between the features as shown in Fig 2.

Chart

Description automatically generated

Fig. 2 Corelation Matrix between features.

* + Also we dropped features with highly correlated features as shown in Fig 3.

Chart

Description automatically generatedFig 3. Corelation Matrix after dropping highly correlated features.

* + Out of 51 features we selected 17 features for training and testing . These features helped in trained the model: Flag, Avg min between sent tnx, Avg min between received tnx, Time Diff between first and last (Mins), Sent tnx, Received Tnx, Number of Created Contracts, max value received, avg val received, avg val sent, total Ether sent, total ether balance, ERC20 total Ether received, ERC20 total ether sent, ERC20 total Ether sent contract, ERC20 uniq sent addr, ERC20 uniq rec token name.

*CLASS IMBALANCE*

As the dataset is highly imbalanced which could lead us to create an algorithm resulting in predicting only 1 type of FLAG. Thus, we have to try different strategies as a solution to imbalance class problem. Here are the few techniques that can be used for such a problem:

* Try Changing Performance Metric
* Try Resampling Dataset
* Try Generating Synthetic Samples
* Try Different Algorithms

So we approached the problem with resampling the dataset. As the data is imbalanced, it can lead to either Undersampling or Oversampling as shown in Fig.4.

Chart, bar chart, waterfall chart

Description automatically generated

Fig 4. Example of Undersampling and Oversampling

Imbalanced classification involves developing predictive models on classification datasets that have a severe class imbalance which turn to have poor performance on, the minority class, although typically it is performance on the minority class that is most important. One approach to addressing imbalanced datasets is to oversample the minority class. So to overcome this problem we used Synthetic Minority Oversampling Technique (SMOTE) method. This is a type of data augmentation for the minority class to oversample the examples in the minority class. It is very effective for class imbalance.

We split the dataset into training (80%) and testing (20%) and used SMOTE method on training data. We noticed the result of oversampling before and after implementation of SMOTE method as shown in Fig 5.

Graphical user interface, text, application, chat or text message

Description automatically generated

Fig 5 . Before and After oversampling with SMOTE method

As observed it shows a huge difference between Fraudulent accounts before and after implementing SMOTE method on training data.

*MODELING*

Now that we resolved issue with imbalance data we can move on to modeling the data and see that how accurate our model performed to detect the fraud and non-fraud accounts.

We implemented three different learning methods :

* + **Logistic Regression (LR)**

LR is used when the dependent variable(target) is categorical. For ex. to predict the output as 0 or 1. The categorical response has only two 2 possible outcomes. Example: Spam or Not. The logistic model is a statistical model that models the probability of one event taking place by having the log-odds for the event be a linear combination of one or more independent variables [8]. In regression analysis, logistic regression is estimating the parameters of a logistic model.

Hypothesis => Z = WX + B

hΘ(x) = sigmoid (Z)

where, if ‘Z’ goes to infinity, Y(predicted) will become 1 and if ‘Z’ goes to negative infinity, Y(predicted) will become 0.

* + **Random Forest Classifier (RF)**

RF is a meta estimator that fits a number of decision

tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max\_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree. It consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction [9] as shown in Fig 6.

Diagram

Description automatically generated with medium confidence

Fig 6 Random Forest

* + **XGB Classifier**

XGBoost is an optimized distributed gradient boosting library designed highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way [10].

V. RESULTS

In this section we will show about the experimented evaluation of the three models comparing with the confusion matrix.

As we split the dataset into training and testing set. We trained the model with 80% and tested with 20% of the set. In total there were 8981 accounts in which the test set contains 1797 accounts., out of which 1550 were non-fraudulent accounts marked as Flag - 0 and 247 were fraud accounts marked as Flag – 1.

To evaluate the result a confusion matrix is the best way to show the correct and incorrect predictions for each class. Higher the diagonal value from top left to bottom right better is the prediction.

For the logistic regression as shown in Fig. 7 predicts 1342 as correctly predicted accounts out of 1550 which is very low. 219 accounts were predicted correctly as fraud accounts whereas unable to identify 28 accounts as fraud accounts. Also 208 accounts were predicted wrong as false positive which is highly inaccurate.

