Bitcoin Transaction Confirmation Time Prediction: A Classification View

Limeng Zhang[1,2], Rui Zhou[1(✉)], Qing Liu[2], Jiajie Xu[3], and Chengfei Liu[1]

[1] Swinburne University of Technology, Melbourne, Australia
{limengzhang,rzhou,cliu}@swin.edu.au

[2] Data61, CSIRO, Hobart, Australia
Q.Liu@data61.csiro.au

[3] Soochow University, Suzhou, China
xujj@suda.edu.cn

Abstract. With Bitcoin being universally recognised as the most popular cryptocurrency, more Bitcoin transactions are expected to be populated to the Bitcoin blockchain system. As a result, many transactions can encounter different confirmation delays. One of the most demanding requirements for users is to estimate the confirmation time of a newly submitted transaction. In this paper, we argue that it is more practical to predict the confirmation time as falling into a time interval rather than falling onto a specific timestamp. After dividing the future into a set of time intervals (i.e. classes), the prediction of a transaction's confirmation can be considered as a classification problem. Consequently, a number of mainstream classification methods, including neural networks and ensemble learning models, are evaluated. For comparison, we also design a baseline classifier that considers only the transaction feerate. Experiments on real-world blockchain data demonstrate that ensemble learning models can obtain higher accuracy, while neural network models perform better on the f1-score, especially when more classes are used.

Keywords: Transaction confirmation time · Bitcoin · Blockchain · Ensemble learning · Neural network

1 Introduction

Bitcoin, invented by Satoshi Nakamoto in 2008 [23], has become one of the most popular cryptocurrencies, with its market capitalization reaching 1100 billion in August 2021[1]. Meanwhile, many worldwide businesses, like Paypal, Microsoft, Overstock, etc., have embraced Bitcoin as one method of payment. As a result, more Bitcoin transactions are anticipated to be propagated into the Bitcoin blockchain system. However, the bulk of new transactions cannot be included together in the next block due to the limited capacity of a block. Transactions submitted to the Bitcoin system, therefore, incur confirmation delays. Concerned

[1] https://www.coindesk.com/price/bitcoin.

by this, it becomes vital to help a user to understand how long it may take for a transaction to be confirmed in the Bitcoin blockchain.

Most previous attempts on estimating the confirmation time for a transaction focus on predicting a specific timestamp or predicting the number of blocks a transaction needs to wait for before it is confirmed [3,7,10,12,13,18,32,33]. However, it is usually more practical to predict the confirmation time as falling into which time interval in future (e.g., within 1 h, between 1 h and 4 fours, and more than 4 h). Users also find such prediction informative and helpful. When estimating a specific timestamp, the first drawback to consider is the confirmation time variance, particularly for transactions that will be confirmed in the next block. Their confirmation times are affected by the remaining waiting time before the next block is generated. Even transactions with the highest fees need to consume such periods before get confirmed. This means, due to later submission, a transaction with a higher fee may take longer confirmation time than one with a lower fee in the same block. The second factor is that the mining time of a new block is unpredictable (ten minutes is in average [1]), so a relatively long time interval may correspond to different number of blocks in reality, resulting in another type of estimation variation. When utilizing the block interval as the confirmation time, the primary issue is the unbalanced distribution of transactions throughout each confirmation time (block interval), which may cause the estimation to be heavily dependent on a single transaction, particularly when transactions are scarce for a given block interval. For example, for the transactions confirmed in the block range 621001–621500, the maximum confirmation time is 162 blocks. 85% of transactions are confirmed within 5 blocks, with the remaining 15% scattered among the remaining 157 blocks. Consequently, a transaction falling in one of the remaining 157 blocks could have a considerable impact on the estimation result for that particular block. Furthermore, when the estimated confirmation time (in terms of both a time interval and a block interval) surpasses a certain level, users usually choose to pay a higher transaction fee in order to accelerate transaction confirmation. In conclusion, we argue that as long as the confirmation time falls within an acceptable range, a confirmation time range offered to system users is more practical. In this scenario, the prediction task can be considered as a classification problem.

