Hindawi
Security and Communication Networks
Volume 2022, Article ID 1554752, 10 pages
https://doi.org/10.1155/2022/1554752

Research Article

CTRF: Ethereum-Based Ponzi Contract Identification

Xuezhi He, Tan Yang, and Liping Chen

State Key Laboratory of Networking and Switching Technology School of Computer Science(National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing 100876, China

Correspondence should be addressed to Tan Yang; tyang@bupt.edu.cn

Received 30 December 2021; Revised 27 January 2022; Accepted 2 March 2022; Published 29 March 2022

Academic Editor: Yuling Chen

In recent years, blockchain technology has been developing rapidly. More and more traditional industries are using blockchain as a platform for information storage and financial transactions, mainly because of its new characteristics of non-tamperability and decentralization compared with the traditional systems. As a representative of blockchain 2.0, Ethereum has gained popularity upon its introduction. However, because of the anonymity of blockchain, Ethereum has also attracted the attention of some unscrupulous people. Currently, millions of contracts are deployed on Ethereum, many of which are fraudulent contracts deployed by unscrupulous people for profit, and these contracts are causing huge losses to investors worldwide. Ponzi contracts are typical of these contracts, which mainly reward the funds invested by later investors to early investors, and later investors will have no gain. However, although there are some studies for identifying Ponzi contracts on Ethereum, there is some room for progress in the research. Therefore, we propose a method to detect Ponzi scheme contracts on Ethereum-CTRF. This method forms a dataset by extracting the word features and sequence features of the smart contract's code and the features of transactions. The dataset is divided into a training set and a test set. Oversampling is performed on the training set to deal with the problem of positive and negative sample imbalance. Finally, the model is trained on the training set and tested on the test set. The experimental results show that the model has significantly improved recall compared with existing Ponzi contract detection methods.

1. Introduction

Blockchain technology was proposed by Nakamoto [1], and since its introduction in 2008, it has received widespread attention. Due to its decentralized and tamper-evident features, blockchain technology has now been applied in the fields of finance, healthcare, and social governance. Based on these advantages, blockchain technology will have even broader application prospects in the future.

In the field of financial transactions, the underlying blockchain technology-based electronic cash systems (Bitcoin and Ethereum) have attracted many investors. However, they have also attracted many unscrupulous people. Owing to the inherent anonymity and tamper-evident nature of the blockchain, the Bitcoin and Ethereum platforms have also become a breeding ground for unscrupulous transactions to take place. On the Silk Road website, which was shut down in 2013, as much as $300,000–$500,000 per day was traded in Bitcoin regarding drugs and private data, and at the time of Silk Road's closure in 2013, approximately 9.5 million Bitcoins worth $1.2 billion had been traded on Silk Road [2].

Ethereum is known as Blockchain 2.0, which mainly solves the problem of lack of scalability of Bitcoin system; however, while people add new features to Ethereum, new risks are also introduced. Phishing, fraud, theft, and other illegal criminal activities launched by taking advantage of the flaws in the Ethereum blockchain technology have emerged. On June 17, 2016, the DAO of Ethereum was attacked by hackers. The attacker exploited a vulnerability in a contract written by the DAO to transfer more than 3 million Ether coins equivalent to $60 million from its asset pool to its sub-DAO [3].

Among the plethora of scams, Ponzi schemes, a classic scam in the real world, are also happening on the blockchain. Ponzi schemes on Bitcoin usually actively advertise their on-

chain projects on social media as having high rewards and low risk to attract investors. Ponzi schemes on Ethereum are smart contracts written in the high-level programming language, Solidity [4]. They are generally packaged as investment projects or gambling games that also promise huge returns to investors. Some Ponzi schemes even create their promotional websites to attract investors through aggressive marketing campaigns. Usually, investors know little about blockchain, and it can be difficult for them to tell which are meaningful smart contract investment projects and which are smart contracts disguised as high-yield investment schemes. According to Chainalysis, scams on Ethereum from 2017 to 2019 affected millions of people and caused $4.3 billion worth of losses. The majority of these came from Ponzi schemes, accounting for 92% of the total. The number of victims of Ethereum Ponzi scams alone was 2.4 million, with the average amount transferred by victims being $1,676 [5]. According to Bartoletti et al., 191 smart Ponzi schemes active on Ethereum raised almost $500,000 from more than 2,000 different users from August 2015 to May 2017 [6].

It is evident from reading the previously mentioned studies that there is an urgent need to strengthen the regulation and monitoring of the blockchain market. Although there are some studies in this area, they do not focus on the recall value of the experimental results. In fraud detection applications, as in many fields with unbalanced class distributions, it is more important to correctly classify the true classes (i.e., the "Ponzi" classes in our problem) than to correctly classify the majority classes. Therefore, the recall is more important than the precision of the prediction.

We have achieved good results by dealing with the data imbalance problem and by setting up a large number of experiments in different environments with more effective features. The recall values have improved compared to existing studies. To summarize, our contributions are as follows:

(1) dealing with the imbalance between positive and negative samples of the dataset by expanding positive samples and oversampling,

(2) evaluation and comparison of models for classifying Ethereum Ponzi schemes, ultimately our model has a higher recall,

(3) ssessment of feature contribution.

