2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) | 978-1-6654-1658-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/TRUSTCOM53373.2021.00156

Wait or Reset Gas Price?: A Machine Learning-based Prediction Model for Ethereum Transactions' Waiting Time

Akshay M. Fajge*, Subhasish Goswami†, Arpit Srivastava‡ and Raju Halder*

*Indian Institute of Technology Patna, Patna, India
Email: {fajge_1921cs12, halder}@iitp.ac.in
†Tezpur University, Tezpur, India
Email: subhasish_csb18@agnee.tezu.ernet.in
‡Indian Institute of Information Technology Vadodara, Gandhinagar, India
Email: 201851027@iiitvadodara.ac.in

***Abstract*—The gas mechanism on the Ethereum blockchain attempts to set charges for every smart contract operation in order to prevent infinite control of computational resources by the executing transactions. To satisfy this requirement, users need to specify in their transactions how much gas fees (in terms of gas limit and gas price) they would like to pay for. In essence, these gas fees are paid to the miners in return for their computational services. Naturally, miners tend to maximize their profits by considering the transactions with higher gas fees, and as a result, the transactions with lower gas fees remain in the waiting pool for a long time. This paper proposes a machine learning-based approach to predict whether a transaction with offered gas fees is likely to be included in the blockchain within the expected time or not. Such prior prediction of transactions' waiting time definitely assists users to reset their gas fees accordingly. The proposed model is evaluated on nearly one million real transactions from Ethereum mainnet, and the experimental results demonstrate a better performance than the existing one in the literature, with an achievement of 90.18% accuracy and 0.897 F1-score when the model is trained with Random Forest on the dataset balanced with SMOTETomek.**

***Keywords*-Blockchain; Machine Learning; Ethereum; Gas Fees; Transaction Mining; Waiting Time;**

I. Introduction

Post the advent of the Bitcoin by Satoshi Nakamoto in 2008 [1], Blockchain has been positioned as a revolutionary technology with its wide range of applications in various sectors including healthcare, finance, government, real estate and many more [2]. While the bitcoin supports only non-Turing-complete scripting language, Ethereum [3], on the other hand, is fueled by the decentralized execution of smart contracts defined in a Turing-complete programming language. This leads to the development of decentralized applications driven by blockchain technology.

In general, Ethereum transactions involve either cryptocurrency transfer between externally owned account (EOA) or smart contract code executions. Ethereum relies on Ethereum Virtual Machine (EVM) as the executing platform for smart contracts. Its gas mechanism [3] attempts to set charges for every smart contract operation as a way to avoid infinite control of computational resources by the executing transactions. More precisely, gas is the fundamental unit used to quantify the computing effort necessary to perform various operations on the Ethereum network. Users must define the amount of gas fees (in terms of gas limit and gas price) they wish to pay for their transactions. These gas fees are compensated to the miners for their computational services. A gas requirement is fulfilled implicitly from the transaction initiator's account balance at the rate which the transactor is willing to pay for each gas unit. These transactions are broadcast over the network where miners are free to select the transactions for the block they mine. Miners may prefer transactions with a higher gas price than those that offer a lesser gas price to maximize their profits. As a result, transactions involving lesser gas fees may languish in the waiting pool for an extended time. Thus, it becomes critical to determine whether the transaction is likely to be included in the block within the anticipated duration. Prior prediction of transactions' waiting times on the network at each given time will undoubtedly benefit users in adjusting gas prices and limits appropriately to ensure transactions are included in the underlying blockchain on time.

A. Related Work

In recent times, there is a growing interest in exploring how to adopt machine learning to blockchain applications and vice versa [4]–[7]. Given the significant importance of the gas price and gas limit in the execution of Ethereum transactions, researchers are now trying to get their hands on the adoption of machine learning approaches that attempt to predict the gas price, gas cost, and waiting time associated with transactions [6]–[8]. The authors in [7] applied random forest regression and multilayer perceptron to predict confirmation time for transactions on the Ethereum mainnet. Bouraga [8] explored the correlation between the transaction initiator's address and the transaction's gas consumption and provided a model for predicting the users' future transac-

2324-9013/21/$31.00 ©2021 IEEE
DOI 10.1109/TrustCom53373.2021.00156

tion's gas consumption based on their transaction history. Werner et. al. [6] proposed an approach that uses a deep-learning based price forecasting model for recommending the Ethereum gas price. Sousa et. al. [9] studied the correlation of gas fees with transaction waiting time and concluded that transaction features cannot determine the waiting time of transactions.