Existing efforts on transaction confirmation estimation suffer from the following four types of drawbacks: (1) Estimation is not tailored for an individual transaction. The works in [12,13,33] estimate the confirmation time for a group of transactions as opposed to a single transaction. Among them, [12,13] estimate the average confirmation time of high-feerate class transactions and low-feerate transactions. Zhao et al. [33] estimate the average confirmation time of all unconfirmed transactions. (2) Models proposed in [7,16] only predict whether a transaction can be confirmed in the next block, addressing the issue as a binary classification problem. They might not be sufficient in practice because they are unable to provide more confirmation information. (3) Some assumptions are not realistic. The confirmation process is modeled as a steady-state queueing system [15], with the assumption that system transactions arrive at a slower rate than they are confirmed, and each time a fixed number of transactions can be confirmed [3,10,12,13,18,33]. In fact, there can be different number of transactions in a block, and the submission rate of transactions establishes a periodic pattern for each day and each week[2], and the rate of submission can exceed the rate of confirmation. (4) There is insufficient utilisation of information on transactions, blocks, and mempool, which can provide further information on the current blockchain system. For example, information in the block sequence can disclose the size and generation rate of future blocks.

In this paper, we compare the prediction performance of three different classification techniques. The first are neural network models, which have been demonstrated to be promising in a range of classification tasks [2,6,11,19,29,31]. It classifies based on neuron layers to discover intrinsic patterns among features. The second are ensemble learning models, which are another type of powerful classification techniques [25]. It makes predictions by combining the results of multiple classifiers. The final is a feerate-ranking baseline classifier that classifies transactions simply based on feerate.

To summarize, we have made the following contributions: (1) summarize and extract features related to transaction confirmation; (2) propose a strategy for discretizing confirmation time; (3) compare the performance of neural network models, ensemble learning models, and a baseline classifier in predicting transaction confirmation time; and (4) demonstrate the importance of incorporating additional features, such as block and mempool features.

The rest of this paper is organized as follows: Sect. 2 reviews the related work on transaction confirmation time estimation. In Sect. 3, we define the problem studied in this paper. In Sect. 4, we outline the selected confirmation features, confirmation time discretization strategy, and the studied classification models. Section 5 presents the classification performance of different models, and Sect. 6 concludes this paper.

2 Related Work

In the Bitcoin blockchain, a transaction will be broadcast across all system nodes once it is submitted to the blockchain system. If a transaction meets the requirements for validity [1], the miner nodes will add it to the mempool. Transactions in the mempool compete to be included in each miner node's candidate block. Then, each miner node will compete to link its own candidate block to the blockchain, a process known as "mining". Once a new block is successfully linked to the blockchain, its transactions are confirmed and its miner is rewarded. These transactions will then be removed from the miners' mempool [21]. In the Bitcoin blockchain, some works have been done on estimating transaction confirmation time. In some studies [7,16], the estimation issue has been modeled as a binary classification problem, predicting whether a transaction could be confirmed in the following block. Then, a number of traditional machine learning algorithms, including Support Vector Machine, Random Forest, AdaBoost, etc., are tested.

[2] Blockchain.com, https://www.blockchain.com/charts/transactions-per-second.

Some studies [3,10,12,13,18,33] choose to base their predictions on analysing the distribution of the transaction submission and transaction confirmation. Among them, the authors of [12,13] approach this estimation problem by modeling it as a bulk service queueing system $M/G^B/1$, with transaction arrival following the Poisson distribution and batches of transactions (B) being confirmed at a rate with a specified distribution. Balsamo et al. [3] describe it as another type of bulk service queueing systems, $M/M^B/1$, with transaction arrival following the Poisson distribution and the confirmation of batches following an exponential distribution. Zhao et al. [33] introduce a possible zero-transaction service to the traditional bulk queueing system, which adds the case of a zero-transaction block in the model. It assumes that transaction arrival follows a Poisson distribution and the batch confirmation follows a stochastic density function. Except for the queueing system solutions, researchers in [10,18] model the confirmation process as a Cramér-Lundberg process with a fixed rate of transaction arrival and an exponential distribution for the confirmation of a fixed number of transactions. Existing works provide insights on estimating the confirmation time based on different source of information. However, these solutions are constrained by either the model's preliminary assumptions or insufficient consideration of the balance between output precision and user expectations. To address these issues, we discretize the original transaction confirmation time into several intervals and do estimations based on these intervals.

3 Problem Definition

The confirmation of a transaction is a complex procedure affected by multiple factors, including the transaction itself, unconfirmed transactions in the mempool, mining policy, and system resources. Given a newly submitted transaction $\hat{tx}$, the studied problem is to predict its confirmation time interval $y \in \{y_1, \cdots, y_n\}$, where $\{y_1, \cdots, y_n\}$ are a set of non-overlapping confirmation time intervals, and they together constitute the future. The goal is to find a function $\mathcal{F}$ that can predict in which time interval a submitted transaction will be confirmed, based on various sources of information. We mainly consider three types of features listed below:

$$y = \mathcal{F}(\text{FeaInfo } (\hat{tx}), \text{ BlockInfo, MemInfo})$$

- **FeaInfo** $(\hat{tx})$ describes the transaction itself, including transaction feerate, transaction weight, transaction inputs, submission time, etc.
- **BlockInfo** refers to the characteristics of mined blocks. It maintains the information of prior blocks, such as historical transactions feerate, block size, block generation speed, etc., implicitly reflecting the volume and speed of future mining information.
- **MemInfo** provides information on unconfirmed transactions in the mempool, revealing the competition among the unconfirmed transactions in the mempool.