2. Related Work

In terms of Ponzi scheme research on Bitcoin. Vasek and Moore analyzed the supply and demand of Bitcoin-based Ponzi schemes, identified 1780 Ponzi schemes, and derived the determinants affecting the life cycle of Bitcoin Ponzi schemes [7]. Boshmaf et al. analyzed MMM, one of the oldest Ponzi schemes on Bitcoin, and proposed analytical criteria and metrics for the Ponzi scheme of cryptocurrencies [8]. It is worth mentioning that they counted the daily Gini coefficients of MMM to measure the income gap between investors. Bartoletti et al. designed a set of relevant characteristics to classify Bitcoin Ponzi schemes, such as average amount invested by users, maximum daily trading volume, number of active days of contracts, number of users, and Gini coefficients. Metrics such as F-score and AUC were then used to evaluate the effectiveness of different supervised learning classification algorithm models and finally succeeded in finding 31 of the 32 Ponzi schemes [9]. Although the imbalanced dataset was treated by them, the model may still suffer from overfitting by reason of the large gap between positive and negative sample size, so there is still room for improvement in their experiments.

In the study of Ponzi schemes on Ethereum, Zheng et al. surveyed the challenges and recent advances in smart contracts, giving a complete picture of the challenges smart contracts by dividing the smart contract lifecycle into four phases: creation, deployment, execution, and completion, where scams like Ponzi contracts are classified as the last phase of the contract's lifecycle, and most scams cause harm to contract users during the contract completion phase [10]. Chen et al. analyzed the current problems of Ethereum from three perspectives: vulnerability, attack, and defense. In the paper, the Rubixi contract is used as an example to classify Ponzi contracts as an attack means in the application layer of Ethereum [11]. Hu et al. analyzed the transaction behavior pattern between Ponzi contracts and other scam contracts to classify contracts from the perspective of transactions [12]. Jung et al. used the 0-day model to analyze the model based on the bytecode features of the contract and finally determined whether the contract is a Ponzi contract [13]. However, they did not consider the transaction features of the contract in their experiments, which may lead to a decrease in accuracy compared with the model that incorporates transaction features. Yujian and Bo classified Ponzi contracts into tree-shaped, chain-shaped, waterfall-shaped, and handoff-shaped by analyzing the Ponzi contract source code. They proposed that the similarity between contract bytecodes can be measured by using NLD [14] (Normalized Levenshtein Distance) and setting the corresponding threshold to determine whether two contracts are similar and whether the contract is a Ponzi contract. Subsequently, they measured the impact of Ponzi contract on Ethereum by counting the total transaction amount [6]. Sun et al. were inspired by the flowchart of traditional test domain code; the bytecodes generated during the operation of a Ponzi contract are concatenated and plotted as a tree of invocation behaviors. The model is trained by comparing the similarity of the behavior trees [15]. Fan et al. solved the prediction bias problem which was made of target leakage during training and improved the generalization ability of the model by analyzing the imbalance and repetition of Ponzi contracts in the dataset [16]. Chen et al. extracted the transaction features from the transaction data of smart contracts and combined them with the opcode of smart contracts in extracted opcode frequency features and used XGBoost to train these data features. Eventually, 434 Ponzi contracts were found by detecting contracts deployed before May 7, 2017 [17].

Although the aforementioned studies achieved good results, most of their experiments aimed at improving the model accuracy without considering improving the recall of the model. In contrast, our CTRF (Code and Transaction Random Forest) model improves the recall of the model by

adding sequence features of the opcodes to the code features and extracting more efficient features of transactions.

3. Smart Ponzi Contracts

The definition of smart contract can be traced back to 1994 by Szabo [18]. It was first defined as an alternative to traditional paper contracts and a digital representation of the transaction agreement to help the parties to fulfill the contractual project. However, due to the immaturity of the technology and the lack of a trustworthy platform for execution, smart contracts did not attract much attention. The establishment of blockchain and the rise of decentralized platforms have brought smart contracts back into the public eye. Ethereum is known as blockchain 2.0 differs from blockchain 1.0 in that Ethereum has a Turing-complete programming language [19]. Developers can implement smart contracts in the high-level programming language Solidity or Golang and compile them for deployment on the EVM (Ethereum virtual machine) [4]. Smart contracts are the basis for implementing blockchain-based information systems in various domains. It can execute transactions without a trusted third party by triggering conditions through program code. For example, the following is a simple example of a smart Ponzi contract–0x83Fccc659EeeeE98ca9764B7B34409347DFbc98b from the source code; we can know that every investment received by the contract will transfer 1% of the balance to the contract creator, and this contract every 5900 blocks (24 hours) will pay 5% of the balance of the contract to the investor.

```
pragma solidity ^0.4.24;
contract ethernity {
  address pr = 0 x 587a38954a
D9d4DEd6B53a8F7F28D32D28E6bBD0;
  address ths = this;
  mapping (address => uint) balance;
  mapping (address => uint) paytime;
  mapping (address => uint) prtime;
  function () external payable {
    if ((block.number-prtime[pr]) >= 5900){
      pr.transfer(ths.balance/100);
      prtime [pr] = block.number;
    }
    if (balance[msg.sender] ! = 0){
      msg.sender.transfer ((block.number-paytime
[msg.sender])/5900*5);balance[msg.sender]/100*5);
    }
    paytime[msg.sender] = block.number;
    balance[msg.sender] += msg.value;
  }
}
```

