Computer Systems
Science and Engineering

Tech Science Press

DOI: 10.32604/csse.2023.039569
Article

Detecting Ethereum Ponzi Schemes Through Opcode Context Analysis and Oversampling-Based AdaBoost Algorithm

Mengxiao Wang[1,2] and Jing Huang[1,2,*]

[1]Faculty of Information Technology, Beijing University of Technology, Beijing, 100124, China
[2]Beijing Key Laboratory of Computational Intelligence and Intelligence System, Beijing, 100124, China
*Corresponding Author: Jing Huang. Email: huangjing@bjut.edu.cn
Received: 06 February 2023; Accepted: 13 April 2023; Published: 26 May 2023

Abstract: Due to the anonymity of blockchain, frequent security incidents and attacks occur through it, among which the Ponzi scheme smart contract is a classic type of fraud resulting in huge economic losses. Machine learning-based methods are believed to be promising for detecting ethereum Ponzi schemes. However, there are still some flaws in current research, e.g., insufficient feature extraction of Ponzi scheme smart contracts, without considering class imbalance. In addition, there is room for improvement in detection precision. Aiming at the above problems, this paper proposes an ethereum Ponzi scheme detection scheme through opcode context analysis and adaptive boosting (AdaBoost) algorithm. Firstly, this paper uses the n-gram algorithm to extract more comprehensive contract opcode features and combine them with contract account features, which helps to improve the feature extraction effect. Meanwhile, adaptive synthetic sampling (ADASYN) is introduced to deal with class imbalanced data, and integrated with the Adaboost classifier. Finally, this paper uses the improved AdaBoost classifier for the identification of Ponzi scheme contracts. Experimentally, this paper tests our model in real-world smart contracts and compares it with representative methods in the aspect of F1-score and precision. Moreover, this article compares and discusses the state of art methods with our method in four aspects: data acquisition, data preprocessing, feature extraction, and classifier design. Both experiment and discussion validate the effectiveness of our model.

Keywords: Blockchain; smart Ponzi scheme; n-gram; oversampling; ensemble learning

1 Introduction

Blockchain stores all value transfer processes based on cryptocurrency transactions [1]. Due to the characteristics of openness and transparency, tamper-proof and traceability, blockchain technology has laid a solid foundation of trust and created a reliable cooperation mechanism, which has a very broad application prospect. For example, Kumar et al. apply blockchain to Industrial

Internet of Things (IIOT) [2] and Softwarized Unmanned Aerial Vehicles (UAV) environments [3] to protect data privacy and build trusted communication entities. Ethereum is a blockchain-based decentralized application platform, known as Blockchain 2.0 [4]. Smart contracts are applications that run on an Ethereum virtual machine. In smart contracts, when preset conditions are met, contract terms written in computer programs will be automatically executed. Traditional contracts need to be overseen by a trusted third party, so they take longer and incur additional costs. In contrast, smart contracts are stored and updated in a distributed blockchain, enabling secure and efficient automatic peer-to-peer trusted transactions [5]. In a nutshell, smart contracts are agreements between participants who do not trust each other and are automatically executed by the blockchain's consensus mechanism, rather than relying on a trusted authority [6].

As the growing popularity of blockchain has attracted more and more users, Kumar et al. have provided a variety of solutions to the large-scale data storage problems it poses [7,8]. However, due to the complexity and lack of supervision of this new technology, criminals take advantage of blockchain's anonymity, immutable, automatic execution of smart contracts, and high credibility to carry out propaganda to obtain false trust and launch a series of fraud activities [9,10]. Many of the frauds are related to Ponzi schemes, which are a common scheme in traditional financial investment, also known as "empty handed white wolf". Simply put, it is to use the money of new investors to pay short-term returns to old investors, creating the illusion of making money, and then cheating more investment [11]. Nowadays, criminals are introducing Ponzi schemes into the blockchain, causing huge economic losses to investors [12]. Studies estimate that from September 2013 to September 2014, bitcoin-based Ponzi schemes raised more than $7 million [10]. Ponzi schemes use smart contracts as camouflage, which brings extremely expensive losses to users and seriously endangers the security of the blockchain ecosystem. Therefore, it is urgent to find a way to realize the detection of smart contracts of Ponzi schemes.

Detecting a Ponzi scheme on Ethereum is not an easy task. The Ethereum-based smart contract is a series of EVM bytecodes, and smart contract source code written in a high-level language needs to be compiled into bytecode by a compiler before it can run on an Ethereum virtual machine. Bytecode, on the other hand, is a string of byte arrays encoded by hexadecimal digits, from which it is difficult to extract effective features. Each byte of EVM bytecode corresponds to a human-readable operation, and by decomposing the bytecode, it can be converted into an easy-to-understand opcode [13,14]. In simple terms, bytecode is produced by compiling contract source code, and opcode is the result of disassembling contract bytecode. Fig. 1 shows the conversion process between the source code, bytecode and opcode of the ZeroPonzi contract. According to this transformation relationship between contract source code, bytecode and opcode, we can analyze the code logic of smart contracts by parsing contract opcode when contract source code is not available. These transactions of smart contracts are transparent and traceable, so their trading behavior is a good reflection of smart contracts.

Based on the above characteristics, we can detect a Ponzi scheme from the two dimensions of contract code and contract transaction. At present, many scholars detect Ponzi scheme contracts from these two perspectives. Relevant studies can be divided into three types. The first type is based on source code inspection, which mainly checks and identifies Ponzi schemes by manually checking contract source code [10,12,15]. The second type is based on feature engineering, which mainly represents the contract by designing some features and then inputting them into the machine learning model to realize the detection of a Ponzi scheme [13,14,16–24]. The main difference between different feature-based methods is how to select effective features. The third type is based on network embedding. This method mainly integrates the formation process of the trading network, the operation code of

the smart contract, and other information into a low-dimensional continuous vector through node embedding technology, and then models the detection problem of Ponzi scheme contract into a node classification task [25,26].

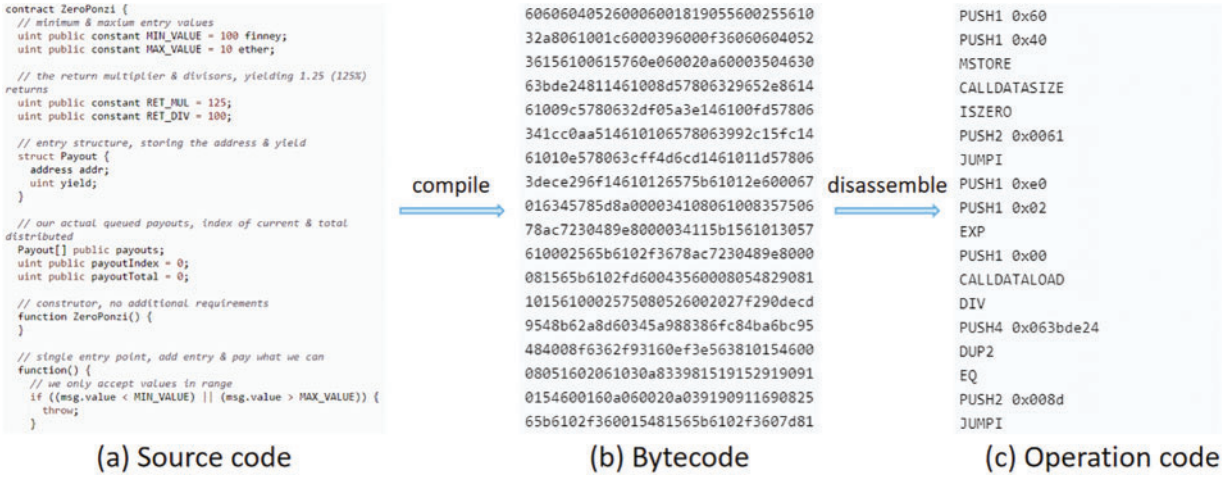

Figure 1: Smart contract transcoding

However, current detection methods still have shortcomings. First, the code features selected do not take into account the logical relationships of the contract source code [13,14,16,18]. A smart contract is a logical code protocol and a single opcode frequency does not reflect this contract logic. Second, most models do not address the problem of imbalanced datasets well [13,14,21]. The number of Ponzi contracts on Ether is very small compared to the total number of contracts, and there is a class imbalance problem. The detection of Ponzi schemes is essentially an imbalanced binary classification problem, but many research methods do not currently address this. Third, there is room for improvement in the detection accuracy of the model, and a more effective method for processing and classifying features is needed to better improve the detection effect.

