An Exploration on Blockchain Unfair Transaction Order Prediction

Zheng Dong[1,*], Junyu Lv[2,*], Xuetao Wei[1]
[1]Department of Computer Science and Engineering, Southern University of Science and Technology, China
[2]School of Innovation and Entrepreneurship, Southern University of Science and Technology, China
Email: {12232414, 12233167}@mail.sustech.edu.cn; weixt@sustech.edu.cn

***Abstract*—Blockchain is basically a chain of blocks that store all committed transactions using a public ledger, which is originally designed for Bitcoin. Blockchain works in a decentralized environment that is enabled by comprising several core technologies. All the transactions occur in a decentralized manner that eliminates the requirement for any intermediaries to validate and verify the transactions. Therefore, it is used in various financial services. However, it suffers scalability problem. In Bitcoin, the limited size and frequency of the blocks along with the number of transactions the network can process can be considered a scalability problem. And the rapid growth of user base will increase and create an adverse impact on the network's throughput, causing many issues like transaction order fairness. Due to network congestion and delay, the transactions received by miners may not be as the same order as they are created. In our work, we predict the unfair transaction orders, to help miners record the transactions properly. First, we define the unfair order prediction problem as a binary classification task. Second, we create datasets to simulate the real situations as close as possible. Having the datasets, we apply several machine learning (ML) and deep learning (DL) models to predict if a transaction is misplaced. Extensive experiment results prove the effectiveness and rationality of our proposed approach.**

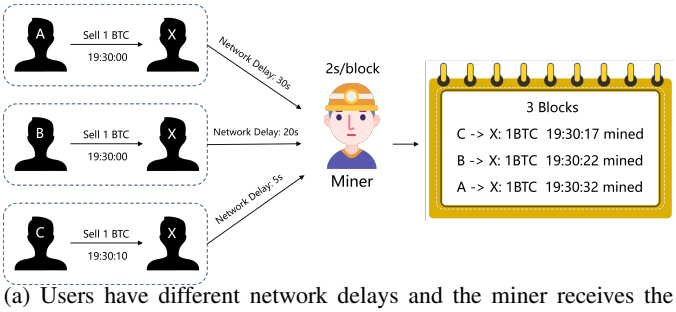
***Index Terms*—Blockchain, Transaction Order Fairness, Machine Learning, Deep Learning**

I. Introduction

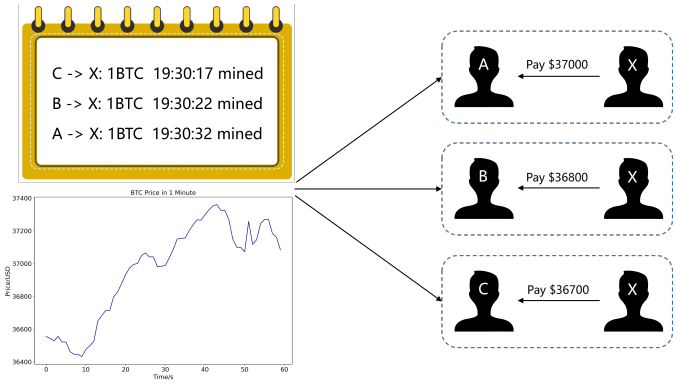
Blockchain technology allows for the peer-to-peer transfer of digital assets without the need for intermediaries [1]. It was originally created to support the popular cryptocurrency Bitcoin, which was proposed in 2008 and implemented in 2009 [2]. Since then, it has experienced significant growth in the capital market, reaching a value of 10 billion dollars in 2016. Blockchain is a chain of blocks that stores committed transactions using a public ledger. This chain grows as new blocks are added to it. Blockchain operates in a decentralized environment that uses technologies such as digital signatures, cryptographic hashing, and distributed consensus algorithms. This allows for transactions to be processed in a decentralized manner, eliminating the need for intermediaries to validate and verify them. Blockchain has several key characteristics, such as decentralization, transparency, immutability, and auditability [3].

Blockchain technology allows for payments to be completed without the need for a bank or any other intermediary. This makes it a potential candidate for use in various financial services, such as digital assets, remittance, and online payment [4]. However, blockchain technology also has some technical constraints, one of which is scalability [5]. In the case of Bitcoin, the limited size and frequency of blocks, along with the number of transactions the network can process, can create scalability issues. The average time it takes to create a block in Bitcoin is 10 minutes, and the block size is limited to 1 megabyte [6], which limits the network's overall throughput. In general, Bitcoin's transaction processing capacity is between 3.3 and 7 transactions per second [7]. However, the increased size of recently generated blocks is effectively limiting the transaction throughput to just 2-4 transactions per second, which is not sufficient for high-frequency trading. In 2019, there were over 36 million wallet users, and this number is only expected to grow, which could create further issues for the network. These issues, such as blockchain congestion,

Fig. 1: Transaction order fairness.



(a) Users have different network delays and the miner receives the transactions in wrong orders.



(b) Users are paid in wrong prices, causing unfairness.

* Equal Contribution.