The smart contract runs on the EVM. After compiling the smart contract code into bytecode and uploading it to Ethereum through transactions, Ethereum will automatically return the generated contract account address, and finally, investors can interact with the contract through transactions [20]. The code of smart contracts can implement a wide variety of functions, which provides investors with a wealth of investment options. However, this can also confuse investors, who may fall into the trap of scams without being fully familiar with these smart contracts.

The Ponzi scheme was invented by Charles Ponzi, an Italian businessman, in which he promised investors a 40% profit return within three months. After attracting investors, he paid the new investors' money as a return to those who initially invested and then enticed more people to invest. He eventually attracted 30,000 investors in seven months. The more official definition from the SEC (United States Securities and Exchange Commission) is "A Ponzi scheme is an investment fraud that involves the payment of purported returns to existing investors from funds contributed by new investors. Ponzi scheme organizers often solicit new investors by promising to invest funds in opportunities claimed to generate high returns with little or no risk. With little or no legitimate earnings, Ponzi schemes require a constant flow of money from new investors to continue. Ponzi schemes inevitably collapse, most often when it becomes difficult to recruit new investors or when a large number of investors ask for their funds to be returned" [21].

From the definition, the typical feature of the Ponzi scheme is to pay the existing investors the so-called returns with the funds provided by the new incoming investors. Compared with other financial frauds, Ponzi schemes are characterized by many victims, wide impact, deep damage, high concealment, and serious social harm. In the Ethereum smart contract, the Ponzi scheme has some new characteristics.

(1) The most obvious thing is that it is based on the anonymity of the blockchain; people cannot know the real identity of the contract initiator. For the unscrupulous this greatly reduces the risk of them committing a scam, but for the average contract user, the risk of their money being compromised is greatly increased.

(2) Ponzi contracts are simpler and more efficient than traditional Ponzi schemes. The scammers only need to deploy Ponzi contracts to Ethereum and they can effortlessly reap the benefits when transactions occur.

(3) Ponzi contracts are easier to replicate and implement. By reviewing the Ponzi contract code, we found that many of the contract codes are identical.

(4) Due to the anonymity of blockchain and the difficulty of traceability, the defrauded funds cannot be successfully recovered, leaving the investors to suffer losses.

4. Methodology

4.1. Dataset. The dataset used in this paper is based on the open-source shared address set [6, 17, 22]. One of the publicly available address sets in [17] originally contained 3590 common contract addresses and 200 Ponzi contract addresses and three incorrect addresses. In our experiments, we eliminated two non-Ponzi contract addresses that were not successfully deployed and found two addresses with exactly opposite labels to other address sets. After careful inspection of the contract source code, the corrected address set situation is as follows: 3586 common contract addresses and 202 Ponzi contract addresses, a total of 3788 smart contracts, where the ratio of positive to negative cost is 1 : 18. We labeled this address set as D1.

Since the gap between the positive and negative sample ratios in D1 is too large, we first dealt with the positive and negative sample imbalance by expanding the positive sample data. The publicly available address set in [6] contains 184 Ponzi contracts, and after removing two of the duplicate addresses, its correction includes 182 Ponzi contract addresses. We collected another 50 Ponzi contract addresses from [22]. The address sets of [6, 22] are combined and then deduplicated against D1 to finally obtain 96 Ponzi contract addresses, which are added to D1 to obtain the expanded address set D2. The expanded address set D2 contains 3586 ordinary contract addresses and 298 Ponzi contract addresses, totaling 3884 smart contracts, where the ratio of positive to negative samples is 1 : 12.

The address dataset remains unbalanced after expanding the positive samples. Besides expanding the data from the perspective of the data source, the other two solutions to deal with the imbalance of the dataset at present are oversampling and undersampling. Oversampling means balancing the positive and negative sample ratios by generating samples from minority classes. And undersampling means reducing the samples of most classes to balance the positive and negative sample ratios. Here, we chose SMOTE oversampling based on oversampling; the basic idea of SMOTE algorithm is to analyze the minority class samples and synthesize new minority class samples added to the dataset according to the KNN algorithm, thus enriching the number of minority class samples and avoiding the problem of overfitting caused by oversampling by copying minority class samples in the past [23]. As in Figure 1, we constructed the feature dataset in two main ways.

(1) Get the bytecode of the contract by Etherscan, then disassemble the bytecode into opcode, and finally convert it into code features.

(2) Obtain the transactions of the contract and calculate the corresponding transaction features, such as life time and Gini coefficient.