B. Motivation and Contributions

In general, a miner's objective is to maximize profit by preferring transactions with a higher transaction fee during the block mining process. However, selecting those transactions from the transactions' waiting pool considering the block's upper gas limit is equivalent to the Knapsack Problem, an NP-Complete problem [10]. Given the miners' competitiveness to mine the block with their limited computational power, it makes sense for them to concentrate their computing resources on mining the block rather than finding the most profitable transactions. As a result, it is normal to find blocks on the Ethereum mainnet that contain no transactions. This behavior of miners further increases the waiting time for the transactions. In this direction, the only existing work which we find related to our proposal is [7] where the learning process involves transaction's basic attributes such as gas price and gas limit. Intuitively, they are not enough for the model to learn about transactions' waiting time [9]. In general, various block- and network-specific attributes (such as difficulty level, the mining rate, transaction time) may also affect the block mining process on the Ethereum network. These attributes are significant for predicting the waiting time for transactions on the network. Since miners are free to choose any transactions (or even no transaction) from the waiting pool to include in the blocks they are mining, transaction initiators must offer an attractive gas price and gas limit in order to tempt miners. As previously stated, the purpose of this article is to estimate the waiting time for transactions on the Ethereum blockchain network using the offered gas price and gas limit.

To address these challenges, we propose a machine learning-based model for predicting the waiting time of transactions in the Ethereum mainnet, taking into account not just transaction-level attributes but also block- and network-specific attributes. To summarize, this paper makes the following contributions:

- We design a novel machine learning-based prediction model for transactions' waiting time in Ethereum mainnet.
- To better characterize features of transactions, we prepare a dataset by collecting nearly one million real transactions from Ethereum mainnet.
- With respect to the existing work in the literature, our proposed system considers both block- and network-specific features, in addition to transaction's basic attributes, as a way to improve the prediction results.
- We employ machine learning algorithms, namely k-nearest neighbors, random forest, multilayer perceptron, and support vector machine, and we experimentally demonstrate that our proposed system performs better than the existing one in the literature, with an achievement of 90.18% accuracy and 0.897 F1-score in case of Random Forest.

II. Preliminaries

Let us now provide relevant background details of Ethereum transactions and their lifecycle in the blockchain network.

A. Ethereum

Ethereum [3] is a transaction-based state machine where the states represent any digital information. In the Ethereum network, every successful transaction is recorded on a secure ledger without the need for a centralized authority. A secure ledger is a type of distributed database that is designed to be immutable and irrevocable. Nodes on the ethereum network are responsible for processing transactions, called *miners*. Miners collect the pending transactions into blocks and add the block into a chain with a cryptographic signature referring to the previously mined block depicted in Figure 1.

Unlike Bitcoin, Ethereum is revolutionized with the support of smart contracts which are programs written in high-level Turing-complete programming languages (e.g. Solidity). Smart contracts are compiled into bytecode and are executed on *Ethereum Virtual Machine* (EVM) that supports nearly 140 operations. Gas is a basic unit of EVM's computation. Readers may refer to the Ethereum yellow paper [3] for more details about the gas requirement for each operation.

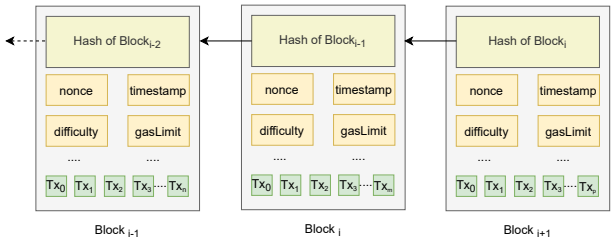


Figure 1: Blockchain Ledger [3]

B. Transaction Lifecycle

A transaction is a cryptographically signed instruction that contains basic information such as sender's and recipient's address, sender's signature, amount of Ether (ETH) to be transferred, a sufficient amount of gas units required to execute the transaction, a price offered per gas unit, and data [3]. Figure 2 depicts the structure of a Ethereum transaction. A transaction that changes the state of Ethereum needs to be

broadcast on the network. Nodes on the Ethereum network maintain a transaction pool (known as *mempool*) for newly broadcasted transactions. These transactions in *mempool* are called *pending transactions*. The required amount of gas is purchased implicitly from the sender's account according to the gas price. In the mempool, transactions bid with their offered gas price to be included in the block. However, there are no predefined criteria for picking a transaction for inclusion in the block, a miner can choose any transaction for inclusion. A transaction with a higher gas price offers more in terms of transaction fees to the miner and thus will more likely to be selected for inclusion by most of the miners. [3].

