Analyzing Transaction Confirmation in Ethereum Using Machine Learning Techniques

Vinicius C. Oliveira
UFJF - DCC, Brazil
vinicius.oliveira@ice.ufjf.br

Júlia Almeida Valadares
UFJF - DCC, Brazil
juliavaladares@ice.ufjf.br

José Eduardo A. Sousa
UFJF - DCC, Brazil
eduardo2@ice.ufjf.br

Alex Borges Vieira
UFJF - DCC, Brazil
alex.borges@ufjf.edu.br

Heder Soares Bernardino
UFJF - DCC, Brazil
heder@ice.ufjf.br

Saulo Moraes Villela
UFJF - DCC, Brazil
saulo.moraes@ufjf.edu.br

Glauber Dias Gonçalves
UFPI - CSHNB, Brazil
ggoncalves@ufpi.edu.br

ABSTRACT

Ethereum has emerged as one of the most important cryptocurrencies in terms of the number of transactions. Given the recent growth of Ethereum, the cryptocurrency community and researchers are interested in understanding the Ethereum transactions behavior. In this work, we investigate a key aspect of Ethereum: the prediction of a transaction confirmation or failure based on its features. This is a challenging issue due to the small, but still relevant, fraction of failures in millions of recorded transactions and the complexity of the distributed mechanism to execute transactions in Ethereum. To conduct this investigation, we train machine learning models for this prediction, taking into consideration carefully balanced sets of confirmed and failed transactions. The results show high-performance models for classification of transactions with the best values of F_1-score and area under the ROC curve approximately equal to 0.67 and 0.87, respectively. Also, we identified the gas used as the most relevant feature for the prediction.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Ethereum, blockchain, machine learning, transaction

1. INTRODUCTION

Ethereum [4,15] has emerged as a popular cryptocurrency. It has become one of the largest cryptocurrency currently in terms of the number of registered financial transactions. Its network has recorded more than 470M transactions (about 9 transactions per second) since February 2020, leading to a market capitalization above 27bn US Dollars[1]. Moreover, Ethereum adds to its financial transactions the capacity to process programmable contracts, namely smart contracts, which has been considered its main advantage compared to Bitcoin, the pioneer cryptocurrency.

In a cryptocurrency system, transactions can be either confirmed or failed after their execution by a distributed consensus mechanism. In this sense, the confirmation of transactions is a critical issue since it can yield losses of users' money in the case of failure, i.e., the fee that is paid to the network operators (miners) to process a transaction is not given back. Additionally, no risks of failure are provided as soon as a transaction enters the network to be processed, thus the occurrence of failures can compromise the experience of users and prevent them from making future transactions. On the other hand, developing models to predict transaction confirmation is challenging due to the small fraction of failures considering millions of recorded transactions and the complexity of the distributed consensus mechanism to execute and validate a transaction in Ethereum [15].

We provide here an investigation of this issue, taking into consideration the features of transactions. We start from the following research question: *How to explore the features of transactions to help users in assessing the final state of their transactions (confirmation or failure)?* We analyze how machine learning models can be used to predict the confirmation of a transaction in Ethereum, and discuss the key features of transactions that impact the transaction final state, thus, providing insights on the complex mechanism behind the Ethereum network for its users. More than two million transactions were collected from the Ethereum network using the most popular service's APIs. We observed that the number of confirmed and failed transactions is unbalanced: about 3% of them failures. As a given class has a considerably higher number of occurrences than the other one, we perform a careful balancing process for properly training the machine learning models. Given such training, we show high-performance models (F_1-score and area under the ROC

[1] `https://etherscan.io`

Symposium on Cryptocurrency Analysis (SOCCA) 2020 Milan,Italy

curve approximately equal to 0.67 and 0.87, respectively) and the most relevant features for failure detection.

Our contributions are as follows: (i) a wide performance analysis of machine learning models to predict the confirmation of transactions in Ethereum, and (ii) a discussion about the key features that impact on such confirmation.

2. RELATED WORK

Machine learning models have been widely applied for predictions in Ethereum. The vast majority of studies, however, focus on the prediction of price trends [2, 6, 9]. All those studies are based on external market features and do not explain the internal mechanism of Ethereum. In fact, [13] is the single research effort align to our work, where models were proposed to predict the time to accept a transaction in addition to explore the impact of unbalanced data on the classifiers. In line with this effort, we investigate another important issue for users: the confirmation of transactions.