The D1-code dataset and D2-code dataset are obtained by extracting opcode features on D1 and D2, respectively. The D1-codeAndTran dataset and D2-codeAndTran dataset

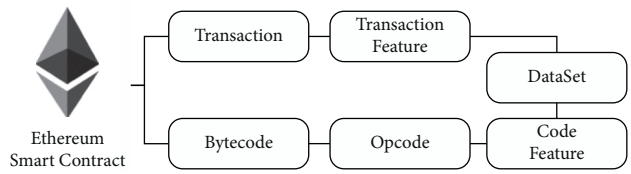

Figure 1: Flowchart of feature extraction.

are obtained by merging the opcode features and transaction features. The experiments are mainly focused on these four datasets. The datasets have been open-sourced in this paper, which can be found at https://github.com/BuptHxz/DetectionOfPonziContract.

4.2. Feature Extraction

4.2.1. Code Feature. Initially, we wanted to get the code features of the contracts by getting the internal implementation logic and keywords directly from the source code. But since the code of most of the contracts in the dataset is not publicly available, we only got the code features of a very small fraction of the contracts, which made it impossible to build a complete dataset. Although most of the contracts' code is not available, we can get the contracts' bytecode, which is compiled from the code, through Etherscan.io. And according to Kiffer et al. [24, 25], the similarity between contracts can be effectively detected by detecting the similarity of contract bytecode. Therefore, we started from the bytecode and got the code features of the contracts by disassembling and other techniques.

As described in Section 2, if developers want to run a contract on Ethereum, they first need to write a smart contract by Solidity, and then, the code is transformed into bytecode after compiling the code with the corresponding specific version of the compiler. Finally, the compiled bytecode is deployed to Ethereum. The bytecode is represented by a string of hexadecimal codes. The following is the compiled bytecode of the Solidity code shown in Section 3.

"bytecode":"6080604052600080546001600160a060020a031990
811673587a38954ad9d4ded6b53a8f7f28d32d28e6bbd0179091
556001805490911630179055348015610044576000 80fd5b5061
017880610054600039600 0f300608060405260008054 73ffffffff
ffffffffffffffffffffffffffffffff168152600460205260409020546117 0c43
91909103106100b557600054600154 73ffffffffffffffffffffffffffffffff
ffffff918216916108fc916064911631049081150290604051600
060405180830381858888f193505050501580156100 8b573d6000
803e3d6000fd5b506000805473ffffffffffffffffffffffffffffffff16
815260046020526040902043905 55b33600090815260026020 5
260409020541561012857336000818152600260209081526040 8
083205460039092529091205461 08fc9160649161170c4391909
1030402046005029081150290604051600060405180830381 85
8888f193505050501580156101 26573d6000803e3d6000fd5b50
5b336000908152600360209081526040808320439055600290 9
152902080543401905500 00a165627a7a72305820bccf7dff930c
d8237a3d56127f741c545a3f33447c34351f3009e937ea335baf0
029"

These bytes correspond to EVM operations and thus instruct the EVM to run the code. To make it easy to distinguish them, Ethereum officials convert these bytes into corresponding easy-to-remember opcodes (Opcodes) and record them in the Yellow Book [20]. For example, 0x60 converted to Opcode is PUSH1; 0x80 converted to Opcode is DUP1. The corresponding partial conversions are shown in Table 1.

The bytecode can be converted to an opcode sequence as follows.

"opcodes": "PUSH1 0x80 PUSH1 0x40 MSTORE PUSH1 0x00 DUP1 SLOAD PUSH1 0x01 PUSH1 0xa0 PUSH1 0x02 EXP SUB NOT SWAP1 DUP2 AND PUSH20 0x587a38954ad9d4ded6b53a8f7f28d32d28e6bbd0 OR SWAP1 SWAP2 SSTORE PUSH1 0x01 DUP1 SLOAD SWAP1 SWAP2 AND ADDRESS OR SWAP1 SSTORE CALLVALUE DUP1 ISZERO PUSH2 0x0044 JUMPI PUSH1 0x00 DUP1 REVERT JUMPDEST POP PUSH2 0x0178 DUP1 PUSH2 0x0054 PUSH1 0x00 CODECOPY PUSH1 0x00 RETURN STOP PUSH1 0x80 PUSH1 0x40 MSTORE PUSH1 0x00 DUP1 SLOAD ...".

By counting the number of bytecodes corresponding to ordinary contracts and Ponzi contracts, we find that there are significant differences between Ponzi contracts and ordinary contracts in the number of some bytecodes, as shown in Figure 2. It shows the comparison of the average number of opcodes between Ponzi contracts and ordinary contracts. It is clear that Ponzi contracts and ordinary contracts have the same trend in the number of opcodes and overall Ponzi contracts have fewer opcodes than ordinary contracts. By looking at the numerous source codes, we found that Ponzi contracts tend to implement all the functions through less code, while ordinary contracts functions are more abundant and therefore have more code and a higher average number of opcodes.