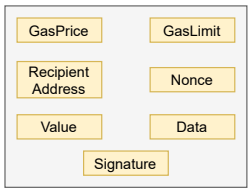


Figure 2: Structure of Ethereum Transaction [3]

III. Prediction Models

In this section, we describe our proposed machine learning-based model for predicting the waiting time of Ethereum transactions. A schematic diagram of our proposed approach is depicted in Figure 3. Let us now describe each of the phases in detail.

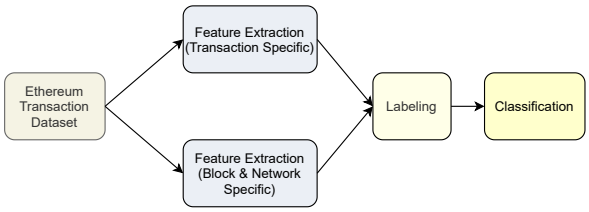


Figure 3: Schematic Diagram of the Proposed Approach

A. Ethereum Transaction Dataset

The ethereum transactions dataset used in this paper is collected from Etherscan, a popular and trusted block explorer and analytics platform for Ethereum [11]. We fetch nearly one million transactions using Etherscan API with the provided transaction hash. Our dataset contains following attributes for a transaction T:

- **hash ($\mathsf{T_{id}}$):** It is a keccak-256 bit hash of the transaction used to uniquely identify the transaction.
- **status ($\mathsf{T_{st}}$):** It indicates a status of the transaction. This field contains 'Success' for a successful transaction and 'Fail' for a failed transaction along with an error message.
- **blockNumber ($\mathsf{T_{bn}}$):** It represents the block number in which transaction has been recorded.
- **timestamp ($\mathsf{T_{bt}}$):** The date and time at which the block containing the transaction is mined.
- **value ($\mathsf{T_{v}}$):** This parameter represents the amount of ETH transferred to the recipient.
- **gasPrice ($\mathsf{T_{gp}}$):** A amount of ETH paid per unit of gas incurred against the execution of the transaction.
- **gasLimit ($\mathsf{T_{gl}}$):** A maximum amount of the gas allowed to execute the transaction.
- **blockDifficulty ($\mathsf{T_{bd}}$):** It denotes difficulty level of the block containing the transaction.
- **appearTime ($\mathsf{T_{at}}$):** This is the time in ISO standard when the transaction first appeared in the waiting pool.
- **receiverAddress ($\mathsf{T_{ra}}$):** This represents 20-byte recipient's address.

B. Feature Extraction

To predict the waiting time of the transactions, we use 13 features that can be divided into two categories.

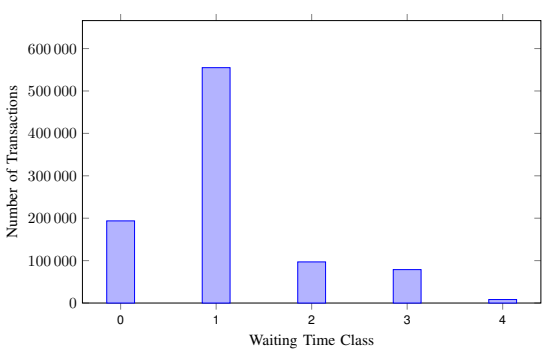
Transaction-Specific Features: The transaction-specific attributes play an indispensable role in deciding the miners' behavior towards the transaction. These include $\mathsf{T_{at}}$, $\mathsf{T_{gt}}$, $\mathsf{T_{gp}}$, and $\mathsf{T_{v}}$, and they are extracted directly from the dataset as mentioned in Section III-A.

Block- and Network-Specific Features: As already mentioned in Section I-B, there are few other factors which influence the miners' behavior in the network and hence the transactions' waiting time. These are specific to blocks and Ethereum network. Let us derive them from the basic attributes of the transactions.