Several investigations focus on other important issues of Ethereum as users behavior and its performance by using (un)supervised machine learning models as well as statistical methods. For example, Payette et al. [12], categorize Ethereum transactions from anonymous addresses, using the K-means algorithm. The relationship between the fee users pay for transactions and the confirmation time has been investigated in [14], by mean of correlations coefficients.

A relevant research effort has been dedicated to the security aspects of Ethereum. Li et al. [10] surveyed real attacks on popular cryptocurrencies, from 2009 and an analysis of the vulnerabilities exploited in these cases. Other recent pernicious threats were identified in Ethereum. Xu et al [7], for instance, propose a random forest classification model to distinguish normal from malicious traffic in a type of attack identified as Eclipse. Bartoletti et al. [1] investigate Ponzi schemes, a type of fraud in smart contracts. The authors identified several frauded smart contracts applying statistical methods used to investigate fraud in financial markets.

The performance aspects of programming transactions and smart contracts in Ethereum were analyzed in [3, 5]. An investigation on Solidity, the Ethereum main language, was performed in [3], which revealed faults to optimize costly programming patterns. In turn, Ethereum smart contracts were studied in [5], where authors inspected the source code of more than 10,000 smart contracts and proposed software indicators to analyze the evolution and main characteristics of the developer's community in Ethereum.

Finally, graph analysis has been used to analyze transaction behavior in Ethereum. For example, the Ethereum financial transactions were characterized in [4] as a multi-flow graph. Also, the transactions of Ethereum and other four cryptocurrencies are analyzed as an evolving graph in [11].

3. DATA COLLECTION METHODOLOGY

We rely on Etherchain and the Etherscan[2], which are well-known and widely used APIs to collect Ethereum data. The use of two distinct APIs allows us to retrieve data from the Ethereum network in a faster way through distinct vantage points and also distributing the retrievals between both APIs, given their limitations of requests.

Our crawling methodology presents two main steps. First, we collect transactions just waiting to be processed in the Ethereum *mempool*. We perform this first step via the Etherchain API. Periodically, the network operators, named *miners*, perform the Ethereum consensus protocol to process a block of transactions picked at the mempool, thus executing and validating each transaction before including it in the blockchain. Then, in the second step, we collect transactions from the Ethereum blockchain. In this case, we retrieve information from the current state of the blockchain, by checking newer blocks whether the state of each processed transaction is either failed or confirmed in these blocks. We perform this second step via the Etherscan API.

Table 1: Attributes extracted from Ethereum transactions

Attribute	Description
hash	A unique string identifying each transaction.
gas offered	Amount of gas a user offers (limit) to execute a transaction.
gas used	Amount of gas a used by the EVM to execute a transaction.
gas price	Amount of Ether user offers per gas unit.
value	Amount of Ether transferred between 2 accounts.
pending time	Interval between the observation of a transaction in the pool and its inclusion in the blockchain.

Transactions from the mempool and the blockchain contain a unique hash. Hence we use this hash to uniquely identify and match the data collected both sources. Table 1 presents all attributes we extracted from Ethereum transactions in the current work. Most of them are directly obtained by the aforementioned APIs, whereas the *pending time* is calculated from the difference between the time a transaction is included in the blockchain and the time it is first observed in the mempool. To reduce the effects of inaccurate time synchronization between the two APIs (Etherscan and Etherchain), we discharge negative pending times.

We retrieved 20,857,783 transactions from April 15th to July 26th, 2019. However, we filter out 2,519,711 transactions for our analyses, uniformly distributed along the period, to obtain accurate features, in particular, pending time. Overall, we analyze 2,442,169 confirmed transactions (96.9%) and 77,542 failed transactions (3.1%) that keep the characteristics of all retrieved data. This corresponds to a sample of roughly 3.2% of the number of transactions that Ethereum is able to process within the data period of 103 days, taking into consideration the average of 8.6 transactions per second reported by the Etherscan web service. All data, scripts and crawling tools are available publicly.[3]

4. PROPOSED FRAMEWORK

In this section, we present the framework to generate the models that evaluate Ethereum transactions confirmation. We also describe the data preprocessing, the classification models and, the techniques to handle imbalance data.

A flowchart of the framework can be shown in Figure 1. On its left side, we have the data collection methodology (Section 3). The procedure starts by collecting data from the Ethereum platform, so the transactions are grouped into blocks containing several transactions. Thus, the crawlers (through API Requests) generate the raw data and the most important features are extracted. This first part concludes with data preprocessing: additional features are generated using the original ones (such as pending time) and data is standardized (each feature is normalized in $[0, 1]$).