We obtained the sequence of opcodes corresponding to the contract bytecodes by disassembling the bytecodes into opcodes. Then, we computed the code features of each contract by using the bag-of-words model [26]. There are opcodes such as PUSH1, PUSH2, DUP1, and DUP2, which we combine into one opcode such as PUSH and DUP, and then, do the statistics, and finally, we select a total of 77 code features by combining the calculations.

However, the features extracted by the bag-of-words model do not take into account the sequence of opcodes and ignore the semantic information of opcodes. Therefore, we use the Doc2Vec model to obtain the sequence features and semantic features of the opcodes to make up for the deficiency of the bag-of-words model in extracting the special diagnosis [27]. The final code features are 77 features extracted by the bag-of-words model and 20 features extracted by the Doc2Vec model.

4.2.2. Transaction Features. According to the characteristics of Ponzi schemes, it is known that most of the later investors incurred losses, and only some early investors may get the gains. Past studies tend to extract the features of trading from the perspective of Ethereum, and we added some new features on top of this. For example, in the Gini coefficient, as the funds of later investors in a Ponzi scheme are often transferred to the accounts of earlier investors, this characteristic of an unbalanced distribution of funds will make the Gini coefficient larger. The Gini coefficient is a number between 0 and 1. The closer the Gini coefficient is to 1, the more unbalanced the distribution of funds is, and the closer it is to 0, the more balanced the distribution of funds is.

We selected the following transaction characteristics:

(1) Bal: the balance of the contract after the last trade
(2) TotalGet: the number of all ETH received by the contract
(3) TotalSend: the number of all ETH sent by the contract
(4) MaxSend: the maximum amount of ETH sent in a single contract
(5) AvgFee: the average cost of all transactions in the contract
(6) LifeTime: the survival time of the contract
(7) GetDivSend: TotalGet/TotalSend
(8) AddrGetProfit: the number of addresses that receive proceeds from the addresses traded with the contract
(9) Gini: Gini coefficient

4.3. CTRF Model. We proposed CTRF (Code and Transaction Random Forest) to identify and classify our contracts. The specific CTRF model structure diagram is shown in Figure 3.

In the data preprocess phase, we first obtained the contract transactions and bytecode and then disassembled the bytecode to obtain the contract opcodes. Inspired by NLP, we used the bag-of-words algorithm to obtain the word features of the opcodes and then used the Doc2Vec algorithm to obtain the sequence features of the opcodes. The word features of the opcode and the sequence features together form the code features of the contract. For transaction features, we chose such as Gini coefficient to represent the contract.

In the model train phase, we first synthesized a new sample of Ponzi contracts by SMOTE oversampling, thus solving the positive and negative sample imbalance problem. Subsequently, we composed decision trees by randomly selecting the code features and transaction features of the contract and used the idea of integrated learning to statistically vote on the results using a forest composed of decision trees to obtain the classification results. Finally, we got our training model.

In the model test phase, we inputted the test set into our model trained in the previous phase to finally get the classification results of the contract.

In order to improve the recall value of the model as much as possible, we did the following. First, we solved the problem of imbalance in the number of positive and negative samples in the dataset by expanding the samples of Ponzi contracts. Then, we extracted the word features and

Table 1: Examples of converting bytecode to opcode.

Bytecode	Opcode	Bytecode	Opcode
0x60	PUSH1	0x5b	JUMPDEST
0x80	DUP1	0x52	MSTORE
0x56	JUMP	0x50	POP
0x90	SWAP1	0x17	OR

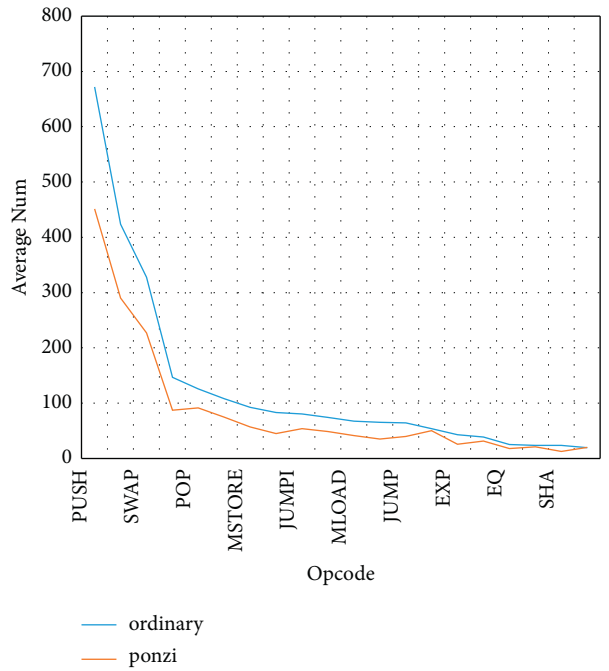

Figure 2: The average number of opcodes for ordinary contracts vs. Ponzi contracts.