4 Methodology

This section starts with a summary of the transaction confirmation features. The technique of discretization is then utilised to discretize the confirmation time. Finally, neural network models, ensemble learning models, and a feerate-ranking baseline classifier are described for this prediction problem.

4.1 Feature Selection

Based on the confirmation process in the Bitcoin blockchain, we summarize three factors that contribute to transaction confirmation:

- **Transaction features** describe the unique details of a submitted transaction.
  - *transaction weight* measures transaction size owing to the Segwit upgrade[3].
  - *transaction feerate* is the transaction fee per size unit (each unit is approximately equivalent to a quarter of a transaction weight unit). Typically, transactions with a higher feerate are considered confirmed earlier than those with a lower feerate.
  - *number of inputs* and *number of outputs* are related to the validation cost. Miners need to check the legitimacy of the assets stated in each transaction input.
  - *transaction first-seen time* is the first time that a transaction is noticed by the blockchain. The first-seen time is used since it is difficult to determine the exact submission time of a transaction.
  - *mempool position* indicates the unconfirmed transactions that are typically expected to be processed earlier than this one. It sums up the weight of unconfirmed transactions with higher feerates.
- **Block states** reflect the characteristics of the mined blocks, including block size, block generation speed, transaction confirmation distribution, etc.:
  - *block weight* and *number of transactions* represent the capacity of a block in terms of transaction weight and transaction number included in this block.
  - *difficulty* refers to the mining difficulty, which is related to the computational cost of a miner node.
  - *block interval* is the interval between this block and the previous one, indicating the rate at which blocks are generated.
  - *average feerate* is the average feerate of transactions included in this block. It is calculated by dividing the total transaction fee by one-fourth of the total transaction weight. By introducing this feature, we aim to capture the feerate trend in continuous blocks.

[3] Segwit transactions relocate the unlocking script (witness) from within the transaction to an external data structure, resulting a smaller size in terms of its raw data.

  - *transaction confirmation distribution* is the distribution, in terms of transaction weight, of transactions in this block at each feerate interval.
- **Mempool states** indicate the competition among the unconfirmed transactions. It is modeled in terms of the weight of these unconfirmed transactions at each feerate interval, due to the limited capacity of a block and the precedence of transactions with greater feerates.

4.2 Confirmation Time Discretization

First of all, we choose to discretize the future time according to block intervals, rather than time intervals. This is mainly due to the unpredictability of a block's mining time, which records the confirmation time of many transactions confirmed together within a single block. For example, Fig. 1 illustrates the time range of all confirmed transactions with a 2-block confirmation interval. We can find that the duration of the 2-block interval could range from a few seconds to several hundred seconds, causing the time interval to potentially overlap with other block intervals. In addition, the index of block interval can handle submission time fluctuation. Due to later submission, a transaction with a higher feerate may take longer confirmation time than one with a lower feerate in the same block.

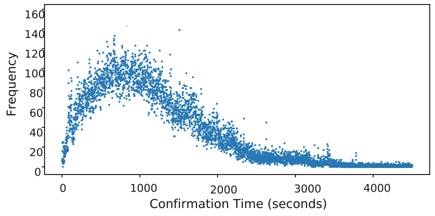


Fig. 1. The frequency of each confirmation time (seconds) with 2-block interval

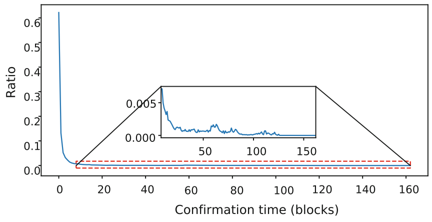


Fig. 2. The ratio of confirmation time (blocks) distribution in the blockchain

We design two guidelines to discrete the future time into a set of classes. The first is to make the number of transactions balanced for each class. The Bitcoin blockchain system exhibits the long-tail effect of transaction confirmation time, as shown in Fig. 2. Few transactions are confirmed after 10 blocks, with the majority of transactions being confirmed within 10 blocks. The second is that transactions with the same confirmation block interval are better to be grouped in the same class.