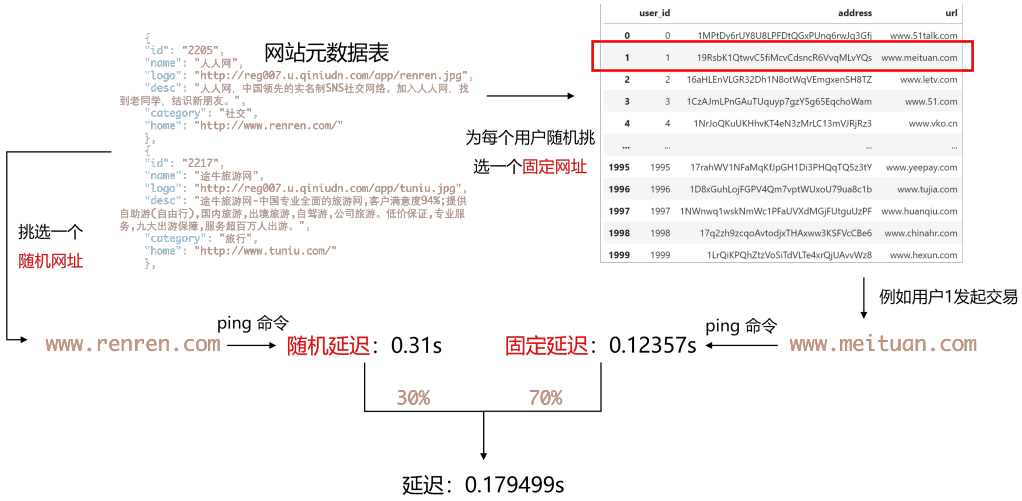


Fig. 2: DelayWeb latency calculation (in Chinese).

transaction delays, and increased transaction fees, may become a cause for concern.

Transaction order fairness is one of the problems. Due to congestion and delay, the transactions received by miners may not be as the same order as they are created. As shown in Figure 1, considering a special case, suppose users A, B and C are willing to sell 1 BTC to buyer X. They in total propose three transaction at different time. However, their network delays have huge difference. Therefore, for the miner, the three transactions he receives will have totally different orders. Possibly they are recorded in different blocks, thus, when X pay the three sellers, the price will not be what they expected before because the value of BTC changes in every second, causing unfairness.

In this paper, we target at predicting the unfair transaction orders, to help miners record the transactions properly. First, we define the unfair order prediction problem as a binary classification task. Second, since there is no public dataset about transaction orders, we have to create datasets to simulate the real situations as close as possible. Having the datasets, we apply several machine learning (ML) and deep learning (DL) models to predict if a transaction is misplaced. Extensive experiment results prove the effectiveness and rationality of our proposed approach.

The contributions of our paper are:

- We formulate the unfair transaction order problem into a binary classification task.
- We generate two datasets that can simulate the real-world conditions.
- We used different ML and DL models and validate them on our datasets.

The rest of this paper is organized as the following. Section II provides the definitions about our task. Section III introduces how we create the datasets. The adopted methodology and models will be given in section IV. In section V, the models is verified by our dataset, and the experiment results demonstrate its practicability. A brief conclusion and several potential directions are provided in the last section VI.

II. Preliminaries

Cross-sectional data and time series data. Cross-sectional data is a type of data that is collected at a specific point in time, rather than over a period of time. This type of data is often used in the social sciences to study a particular population and understand their characteristics, behaviors, and attitudes at a given point in time, but it does not provide information about how those characteristics may change over time. Cross-sectional data can be collected through various methods, such as surveys, experiments, or observations, and it is often used in conjunction with other types of data to provide a more complete picture of a population. Time-series data is a sequence of data points, measured typically at successive time points spaced at uniform time intervals. It is often used in statistics, signal processing, and machine learning. The data points in a time series may be observations of a variable over time, or they may be derived from another type of data through a mathematical transformation. The individual data points in a time series can provide insights into trends and patterns over time, which can be useful for forecasting future values of the variable. One typical difference between cross-sectional data and time-series data is that time-series is a continuous observation on a single object, such as the weather of Shenzhen in each month. Cross-sectional data can be a description on different objects at the same time, for example, the weather of Shenzhen, Guangzhou and Dongguan in December.

Binary classification task. Binary classification is a type of supervised machine learning problem where the goal is to predict the outcome of a binary variable, which can take on one of two possible values (e.g. 0 or 1). This type of classification task is commonly encountered in many fields, including finance, medicine, and marketing.

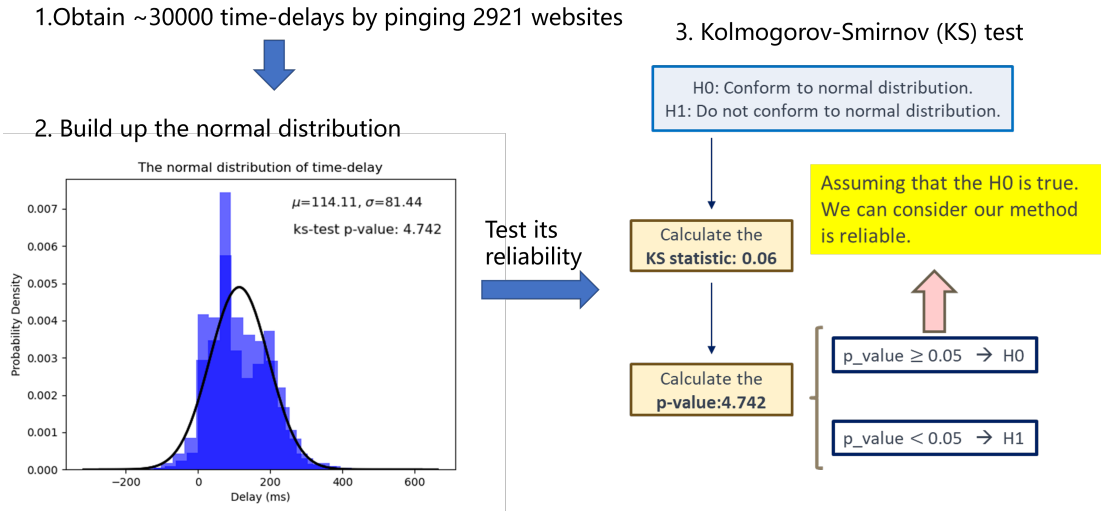


Fig. 3: The generating procedure of the distribution in DelayDist dataset.