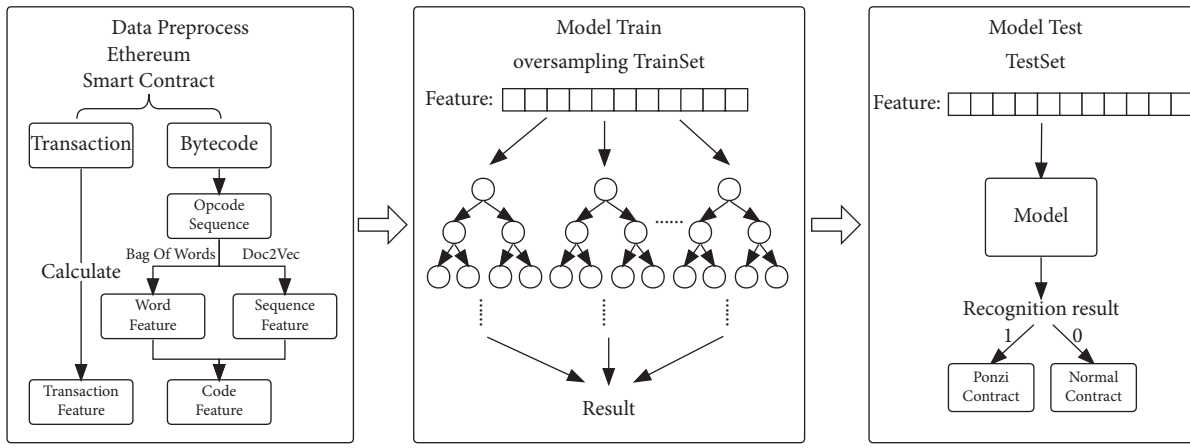

Figure 3: CTRF model.

sequence features of the code inspired by NLP in the feature extraction stage, and designed more effective transaction features to represent the Ponzi contracts better. Next, we performed SMOTE oversampling on the positive samples of the dataset to further increase the samples of a few classes and to avoid the overfitting problem of the model. We then penalized its classification errors more severely by increasing the weights of the Ponzi contract classes in the training phase of the model.

The dataset for this experiment contains a total of 3884 data, which is not enough for deep learning. We tested some deep learning algorithms on the D2 dataset, and the result is that the overfitting of the model leads to poor generalization ability. For this paper, we need to

divide the samples into Ponzi contracts and ordinary contracts, which is equivalent to a dichotomous problem. Therefore, we chose KNN, CNN, DT, SVM, and XGBoost to compare with CTRF and used these six models to experiment on the D1-code dataset, D2-code dataset, D1-codeAndTran dataset, and D2-codeAndTran dataset, respectively. KNN classification algorithm is one of the simplest methods in data mining classification techniques. The so-called K nearest neighbors, which means K nearest neighbors, means that each sample can be represented by its closest K neighbor values to achieve classification. The CNN algorithm is widely used in the field of graph classification, but in recent years it has also been applied to the field of NLP. So, we applied CNN to detect Ponzi contract for comparing with CTRF. DT (decision tree) is a tree-structured algorithm that starts from the root node according to the corresponding features and thus selects branches until it reaches the leaf nodes, taking the category stored in the leaf nodes as the decision result. SVM (support vector machine) is a class of generalized linear classifier that performs binary classification of data in a supervised learning manner, where the decision boundary is the maximum margin hyperplane solved for the learned samples, and the elements are classified after this plane is finally determined. XGBoost is one of the Boosting algorithms. The idea of boosting algorithm is to integrate many weak classifiers to form a strong classifier. Since XGBoost is a boosting tree model, it is integrating many tree models to form a very strong classifier. And the tree model used is the CART regression tree model.

5. Experimental Results and Feature Analysis

5.1. Experiment Setting. Datasets. To compare the validity of the datasets and features, we did experiments on four main datasets.

(1) D1-code: code features extracted from the corrected Chen's address set as the dataset

(2) D1-codeAndTran: a dataset consisting of code features and transaction features extracted from the modified Chen's dataset

(3) D2-code: code features extracted from the expanded dataset as a dataset

(4) D2-codeAndTran: a dataset consisting of code and transaction features extracted from the expanded dataset

We conducted independent experiments on these four datasets: first cross-validating to find the best experimental parameters, then using 70% of the dataset for training and 30% for testing, and finally conducting 20 experiments to calculate the average results.

Evaluation Metrics. In this paper, precision, recall, and F-score are used as the evaluation criteria for the experimental results. Among them, precision is the proportion of all contracts judged as a certain category that are contracts of that category. The recall is the proportion of the number of detected Ponzi contracts to the total number of Ponzi contracts. The F-score is a summation of the precision and recall values. The solution formula for the three selected metrics is shown in (1)–(3).

$$\text{Precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}, \tag{1}$$

$$\text{Recall} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}, \tag{2}$$

$$F - \text{score} = 2 \times \frac{\text{Precision}}{\text{Precision} + \text{Recall}}. \tag{3}$$

True positive is the number of Ponzi contracts that are correctly determined. False positive is the number of non-Ponzi contracts that are misclassified as Ponzi contracts. False negative is the number of Ponzi contracts that are misclassified as non-Ponzi contracts.