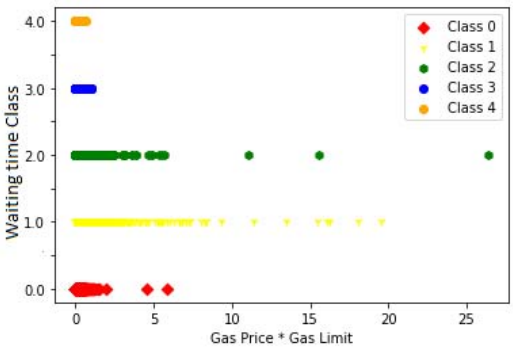
- **waitingTime ($\mathsf{T_{wt}}$):** This is the time difference (in seconds) between $\mathsf{T_{bt}}$ and $\mathsf{T_{at}}$.
- **averageDifficulty ($\mathsf{B^{n}_{ad}}$):** It is an average of the last n-blocks' difficulty.
- **toAddressType ($\mathsf{T_{mt}}$):** It denotes whether $\mathsf{T_{ra}}$ is a contract address or an external account address.
- **appearHour ($\mathsf{T_{ah}}$):** It indicates the hour of the day between 0 to 23 extracted from the $\mathsf{T_{at}}$.
- **miningTime: ($\mathsf{B_{mt}}$):** This is the block timestamp difference (in seconds) between the block containing the transaction and its previous block.
- **miningRateX: ($\mathsf{B_{rx}}$):** It represents the number of blocks added in the last x seconds.
- **gasCost: ($\mathsf{T_{gc}}$):** This is the gas purchase cost, calculated as the product of $\mathsf{T_{gp}}$ and $\mathsf{T_{gl}}$.

C. Labeling and Data Balancing

To predict the waiting time of the transaction, we express the problem as a predictive modeling problem for that we label the transactions in four different categories based on the waiting of transactions. Considering the miners' behavior discussed in Section II-B and the average time of 12 to 14 seconds [11] for a block to be included in the Ethereum blockchain, we create the classes with the duration of 30 sec, 1 min, 2 min, 1 hr, and longer. After labeling, the statistical



(a) Number of transactions in various classes of waiting time



(b) Plot of Gas Price * Gas Limit and waiting time group

Figure 4: Transaction Distribution

analysis of the dataset is presented in Table I. Observe that the dataset distribution is right-skewed since the $mean > median$ for all the classes.

Table I. Statistical Analysis of our Dataset

Class	T_{wt}	Mean	Mode	Median
0	0-29	10.6	1	8
1	30-59	31.53	30	30
2	60-119	72.34	60	61
3	120-3599	568.74	179	304
4	$>$ 3599	56258.6	4098	7355

As shown in Figure 4(a), we observe that the dataset is highly imbalanced since more than 50% of the transactions are labeled with class 0 and class 1. This is because our dataset contains realistic Ethereum mainnet transactions. Figure 4(b) shows the raw transaction distribution w.r.t. transaction cost. We take several measures to mitigate data imbalance before training. In particular, we employ three techniques: Random Undersampling [12], SMOTE [13], and SMOTETomek. The random undersampling reduces the sample size of majority classes to that of minorities. In comparison, the Synthetic Minority Oversampling Technique (SMOTE) increases minority class samples whereas SMOTETomek employs both undersampling and oversampling approaches.

D. Classification Algorithms

We consider various classifiers such as k-Nearest Neighbor (KNN), Random Forest (RF), Multi-layer Perceptron(MLP), Support Vector Machine (SVM), and Ensemble Approach (EA) to predict the waiting time of the transactions.

k-Nearest Neighbour (KNN): k-Nearest Neighbour [14] is a supervised learning algorithm based on the assumptions that similar data are relatively closer to each other. The key step of the KNN algorithm is to first select k nearest neighbors for each test sample and to classify them based on the nature of the nearest neighbors. Figure 5 depicts two significant features in our dataset. First, the time to mine the last block which is inversely proportional to the mining power of a blockchain network at any given time (denoted along the y-axis). The second one is the product of gas price and gas limit set by the sender of the transaction (denoted along the x-axis). It is observed that the random data points are classified into two classes (denoted by red and blue) based on their absolute distance. Based on the behavior of the dataset in the scatter plot it can be theorized that KNN may act as a suitable algorithm for the dataset.