This paper proposes an ethereum Ponzi scheme contract detection model using AdaBoost algorithm based on oversampling combined with opcode context characteristics. First, to better extract the logical relationship of the contract code, we use the n-gram algorithm to extract more comprehensive contract opcode features and combine them with account features that represent the characteristics of the contract transaction to construct the contract feature data set. Secondly, to deal with the class imbalance in the Ponzi scheme smart contract dataset, the adaptive synthetic sampling (ADASYN) method [27] is adopted to extract effective characteristic data for detection. Third, we trained the dataset using the better-performing AdaBoost [28] and combined oversampling with AdaBoost. The weight assignment of the AdaBoost classifier to the training samples is indirectly changed by introducing ADASYN in each round of augmentation, thus allowing each weak classifier to better learn features from the Ponzi scheme contract samples. Finally, the detection of Ponzi scheme contracts is improved.

The main contributions of this paper are as follows:

- The opcode feature extraction method has been improved. By using the n-gram algorithm to extract opcode context features, we can better represent the logical relationship of the contract source code.

- We use the ADASYN algorithm to oversample characteristic datasets to solve the detection problem of the ethereum Ponzi scheme with class imbalance.
- The AdaBoost classifier has been improved by combining with ADASYN, so as to better learn the features of Ponzi contracts and improve the detection of the model.
- Experiments on large-scale datasets and comparisons with other representative methods. The results verify the validity of the model.

The rest of this paper is organized as follows. In Section 2, we summarize the related works. In Section 3, we give a detailed description of the design and construction of the model. In Section 4, we show experimental studies and results. In Section 5, we have a discussion. Finally, we conclude this article and propose future research directions in Section 6.

2 Related Work

2.1 Ponzi Scheme Detection

The combination of Ponzi schemes and smart contracts has created new forms of fraud that have serious implications for the Ethereum ecosystem. Combining the characteristics of the two, we can analyze the Ponzi scheme contract from two dimensions: first, by analyzing the code, mining the business logic of the contract; second, by analyzing the transaction records of the contract, the transaction characteristics of the contract are explored. Song et al. [29] summarized and analyzed the important features of Ponzi scheme contracts based on relevant literature, and established an effective method for investors to distinguish Ponzi scheme contracts in blockchain. At present, research related to Ponzi scheme contract detection can be divided into three categories: The first type is based on source code inspection, which mainly analyzes the contract control logic by manually checking the contract source code. Chen et al. [15] analyzed four types of Ponzi scheme contract bytecode level patterns by analyzing contract source code logic. Bartoletti et al. [12] established a standard for classifying a contract as a Ponzi scheme by manually analyzing open source code to retrieve Ponzi scheme contracts. Normalized Levenshtein Distance (NLD) is proposed to compare contract similarities, so as to further dig out the Ponzi scheme hidden in non-open source contracts. The downside of this approach is that it only finds Ponzi contracts that are similar to the bytecode of known Ponzi contracts, and the code is cumbersome to check and requires a lot of human resources. The second type is based on feature engineering. With the development of data mining methods and machine learning technology in this field, a set of features is designed to represent a smart contract, and then it is input into the machine learning model to realize Ponzi scheme detection in smart contracts. Chen et al. [13,14] first extracted account characteristics and frequency characteristics of individual opcodes were extracted from transaction data and opcodes of smart contracts, and then classification models based on machine learning were constructed to detect potential Ponzi schemes in smart contracts. Zhang et al. [16] innovatively added bytecode features and then used improved LightGBM to identify Ponzi scheme contracts. Zhang et al. [17] proposed to use 2-gram opcodes to more objectively reflect the correlation of adjacent operation instructions. Wang et al. [18] consider account characteristics and code characteristics at the same time and propose a combination of SMOTE and LSTM for Ponzi scheme contracts with sample imbalance problem. Sun et al. [19] captured the changing characteristics of contracts in their transaction process through behavior forest, so as to realize the identification of newly deployed Ponzi scheme contracts. Fan et al. [20] proposed a model using the idea of ordered enhancement. They used ordered target statistics (TS) to deal with class characteristics, avoid prediction bias problems, and improve the model's detection performance. Chen et al. [21] extracted semantic and structural features of codes and a method based on Text

Convolutional Neural Networks. Jung et al. [22] used data mining methods to detect Bitcoin addresses related to Ponzi schemes. Lou et al. [23] proposed an improved convolutional neural network as a Ponzi scheme detection model in smart contracts to overcome the training difficulties caused by different bytecode lengths of smart contracts. Bian et al. [24] processed contract features into grayscale images and proposed a capsule network to identify Ponzi scheme contracts. Although these feature engineering detection methods can realize the automatic detection of the Ponzi scheme contract well, there are still some problems such as an imbalanced data set, the selected training features do not take into account the logical relation of the contract source code, and the poor portability of the model. The third type is based on network embedding. This method mainly integrates the transaction formation process and the operation code of the smart contract into a low-dimensional continuous vector through dynamic node embedding techniques and uses a binary classifier to identify the Ponzi scheme. Liang et al. [25] proposed for the first time the realization of intelligent contract Ponzi scheme detection through dynamic graph embedding techniques, and proposed a data-driven intelligent Ponzi scheme detection system DSPSD, which can directly predict whether a contract account has realized a Ponzi scheme according to the account operation code and transaction data. Yu et al. [26] modeled the identification and detection of Ponzi schemes as a node classification task. By manually extracting the basic features of smart contracts and combining these features with the topology of the trading network, to identify a Ponzi scheme. However, existing node embedding methods will lose some node information, and cannot combine the node characteristics and topology of the transaction network well.

2.2 Imbalanced Data Classification

In smart contract transactions, fraudulent Ponzi scheme transactions account for only a small part, while most of them are normal transactions. In essence, Ponzi scheme detection on Ethereum is an imbalanced classification problem. If the sample numbers of different categories are very different, the model learning process will be troublesome. Therefore, before constructing the classification model, it is necessary to deal with the imbalance of the data set. The solution to the class imbalance problem in this type of data set preprocessing is to change the class distribution. Because of the class imbalance problem in data sets, some scholars are committed to eliminating most class samples to solve the class imbalance problem. After improving and adjusting the undersampling method, better classification results can be obtained. Random Under Sampling [30] is a relatively simple method. However, ignoring most sample data can lead to the loss of some important information and reduce the learning effect of the model. Another group of scholars focuses on minority samples to address the class imbalance problem. Random Over-Sampling [31] is the earliest proposed over-sampling method, which primarily aims to balance data by randomly copying minority samples. However, this simple replication method easily causes the model's overfitting problem, and the data after random replication is not representative. SMOTE can effectively reduce the risk of overfitting due to random copying of data, but when there are significant noise and boundary samples in the dataset, SMOTE cannot control the smote sample and may lead to overgeneralization of the noise sample and increase overlap between different decision boundary classes. This does not significantly improve the forecast [32]. BorderLine SMOTE is an improvement on SMOTE. This algorithm uses only a few class samples on the boundary to make new samples and does nothing with the rest [33]. Similar to Borderline SMOTE, ADASYN uses a weight distribution based on minority learning difficulties. The strongest feature of this is some mechanism to automatically determine how many SMOTE smote should be produced per minority class, instead of Smote Smote by the same number per minority class, as in Smote, ADASYN can adapt itself to make the classification boundary more reasonable [34] and reduce the deviation caused

by class imbalance. In this model, we use the ADASYN algorithm to process a small number of Ponzi scheme contract samples. By oversampling the characteristic data of the Ponzi scheme contract in the training set, we can balance the two types of sample data in the training set quantitatively, thus enhancing the model's effect.