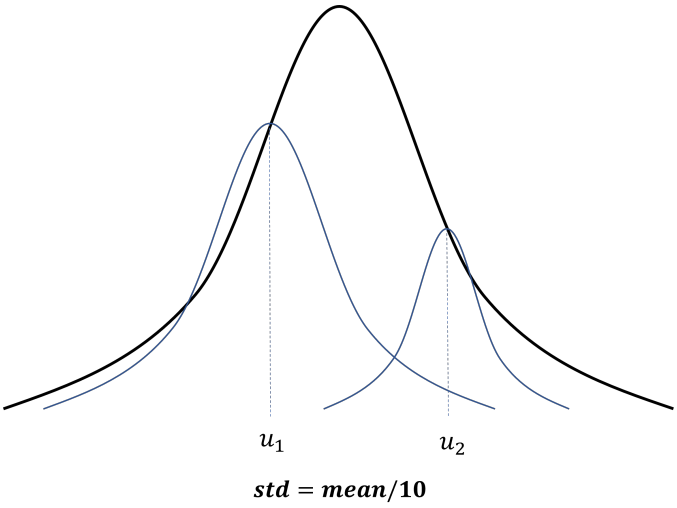


Fig. 4: DelayDist latency generation.

In a binary classification problem, the goal is to train a model on a dataset where each data point is associated with a set of features and a binary label. For example, if we were trying to predict whether an email is spam or not, the dataset would consist of a collection of emails, where each email is represented by a set of features (such as the sender, the subject, and the content of the email) and a binary label (0 for "not spam" and 1 for "spam"). The model would be trained on this dataset in order to learn the patterns that differentiate spam emails from non-spam emails.

Once the model is trained, it can be used to make predictions on new data points (i.e. new emails) by using the same set of features. The model uses the features of the new data point to make a prediction about its label. This prediction can be either 0 or 1, depending on the model's decision. The model's decision is based on the rules it learned during training, which are derived from the patterns it observed in the training data.

There are many algorithms that can be used for binary classification, and the choice of algorithm often depends on the specific characteristics of the dataset and the nature of the problem. Some common algorithms for binary classification include logistic regression, support vector machines, and decision trees. These algorithms differ in their approach to learning from the training data and making predictions, but they all have the goal of accurately predicting the outcome of a binary variable.

In order to evaluate the performance of a binary classification model, we typically use metrics such as accuracy, precision, recall, and F1 score. These metrics allow us to assess how well the model is able to make correct predictions on new data points. For example, if we were using a binary classification model to predict whether an email is spam or not, we would want to have a high precision (i.e. a low rate of false positives), because we do not want to accidentally mark a legitimate email as spam. On the other hand, we would also want to have a high recall (i.e. a low rate of false negatives), because we do not want to miss any spam emails. The F1 score is a balance between precision and recall, and it is a commonly used metric for evaluating binary classification models.

Unfair order prediction. We define the transaction order prediction problem as a binary classification task. Assume there are n users in the blockchain system, which are $\{u_1, u_2, \ldots, u_n\}$. Suppose a user u_i initiates a transaction with user u_j at timestamp ts^{send}, and the transaction amount is v. The miner receives this transaction at timestamp ts^{recv} after the network latency ltn, then the whole transaction can be defined as

$$t^{all} = (ts^{send}, ts^{recv}, ltn, u_i, u_j, v) \tag{1}$$

However, from the view of the miner, ts^{send} and ltn is not visible since the transmitted data does not contain them.

Therefore, the received transaction t is defined as a quadruple:

$$t = (ts^{recv}, u_i, u_j, v) \tag{2}$$

A transaction sequence $T = [t_1, t_2, \ldots, t_l]$ is a series of transactions of length l received by the miner, where $\forall i = 1, 2, \ldots, l-1, ts_i^{recv} < ts_{i+1}^{recv}$. Each transaction is associated with a binary label, where 0 for "right order", and 1 for "misplaced". To be specific, if a transaction is created early but received late, then it is misplaced. For the transaction $t_i, i > 1$, we retrieve its sending timestamp t_i^{send} in the corresponding whole transaction t_i^{all}. Similarly, we get t_{i-1}^{send} of t_{i-1}. Then the label of t_i should be

$$y_i = \mathbb{I}(t_i^{send} < t_{i-1}^{send}) \tag{3}$$

where $\mathbb{I}$ stands for the indicator function.

Therefore, given a transaction sequence T, the unfair order prediction task is to obtain a probability distribution $\hat{P} \in \mathbb{R}^2$ on binary labels for each transaction that best predicts if it is misplaced. In conclusion, our goal is to build a model with parameters Θ^* that satisfies

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}}\, CrossEntropy(\hat{P}, P) \tag{4}$$

III. Datasets

One of the primary challenges is that there is no publicly available dataset related to the unfair transaction order problem, which means we have to create datasets that as closely resemble real-world scenarios as possible.