Specifically, the discretization is done in the following steps: (1) Initiate the remaining intervals using the required intervals k and the smallest confirmation time by a 1-block interval. (2) Determine the split ratio to classify the unclassified confirmation time range. It is calculated by dividing the remaining proportions by the remaining intervals. (3) Create a new discretized confirmation interval by adding the ratios beginning with the shortest confirmation time until their sum reaches the split ratio from step 2. (4) Replace the shortest confirmation time with the next confirmation time and reduce the remaining intervals by one. (5) Repeat steps 2–4 until k-1 intervals are obtained, at which point the remaining confirmation time range corresponds to the last discretization interval.

4.3 Classification Methods

Baseline Classifier (Baseline). The baseline classifier assumes that a transaction with a higher feerate will be confirmed earlier than one with a lower feerate. Specifically, it operates by first sorting historical confirmed transactions by feerate and then classifying them based on the fraction of the related class. For example, according to Table 1, transactions in Class 1 (1-block confirmation time) account for 62%, indicating that 62% of transactions are confirmed in the next block after submission. The baseline classifier will then assume that 62% of transactions with the highest fees are confirmed in the next block. Therefore, the criteria for classification will be based on the lowest feerate, 22.24, generated by the top 62% of transactions. Thus, a new transaction with a feerate higher than 22.24 will be classified as Class 1.

Table 1. Prediction results of Baseline under 4 classes

Classes	Discretization		Baseline
	Range_block	Ratio	(corresponding feerates)
Class 1	1	62%	≥22.24
Class 2	2	13%	[17.59, 22.24)
Class 3	3–7	13%	[7.09, 17.59)
Class 4	≥8	11%	<7.09

Neural Network Models (NN). Deep neural networks (DNNs) have emerged as a major force in the machine learning community, with applications in many areas [19,31], such as speech recognition, image classification, medical diagnosis, etc. DNNs are known for their capacity to discover complicated structures and acquire high-level concepts in data. Additionally, DNN makes it easier to incorporate additional information owing to its structure flexibility. Consequently, we adopt three major types of deep neural networks: the first is a fully connected network, Multi-Layer Perceptron (MLP), which predicts only based on transaction features, and the other models are neural networks with Long Short-Term Memory (LSTM) [11] and attention mechanisms [2,6,29]. LSTM and attention mechanisms are applied to capture inherent patterns in blocks and mempool, and both have showed remarkable performance in the processing of time series data

[8,22,26]. LSTM aggregates information on a token-by-token basis in sequential order, whereas attention mechanisms attempt to capture the relationships between different positions of a single sequence to generate a representation for the sequence. In this work, we employ three popular attention mechanisms: additive attention (Adv) [2], self attention (Self) [29] and weighted attention (Wht) [6].

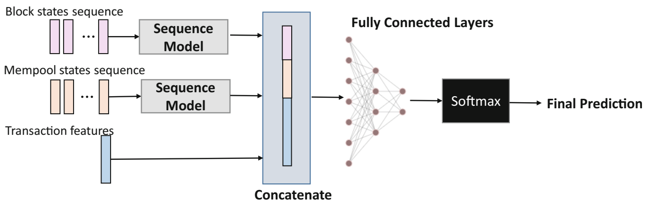


Fig. 3. The general architecture of neural network models.

The general structure of neural network models is shown in Fig. 3. It comprises two models: a feature extraction module and a prediction module. The function of the feature extraction module is to extract inherent patterns from transaction features, block states, and mempool states. Specifically, sequence processing models (only applicable in LSTM and attention models) are initially employed to derive patterns from historical block states and mempool states. The last time-step features derived from both block and mempool state sequence are then combined with transaction characteristics (only transaction features are involved in MLP) for further prediction. In the prediction module, fully-connected layers are stacked to handle the combined features from the feature extraction module. Then a softmax function is applied to generate the classification results.