Chart, treemap chart

Description automatically generated

Fig 7. Logistic Regression result

For RF, as shown in Fig. 8 predicts 1523 as correctly predicted accounts out of 1550 which is better than LR. 225 accounts were predicted correctly as fraud accounts whereas unable to identify 22 accounts as fraud accounts. Also 27 accounts were predicted wrong as false positive which shows very effective results than LR.

Chart, treemap chart

Description automatically generated

Fig 8. Random Forest result

As the final model XGBoost Classifier is not much different from RF confusion matrix but still unable to identify 65 true positive accounts as shown in Fig. 9.

Chart, treemap chart

Description automatically generated

Fig 9. XGB Classifier result

Random forest classifier shows the best result among three models with a little difference with the XGB Classifier. LR depicts the worst result among the three models.

So to visualize the performance of the binary classifier we used Receiver Operating Characteristic (ROC) plot which gives us the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) at different classification thresholds for Random Forest classifier as shown in Fig 10, in which it shows AUC is 0.99 which is very close to 1.0, hence depicting the best result among other three models.

Line chart

Description automatically generated

Fig 10. ROC plot for tuned RF classifier

VI. CONCLUSION

We wanted to make a classification, and to understand which Ethereum transaction is a fraud, and which is not. We did EDA, we chose which attributes we should use to predict the model, and which features we should avoid. We had to deal with outliers in the data set, and we had to be aware of correlation issues, that can make us decide to avoid some features, that are highly correlated with others. Then we used 3 different models – Logistic Regression, Random Forest and XGB Classifier. The Random Forest Model did the best work for us. to conclude, we think that we answer the question well, and we could predict fraudulent transactions. Important factors for fraudulent behaviour identification are total value in ETH sent by a wallet, the average value in ETH sent by a wallet, the average time between outgoing transactions, the standard deviation of time between outgoing transactions and the frequency of outgoing transactions. Also accuracy is not the best metric to use when evaluating imbalanced datasets as it can be very misleading.

For future work, we suggest adding the volatility of Ethereum to the consideration, for example in times there is the high volatility of the crypto, it may be the time that the fraudulent transactions are increasing. Experimenting with the different models like XGB hyperparameters tuning for XGB Classifier, different normalization techniques. Also generating synthetic samples can increase the efficiency of the model. Data augmentation could be introduced to increase the complexity of dataset. Also, more time could be used to test different hyperparameters. Testing with the larger dataset could give more insights with efficient feature selection and resampling technique can predict good results.

VII. REFERENCES

[1] E. Jung, M. Le Tilly, A. Gehani, and Y. Ge, “Data Mining-Based Ethereum Fraud Detection,” in 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, Jul. 2019, pp. 266–273. doi: 10.1109/Blockchain.2019.00042.

[2] Runnan Tan, Qingfeng Tan, Peng Zhang, Zhao Li, “Graph Neural Network for Ethereum Fraud Detection”, 2021 IEEE International Conference on Big Knowledge (ICBK).

[3] “What is Blockchain?”, By Adam Hayes, Reviewed by Jefereda R. Brown Fact checked by Suzanne Kvilhaug, Updated March 05, 2022. <https://www.investopedia.com/terms/b/blockchain.asp>

[4] J. Wu et al., “Who Are the Phishers? Phishing Scam Detection on Ethereum via Network Embedding,” IEEE Trans. Syst. Man Cybern, Syst., vol. 52, no. 2, pp. 1156–1166, Feb. 2022, doi: 10.1109/TSMC.2020.3016821.

[5] M. Bartoletti, S. Carta, T. Cimoli and R. Saia, “Dissecting ponzi schemes on ethereum: identification analysis and impact”, 2017, [online] Available: <http://arxiv.org/abs/1703.03779>.

[6] Rahmeh Fawaz Ibrahim, Aseel Mohammad Elian, Mohammed Ababneh, ” Illicit Account Detection in the Ethereum Blockchain Using Machine Learning”, Published in: 2021 International Conference on Information Technology (ICIT), 14 July 2021.

[7] Vagif Aliyev, “Ethereum Fraud Detection”, 2021. <https://www.kaggle.com/vagifa/ethereum-frauddetection-dataset>

[8] Saishruthi Swaminathan, “Logistic Regression – Detailed Overview”, 15 March 2018.

https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc

[9] Tony Yiu, ”Understanding Random Forest”, 12 June 2019.

https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[10] [Online] Available: https://xgboost.readthedocs.io/en/stable/