Random Forest (RF): Random forest [15] classifier is a supervised learning algorithm that operates on the ensemble voting principle. RF is a set of multiple decision trees that form an ensemble to produce better classification results. RF relies on selecting a random subset of features and data for different decision trees, which helps make a balanced, unbiased prediction. Because of this random nature, RF is one of the most preferred algorithms for large non-linear datasets. It aggregates the predictions of multiple randomized decision trees and has demonstrated significant performance in settings with a large number of

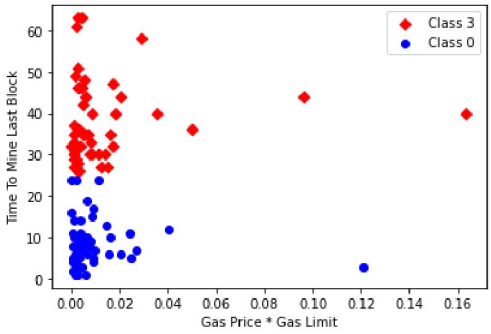


Figure 5: Scattered plot of random data points

features [15]. Additionally, it is adaptable to a variety of informal learning tasks. Another attractive property of RF that makes it suitable for adoption in this paper is the variable importance that gives a quantitative idea of how different dataset features contribute to the prediction. For non-linear relationships, like in our case between gas cost and waiting time, RF is expected to perform better than the other fundamental algorithms like naive bayes [16] and logistic regression [17].

Multi-layer Perceptron (MLP): Multi-layer Perceptron [18] is a feedforward artificial neural network-based classification algorithm. MLP neural network contains one input layer, one output layer and atleast one hidden layer. MLPs are well suited to classification prediction problems in which inputs are classified or labelled. For the scope of this paper, MLP has been used to test the efficiency of neural networks on our dataset having low correlation.

Linear Support Vector Machine (LSVM): Support Vector Machine [19] is a learning algorithm based on the concept of differentiating classes by making hyperplanes. As the the complexity of the data increases, the degree of hyperplane can be adjusted. Linear SVM [20] is a newer and faster version of SVM that is easily scalable for larger datasets.

Ensemble Approach: We consider an ensemble hybrid model which has been developed based on the weighted approach because it is more suitable for unbiased computation [21]. The ensemble model consists of KNN, RF and MLP models with the final result being the weighted voting of all these three models. To decide the weight of various models in voting, we give the highest weightage to the model which have the highest accuracy. The final result of the classification is the class that gets majority number of votes from these three models. In the case when all the three models vote for different classes, we consider the class voted by the model with highest accuracy as the final result.

IV. Experimental Evaluation

Let us describe the experimental setup and the experimental evaluation results. The experiments are performed on a large dataset, and therefore they require a device with high CPU performance, memory size, and hard disk capacity. Our experimental environment is described in Table II.

Table II. Experimental Environment

Software and Hardware	Configuration
Operating System	Ubuntu 18.04
CPU	Xeon(R) E5-2620 v3
Memory Size	320 GB
Disk Capacity	5 TB

A. Evaluation Criteria

In order to evaluate the experimented results, we consider the following criteria which demonstrate the performance of the machine learning models.

Accuracy: Accuracy is one of the most widely used score for the evaluation of classification problems as it gives a clear quantitative idea of the performance of the model. Accuracy score is the percentage measure of how many labels are predicted correctly for given samples and is calculated based on the following function:

$$accuracy(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(\hat{y_i} = y_i) \quad (1)$$

where $n_{samples}$ denote total number of samples and $\hat{y_i}$, y_i represent the correct value and the predicted value of i^{th} sample respectively.

Precision: Precision gives the idea of how precisely a model can predict correct values. This is computed as follows

$$Precision = \frac{true\ positives}{true\ positives + false\ positives} \quad (2)$$

Recall: Recall gives a good idea of how accurately a model can predict the classes of interest for the specific problem. This is computed as follows

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives} \quad (3)$$

F1 Score: F1 Score is the harmonic mean of precision and recall and it is used to give a better measure of incorrectly predicted labels compared to accuracy. Use of harmonic mean makes sure extreme values are ignored and hence a better evaluation result.

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (4)$$

F1 score is calculated for individual classes taking one class at a time while taking rest of the samples as separate single class. On the other hand, Macro average is used as Macro F1 score gives equal weightage to all the classes and is computed as follows:

$$Macro\ F1\ Score = \frac{1}{n_{classes}} \sum_{i=0}^{n_{classes}-1} (F1\ Score)_i \quad (5)$$

Where $(F1\ Score)_i$ denotes the F1 Score of i^{th} class used for Macro average.