Ensemble Learning Models (EL). Ensemble learning is known as the crowd wisdom of machine learning techniques. It enhances the performance of prediction by training multiple estimators and integrating their predictions. Figure 4 provides a general illustration of the structure. When each base classifier has finished producing a prediction result, output fusion is used to integrate all of the base model outputs into a single output [25]. In this paper, we study the classification performance of four state-of-the-art ensemble approaches: XGBoost [5], lightGBM [14], Random Forest (RF) [4] and Rotation Forest (RoF) [24], all of which are well-known for their outstanding performance in handling classification tasks. XGBoost is a cutting-edge gradient boosting framework of decision trees, which gains popularity in the 2015 Kaggle classification challenge. Compared to XGBoost, LightGBM employs histogram-based algorithms to reduce execution time and memory consumption. RF, ensembling decision trees based on the bagging technique, is popular owing to its generalized performance, high prediction

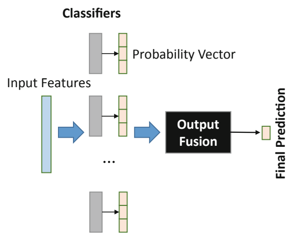


Fig. 4. The general architecture of ensemble learning models.

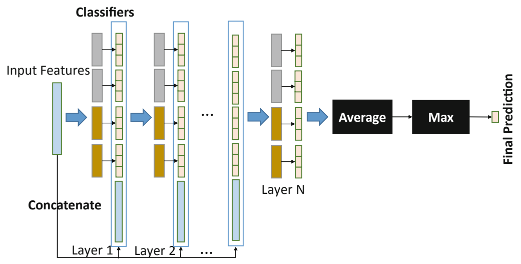


Fig. 5. The framework of deep forest (each level is composed of two random forests (grey) and two extremely randomized trees (yellow)) (Color figure online)

accuracy, and quick operation speed. Meanwhile, RoF has been demonstrated to score much better on classification tests than other ensemble approaches such as Bagging, AdaBoost, and Random Forest [24].

In addition, extensive studies have been made on the coupling of ensemble learning models with DNN techniques, driven by the outperformance of the neural networks and ensemble learning methods [17,20,30,36]. Among these methods, deep forest (DF) [36] has been proven effective in handling a range of classification tasks, including crop detection [34], medical diagnosis [27,28], software defect prediction [35], etc. Figure 5 illustrates the general framework of Deep forest (DF). It maintains the layer structure of DNN while replacing the neurones in the fully connected layers with base estimators (some ensemble learning models). In the work [9], the based estimators consist of two random forest models and two extremely randomised trees classifiers. In addition, it combines the output of the previous layer with the raw input feature as the new input for the subsequent layer. Finally, DF makes its prediction based on the prediction results of each base estimator in the final layer. In the training process, DF adaptively controls its layer complexity by terminating training when the required accuracy is achieved.

Further, among existing attempts with DF, The work [20] addresses the problem of price prediction, which is analogous to confirmation time prediction. Inspired by it, we adapt its framework as well as its penalty mechanism (DF_cost). The penalty mechanism operates as follows: if a sample of one class i is misclassified as class j, the model will incur a misclassification cost c_{ij}. The predicted class $\hat{y}$ is acquired by optimizing the objective function as follows:

$$\hat{y} = \underset{\hat{y}\in\{I_1,I_2,\dots I_n\}}{argmin} \mathcal{L}(y,\hat{y}) \tag{1}$$

$$\mathcal{L}(y,\hat{y}) = \sum_{i=1}^{n} p(\hat{y}|y)c_{ij} \tag{2}$$

$$c_{i,j} = |o_i - o_j| \tag{3}$$

where I_j denotes misclassifying class i as class j, and $p(\hat{y}|y)$ refers to the posterior probability of predicting the class y as $\hat{y}$. Meanwhile, $c_{ij} = c_{ji}$ and $c_{ii} = 0$. The cost c_{ij} is determined by the distance from the class centre (the mean of samples in each class), and the classification cost for the model is the overall cost of misclassification for all samples.

5 Experiments

5.1 Experiment Settings

Datasets. We collect transaction data from block range 621001–622500 via Blockchain.com[4]. Each dataset consists of 225 continuous blocks picked from every 250 blocks. The first 80% blocks in each dataset are utilised for training (about 400,000 transactions), while the remaining 20% are used for testing (about 100,000 transactions). The information regarding testing can be found in Table 2. In the experiments, only newly submitted transactions are selected for both training and testing.

Table 2. Testing data

Interval	Block information	Interval	Block information
1	621185–621229	4	621935–621979
2	621435–621479	5	622185–622229
3	621685–621729	6	622435–622479

Evaluation Metrics. To evaluate the performance of different models, we calculate the overall accuracy:

$$\text{accuracy (acc)} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{4}$$

We also utilise the macro-averaged f1-score, which is the arithmetic mean of all the per-class f1-score, as an indicator of classification performance:

$$\text{recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{5}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{6}$$

$$\text{f1-score} = \frac{2\text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \tag{7}$$

where TP (true positive), FP (false positive), FN (false negative), and TN (true negative) are observed classification results.