First, we generate a user set according to the encrypted address generating algorithm used in Bitcoin. Each user is represented by a string of address. To generate a transaction, we randomly choose a sender user u_i and a receiver user u_j and record the current system time as ts^{send}. The transaction value v is also a randomly generated number. The receive time of the miner is calculated by $ts^{recv} = ts^{send} + ltn$, where ltn is network latency to transmit transaction data to the miner. However, ltn cannot be a random number because our goal is to simulate the real-world scenarios. Therefore, the core of dataset generation is to simulate ltn.

A. Network Time-delay / Latency

Because the order of a blockchain transaction is determined by the time it was received by the miner, the time-delay / latency ltn of the users determines the transaction order. Consequently, generating adequate time-delay for users is the focus of the dataset-creating task.

One viable method for generating time-delay is to simulate the network operation environment. However, due to the network's heterogeneity and rapid development, simulating the Internet is a challenging task [8]. The heterogeneity of the Internet is reflected in several aspects, such as the individual links that transmit network traffic, the interoperability protocols of links, and the degree of congestion on different links, all of which make it more difficult to obtain the time-delay of the network traffic [9].

	send_timestamp	recv_timestamp	latency	label	from	to	value
0	2022-10-25 09:48:30.414545	2022-10-25 09:48:30.432322	0.017777	0	1KYsPXSLA4s6eQAAFm3849g3mksDE4CWFK	1HRUm7h7EyP4qXm4Jr8d3M1koBGyQHXKdo	5.112375
1	2022-10-25 09:48:30.331380	2022-10-25 09:48:30.434821	0.103441	1	1GywTZkeUMWf9J3NaJhCbq6Lt5eUkC6Rid	17aUdKKWk8uoDtzGUcfXnmwgtL4imsxv9Z	8.107424
2	2022-10-25 09:48:30.383802	2022-10-25 09:48:30.435350	0.051548	0	12gtzJc3qrQThEk3TcUxCukmF6CfbEMdBm	1PpHffFCJa8PLMRUskxLi2KaCWATrS78zi	0.261991
3	2022-10-25 09:48:30.438316	2022-10-25 09:48:30.478155	0.039839	0	13LLxejwz7YKCrvDAvGA5a1V1r36vtJoXc	18MXbFhYMmcorHBdh5WJRCDUNwd5pTRiPg	0.848824
4	2022-10-25 09:48:30.358478	2022-10-25 09:48:30.526710	0.168232	1	1HCMQasEvt5K9Lm555MbJzYUJj7jCs1j2E	12qYZB54ici8jVgLAWap4dDtNRM77qQdj4	1.843508
...	...	...	...	...	...	...	...
99766	2022-10-25 11:09:51.654577	2022-10-25 11:09:51.694776	0.040199	0	1JtVnzqSfnhKFzgNhUbzBe4N6qYSLW1pXhv	19Jb2yVGpvJxkxN98fcp4XJLn1uCMakxTj	8.893335
99767	2022-10-25 11:09:51.689212	2022-10-25 11:09:51.746803	0.057591	0	1G7jLPXC4DPYQ4BXTPacuAEAqJQBx3siHJ	1KfBqnpyazENxBDR736jCMH9yroPnR6vr3	7.300961
99768	2022-10-25 11:09:51.721649	2022-10-25 11:09:51.761165	0.039516	0	1C7U3FErAtCJWeVRmYtZbd4uh2eXzXGjVG	1PhWATQnDnVSLAMa93ik8mKpDaNzdQ1PhtJ	9.941156
99769	2022-10-25 11:09:51.747805	2022-10-25 11:09:51.763729	0.015924	0	1BSi3qJE4LnCXaUNof7GQ6PqmpyYjeWZCE	1F4BzsrnbJjYcU5fTYmSgJazSpVicutV75	1.019992
99770	2022-10-25 11:09:51.630142	2022-10-25 11:09:51.811320	0.181178	1	1EVP9M7CquaieMg1bMAEABqJhckSdndMvV	1HCMQasEvt5K9Lm555MbJzYUJj7jCs1j2E	8.492597

交易发起时间 = 交易记录接收时间 网络延迟 随机挑的给钱的用户A 随机挑的收钱的用户B 给了多少

按这个排序

Fig. 5: DelayWeb data slice.

Considering the complexity, we plan to generate the delays by the following guidelines:

- The network environment of a user is mostly stable. For example, a user using a 1000Mb fiber cable is definitely better than a 1Mb WiFi network.
- The network can often be disrupted. For example, when a user is downloading something, the latency will increase.

B. DelayWeb Dataset

A brute-force solution is to use real-world latency. To be specific, we first acquired a dataset of over 200 websites' URLs. Then, each user is randomly assigned with an fixed website URL. When generating a transaction, we ping the corresponding fixed website of the sender u_i and record the latency as ltn_f. And to simulate the network disturbance, a random website URL is selected and pinged to get the random latency ltn_r. Then the total latency ltn is determined as

$$ltn = w_1 \cdot ltn_f + w_2 \cdot ltn_r \tag{5}$$

where the weights satisfy $w_1 + w_2 = 1$. The latency calculation procedure is provided in Figure 2.