3 Model

This section summarizes the implementation process of the model, including data acquisition, feature extraction, class imbalance processing of the dataset, and classifier training. Fig. 2 is the overall process of our approach. First, we obtained valid contract information from the Etherscan website[1], mainly having contract opcodes and contract transaction records. Then, the corresponding opcode and account characteristics are extracted to construct the smart contract feature data set. Here, we focus on extracting opcode context features to better reflect contract logic. Finally, combining ADASYN technique [34,35] with the AdaBoost algorithm [27,28], the introduction of ADASYN in each round of boosting enables each weak learner to learn from a sample of Ponzi scheme contracts. To complete model training by better modeling minority classes in the training set. After the training, the test set data are predicted to verify the effectiveness of the model.

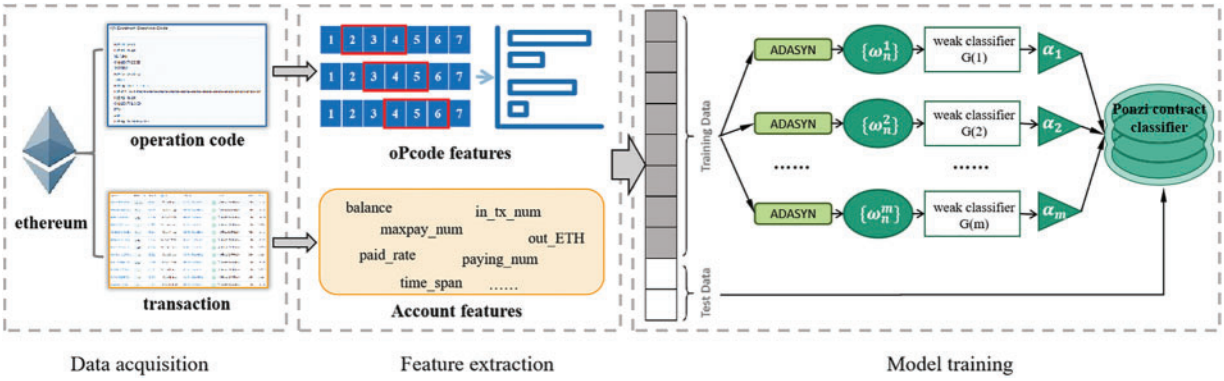

Figure 2: Framework of the model

3.1 Data Acquisition

We downloaded validated labeled datasets from the XBlock website[2] provided by Chen et al. [13]. This dataset currently includes 200 Ponzi scheme contract addresses and 3590 normal contract addresses, which is a class-imbalanced dataset. We crawled the bytecode of each contract and all transaction records (both external and internal) from Etherscan website based on the contract addresses provided in this dataset. Contract bytecode (see Fig. 1b) is a string from which it is difficult to obtain valid features, which requires converting them into human-readable opcodes (see Fig. 1c). Therefore, we split the bytecode into opcodes according to the Ethereum Yellow Book to better read the functional features of the contract.

3.2 Feature Extraction

The features of Ponzi scheme smart contracts are extracted from the perspective of contract code and contract account. In terms of contract code, since most source code is not published and is difficult

[1] https://etherscan.io/
[2] http://xblock.pro/

to obtain, it is not practical to use source code to extract contract characteristics. In contrast, the bytecode of contracts is public and easily available. It can be transformed into opcode, which contains some semantic information from the source code, so that it can be used in feature extraction. Therefore, we choose to disassemble the contract bytecode and convert it into opcode to extract the code features. On the other hand, contract transaction data records the transaction relationship between the user and the contract, as well as a series of actions triggered when the contract meets the conditions. Therefore, we can also extract account characteristics from transaction data to detect abnormal trading behavior. In a word, we choose to extract opcode features and account features to describe Ponzi scheme contracts.

3.2.1 Opcode Features

The smart contract opcode is disassembled from EVM bytecode, which is compiled from the source code. Contract opcodes may contain the logic of the source code, so we can dig into the potential problems of smart contracts from the contract opcodes. To intuitively feel the opcode difference between Ponzi scheme contracts and normal contracts, we compared the opcodes of different contracts and found that there is a significant difference in opcode frequency between Ponzi scheme contracts and normal contracts. Therefore, opcode frequency can be used as a valid feature of Ponzi scheme smart contract detection.

Most of the existing research has extracted a single opcode frequency of a contract, and then used it as an opcode feature to train the classification model. However, a single opcode may not reflect the logical relationship of the contract. Therefore, we consider using the n-gram algorithm to extract the context logic between contract operation codes, so as to reflect the code characteristics of the contract more comprehensively.

N-gram is an algorithm based on a statistical language model [36]. It will perform a sliding window operation of size n for the contract opcode to form a sequence of opcode fragments of length n. Each fragment is called a gram, and then the frequency of occurrence of all grams is counted. Specifically, we used an n-size sliding window for word segmentation of the contract opcode, and then used the term frequency-inverse document frequency (TF-IDF) algorithm to calculate the frequency of each opcode string. In this way, each operation code will be associated with the adjacent operation code, so that the context of the contract operation code can be extracted to reflect the logical characteristics of the contract source code more objectively and accurately.

3.2.2 Account Features

Smart contract transaction data records the money flow relationship between investors and smart contracts, including external and internal transactions. In terms of trading, there are many differences between Ponzi scheme contracts and ordinary smart contracts. Vasek et al. analyzed the supply and demand of a Bitcoin Ponzi scheme, determining factors influencing the persistence of fraud, and noted that the frequency of interaction between scammers and victims would have an impact on the life of the scheme [37]. Chen et al. [13] summarized the characteristics of contract trading in a Ponzi scheme through hand capital examination. Based on a summary and observation of the transaction behavior of Ponzi scheme contracts, we extract the following 17 characteristics from contract transaction records, and Table 1 shows the statistics for these characteristics.

1) Balance: the balance of smart contracts.
2) Maxpay_num: the maximum number of returns received by the investor.
3) Paid_rate: the percentage of investors who received one or more returns.

4) Difference_index: indicates the difference between the income and investment amount of participants.
5) Difference_counts_mean: the average value of the difference between the number of investments.
6) Difference_counts_standard: standard deviation of the difference of investment quantity.
7) Difference_amounts_mean: indicates the average difference of the investment amount.
8) Difference_amounts_skewness: difference of the investment amount.
9) Known_rate: the percentage of payers who have invested before receiving returns.
10) Payment_time: investment time of the participant.
11) In_tx_num: the number of transactions invested in the contract.
12) Out_tx_num: the number of transactions in which the contract pays returns to the investor.
13) In_ETH: the amount invested in the contract.
14) Out_ETH: the amount paid by the contract to the investor.
15) Paying_num: the number of users invested in the contract.
16) Paid_num: the number of users to be paid by the contract.
17) Time_span: the time interval between the first and last trade of the contract.