The right side of the framework contains the steps for creating models from data. The set of machine learning techniques is defined in the first step. Here, Decision Trees

[2]https://www.etherchain.org, https://etherscan.io

[3]`http://netlab.ice.ufjf.br/ethereum`

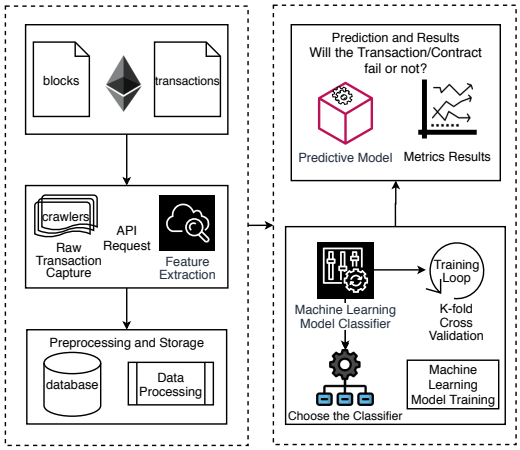

Figure 1: Proposed framework to generate models.

(DT), Random Forest (RF), Logistic Regression (LR), and Support Vector Machine (SVM) were used. DTs are trees formed by if-then-else instructions in the inner nodes and classes in the leaves and are normally generated by a greedy strategy to optimize the splitting of data. Here, the Gini index was used. RF is a set of DTs that work as an ensemble, where the class is inferred by its models. LR is a statistical model that uses a logistic function. SVM finds the hyperplane with the maximal margin to the data of different classes to perform classification. In SVM, kernels transform data and we used (i) linear, (ii) gaussian, and (iii) sigmoid.

The data is unbalanced as only 3.1% of the instances represent failed transactions. Thus, the following balancing strategies were applied to avoid bias: Undersampling, Oversampling, and SMOTE [8]. We also generate the models without using a balancing approach for comparison.

We perform a cross-validation loop for training and validating the models with 10 folds. These models are evaluated using the validation datasets according to accuracy, precision, recall, F_1-score, F_β-score with $\beta = 2$, Matthews Correlation Coefficient (MCC), number of true positives (TP) and negatives (TN), and area under the ROC curve (AUC-ROC). Note that accuracy is not a proper performance metric when data is unbalanced. On the other hand, F_β-score, AUC-ROC, and the *Matthews Correlation Coefficient* (MCC) are proper performance metrics when data is unbalanced. MCC $\in [-1, 1]$, where $+1$ represents a perfect prediction.

One can also analyze the DT and RF models to extract knowledge regarding the decision making process. For instance, the conditions that lead to failure can provide insights from situations that should be avoided.

5. RESULTS

The results obtained in the computational experiments and the models created by the proposed framework are discussed here. We randomly selected 50,000 transactions from the original data (Section 3) to reduce the amount of data analyzed. As in entire data, this sample is highly unbalanced as about 97% of the instances are confirmed transactions. The *positive* class is represented by the transactions that have failed (minority class) and the *negative* ones are those with success (majority class).

The experiments were performed using Scikit-learn[4], a well-known library of machine learning in Python. Also, the unbalanced-learn package[5] was adopted to balance data with SMOTE, undersampling, and oversampling. In all the cases, the default parameters of the libraries were used.

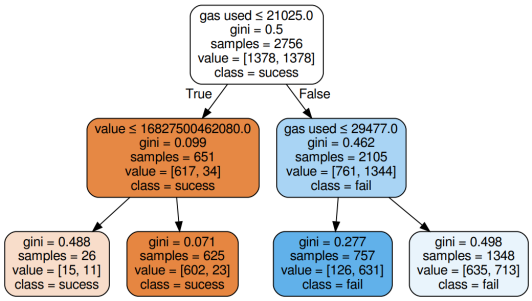

Figure 2: DT created using undersampling.

5.1 Performance Analysis of the Techniques

Table 2 presents the performance results, where "None" groups the cases without a balancing method. Accuracy is not appropriate to evaluate models when data is unbalanced. Indeed, methods without balancing technique has the highest accuracy values but also, in these cases, the other metrics (e.g., precision, recall, F1, etc) assume small values (most of them zero). The only exceptions, when dealing with no balancing method, are the cases for RF or DT which presents high values for accuracy and other metrics. However, for these two RF/DT metrics, we observe better values of AUC-ROC and TP when dealing with the balancing strategies.