After generating the latency, we can calculate $t^{recv} = t^{send} + ltn$, then the whole transaction t^{all} can be created. In total, we generated near 100,000 t^{all} to build the dataset DelayWeb. Some transactions in the dataset is given in Figure 5.

C. DelayDist Dataset

The more of the various parameters in the network are known, the more favorable it is for the simulation of the network. However, due to the lack of sufficient statistical invariants of network parameters [10], the simulation work has encountered great difficulties. Nowadays, the majority of research in this field focuses on statistical models of network flow, including the generalized Cauchy process, which provides a technique to characterize the multi-fractal phenomena of traffic and find the Hurst exponent [11]; the auto-correlation function model [12], which can obtain the optimal approximation of the network traffic in Hilbert space; the stochastic traffic delay bound model(σ, ρ) [13] and (σ', ρ') [14]. By using a large amount of network traffic data to model the statistical distribution of network delay, [15] proposed that the time-delay of network traffic may follow normal distribution.

In order to obtain a lot of network traffic data while also ensuring its reliability, we collect 2921 valid websites with different domain names from real-world scenarios. Then, we use the ping command, which operates by sending a small packet of data to a specific domain to test the connectivity between two servers in the Internet network, and obtain the time-delay of data transmission. To eliminate potential interference, we use ping commands on each website for ten times, yielding 29210 time-delays.

After acquiring enough time-delay data, we begin utilizing them to establish a normal distribution. Firstly, we group the sampled time-delay points with a step size of 0.01 and an abscissa. Secondly, we calculate the mean and variance of the data and then build up the normal distribution N:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{6}$$

where the obtained mean is 114.11 and the variance is 81.44.

To determine the reliability of the normal distribution of time-delay, we employ KS test. Kolmogorov-Smirnov (KS) test [16], [17] is a statistical test that is used to compare the cumulative distribution function (CDF) of a variable with a reference CDF. KS test is widely used in a variety of fields, including economics, finance, and engineering, to test hypotheses about the underlying distribution of a dataset. It is a powerful tool to detect differences between two distributions, and it can be used to compare datasets with a wide range of shapes and characteristics. The steps to perform a Kolmogorov-Smirnov (KS) test are as the following:

1) Specify the hypotheses. The null hypothesis is that the data conform to the normal distribution, and the alternative hypothesis is that the data do not conform to the normal distribution.
2) Calculate the maximum distance between the cumulative distribution functions (CDFs) of the sample and the reference distribution. Here we will get the maximum distance(KS statistic) and the p-value, where the p-value is the probability of obtaining a distance as large as the observed distance if the null hypothesis is true.
3) Compare the p-value to the desired level of statistical significance. If the p-value is less than the desired level of significance, we can reject the null hypothesis. The most commonly used alpha levels are 0.05 and 0.01, which correspond to a 5% and 1% risk of making a Type I error, respectively.

The p-value is 4.742 after calculation. When we compare it with the desired level(0.05) we choose, we can assume that there is sufficient evidence to conclude that the time-delay conforms to the normal distribution, which indicates the way we generate the dataset is reliable. The whole generating process of the distribution is shown in Fig 3.

After obtaining the distribution, to generate a latency for each transaction under the two guidelines mentioned above, we assign each user with a sub-distribution $N'_u \sim (\mu'_u, \sigma'_u)$, where μ'_u is sampled from N, and

$$\sigma'_u = \frac{\mu'_u}{\alpha} \tag{7}$$

where $\alpha > 1$. The transaction latency is sampled from this sub-distribution, which indicates that if a user has a large average latency, i.e. he has a bad network environment, then the standard variance will also be large, because a bad network also tent to have a large jitter. An example is shown in Figure 4. Suppose user u_1 has a unstable network, then his sub-distribution will have large μ' and σ'. User u_2 has a fast and stable network environment, then his sub-distribution will be stable. Finally, we set $\alpha = 10$ to build the DelayDist dataset.

TABLE I: Dataset description.

Dataset	**DelayWeb**	**DelayDist**
#users	2000	5000
#trasactions	99,771	100,000
%misplaced	20.82	50.02
w_1	0.7	0.7
w_2	0.3	0.3
α		10
μ		114.11
σ		81.44

D. Labeling

As mentioned before, from the view of miners, we can only see the sender, receiver and the transaction value. And the miner can record the time it receives the transaction. Thus, we delete the irrelevant columns from t^{all} to get $t = (ts^{recv}, u_i, u_j, v)$. For the transaction sequence T, we iterate from the first transaction and label them by Eq. 3. A detailed description of the two datasets is shown in Table I.

IV. Methodology

A. Time Series Model

At first, we consider the prediction task as a time series prediction problem, therefore, we apply typical time series prediction models to predict the 0-1 label sequence. In the dataset, the useful input features are the users, because the transaction latency are related to the simulation of network environment of each user. It is a categorical feature, where we can use embedding techniques.