Table 1: Transaction characteristics statistics

	Ponzi			Non-Ponzi		
	Mean	Median	Std	Mean	Median	Std
Balance	1.15	0.00	5.05	142.82	0.00	3777.66
Maxpay_num	23.34	1.00	81.68	84.57	0.00	682.38
Paid_rate	**0.39**	**0.33**	**0.41**	**0.13**	**0.00**	**0.31**
Difference_index	0.22	0.00	1.76	−0.40	0.00	3.09
Difference_counts_mean	0.93	0.00	20.10	−6.82	0.00	112.70
Difference_counts_standard	2.96	0.00	12.37	8.89	0.00	135.78
Difference_amounts_mean	−1.33	0.00	10.78	−397.49	0.00	8720.46
Difference_amounts_skewness	−0.56	0.00	3.80	−0.73	0.00	4.88
Known_rate	**0.27**	**0.00**	**0.36**	**0.10**	**0.00**	**0.27**
Payment_time	**0.28**	**0.15**	**0.34**	**0.12**	**0.00**	**0.29**
In_tx_num	133.64	9.50	416.49	766.56	5.00	2319.54
Out_tx_num	54.39	2.00	165.44	162.60	0.00	1068.62
In_ETH	368.13	1.60	1914.88	13312.85	0.00	292660.54
Out_ETH	367.08	1.25	1913.86	17516.37	0.00	367637.73
Paying_num	42.72	2.00	300.59	67.86	0.00	468.83
Paid_num	6.26	1.00	16.82	20.47	0.00	283.07
Time_span	**16424499.36**	**69316.00**	**36799130.17**	**16059999.38**	**88114.00**	**36685802.54**

By comparison, it can be found that there are obvious statistical differences between normal contracts and Ponzi scheme contracts in the above 17 account characteristics. With the exception of paid_rate, known_rate, Payment_time, and time_span, the standard deviation of the remaining 13 Ponzi scheme contract characteristics is much lower than that of normal contracts, which implies that there are many similar user behaviors in Ponzi scheme contracts. So we can use these trading characteristics to detect Ponzi schemes.

For the four account characteristics whose differences are not intuitive, we use a summary scatter diagram for visualization, as shown in Fig. 3.

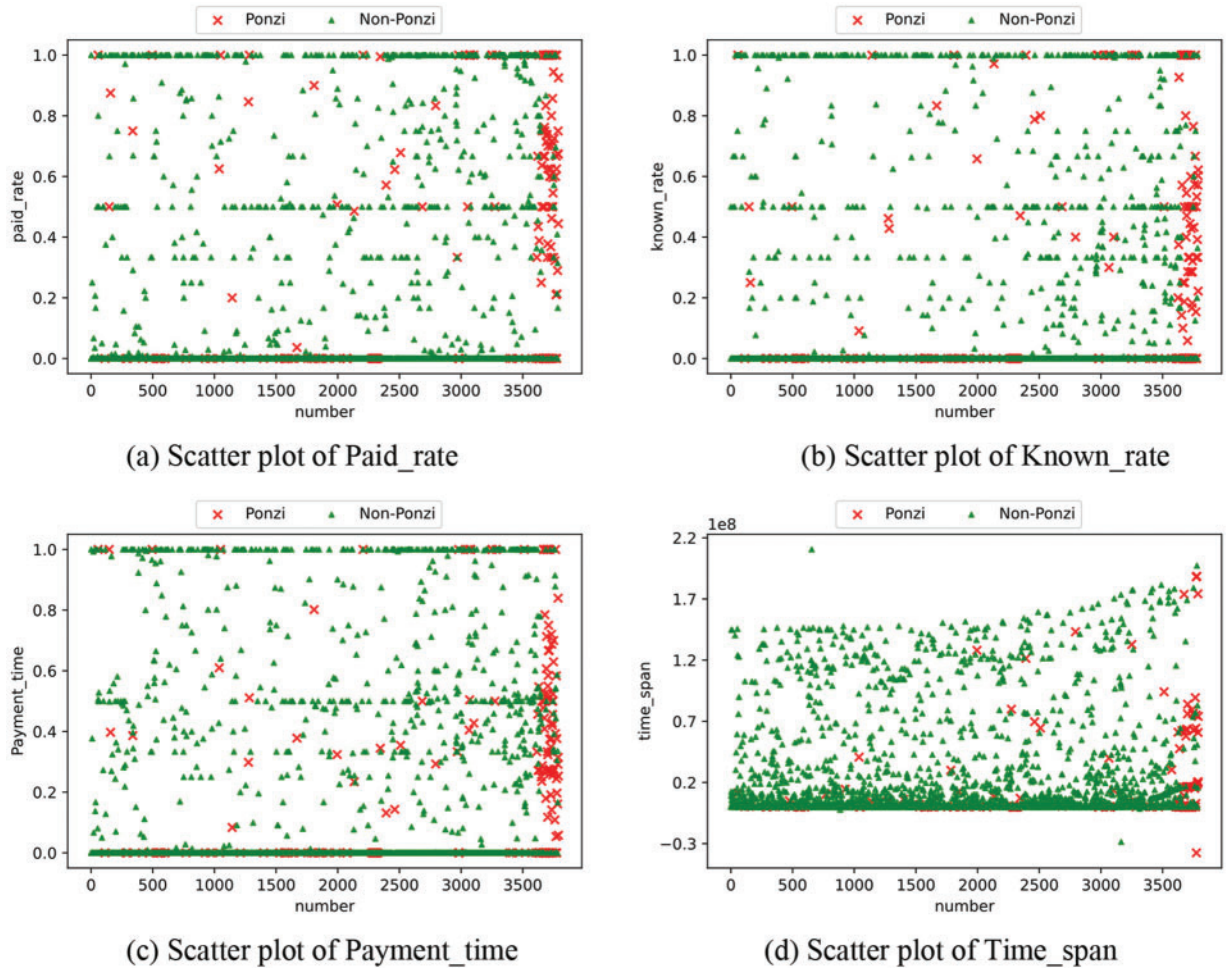

Figure 3: Scatter plot of different account characteristics

As shown in Figs. 3a–3c, for Ponzi scheme contracts, Paid_rate is mainly between 0.3 and 0.8, known_rate is concentrated between 0.2 and 0.6, and Payment_time is concentrated between 0.1 and 0.7. The Paid_rate value, known_rate value, and Payment_time value of normal contracts are concentrated in the 0 or 1 position. As shown in Fig. 3d, Time_span is mainly between 0 and 2 * e7 in Ponzi scheme contracts, while in non-Ponzi scheme contracts it is concentrated around 7 * e7. Therefore, these four account data can also be used as transaction features.

3.3 Class Imbalance Processing of Data Set

Based on the above code and account features, we obtain a data set of contract characteristics containing 3590 normal contracts and 200 Ponzi scheme contracts, which is a seriously imbalanced dataset. Because the majority of samples occupy too much proportion in the total samples, the minority of samples are ignored, and the trained classifier is more inclined to the majority of classes, leading to degradation of classifier performance.

In order to solve this class imbalance problem, we use the ADASYN oversampling algorithm to process the data. ADASYN automatically determines how many to combine per minority sample based on the data distribution, not SMOTE Smote with the same amount per minority sample [27]. Assuming that the majority class samples are m_l and the minority class samples are m_s. First, we need to set the number of minority class samples to be synthesized according to Eq. (1), where $\beta \in [0, 1]$, to control the desired balance level.

$$G = (m_l - m_s) \times \beta \tag{1}$$

For each minority class sample x_i, the Euclidean distance is used to calculate the k neighbors of x_i, and Δ_i is the number of majority class samples in k neighbors of x_i. Then, Eq. (2) can be used to calculate the ratio r_i, and Eq. (3) can be used for normalization processing to obtain the situation of most samples around a few samples x_i. Next, calculate the number of samples to be generated for x_i according to Eq. (4).

$$r_i = \Delta_i / k, \quad i = 1, \ldots, m_s \tag{2}$$

$$\hat{r}_{\mathrm{i}} = r_i \Big/ \sum_{i=1}^{m_s} r_i \tag{3}$$

$$g_i = \hat{r}_{\mathrm{i}} \times G \tag{4}$$

Finally, a minority sample x_{zi} is randomly selected from k neighbors of x_i. Generate the synthesis sample according to Eq. (5). This step is repeated until the number of synthesized samples reaches g_i.

$$s_i = x_i + (x_{zi} - x_i) \times \lambda \tag{5}$$

The ADASYN algorithm is amount to adding a weight to each minority class sample, and the more majority class samples around, the higher the weight. Use the ADASYN algorithm to oversample the characteristic data of Ponzi scheme contracts in the training set, which can balance the sample data in the training set quantitatively, thus improving the effect of the model.