[4] https://www.blockchain.com/api/blockchain_api.

Compared Methods

- **NN** stands for neural network models.
  - **MLP** is a neural network model which only takes transaction features as input.
  - **Lstm** employs LSTM to extract patterns in block states and mempool states.
  - **Adv**, **Wht** and **Self** correspond to using different attention techniques to extract features from block states and mempool states: additive attention [2], weighted attention [6] and self attention [29].
- **EL** stands for ensemble learning models.
  - **RF**, **RoF**, **xgBoost** and **lightGBM** are four state-of-the-art ensemble learning models.
  - **DF** and its variants
    - **DF** refers to deep forest [9], with two random forest classifiers and two extremely randomized tree classifiers in each layer.
    - **DF_cost** introduces penalty into DF for misclassification [20].
    - **DF_xg** and **DF_xgRF** replace the base estimators in each layer of DF with four xgBoost classifiers and two random forest classifiers along with two xgBoost classifiers.
- **Baseline** stands for the baseline classifier.

Discretization. As shown in Table 3, we discretize the transaction confirmation time range into four different class sizes: $k = 2$, $k = 4$, $k = 6$, and $k = 8$. For $k = 2$, confirmation time is split into two categories: confirmed in 1 block interval and confirmed in more than 1 block (≥2 blocks). In such case, the problem can be considered as predicting whether a transaction will be confirmed in the next block [7,16]. Considering that transactions confirmed beyond 50 blocks are very rare at each confirmation time as shown in Fig. 2, and Class 8 in $k = 8$ in Table 3 has included transactions confirmed beyond 59-block interval, we stop discretizing the confirmation time into more classes.

Table 3. Confirmation time discretization (block intervals falling in each class)

Classes	k = 2		k = 4		k = 6		k = 8	
	Range	Ratio	Range	Ratio	Range	Ratio	Range	Ratio
Class 1	1	62.2%	1	62.2%	1	62.2%	1	62.2%
Class 2	≥2	37.8%	2	13.2%	2	13.2%	2	13.2%
Class 3			3–7	13.2%	3–4	8.5%	3	5.2%
Class 4			≥8	11.5%	5–8	5.5%	4–5	5.5%
Class 5					9–28	5.3%	6–9	4.0%
Class 6					≥29	5.3%	10–18	3.4%
Class 7							19–58	3.4%
Class 8							≥59	3.2%

Model Configuration. In the neural networks, the sequence processing model is configured with 8 hidden units and a sequence length of 3. Fully connected layers are a three-layer fully-connected neural network with 64 and 8 hidden units for the first two levels, and then the specified class size. The batch size is set to 1000 if applicable and models are optimised using stochastic gradient descent (SGD) with the Adam optimizer. In the ensemble learning models, RF, RoF and extremely randomized tree classifiers are set with 100 trees.

5.2 Result Analysis

Comparison on Classification Models. The accuracy and f1-score shown in Table 4 are the average of the results obtained from the six datasets. As class size increases, both the accuracy and f1-score performances of each model decrease. xgBoost achieves the most competitive performance among all discretization results, while Baseline performs the worst. The Baseline's worst performance exposes the complexity of the transaction confirmation mechanism, in contrast to the simplicity of the transaction priority.

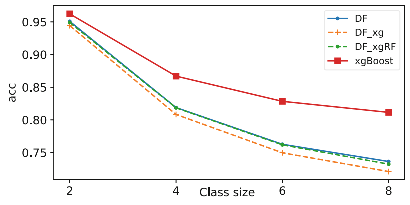
Table 4. An overall performance of models on 6 datasets

Methods		k = 2		k = 4		k = 6		k = 8	
		acc	f1-score	acc	f1-score	acc	f1-score	acc	f1-score
NN	MLP	91.52%	86.70%	82.85%	57.02%	79.20%	43.53%	77.29%	33.79%
	Adv	91.98%	87.61%	84.06%	60.43%	79.42%	**45.65**%	77.45%	36.27%
	Wht	91.89%	87.33%	83.39%	58.15%	78.80%	44.89%	77.38%	**37.37**%
	Self	91.99%	87.86%	82.45%	54.75%	78.82%	38.22%	76.49%	33.98%
	Lstm	91.74%	86.42%	81.11%	51.29%	77.13%	33.63%	76.78%	32.68%
EL	RF	96.03%	94.72%	86.31%	62.51%	82.07%	42.97%	80.29%	32.69%
	RoF	92.76%	89.82%	81.78%	55.62%	77.62%	38.62%	75.87%	29.14%
	xgBoost	**96.23**%	**94.92**%	**86.70**%	63.16%	**82.84**%	44.17%	**81.15**%	33.84%
	lightGBM	96.10%	94.67%	86.38%	62.73%	82.29%	43.20%	80.44%	32.52%
	DF	95.21%	94.49%	82.65%	63.14%	77.87%	45.03%	75.38%	34.02%
	DF_cost	–	–	82.64%	**63.25**%	77.69%	45.06%	73.88%	36.35%
Baseline		62.30%	56.61%	49.54%	29.10%	46.93%	19.87%	45.67%	14.70%