In the NLP area, how to obtain effective representations of text words has long been a research focus. One-hot embedding is a simple solution to represent each word with a one-hot vector whose dimension is equal to the size of the vocabulary. The difference between these vectors is the word index. However, one-hot embedding suffers from dimension curse, making the embedding vectors very large and sparse. More importantly, the vectors cannot reflect the semantic relationships between words. Therefore, word embedding technologies are proposed to efficiently learn a fixed-length real-value vector representation for each word. And the most important advantage is

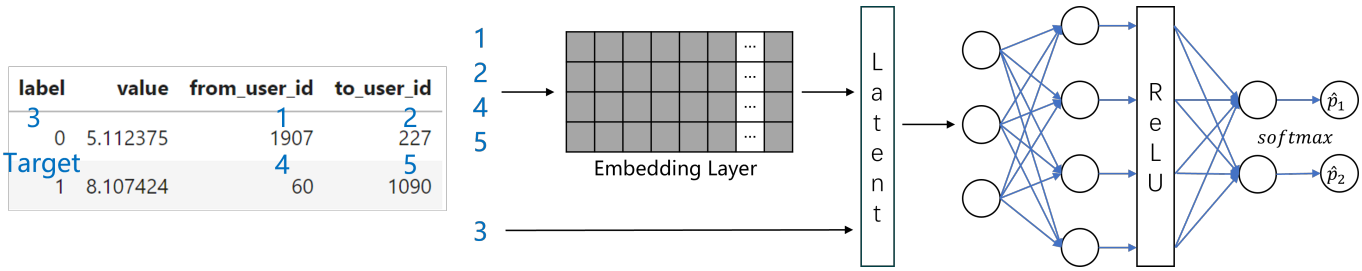


Fig. 6: Cross-sectional prediction framework.

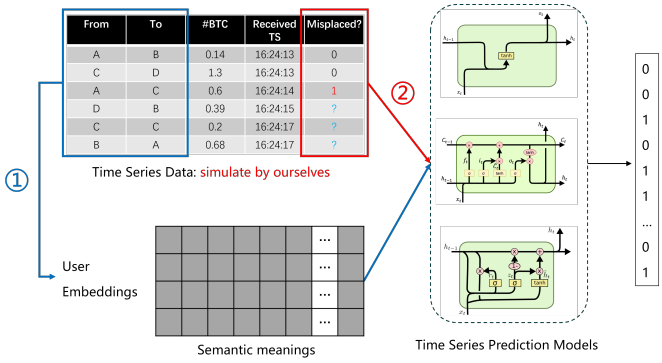


Fig. 7: Time series prediction framework.

that it can catch the contextual similarity of words. If two embedding vectors are close, which means they have high similarity, then the two corresponding words also tend to have similar semantic meanings.

Therefore, we use a embedding layer with an embedding matrix E to represent the users and extract their underlying similarities. In the embedding layer, each user ID is mapped to an continuous real value vector. Together with the label sequence, it is fed into RNN models, where we use the vanilla RNN [18], LSTM [19] and GRU [20]. The output is a hidden vector, then a linear layer with Softmax function is served as the classifier. The brief framework is shown in Figure 7.

B. Cross-sectional Model

However, it is inappropriate to model the task as a time series prediction problem. This is because the definition of time series data is a continuous observation on some specific properties of the same object. But in our dataset, the transactions are proposed by different users that have different network environments, thus, it is wrong to consider the transaction sequence as a time series.

Therefore, we model it as a cross-sectional data, where the label of t_i is determined by the features of t_i and t_{i-1}, and not related to time. Then it becomes a traditional binary classification task, where we can use a multi-layer perceptron [21] (MLP) to solve the problem.

As shown in Figure 6, the input features are the users of t_i and t_{i-1}, as well as the label of t_{i-1}. Then the users are also fed into an embedding layer to generate latent representations. Then we concatenate the label of t_{i-1} and feed it into the MLP to generate the probability distribution $\hat{P}$. Sadly, because our dataset is still too easy, the MLP is overpowered. Finally, we only use a single linear layer. The detailed performance will be given in Section V. Therefore, we plan to try machine learning (ML) models instead of deep learning.

C. Machine Learning Models

Since the task is to do classification with only five features, we try the following typical and widely-used ML models in classification tasks:

- SVM [22] (Support Vector Machine)
- KNN [23] (k Nearest Neighbors)
- DT [24] (Decision Tree)
- ET [25] (Extra Tree)
- RF [26] (Random Forest)
- CBT [27] (CatBoost)

Among these, CatBoost is one of the SOTA ML models that can achieve better performance than many DL models in various situations, especially in data mining competitions. CatBoost is a machine learning algorithm that can be used for both classification and regression tasks. It is based on gradient boosting decision tree [28] (GBDT) and is designed to improve the prediction accuracy of models by reducing overfitting and bias.

CatBoost is particularly well-suited for working with categorical data It can handle missing values and categorical features automatically, without the need for preprocessing. This makes it a good choice for tasks such as recommendation systems and customer segmentation, where there is often a large amount of categorical data. CatBoost is implemented in Python and is available as an open-source library. It can be used as a standalone library or integrated with other machine learning libraries, such as scikit-learn.

V. Experiments

A. Settings

For the time series model, the window size for the input of RNN was set as $w = 5$. The data ratio for training, validation, and testing was set as 7:1:2. For model parameters, the user embedding dimension $d_u = 8$, the hidden size of RNN $d_h = 32$, batch size was 64, and learning rate was 0.00001. Adam [29] algorithm was employed to control the overall training process, and the loss function was *Cross Entropy Loss*. The training process would be early-stopped if the validation loss was not

TABLE II: Model parameters.