3.4 Model Training

We use AdaBoost to learn contract code characteristics and account characteristics to realize Ponzi scheme detection. AdaBoost (Adaptive Boosting) is a kind of Boosting algorithm proposed by Wang [35] in 1995, which has strong practical advantages compared with Boosting the previous algorithm. The adaptability of this algorithm is mainly reflected in the following aspects: in the former basic classifier, if a sample is misclassified, its corresponding weight will be increased; on the contrary, the weight of the correctly classified samples will decrease. Next, these samples are used to train the next basic classifier. At the same time, the algorithm will add a new weak classifier in each iteration according to the preset parameters until the setting is satisfied. In general, the core idea of this algorithm is "divide and conquer". By constantly adjusting the role of samples and weak classifiers in training, the accuracy of model prediction results is finally improved [28].

The training process for the model is divided into three main steps.

The first step is to initialize the weight distribution of the training data. If the number of samples is n, then the initial weights of these samples are all 1/n. This may cause the sampling bias of the training set to the majority classes. Therefore, we introduction of ADASYN in each round of boosting will enable each weak classifier to learn from more minority samples and thus learn wider decision regions for the minority class. The final initial weight distribution of the training set is obtained as shown in Eq. (6). Where N is the number of samples after oversampling.

$$D_1 = \left(w_{1,1}, \ldots w_{1,i}, \ldots w_{1,N}\right), \quad \omega_{1,i} = \frac{1}{N}, \quad i = 1, 2, \ldots, N \tag{6}$$

The second step is to perform iterative training. In the mth round of training, the training dataset with weight distribution D_m is used for learning to obtain the weak classifier $G_m(x)$, and the classification error rate of $G_m(x)$ on the training dataset is calculated according to Eq. (7). Immediately afterward, the coefficient of $G_m(x)$ is calculated using Eq. (8), which is used to characterize the importance of $G_m(x)$ in the final splitter. Finally, the weight distribution of the training dataset is updated by Eqs. (9)–(11).

$$e_m = P\left(G_m\left(x_i\right) \neq y_i\right) = \sum_{i=1}^{N} \omega_{m,i} I\left(G_m\left(x_i\right) \neq y_i\right) \tag{7}$$

$$\alpha_m = \frac{1}{2} \log \frac{1 - e_m}{e_m} \tag{8}$$

$$D_{m+1} = \left(w_{m+1,1}, \ldots w_{m+1,i}, \ldots w_{m+1,N}\right) \tag{9}$$

$$\omega_{m+1,i} = \frac{\omega_{m,i}}{Z_m} e^{-\alpha_m y_i G_m(x_i)}, \quad i = 1, 2, \ldots, N \tag{10}$$

$$Z_m = \sum_{i=1}^{N} \omega_{m,i} e^{-\alpha_m y_i G_m(x_i)} \tag{11}$$

The third step is to combine the M weak classifiers into a strong classifier according to the weak classifier weights α_m. As shown in Eq. (12).

$$G\left(x\right) = sign\left(f\left(x\right)\right) = sign\left(\sum_{m=1}^{M} \alpha_m G_m\left(x\right)\right) \tag{12}$$

We trained this model on the training set and then used the trained model to fit the test set and obtain the predicted results. The detection effects are shown in Part 4.

4 Experiment

In this section, we evaluate the validity of our proposed model through experiments. First, we introduce the data set and related indicators for model evaluation. We then describe in detail the Ponzi scheme detection experiment in our model and introduce the comparison with other representative methods. Finally, the most representative features are analyzed.

4.1 Data Set

In this experiment, we use the same dataset as in [13]. First, we downloaded the dataset from the XBlock website, containing the addresses of 3590 non-Ponzi scheme contracts and 200 Ponzi scheme contracts.

Based on the above contract address, we obtained the corresponding contract bytecode and transaction records from the Etherscan web crawler. Then, according to the list of bytecode and opcode conversion relationships, we decompose the bytecode into opcodes and extract the contextual characteristics of the opcodes from them. At the same time, according to the transaction records of each contract, we obtain the characteristics of each contract transaction.

Our model uses a combination of code characteristics and account characteristics, and the contract characteristics dataset contains 5017 characteristics (5000 opcode context characteristics and 17 account characteristics). To construct the model, we randomly divided the above data set of contract characteristics, 80% for training and 20% for testing.

4.2 Evaluation Metrics

To compare with other representative Ponzi scheme detection methods, we use accuracy rate, recall rate, and F1-score to evaluate these models. The formula for these indicators is shown in Eqs. (13)–(15):

$$Precision = \frac{true\ positive}{true\ positive + false\ positive} \tag{13}$$

$$Recall = \frac{true\ positive}{true\ positive + false\ negative} \tag{14}$$

$$F1{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{15}$$

4.3 Impact of N-Gram Parameter

Unlike previous studies, we choose the n-gram algorithm to extract the context of the contract, so as to more accurately reflect the logic of the contract source code. The n-gram algorithm divides the contract opcodes into several fragments by sliding windows of n size, and then performs frequency statistics on them. In order to observe the influence of different window sizes on model accuracy, we conducted experiments on different values of n, and Table 2 shows the corresponding results.

Table 2: Comparison of n-gram performance

	Precision	Recall	F1-score
1-gram	0.94	0.74	0.83
2-gram	0.94	0.77	0.85
3-gram	**0.94**	**0.79**	**0.86**
4-gram	0.91	0.82	0.86

According to Table 2, with the increase of the sliding window n, the recall rate and F score gradually increase, but when the sliding window increases to 4, the detection accuracy of the model

will decrease. Therefore, we finally set the sliding window size to 3, and select 3-gram with the best performance to extract opcode context features.

4.4 Impact of Different Features

In order to compare the detection performance of different features, we respectively use code features, account features, and the combination of two features for training, and compare the training effect. Table 3 summarizes the performance of the above three characteristics in detecting Ponzi schemes.

Table 3: Performance comparison of different features

	Precision	Recall	F1-score
Account	0.86	0.31	0.45
Opcode	0.94	0.79	0.86
Account + Opcode	**0.97**	**0.85**	**0.90**

According to Table 3, account characteristics are not effective in detecting Ponzi schemes and cannot be used alone for contract classification. In contrast, opcode context features are well detected. Although using only account characteristics does not work well, it may help improve the efficiency of the model. By combining the account feature and the code feature, the model's detection precision, recall rate, and F1-score are all improved. To sum up, the model can learn more valuable information and improve the training effect of the model by combining the account features with the context features of the operation code.

4.5 Imbalanced Classification Processing

This section addresses the problem of data imbalance. Table 4 shows the model classification effect before and after different sampling methods.

Table 4: Model performance before and after oversampling

	Precision	Recall	F1-score
No sampling	0.97	0.85	0.90
Random oversampling	0.97	0.82	0.89
SMOTE	0.95	**0.90**	0.92
SMOTE_Tomek	0.97	0.85	0.90
ADASYN	**0.97**	0.87	**0.92**

In our model, the ADASYN algorithm is used to oversample the training set data. As shown in the table, the detection effect of the model after oversampling has been greatly improved, especially since the F1-score has increased to 0.92, indicating that oversampling of a few classes is effective.

4.6 Impact of Different Classifiers

The experiment verifies the accuracy of the AdaBoost classifier and five classifiers, XGBoost, Random Forest, LightGBM, GBDT, and CatBoost. The features used in the experiment are 17 account features and 3-gram opcode features. We randomly divided the feature data set, 80% for

training, 20% for testing, and then used the above six classifiers for model training. Fig. 4 shows the comparison results.