Area Under Receiver Operating Characteristic Curve: Receiver Operating Characteristic (ROC) curve is a two-dimensional plot of recall (Equation 3) versus the false positive rate (Equation 6). The area under the ROC curve is the numeric metric used to represent the quality and performance of the classifier. Accuracy and Area Under Curve

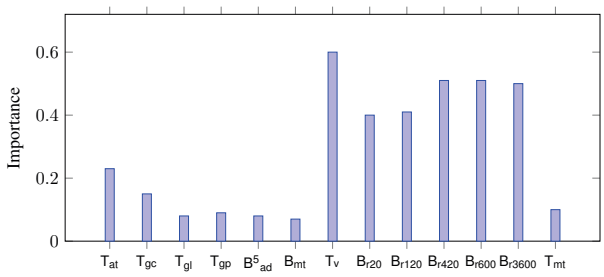


Figure 6: Features Importance for RF Model

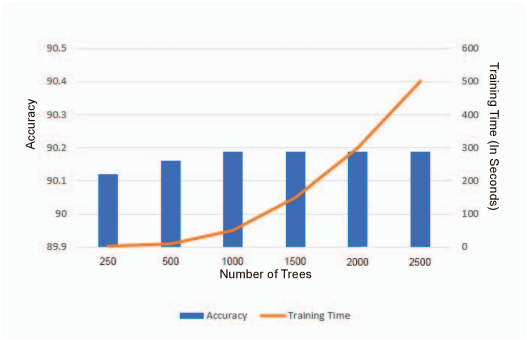


Figure 7: Training Time vs. Accuracy for RF

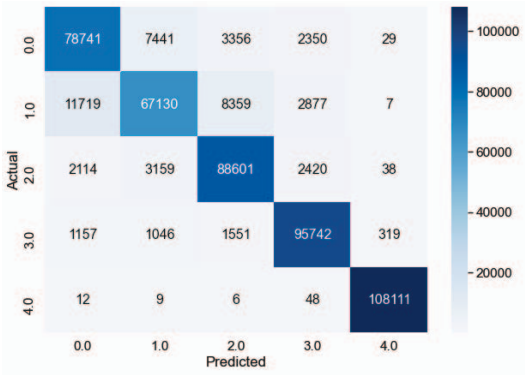


Figure 8: Confusion Matrix for RF

together give a good understanding of the performance of the model [22].

$$False\ Positive\ Rate = \frac{false\ positives}{false\ positives + true\ negatives} \quad (6)$$

B. Evaluation of Classifiers

Experimenting with KNN: As the performance of the KNN algorithm depends on the value of k which represents the number of nearest neighbour, we conducted a number of training and testing phases with different values of k ranging from 3 to 100 to determine its optimal value for our model. The result is depicted in Table III. This is to observe that the algorithm achieves best performance with k value 3. Due to the random nature of the dataset, we used a weighted approach to determine the nearest neighbors by taking the inverse of the distance between points as the weight. In particular, we adopted Manhattan distance [23] to calculate distances between various points.

Table III. Evaluation of KNN Models

k Value	Accuracy	F1 Score	Precision	Recall	ROC
3	86.56	0.86	0.86	0.86	0.91
5	85.75	0.84	0.85	0.85	0.90
7	85.23	0.84	0.85	0.855	0.90
10	84.42	0.83	0.84	0.84	0.89
15	83.55	0.82	0.83	0.83	0.89
20	82.74	0.81	0.82	0.82	0.88
100	77.8	0.76	0.77	0.77	0.85

Experimenting with RF: We evaluated RF models with multiple combination of parameters to get the best possible result. RF Model's N_estimator parameter represents the number of trees in the forest that helps to produce unbiased results by selecting features randomly [24]. The final result is computed based on the ensemble voting by various decision trees.