5.2. Results Summary. Table 2 summarizes the results of the corresponding features of the original and expanded datasets under different methods. After analyzing the data in the table, we got the following conclusions.

It is clear that CTRF outperforms other algorithms in terms of precision and recall, and CTRF and XGBoost also outperform the original dataset D1 on the D2 dataset after our positive sample expansion. Although KNN, CNN, DT, and SVM perform well in terms of recall, their precision is poor.

D1-code. Since we added the sequence feature of the opcode, the experimental results improve the recall value by 11% and the F1 value by 7% compared with those in [28]. This indicates that sequence features of the opcode can help the model to identify more Ponzi contracts.

D1-codeAndTran. The recall value obtained in the experiment of Chen et al. was 0.69 [28]. In contrast, we achieve a recall of 0.85, which is a full 16% improvement. Our selected transaction features are better than those selected by Chen et al., and the recall of the experiment improves slightly after the inclusion of the transaction features. The addition of transaction features does enhance the model's identification of Ponzi contracts.

D2-code. After expanding the data, the recall value is improved by 3%. This means the imbalance between positive and negative samples affects the effect of the model. Therefore, CTRF shows a higher recall value in the D2 dataset after we expand the positive sample and oversample it.

D2-codeAndTran. After adding transaction features to the extended dataset, the experimental results are slightly improved compared with those on the extended dataset D2-code without transaction features, where the recall value is improved by 2%. It proves that our extracted transaction features such as Gini coefficients can indeed help the model identify more Ponzi contracts.

Table 2: The experimental results on four datasets.

Metric		KNN	CNN	DT	SVM	XGBoost	CTRF
Precision	D1-code	0.518	0.486	0.500	0.586	0.918	0.953
	D2-code	0.501	0.630	0.551	0.543	0.926	0.933
	D1-codeAndTran	0.485	0.621	0.571	0.642	0.907	0.929
	D2-codeAndTran	0.478	0.705	0.611	0.545	0.918	0.928
Recall	D1-code	0.803	0.583	0.836	0.836	0.811	0.847
	D2-code	0.813	0.773	0.812	0.788	0.863	**0.875**
	D1-codeAndTran	0.787	0.683	0.721	0.852	0.828	0.852
	D2-codeAndTran	0.801	0.761	0.863	0.763	0.873	**0.891**
F-score	D1-code	0.628	0.530	0.626	0.689	0.862	0.897
	D2-code	0.619	0.694	0.657	0.643	0.893	0.903
	D1-codeAndTran	0.601	0.651	0.638	0.732	0.865	0.889
	D2-codeAndTran	0.598	0.732	0.715	0.635	0.894	0.909

Table 3: The importance of the twenty most significant features.

	D1-code	D2-code	D1-codeAndTran	D2-codeAndTran
1	LT	LT	LT	LT
2	SLOAD	LOG	LOG	maxSend
3	LOG	CALLDATALOAD	AND	totalSend
4	GAS	SLOAD	maxSend	addrGetPro
5	AND	CALL	SLOAD	avgFee
6	CALLDATALOAD	AND	MSTORE	LOG
7	SSTORE	RETURN	totalSend	Gini
8	CALL	STOP	CALLDATALOAD	AND
9	MSTORE	SUB	MUL	SLOAD
10	MUL	GAS	SSTORE	CALL
11	GT	MSTORE	SHA	CALLDATALOAD
12	SHA	MUL	GAS	RETURN
13	DUP	RETURNDATASIZE	SUB	totalGet
14	RETURN	CALLDATASIZE	DUP	GAS
15	SUB	SSTORE	GT	STOP
16	STOP	GT	STOP	SUB
17	TIMESTAMP	CALLVALUE	avgFee	MSTORE
18	ADD	CODECOPY	CODECOPY	SHA
19	MLOAD	MLOAD	CALL	GT
20	OR	EXTCODESIZE	ISZERO	MUL

5.3. Feature Analysis. We obtained the corresponding feature importance rankings by analyzing the performance of the CTRF model on the four datasets, as shown in Table 3.

On D1-code and D2-code, our operand sequence features (D2V) extracted by Doc2Vec can better help the model to identify the Ponzi contracts. Among the more important word features extracted by bag-of-words are LT and log.

LT represents less than judgment in EVM. After calculation, we concluded that on average each Ponzi contract has 14 LT opcodes, while each non-Ponzi contract has only 11 on average. And most of the Ponzi contracts have fewer opcodes corresponding to them than non-Ponzi contracts. Only the number of LT opcode is more in Ponzi contracts than in non-Ponzi contracts, which is why LT opcode is so important for detecting Ponzi contracts. The importance of LT opcode is also reflected in the Ponzi contract code. We observed a large number of Ponzi contracts and found that when most of them receive a transfer from an external account, they will determine whether the amount of the transfer is less than the minimum investment threshold set by the contract. If it is less than that, the investment will be swallowed directly and no subsequent returns will be made to the investor.