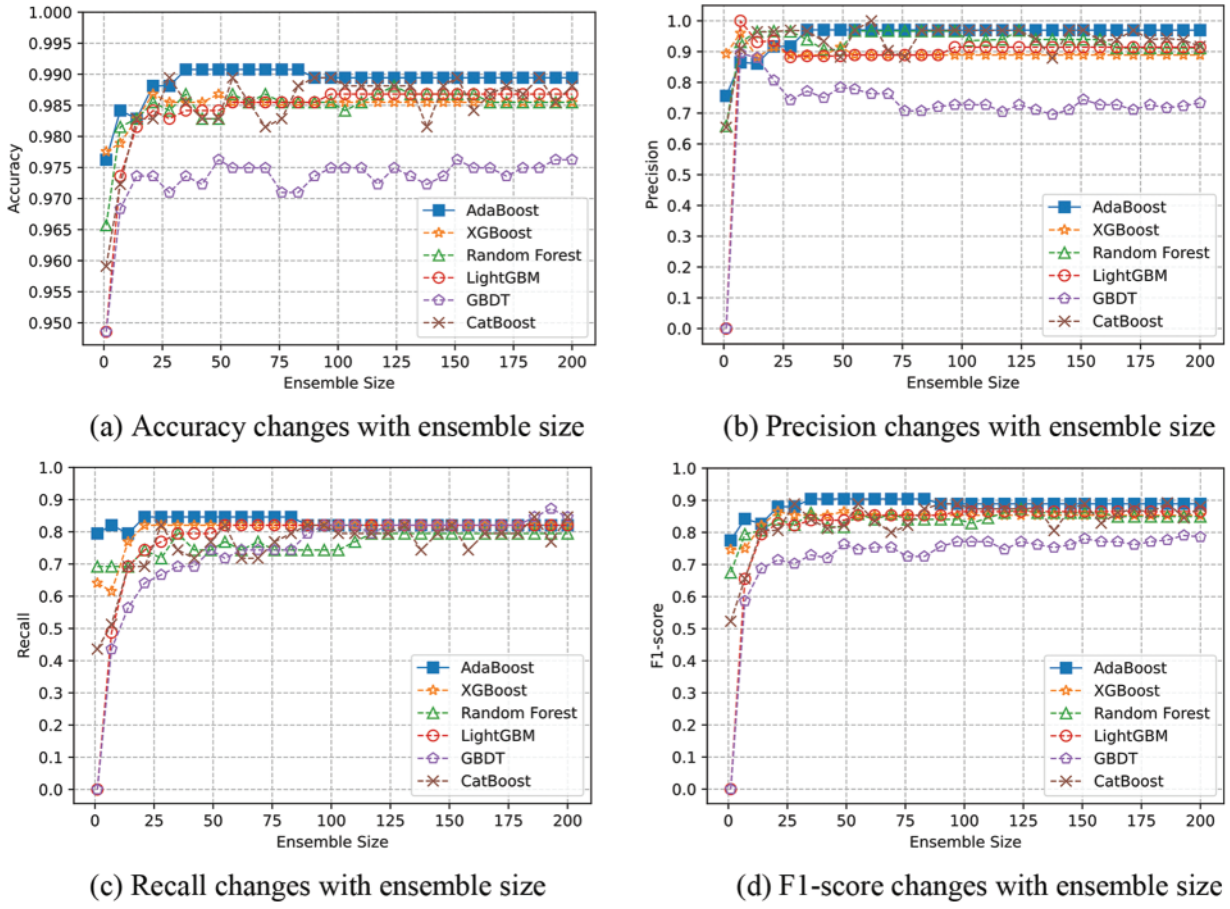

Figure 4: Comparison of detection effect of different classifier methods

As can be seen from Fig. 4, AdaBoost's performance is superior to the other five classifiers in accuracy, precision, recall, and F1-score. Adaboost obtained the best prediction when the ensemble size reached about 80, but as the ensemble size increased, overfitting would occur, resulting in slightly poor model prediction performance.

4.7 Performance Comparison

We compared the proposed method with a more representative Ponzi scheme detection model, and Fig. 5 shows the results.

It can be seen from the above experimental results that the proposed method significantly improves the performance of the detection model. The accuracy rate reached 0.97, and the F1-score reached 0.92, indicating that our method can effectively detect the Ponzi scheme smart contract on Ethereum.

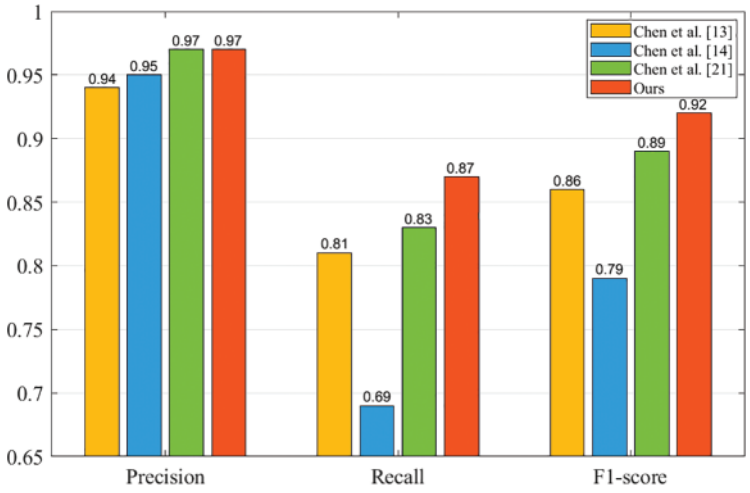

Figure 5: Performance comparison of different models

4.8 Feature Importance Ranking

To analyze the impact of the selected features on the model, AdaBoost's feature importance calculation method, feature_importances_, is used to calculate the importance of the above features to the model. Fig. 6 shows the 10 most important features of AdaBoost model training.

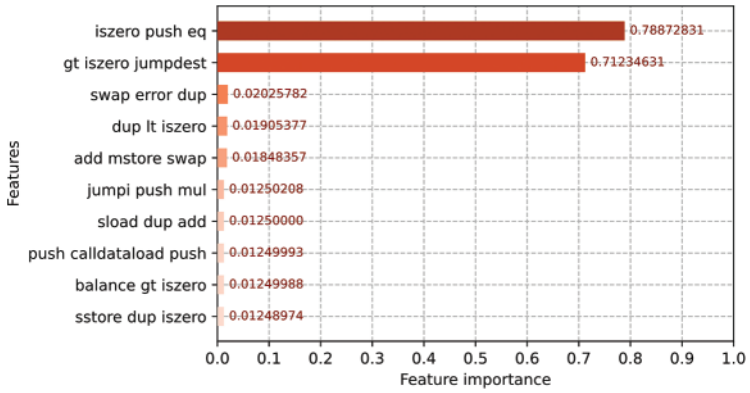

Figure 6: Feature importance ranking

The results show that iszero_push_eq and gt_iszero_jumpdest are the two most important opcode fragments, where iszero, eq, and gt are all conditional instructions that determine whether they are 0, equal, or greater, respectively. Whereas push is associated with a stack operation, jumpdest is the jump instruction. We speculate that the combination of these two orders may be effective because Ponzi contracts typically place investor addresses in the stack in order according to some judgment. Or when certain conditions are met to jump, in order to complete the contract to a specific user payment operation.

5 Discussion

Cryptocurrency fraud detection is an important research issue in blockchain transaction security. Ponzi scheme contracts, as a hidden fraudulent activity, have brought a lot of economic losses to people. In order to protect the security of the blockchain system, many researchers have proposed a variety of detection models for Ponzi schemes. Related research is mainly carried out from four aspects, including data acquisition, imbalanced class processing, feature extraction, and classifier design, as shown in Fig. 7.