RF outperformed the other methods for the tested balancing approaches, with the best AUC-ROC, F_1-score and F_β-score values. DT also achieves good results. On the other hand, the three SVM variants achieve the worst results in general: TP increases but TN decreases largely.

Both balancing methods, i.e, undersampling and SMOTE, provided good results. Despite the SMOTE complexity to handle unbalanced classes, undersampling obtained the best AUC-ROC value and good values of F_1-score and F_β-score.

The best model with respect to AUC-ROC is RF with undersampling. This model has also presented good values for the other metrics considered here. Moreover, DT with undersampling obtained a good performance, compared to the remaining methods. Thus, the importance of the features of these models are analyzed in the following section.

5.2 Analysis of the Most Relevant Features

Now, we investigate the most important features and how they are used in the decision process. Fig. 2 presents a DT obtained when the data is balanced by undersampling. Each DT node contains its condition (associated with one feature), the Gini index (the splitting criterion used here), the number of transactions instances before data splitting into 2 sets (one for each edge), the number of transactions instances of each class, and the most frequent class. The DT edges are the results of the conditions (true or false) and the colors of the node represent its portion of failures (blue) and success (orange). The DT shown in Fig. 2 is able to identify both classes and the gas used is its most important feature. After splitting the data according to the gas used, the feature value can also be considered relevant.

[4]`https://scikit-learn.org/`

[5]https://imbalanced-learn.readthedocs.io/

Table 2: Results obtained when using different machine learning methods and approaches for balancing data.

Balancing Technique	Classifier	Accuracy	Precision	Recall	F_1-score	F_β-score	MCC	TP	TN	AUC-ROC
None	Decision Tree	0.97716	0.617754	0.667755	0.641782	0.657117	0.506899	1023 (66.8%)	47835 (98.7%)	0.827347
	Random Forest	0.98384	0.894336	0.535901	0.670204	0.582600	0.760797	821 (53.6%)	48371 (99.8%)	0.766950
	Logistic Regression	0.96936	0.000000	0.000000	0.000000	0.000000	0.000000	0 (0.0%)	48468 (100%)	0.500000
	Linear SVM	0.96936	0.000000	0.000000	0.000000	0.000000	0.000000	0 (0.0%)	48468 (100%)	0.500000
	Gaussian SVM	0.96936	0.000000	0.000000	0.000000	0.000000	0.000000	0 (0.0%)	48468 (100%)	0.500000
	Sigmoid SVM	0.94328	0.028902	0.026110	0.027435	0.026624	-0.006184	40 (2.6%)	47124 (97.2%)	0.499190
Undersampling	Decision Tree	0.86186	0.162332	0.843342	0.272258	0.458579	0.300507	1292 (84.3%)	41801 (86.2%)	0.852894
	Random Forest	0.88988	0.198848	0.856397	0.322755	0.515480	0.381767	1312 (85.6%)	43182 (89.1%)	0.873668
	Logistic Regression	0.79248	0.054054	0.349869	0.093641	0.167041	0.018869	536 (35.0%)	39088 (80.6%)	0.578170
	Linear SVM	0.88268	0.050228	0.157963	0.076220	0.110543	0.020082	242 (15.8%)	43892 (90.6%)	0.531775
	Gaussian SVM	0.69352	0.060092	0.614883	0.109484	0.216015	0.175745	942 (61.5%)	33734 (69.6%)	0.655444
	Sigmoid SVM	0.58628	0.043561	0.596606	0.081194	0.168572	0.104084	914 (59.7%)	28400 (58.6%)	0.591280
Oversampling	Decision Tree	0.97854	0.663812	0.607050	0.634163	0.617612	0.598223	930 (60.7%)	47997 (99.0%)	0.798666
	Random Forest	0.98272	0.797683	0.584204	0.674454	0.617241	0.702714	895 (58.4%)	48241 (99.5%)	0.789760
	Logistic Regression	0.77120	0.060036	0.441253	0.105691	0.194387	0.130968	676 (44.1%)	37884 (78.2%)	0.611441
	Linear SVM	0.86702	0.048208	0.178198	0.075886	0.115766	0.026896	273 (17.8%)	43078 (88.9%)	0.533496
	Gaussian SVM	0.78474	0.083702	0.605744	0.147080	0.269532	0.208645	928 (60.6%)	38309 (79.0%)	0.698071
	Sigmoid SVM	0.62214	0.049977	0.629243	0.092599	0.189637	0.086424	964 (62.9%)	30143 (62.2%)	0.625579
SMOTE	Decision Tree	0.43134	0.049925	0.973890	0.094980	0.207147	0.130639	1492 (97.4%)	20075 (41.4%)	0.694041
	Random Forest	0.53560	0.057496	0.919713	0.108226	0.229973	0.145633	1409 (92.0%)	25371 (52.3%)	0.721586
	Logistic Regression	0.75716	0.057622	0.451044	0.102189	0.190673	0.023828	691 (45.1%)	37167 (76.7%)	0.608940
	Linear SVM	0.86650	0.048146	0.178851	0.075869	0.115915	0.029470	274 (17.9%)	43051 (88.8%)	0.533543
	Gaussian SVM	0.80234	0.090998	0.606397	0.158249	0.284324	0.217028	929 (60.6%)	39188 (80.9%)	0.707465
	Sigmoid SVM	0.54840	0.040717	0.609008	0.076332	0.160629	0.109145	933 (60.9%)	26487 (54.6%)	0.577746