Notation	Parameter	Value
Time-series Models		
w	Window size	5
d_u	User embedding dimension	8
d_h	Hidden size	32
d_o	Linear output dimension	2
	Batch size	64
	Early stopping epochs	10
	Learning rate	$1e-5$
Cross-sectional Model: MLP		
d_u	User embedding dimension	8
d_o	Linear output dimension	2
	Batch size	64
	Early stopping epochs	30
	Learning rate	$1e-4$
CatBoost		
	Max depth	6
	Loss function	Cross Entropy
	Early stopping epochs	100
	Learning rate	0.05

decreasing for 10 epochs, then the best model on validation data would be saved. We performed a thorough search on vector dimensions, and the complete parameter selection is given in Table II. For cross-sectional model, the learning rate was set to 0.0001, and other parameters are mostly kept unchanged. For CatBoost, the learning rate was set to 0.05 and the max depth of the tree was 6.

Our experiments were performed on a M1 Pro CPU with 16GB RAM. The PyTorch version is 1.13, the scikit-learn version is 1.1.3 and CatBoost version is 1.1.1.

B. Performance Analysis

The performance of the classification methods was compared in Table III, where the prediction accuracy was reported. Upon examining the table, several observations can be made.

- First, the accuracy of deep learning models was found to be significantly better than that of traditional machine learning models.
- To further demonstrate the ineffectiveness of time series modeling for this task, the training accuracy was also reported, showing that recurrent neural networks (RNNs) were unable to fit the data.
- On the other hand, the high accuracy of the cross-sectional model, a multilayer perceptron (MLP), suggests that our assumption about the suitability of cross-sectional data was correct.
- Additionally, it was found that the state-of-the-art machine learning model, CatBoost, was able to achieve even better performance than the deep learning model, making it a good choice for this dataset as it was not complex enough to warrant the use of deep learning.

In Figure 8, the training time per epoch for vanilla RNN, long short-term memory (LSTM), gated recurrent unit (GRU), MLP, and CatBoost was reported. From the figure, it can be seen that the MLP had the fastest training time due to its simple structure. The time series models were slower because they had to be trained sequentially, while CatBoost was faster than them due to being a deterministic machine learning model. However, CatBoost was slower than the MLP as it required a large amount of computation to generate subtrees.

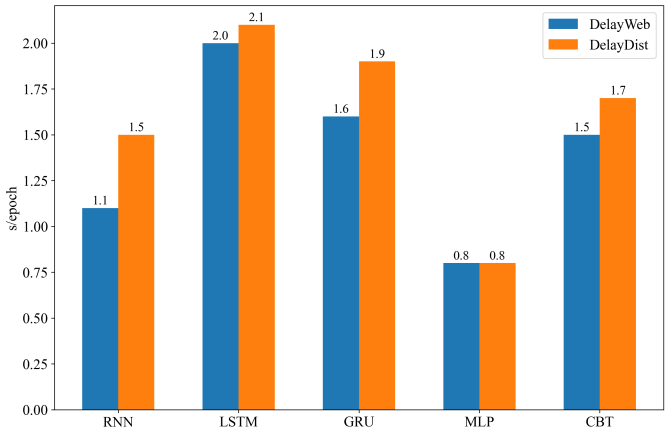


Fig. 8: Training time for each model.

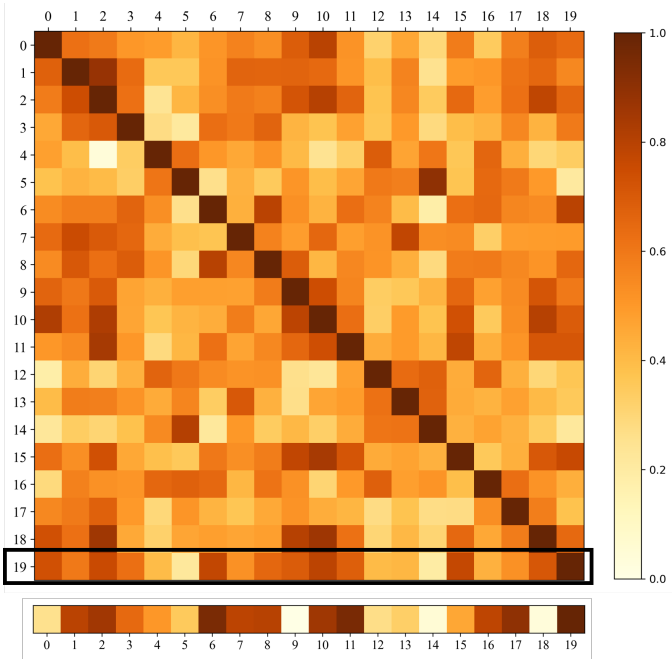


Fig. 9: User embeddings heat map.

C. Interpretability of User Embeddings

As stated before, the embedding layer maps each user to a vector representation that can reflect its semantic meanings. The similarity of embedding vectors can also measure the similarity of the behavior of users. To be specific, two users are similar if their embedding vectors are close in the latent space. In our task, the similarities can be calculated by $E \cdot E^T$. In Figure 9, we select the first 20 users in the DelayDist dataset and compute their similarity by the dot-product of their

TABLE III: Performance evaluation results.