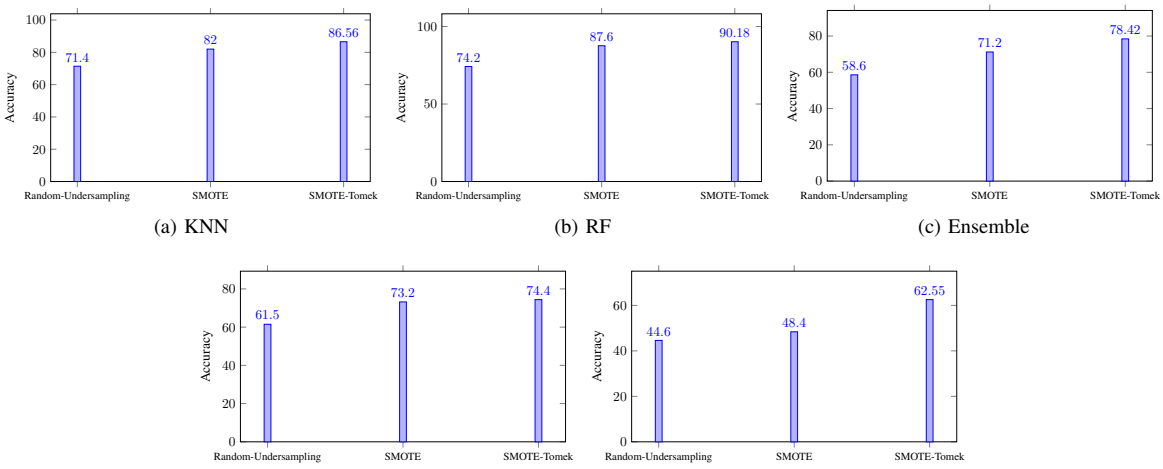
We tested the model's performance by varying the following parameters: depth of a tree, minimum number of samples for each node to get split, minimum number of samples in leaf nodes, and N_estimators. Figure 6 represents the importance of various dataset features described in Section III-A. Note that block- and network-specific attributes are more useful than most of the transaction-specific attributes. The performance of various RF models with different values for N_estimators ranging from 250 to 2500 is shown in Table IV, where the best result is obtained for N_estimators value 1000 or more. Figure 7 shows a relation between the accuracy and training time of the model, whereas the confusion matrix is shown in Figure 8.

We observe that increasing the value of N_estimators increases the accuracy slightly for the model compromising the training time. Therefore, there is a need to maintain a trade-off between the training time and the model's performance.

Table IV. Evaluation of RF Models

No. of Trees	Accuracy	F1 Score	Precision	Recall	ROC
250	90.12	0.89	0.896	0.896	0.935
500	90.16	0.897	0.896	0.895	0.935
1000	90.18	0.897	0.896	0.896	0.936
1500	90.18	0.897	0.896	0.896	0.936
2000	90.18	0.897	0.896	0.896	0.936
2500	90.18	0.897	0.896	0.896	0.936

Experimenting with Other Classification Algorithms: In addition to KNN and RF, we have also conducted our



(a) KNN (b) RF (c) Ensemble

(d) Linear SVM (e) MLP

Figure 9: Accuracy comparison w.r.t. different balancing technique

experiment with the followings: (1) MLP with 1-3 hidden layers by varying the size of each hidden layer ranging from 32 to 254 neurons, (2) Linear SVM with planar and gaussian kernels by considering both l_1 and l_2 penalty [25], and (3) Ensemble model with two different approaches: one with all the models having equal weightage and the other one with weighted approach. As all the transactions in a block are confirmed at the same time, our dataset contains several overlapping data points which leads to a disappointing performance for Linear SVM model. Moreover, MLP requires a high degree of encoding due to large number of features associated with our dataset. The low accuracy and F1-score of MLP eventually resulted into a degradation of the performance of the Ensemble technique as well.

Experimenting with Balancing Techniques: The accuracy comparison for different machine learning models under various balancing techniques is presented in Figure 9, which shows that the SMOTETomek performs better than SMOTE and Random Undersampling.

Overall Observation: Table V shows the performance evaluation of various machine learning techniques with the best combination of parameters. It is clear that the RF and KNN algorithms perform better than MLP and SVM. The model trained by the RF algorithm achieves 90.18% accuracy and 0.897 F1-score which is the highest among all the models.

C. Discussion w.r.t. Literature

As we already mentioned before, the only existing work which is related to our proposal is by Singh et. al [7]. The proposal uses only gas limit, gas price, and ether value of the transactions as the features. The performance of [7] on our dataset is shown in Table VI. Interestingly, we observe that the correlation of waiting time with gas limit and gas price is limited, and therefore, the prediction based on these features is not efficient enough. In comparison to this, our proposed model achieves a better result with 90.18% accuracy and 0.897 F1 score when the model is trained with RF on the dataset balanced with SMOTETomek.

Table V. Performance Evaluation for Classification Algorithms

Model	Accuracy	F1 Score	ROC
KNN	86.56	0.86	0.91
RF	90.18	0.897	0.936
MLP	62.55	0.58	0.61
Linear SVM	74.41	0.74	0.74
Ensemble	78.42	0.76	0.76

Table VI. Comparison w.r.t. Literature

Research Work	Accuracy	F1 Score
Singh et. al. [7]	78.5%	0.78
Our Proposed Approach	90.18%	0.897

V. Conclusion and Future Plans

This paper proposes a machine learning-based model to predict the transaction waiting time on the Ethereum platform. We trained the models by extracting both the block- and network-specific features along with the transaction's basic attributes. Our approach improves the learning process relative to the baseline models, resulting in a better prediction of the waiting time for transactions. The experimental result shows that the transaction's waiting time prediction accuracy of our model for transactions on Ethereum mainnet is up to 90.18% with an F1-score of 0.897. In essence, the inclusion of block- and network-specific attributes delivers a notable improvement compared to the existing model.