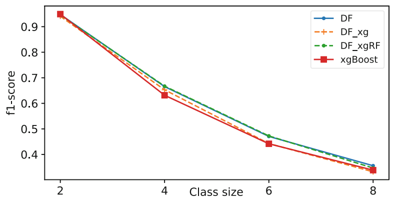
xgBoost, followed by lightGBM and RF, achieves the highest accuracy among all models on all four classification tests. In particular, when $k = 2$, its accuracy and f1-score are superior to those of other models. As the class size increases ($k \in \{4, 6, 8\}$), however, its f1-score advantage over DF and DF_cost diminishes. In addition, based on the performance of RF, DF and DF_cost, the strategy of assembling RF within their framework surpasses RF in terms of f1-score but fails to achieve superior prediction accuracy.

Despite the fact that neural network models perform poorly in terms of accuracy and f1-score on smaller class size ($k \in \{2, 4\}$), Adv achieves a very competitive f1-score on larger class size ($k \in \{6, 8\}$). Moreover, among all neural network models, Adv has the highest prediction accuracy. Moreover, the higher performance of Adv in comparison to MLP demonstrates the importance of block and mempool information.

Performance of DF Variants. This set of experiments aims to test the performance of DF variants by replacing the base estimators in each layer. According to Table 4, XgBoost fails to achieve comparable f1-score performance. However, by assembling RF into DF, DF can achieve a better f1-score than RF. Therefore, we attempt to incorporate xgBoost into DF. Specially, we substitute the original four estimators in DF with two random forests and two xgBoost classifiers (DF_xgRF) or four xgBoost classifiers (DF_xg) in each layer. According to results shown in Fig. 6(a) and Fig. 6(b), DF and its two variants outperform xgBoost in terms of f1-score but still fall short in terms of accuracy. In addition, we find that substituting estimators have no positive effect on the accuracy or f1-score of the DF framework.



(a) Accuracy of DF variants



(b) f1-score of DF variants

Fig. 6. Prediction performance of DF variants

6 Conclusion

In this study, we compare the performance of neural networks, ensemble learning models and a feerate-ranking baseline classifier on the prediction of transaction confirmation time as a classification problem. In terms of prediction accuracy, xgBoost provides the best classification results, whereas the neural network model applying additive attention delivers increasingly competitive f1-score performance as class size increases. In addition, we demonstrate that block and mempool information has a positive effect on improving neural networks prediction performance. Our future work will focus on two areas: incorporating block and mempool information into ensemble learning methods and boosting the predictive accuracy of neural network models.

Acknowledgements. This research is supported by Data61, Australian Research Council Discover (Grant No. DP170104747, No. DP180100212 and No. DP200103700) and National Natural Science Foundation of China (Grant No. 61872258).