Model	DelayWeb			DelayDist		
	Train	Valid	Test	Train	Valid	Test
SVM	79.63	79.08	77.61	51.25	50.44	49.63
KNN	81.30	75.69	74.26	68.82	49.75	50.00
DT	79.63	79.08	77.62	66.76	66.67	67.13
ET	79.68	79.09	77.63	66.77	66.63	67.12
RF	79.63	79.08	77.62	66.75	66.67	67.13
RNN	79.63	79.08	77.62	66.72	66.76	67.06
LSTM	79.64	79.09	77.62	66.72	66.78	67.09
GRU	79.63	79.09	77.62	66.72	66.76	67.11
MLP	85.42	81.64	80.48	88.66	**81.42**	73.78
CBT	82.85	**82.03**	**80.86**	88.59	80.95	**75.32**

embedding vectors, and show them in a heat map. According to the color bar, a darker color represents a higher similarity.

In the dataset, the attribute of each user is their mean latency, and the sub-distribution is generated by it. Therefore, the network environment of two users is similar if their mean latency is close. We select user u_{19} and visualize the closeness to other users' mean latency in the bottom of the figure. From the two heat maps, we can observe that (1) if the mean latency of two users are close, the similarity of their embedding vector will also be high. However, (2) high similarity does not conduct close mean latency. This is the ability of embedding technique to extract the underlying correlations between users, which cannot be calculated directly by the visible features.

VI. Conclusion

In this paper, we first researched the existing methods and shortages of blockchain. We propose that the network delays can lead to unfair order transaction problem. Then, we give a formal definition of the problem and model it as a binary classification task. Due to the lack of public datasets, we have to generate dataset by ourselves. To model the real network scenarios as close as possiable, we propose two approaches and their corresponding dataset DelayWeb and DelayDist. When developing methods, we first model it as a time series prediction task and proposed a traditional time series forecasting framework with an embedding layer and recurrent neural networks. However, after further analysis, we realize that the data should be modeled as cross-sectional. Therefore, we revise our approach to multi-layer perceptron. What's more, we also try several machine learning models due to the simpleness of our datasets. Extensive experiment results prove the effectiveness and rationality of our proposed frameworks. Following, a visualization of the embedding layer provides the interpreability of the model.

There are still many aspects that should be improved. However, we think the most important problem is how to create the dataset. Our datasets are still too simple that we cannot add any real world blockchain features into them, and it also results in the overpowering of MLP. Therefore, the improvement of datasets is the most urgent task in the future.

References

[1] A. A. Monrat, O. Schelén, and K. Andersson, "A survey of blockchain from the perspectives of applications, challenges, and opportunities," *IEEE Access*, vol. 7, pp. 117 134–117 151, 2019.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[3] M. N. M. Bhutta, A. A. Khwaja, A. Nadeem, H. F. Ahmad, M. K. Khan, M. A. Hanif, H. Song, M. Alshamari, and Y. Cao, "A survey on blockchain technology: Evolution, architecture and security," *IEEE Access*, vol. 9, pp. 61 048–61 073, 2021.

[4] G. W. Peters, E. Panayi, and A. Chapelle, "Trends in crypto-currencies and blockchain technologies: A monetary theory and regulation perspective," *arXiv preprint arXiv:1508.04364*, 2015.

[5] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International journal of web and grid services*, vol. 14, no. 4, pp. 352–375, 2018.

[6] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 3–16.

[7] A. Gervais, G. O. Karame, V. Capkun, and S. Capkun, "Is bitcoin a decentralized currency?" *IEEE security & privacy*, vol. 12, no. 3, pp. 54–60, 2014.

[8] S. Floyd and V. Paxson, "Difficulties in simulating the internet," *IEEE/ACM Trans. Netw.*, vol. 9, pp. 392–403, 2001.

[9] W. H. Inmon and T. J. Bird, "The dynamics of data base," 1986.

[10] L. Breslau, D. Estrin, K. R. Fall, S. Floyd, J. S. Heidemann, A. G. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, "Advances in network simulation," *Computer*, vol. 33, pp. 59–67, 2000.

[11] M. Li and S. C. Lim, "Modeling network traffic using generalized cauchy process," *Physica A-statistical Mechanics and Its Applications*, vol. 387, pp. 2584–2594, 2008.

[12] M. Li, W. Zhao, W. Jia, D. Long, D. Long, and C.-H. Chi, "Modeling autocorrelation functions of self-similar teletraffic in communication networks based on optimal approximation in hilbert space," *Applied Mathematical Modelling*, vol. 27, pp. 155–168, 2003.

[13] M. Li and W. Zhao, "Representation of a stochastic traffic bound," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, pp. 1368–1372, 2010.

[14] M. Li, W. Zhao, and C. Cattani, "Delay bound: Fractal traffic passes through network servers," *Mathematical Problems in Engineering*, vol. 2013, pp. 1–15, 2013.

[15] A. Wang and M. Li, "Statistical distribution model of network traffic time delay," 2017.

[16] W. Conover, "A practical guide to the kolmogorov-smirnov test as a goodness-of-fit test," *American Statistician*, vol. 35, no. 3, pp. 124–129, 1981.

[17] J. Maindonald and J. Wills, "The kolmogorov-smirnov test for goodness of fit: a review," *International Statistical Review*, vol. 68, no. 3, pp. 261–277, 2000.

[18] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 240–254, 1994.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.

[21] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.
[22] I. Steinwart and A. Christmann, *Support vector machines*. Springer Science & Business Media, 2008.
[23] O. Kramer, "K-nearest neighbors," in *Dimensionality reduction with unsupervised nearest neighbors*. Springer, 2013, pp. 13–23.
[24] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.
[25] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.
[26] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
[27] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," *Advances in neural information processing systems*, vol. 31, 2018.
[28] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.
[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.