In the future, we plan to investigate various mining software and their transaction selection strategies in the absence of offered gas prices for more useful feature extraction and to address the problem of data point overlapping without affecting the model's performance.

ACKNOWLEDGMENT

First author's research work is supported by Visvesvaraya PhD Scheme, MeitY, Govt. of India (MEITY-PHD-3009).

REFERENCES

[1] S. Nakamoto. (2008) Bitcoin: A peer-to-peer electronic cash system. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] M. H. Miraz and M. Ali, "Applications of blockchain technology beyond cryptocurrency," *arXiv preprint arXiv:1801.03528*, 2018.

[3] G. Wood. (2014) Ethereum: A secure decentralised generalised transaction ledger. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[4] H. Kim, S. Kim, J. Y. Hwang, and C. Seo, "Efficient privacy-preserving machine learning for blockchain network," *IEEE Access*, vol. 7, pp. 136 481–136 495, 2019.

[5] S. Tanwar, Q. Bhatia, P. Patel, A. Kumari, P. K. Singh, and W. Hong, "Machine learning adoption in blockchain-based smart applications: The challenges, and a way forward," *IEEE Access*, vol. 8, pp. 474–488, 2020.

[6] S. M. Werner, P. J. Pritz, and D. Perez, "Step on the gas? a better approach for recommending the ethereum gas price," in *Mathematical Research for Blockchain Economy*. Springer, 2020, pp. 161–177.

[7] H. J. Singh and A. S. Hafid, "Prediction of transaction confirmation time in ethereum blockchain using machine learning," in *International Congress on Blockchain and Applications*. Springer, 2019, pp. 126–133.

[8] S. Bouraga, "An evaluation of gas consumption prediction on ethereum based on transaction history summarization," in *2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. IEEE, 2020, pp. 49–50.

[9] J. E. de Azevedo Sousa, V. Oliveira, J. Valadares, G. Dias Gonçalves, S. Moraes Villela, H. Soares Bernardino, and A. Borges Vieira, "An analysis of the fees and pending time correlation in ethereum," *International Journal of Network Management*, p. e2113, 2020.

[10] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, 1979.

[11] Etherscan.io. Ethereum (ETH) blockchain explorer. [Online]. Available: http://etherscan.io/

[12] M. Bach, A. Werner, and M. Palt, "The proposal of undersampling method for learning from imbalanced datasets," *Procedia Computer Science*, vol. 159, pp. 125–134, 2019, knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES2019.

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002.

[14] G. Guo, H. Wang, D. Bell, and Y. Bi, "Knn model-based approach in classification," 08 2004.

[15] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[16] K. P. Murphy *et al.*, "Naive bayes classifiers," *University of British Columbia*, vol. 18, no. 60, 2006.

[17] R. E. Wright, "Logistic regression. (eds.) reading and understanding multivariate statistics," p. 217–244, 1995.

[18] P. Marius, V. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer perceptron and neural networks," *WSEAS Transactions on Circuits and Systems*, vol. 8, 07 2009.

[19] T. Evgeniou and M. Pontil, "Support vector machines: Theory and applications," vol. 2049, 01 2001, pp. 249–257.

[20] Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, and C.-J. Lin, "Training and testing low-degree polynomial data mappings via linear svm." *Journal of Machine Learning Research*, vol. 11, no. 4, 2010.

[21] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*. Springer, 2000, pp. 1–15.

[22] Jin Huang and C. X. Ling, "Using auc and accuracy in evaluating learning algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, 2005.

[23] D. Sinwar and R. Kaushik, "Study of euclidean and manhattan distance metrics using simple k-means clustering," *Int. J. Res. Appl. Sci. Eng. Technol*, vol. 2, no. 5, pp. 270–274, 2014.

[24] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas, "How Many Trees in a Random Forest?" in *Machine Learning and Data Mining in Pattern Recognition*. Springer, Berlin, Heidelberg, Jul. 2012, pp. 154–168.

[25] Y. Koshiba and S. Abe, "Comparison of l1 and l2 support vector machines," in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, vol. 3. IEEE, 2003, pp. 2054–2059.