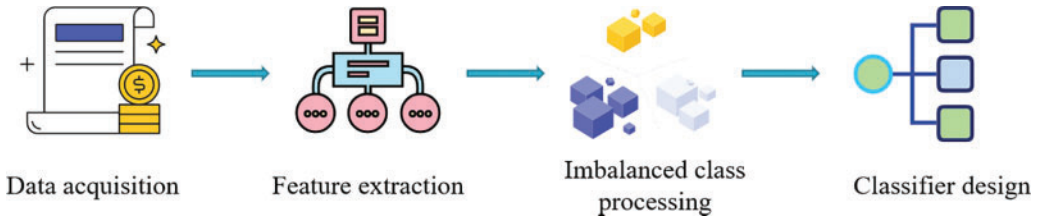

Figure 7: Model design process

The first problem in dealing with Ponzi scheme detection is how to obtain data. Currently, the open source data set about the Ponzi scheme is relatively lacking, and the dataset from the XBlock website is the most widely used. For example, Chen et al. [13] use the dataset to train their models. The data set contains the contract address and corresponding labels, and the number of contract samples is constantly updated with the collection and verification of researchers. To date, the dataset contains 3,790 contract addresses and their labels, including 200 Ponzi scheme contracts. In this paper, we choose the same data set as well. Some researchers have expanded the dataset, e.g., Fan et al. [20] supplemented Ponzi scheme contracts through promotion website inspection and source code analysis, but some of this data is not yet publicly available.

The second problem is feature extraction. Effective features can better characterize the contract function and improve the model's detection performance. Some researchers [17,20,21] take advantage of the code features to describe contracts. The difference is that [21] extracts the semantic features and structural features of the contract code by abstract syntax tree and structure-based traversal, while [17,20] uses the 2-gram algorithm to segment the opcode sequence and obtain the opcode characteristics of the contract. However, the code features cannot represent the transaction characteristics of the contract well, while the transaction records contain the fund flow relationship, which can be used as the basis to identify the Ponzi scheme. Therefore, some researchers [13,14,18] construct the data set of contract characteristics by extracting contract account characteristics and the frequency of a single opcode. In addition, Zhang et al. [16] add contract bytecode features and construct the contract data set by extracting contract bytecode features, opcode features, and account features, so as to extract contract characteristics in a more comprehensive way. The results in Table 2 show that context features (2-gram, 3-gram, and 4-gram) extracted from opcodes are superior to single opcode features (1-gram) in terms of detection accuracy, and 3-gram features can achieve better results. Meanwhile, the results in Table 3 show that the combination of code and account features can improve the model's detection performance. Therefore, our model uses 3-gram algorithm to extract the context logic of the contract operation code, and combines it with account characteristics to represent the characteristics of the contract.

The third work of detection is to deal with class imbalance problems in the dataset because the number of abnormal contracts accounts for a very small proportion. Some early studies [13,14,21] did

not address this problem. In contrast, other subsequent studies note the problem and use oversampling methods to balance classes. Wang et al. [18] used the synthetic minority oversampling technique (SMOTE) to fill in contract characteristics data for class imbalance. However, the SMOTE algorithm cannot overcome the category distribution problem in the imbalanced data set, which tends to produce distribution marginalization and blur the boundary between positive and negative samples. Aiming to solve this problem, some studies [16,17] have used the SMOTE_Tomek global sampling method to balance the dataset. After using the SMOTE algorithm, expand the samples, and use the Tomek Link algorithm to remove the heterogeneous samples glued at the boundary. And other researchers [20] used the modified Borderline-SMOTE technique based on SMOTE to improve the class distribution of samples. Our model uses ADASYN to make the classification boundary more reasonable by assigning different weights to different minority samples. As shown in Table 4, the model will perform better after using reasonable methods to deal with class imbalances in the dataset.

The fourth part of the whole scheme is the classifier. Related methods can be divided into three categories: traditional machine learning methods, deep learning or neural network models, and ensemble learning. Among the first group methods, the decision tree is a popular approach. Fan et al. [20] used the decision tree as a basic predictor to detect Ponzi scheme contracts. In addition to classical machine learning algorithms, some other researchers have used neural networks, such as LSTM (long-short-term memory network) [18] and fully connected neural networks [21], to detect Ponzi scheme contracts. Ensemble Learning uses a variety of compatible learning algorithms to perform a single task and achieve better generalization performance than a single learner. Due to the prominent advantage in classification tasks, most Ponzi scheme contract detection models [13,14,16,17,20] choose ensemble learning algorithms as classifiers. Among them, SMOTE_Tomek mixed sampling was used to replace LightGBM weight allocation, which improved the detection effect of the model [16]. The optimization strategy can significantly improve the predictive performance of the model. The AdaBoost algorithm increases the weight of misclassified samples from the previous base classifier and is used to train the next base classifier again, iterating continuously to obtain the final strong classifier. The algorithm uses the method of adjusting sample weight to highlight the classification error and finally achieves a low generalization error rate. In this paper, we use the AdaBoost algorithm in ensemble learning to adjust sample weights, thus improving the detection accuracy of the model.

Table 5 shows the comparison of the above models.

Table 5: Model comparison (The number in account features (#) indicates the number of features)

	Data set	Feature extraction	Class imbalance processing	Classification model
Chen et al. [13]	Dataset from XBlock	Opcode features (individual); account features (7);	——	XGBoost
Chen et al. [14]	Dataset from XBlock	Opcode features (individual); account features (13);	——	Random forest

(Continued)

Table 5: Continued

	Data set	Feature extraction	Class imbalance processing	Classification model
Chen et al. [21]	Dataset from XBlock	Source code features (structural and semantic)	——	Fully connected neural network
Fan et al. [20]	Open source dataset [12] + dataset from XBlock + DApp	Opcode features (2-gram);	Borderline-SMOTE	Decision tree
Wang et al. [18]	Dataset from XBlock	Opcode features (individual); account features (7);	SMOTE	LSTM
Zhang et al. [16]	Dataset from XBlock	Opcode features (individual); account features (7); bytecode features;	SMOTE_Tomek	LightGBM
Zhang et al. [17]	Dataset from XBlock	Opcode features (2-gram); bytecode features;	SMOTE_Tomek	CatBoost
Ours	Ponzi_label.csv	Opcode features (2-gram); account features (7);	ADASYN	AdaBoost

6 Conclusion

In this study, we propose a new method for detecting Ponzi scheme smart contracts in Ethereum. We use the n-gram algorithm to extract more comprehensive contract opcode features and combine them with account features characterizing contract transaction characteristics to construct the contract feature data set. Then, we indirectly change the weight assignment of the AdaBoost classifier to the training samples by using the ADASYN algorithm. thereby better learning features from a small number of classes and solving the class imbalance problem that exists for Ponzi scheme contract detection in ethereum. Experimental results show that our method can effectively improve the detection effect of Ponzi scheme contracts.

However, the structural features of contract transactions have not been analyzed in depth in our work, so in the next work we will consider capturing topological structural information of contract transactions by constructing a transaction graph. We will also investigate the scalability of the model framework to enable the detection of other similar fraudulent activities and promote the security of blockchain technology.

Acknowledgement: The authors would like to thank Zibin Zheng's research team for sharing the data on XBlock.

Funding Statement: This work was supported by National Key R&D Program of China (Grant Numbers 2020YFB1005900, 2022YFB3305802).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] K. Huang, X. Zhang, Y. Mu, F. Rezaeibagha and X. J. Du, "Scalable and redactable blockchain with update and anonymity," *Information Sciences*, vol. 546, pp. 25–41, 2021.

[2] P. Kumar, R. Kumar, A. Kumar, A. Franklin, S. Garg *et al.,* "Blockchain and deep learning for secure communication in digital twin empowered industrial IoT network," *IEEE Transactions on Network Science and Engineering*, pp. 1–13, 2022.