RF is a committee of DTs and the features in these trees are important for the prediction. In this work, the main attributes in the best RF model are gas used (43.8%), gas offered (30.6%), and gas price (12.5%). The gas used is the most important feature, such as the root of the best DT model (Fig. 2). Also, the gas offered is an interesting feature for the models, as transactions may fail due to out of gas.

6. CONCLUSIONS

In this work, we have introduced the application of machine learning models to analyze confirmation and failures of transactions after processing in the Ethereum current consensus mechanism. These models can be easily trained using features of transactions collected from Ethereum public APIs, and our evaluations show they can reach high accuracy, precision, and recall, i.e., classification of confirmed as well as failed transactions, through the use of data balancing strategies to train machine learning models. Moreover, features related to the Ethereum fee (gas) have higher importance to classify the transaction. For future work, we intend to consider different consensus mechanisms of Ethereum.

Acknowledgements: We acknowledge the partial support of CAPES, CNPq, FAPEMIG.

7. REFERENCES

[1] M. Bartoletti, S. Carta, T. Cimoli, and R. Saia. Dissecting ponzi schemes on ethereum: identification, analysis, and impact. *Future Generation Computer Systems*, 102:259–277, 2020.

[2] M. Chen, N. Narwal, and M. Schultz. Predicting price changes in ethereum. *IJCSE*, 2019.

[3] T. Chen, X. Li, X. Luo, and X. Zhang. Under-optimized smart contracts devour your money. In *IEEE SANER*, 2017.

[4] T. Chen, Z. Li, Y. Zhu, J. Chen, X. Luo, J. C.-S. Lui, X. Lin, and X. Zhang. Understanding ethereum via graph analysis. *ACM Trans. Internet Technol.*, 20(2):1–32, 2020.

[5] A. P. et al. A massive analysis of ethereum smart contracts empirical study and code metrics. *IEEE Access*, 7:78194–78213, 2019.

[6] C. L. et al. Cryptocurrency price prediction using news and social media sentiment. *SMU Data Sci. Rev*, 1(3):1–22, 2017.

[7] G. X. et al. Am I eclipsed? A smart detector of eclipse attacks for Ethereum. *Computers & Security*, 88:101604, 2020.

[8] T. P. et al. Learning imbalanced datasets based on smote and gaussian distribution. *Information Sciences*, 512:1214–1233, 2020.

[9] D. Kumar and S. Rath. Predicting the Trends of Price for Ethereum Using Deep Learning Techniques. In *Proc. of ICAIECES*, pages 103–114. 2020.

[10] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen. A survey on the security of blockchain systems. *Elsevier FGCS*, 2017.

[11] A. P. Motamed and B. Bahrak. Quantitative analysis of cryptocurrencies transaction graph. *Applied Network Science*, 4(1):131, 2019.

[12] J. Payette, S. Schwager, and J. Murphy. Characterizing the Ethereum Address Space, 2017.

[13] H. J. Singh and A. S. Hafid. Prediction of Transaction Confirmation Time in Ethereum Blockchain Using Machine Learning. In *Int. Congress on Blockchain and Applications*, 2019.

[14] J. E. d. A. Sousa, V. Oliveira, J. Valadares, A. B. Vieira, S. M. Villela, H. S. Bernardino, and G. Dias. An Analysis of the Fees and Pending Time Correlation in Ethereum. *Int J Network Mgmt. nem.2113*, 2020.

[15] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.