References

1. Antonopoulos, A.M.: Mastering Bitcoin: Programming the Open Blockchain. O'Reilly Media, Inc., Sebastopol (2017)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
3. Balsamo, S., Marin, A., Mitrani, I., Rebagliati, N.: Prediction of the consolidation delay in blockchain-based applications. In: Proceedings of the ACM/SPEC International Conference on Performance Engineering, pp. 81–92 (2021)
4. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
5. Chen, T., et al.: XGBoost: extreme gradient boosting. R Package Version 0.4-2 **1**(4), 1–4 (2015)
6. Felbo, B., Mislove, A., Søgaard, A., Rahwan, I., Lehmann, S.: Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. arXiv preprint arXiv:1708.00524 (2017)
7. Fiz, B., Hommes, S., State, R.: Confirmation delay prediction of transactions in the bitcoin network. In: Park, J.J., Loia, V., Yi, G., Sung, Y. (eds.) CUTE/CSA -2017. LNEE, vol. 474, pp. 534–539. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-7605-3_88
8. Fu, R., Zhang, Z., Li, L.: Using LSTM and GRU neural network methods for traffic flow prediction. In: 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), pp. 324–328. IEEE (2016)
9. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Mach. Learn. **63**(1), 3–42 (2006)
10. Gundlach, R., Gijsbers, M., Koops, D., Resing, J.: Predicting confirmation times of bitcoin transactions. ACM SIGMETRICS Perform. Eval. Rev. **48**(4), 16–19 (2021)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
12. Kasahara, S., Kawahara, J.: Effect of bitcoin fee on transaction-confirmation process. J. Ind. Manag. Optim. **15**(1), 365 (2019)
13. Kawase, Y., Kasahara, S.: Priority queueing analysis of transaction-confirmation time for bitcoin. J. Ind. Manag. Optim. **16**(3), 1077 (2020)
14. Ke, G., et al.: LightGBM: a highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
15. Kleinrock, L.: Theory, vol. 1. Queueing Systems (1975)
16. Ko, K., Jeong, T., Maharjan, S., Lee, C., Hong, J.W.-K.: Prediction of bitcoin transactions included in the next block. In: Zheng, Z., Dai, H.-N., Tang, M., Chen, X. (eds.) BlockSys 2019. CCIS, vol. 1156, pp. 591–597. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-2777-7_48
17. Kontschieder, P., Fiterau, M., Criminisi, A., Bulo, S.R.: Deep neural decision forests. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1467–1475 (2015)
18. Koops, D.: Predicting the confirmation time of bitcoin transactions. arXiv preprint arXiv:1809.10596 (2018)
19. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
20. Ma, C., Liu, Z., Cao, Z., Song, W., Zhang, J., Zeng, W.: Cost-sensitive deep forest for price prediction. Pattern Recogn. **107**, 107499 (2020)
21. Ma, Y., Sun, Y., Lei, Y., Qin, N., Lu, J.: A survey of blockchain technology on security, privacy, and trust in crowdsourcing services. World Wide Web **23**(1), 393–419 (2020)

22. McNally, S., Roche, J., Caton, S.: Predicting the price of bitcoin using machine learning. In: 2018 26th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), pp. 339–343. IEEE (2018)
23. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. In: Decentralized Business Review, p. 21260 (2008)
24. Rodriguez, J.J., Kuncheva, L.I., Alonso, C.J.: Rotation forest: a new classifier ensemble method. IEEE Trans. Pattern Anal. Mach. Intell. **28**(10), 1619–1630 (2006)
25. Sagi, O., Rokach, L.: Ensemble learning: a survey. Wiley Interdisc. Rev.: Data Min. Knowl. Discov. **8**(4), e1249 (2018)
26. Srivastava, N., Mansimov, E., Salakhudinov, R.: Unsupervised learning of video representations using LSTMs. In: International Conference on Machine Learning, pp. 843–852 (2015)
27. Su, R., Liu, X., Wei, L., Zou, Q.: Deep-Resp-Forest: a deep forest model to predict anti-cancer drug response. Methods **166**, 91–102 (2019)
28. Sun, L., et al.: Adaptive feature selection guided deep forest for COVID-19 classification with chest CT. IEEE J. Biomed. Health Inform. **24**(10), 2798–2805 (2020)
29. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
30. Wen, G., Hou, Z., Li, H., Li, D., Jiang, L., Xun, E.: Ensemble of deep neural networks with probability-based fusion for facial expression recognition. Cogn. Comput. **9**(5), 597–610 (2017)
31. Zhang, G.P.: Neural networks for classification: a survey. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **30**(4), 451–462 (2000)
32. Zhang, L., Zhou, R., Liu, Q., Xu, J., Liu, C.: Transaction confirmation time estimation in the bitcoin blockchain. In: Zhang, W., Zou, L., Maamar, Z., Chen, L. (eds.) WISE 2021. LNCS, vol. 13080, pp. 30–45. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-90888-1_3
33. Zhao, W., Jin, S., Yue, W.: Analysis of the average confirmation time of transactions in a blockchain system. In: Phung-Duc, T., Kasahara, S., Wittevrongel, S. (eds.) QTNA 2019. LNCS, vol. 11688, pp. 379–388. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-27181-7_23
34. Zhong, L., Hu, L., Zhou, H.: Deep learning based multi-temporal crop classification. Remote Sens. Environ. **221**, 430–443 (2019)
35. Zhou, T., Sun, X., Xia, X., Li, B., Chen, X.: Improving defect prediction with deep forest. Inf. Softw. Technol. **114**, 204–216 (2019)
36. Zhou, Z.H., Feng, J.: Deep forest: towards an alternative to deep neural networks. In: IJCAI (2017)