[3] P. Kumar, R. Kumar, A. Kumar, A. Franklin, A. Jolfaei *et al.,* "Blockchain and deep learning empowered secure data sharing framework for softwarized uavs," in *2022 IEEE Int. Conf. on Communications Workshops (ICC Workshops)*, Seoul, Korea, pp. 770–775, 2022.

[4] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[5] L. S. Alotaibi and S. Alshamrani, "Smart contract: Security and privacy," *Computer Systems Science and Engineering*, vol. 38, no. 1, pp. 93–101, 2021.

[6] Z. Zheng, S. Xie, H. Dai, W. Chen, X. Chen *et al.,* "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, 2020.

[7] R. Kumar and R. Tripathi, "Large-scale data storage scheme in blockchain ledger using ipfs and nosql," in *Large-Scale Data Streaming, Processing, and Blockchain Security*, Hershey, Pennsylvania, USA: IGI Global, pp. 91–116, 2021.

[8] R. Kumar and R. Tripathi, "Building an ipfs and blockchain-based decentralized storage model for medical imaging," in *Research Anthology on Improving Medical Imaging Techniques for Analysis and Intervention*, Hershey, Pennsylvania, USA: IGI Global, pp. 916–934, 2023.

[9] W. N. Al-Sharu, M. K. Qabalin, M. Naser and A. Saraerh, "A secure framework for blockchain transactions protection," *Computer Systems Science and Engineering*, vol. 45, no. 2, pp. 1095–1111, 2023.

[10] L. Vasek and T. Moore, "There's no free lunch, even using bitcoin: Tracking the popularity and profits of virtual currency scams," in *Financial Cryptography and Data Security: 19th Int. Conf.*, San Juan, Puerto Rico, pp. 44–61, 2015.

[11] I. Kedar and C. Dannen, "Ponzis and pyramids," in *Building Games with Ethereum Smart Contracts: Intermediate Projects for Solidity Developers*, Berkeley, CA, USA: Apress, pp. 143–170, 2018.

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

[13] W. Chen, Z. Zheng, J. Cui, E. Ngai, P. Zheng *et al.,* "Detecting ponzi schemes on ethereum: Towards healthier blockchain technology," in *Proc. of the 2018 World Wide Web Conf.*, Republic and Canton of Geneva, Switzerland, pp. 1409–1418, 2018.

[14] W. Chen, Z. Zheng, E. Ngai, P. Zheng and Y. Zhou, "Exploiting blockchain data to detect smart ponzi schemes on ethereum," *IEEE Access*, vol. 7, pp. 37575–37586, 2019.

[15] W. Chen, X. Li, Y. Sui, N. He, H. Wang *et al.,* "Sadponzi: Detecting and characterizing ponzi schemes in ethereum smart contracts," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 5, no. 2, pp. 1–30, 2021.

[16] Y. Zhang, W. Yu, Z. Li, S. Raza and H. Cao, "Detecting ethereum ponzi schemes based on improved lightGBM algorithm," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 2, pp. 624–637, 2021.

[17] Y. Zhang, S. Kang, W. Dai, S. Chen and J. Zhu, "Code will speak: Early detection of ponzi smart contracts on ethereum," in *2021 IEEE Int. Conf. on Services Computing (SCC)*, Chicago, IL, USA, pp. 301–308, 2021.

[18] L. Wang, H. Cheng, Z. Zheng, Y. Yang and X. Zhu, "Ponzi scheme detection via oversampling-based long short-term memory for smart contracts," *Knowledge-Based Systems*, vol. 228, pp. 107312, 2021.

[19] W. Sun, G. Xu, Z. Yang and Z. Chen, "Early detection of smart ponzi scheme contracts based on behavior forest similarity," in *2020 IEEE 20th Int. Conf. on Software Quality, Reliability and Security (QRS)*, Macau, China, pp. 297–309, 2020.

[20] S. Fan, S. Fu, H. Xu and X. Chen, "Al-SPSD: Anti-leakage smart ponzi schemes detection in blockchain," *Information Processing & Management*, vol. 58, no. 4, pp. 102587, 2021.

[21] Y. Chen, H. Dai, X. Yu, H. Wei, Z. Xie *et al.,* "Improving ponzi scheme contract detection using multi-channel textCNN and transformer," *Sensors*, vol. 21, no. 19, pp. 6417, 2021.

[22] E. Jung, M. L. Tilly, A. Gehani and Y. Ge, "Data mining-based ethereum fraud detection," in *2019 IEEE Int. Conf. on Blockchain (Blockchain)*, Atlanta, GA, USA, pp. 266–273, 2019.

[23] Y. Lou, Y. Zhang and S. Chen, "Ponzi contracts detection based on improved convolutional neural network," in *2020 IEEE Int. Conf. on Services Computing (SCC)*, Beijing, China, pp. 353–360, 2020.

[24] L. Bian, L. Zhang, K. Zhao, H. Wang and S. Gong, "Image-based scam detection method using an attention capsule network," *IEEE Access*, vol. 9, pp. 33654–33665, 2021.

[25] Y. Liang, W. Wu, K. Lei and F. Wang, "Data-driven smart ponzi scheme detection," *arXiv preprint*, arXiv:2108.09305, 2021.

[26] S. Yu, J. Jin, Y. Xie, J. Shen and Q. Xuan, "Ponzi scheme detection in ethereum transaction network," in *Blockchain and Trustworthy Systems: Third Int. Conf.*, Guangzhou, China, pp. 175–186, 2021.

[27] H. He, Y. Bai, E. A. Garcia and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE Int. Joint Conf. on Neural Networks (IEEE World Congress on Computational Intelligence)*, Hong Kong, China, pp. 1322–1328, 2018.

[28] Y. Cao, Q. Miao, J. Liu and L. Gao, "Advances and prospects of adaBoost algorithm," *Acta Automatica Sinica*, vol. 39, no. 6, pp. 745–758, 2013.

[29] L. Song and X. Kong, "A study on characteristics and identification of smart ponzi schemes," *IEEE Access*, vol. 10, pp. 57299–57308, 2022.

[30] B. W. Yap, K. A. Rani, H. A. A. Rahman, S. Fong, Z. Khairudin *et al.,* "An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets," in *Proc. of the First Int. Conf. on Advanced Data and Information Engineering (DaEng-2013)*, Singapore, pp. 13–22, 2014.

[31] M. Hayaty, S. Muthmainah and S. M. Ghufran, "Random and synthetic over-sampling approach to resolve data imbalance in classification," *International Journal of Artificial Intelligence Research*, vol. 4, no. 2, pp. 86–94, 2020.

[32] 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, 2002.

[33] S. Smiti and M. Soui, "Bankruptcy prediction using deep learning approach based on borderline SMOTE," *Information Systems Frontiers*, vol. 22, no. 5, pp. 1067–1083, 2020.

[34] G. Ahmed, M. J. Er, M. M. S. Fareed, S. Zikria, S. Mahmood *et al.,* "DAD-net: Classification of Alzheimer's disease using ADASYN oversampling technique and optimized neural network," *Molecules*, vol. 27, no. 20, pp. 7085, 2022.

[35] Wang R., "AdaBoost for feature selection, classification and its relation with SVM, a review," *Physics Procedia*, vol. 25, pp. 800–807, 2012.

[36] K. Wang, C. Thrasher, E. Viegas, X. Li, B. Hsu *et al.,* "An overview of microsoft web n-gram corpus and applications," in *Proc. of the NAACL HLT 2010 Demonstration Session*, Los Angeles, California, USA, pp. 45–48, 2010.

[37] M. Vasek and T. Moore, "Analyzing the bitcoin ponzi scheme ecosystem," in *Int. Conf. on Financial Cryptography and Data Security*, Nieuwpoort, Curaçao, pp. 101–112, 2018.