During the training process of the D2-codeAndTran dataset, the importance of our newly added transaction features is located in the top positions, and according to the test results, the newly added transaction features make the model outperform the D2 code, which ultimately leads to a 2% improvement in the recall value. It is worth mentioning that in [28] after Chen et al. added transaction features to the dataset, the accuracy of the model improved slightly, but the recall of the model decreased by 4%. In our experiments, on the other hand, we have significantly improved the recall of the model by adding transaction features such as Gini coefficients. Compared with the transaction features extracted by Chen et al., our extracted transaction features are more helpful for the model to identify Ponzi contracts.

6. Conclusion

Nowadays, the issue of on-chain security on Ethereum is attracting more and more attention. Some researches on Ethereum security have also emerged. In this study, we extracted the classification model CTRF for the identification and analyzed of Ponzi schemes on Ethereum. First, we relied on increasing the number of positive samples to get the original dataset D1 and the expanded dataset D2 and then extracted the code features and transaction features of the two datasets, respectively, to get four datasets D1-code, D2-code, D1-codeAndTran, and D2-codeAndTran, and each dataset is divided into training and testing sets according to the ratio of 7 : 3. The training set of the four datasets is then oversampled to deal with the problem of positive and negative sample imbalance. Finally, the corresponding models are trained on each of the four training sets and tested on the test set to obtain the test results.

From the test results, the expanded dataset D1-codeAndTran, the recall value is improved by about 16% compared with the results in [28]. And the model is still able to produce good results without transaction features, and adding our extracted transaction features improves the recall value of the model identification.

In the future, we will make a deeper study on the identification of Ethereum Ponzi contracts. We expect to extract serialized features from the bytecodes of contracts from a deep learning perspective, and then, train them. Then, by comparing the similarity of bytecode sequences between contracts, we can identify Ponzi contracts. In addition, we also expect to build an Ethereum Ponzi contract detection platform to identify and record Ponzi contracts on Ethereum, so as to prevent investors from being cheated. In conclusion, in the future, we will continue our research on Ponzi contracts on Ethereum and maintain the safe and stable development of the Ethereum system.

Data Availability

The datasets mentioned in the article are available at https://github.com/BuptHxz/DetectionOfPonziContract.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under grant no. 2019YFC1521101.

References

[1] S. Nakamoto and A. Bitcoin, "Bitcoin: a peer-to-peer electronic cash system," *Bitcoin*, 2008, https://bitcoin.org/bitcoin.pdf.

[2] J. T. Lawrence, "Virtual currencies; bitcoin & what now after liberty reserve, silk road, and mt. gox?" *Richmond Journal of Law and Technology*, vol. 20, no. 4, 2014.

[3] E. Gün Sirer, "Thoughts on the dao hack," 2016, http://hackingdistributed.com/2016/06/17/thoughts-on-the-dao-hack/.

[4] "Solidity official documentation," 2020, https://docs.soliditylang.org/en/v0.8.3/.

[5] "Chainalysis," 2020, https://go.chainalysis.com/2020-Crypto-Crime-Report.html Crypto crime report.

[6] 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.

[7] M. Vasek and T. Moore, "Analyzing the bitcoin ponzi scheme ecosystem," in *Proceedings of the International Conference on Financial Cryptography and Data Security*, pp. 101–112, Springer, Nieuwpoort, Curaçao, February 2018.

[8] Y. Boshmaf, C. Elvitigala, H. Al Jawaheri, P. Wijesekera, and M. Al Sabah, "Investigating mmm ponzi scheme on bitcoin," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pp. 519–530, Taipei Taiwan, October 2020.

[9] M. Bartoletti, B. Pes, and S. Serusi, "Data mining for detecting bitcoin ponzi schemes," in *Proceedings of the 2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pp. 75–84, IEEE, Zug, Switzerland, June 2018.

[10] Z. Zheng, S. Xie, H.-N. Dai et al., "An overview on smart contracts: challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, 2020.

[11] H. Chen, M. Pendleton, N. Laurent, and S. Xu, "A survey on ethereum systems security: vulnerabilities, attacks, and defenses," *ACM Computing Surveys*, vol. 53, no. 3, pp. 1–43, 2020.

[12] T. Hu, X. Liu, T. Chen et al., "Transaction-based classification and detection approach for ethereum smart contract," *Information Processing & Management*, vol. 58, no. 2, Article ID 102462, 2021.

[13] E. Jung, M. Le Tilly, A. Gehani, and Y. Ge, "Data mining-based ethereum fraud detection," in *Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 266–273, IEEE, Atlanta, GA, USA, July 2019.

[14] L. Yujian and L. Bo, "A normalized levenshtein distance metric," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1091–1095, 2007.

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

[16] S. Fan, S. Fu, H. Xu, and C. Zhu, "Expose your mask: smart ponzi schemes detection on blockchain," in *Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, IEEE, Glasgow, UK, July 2020.

[17] W. Chen, Z. Zheng, J. Cui, E. Ngai, P. Zheng, and Y. Zhou, "Detecting ponzi schemes on ethereum: towards healthier blockchain technology," in *Proceedings of the 2018 world wide web conference*, pp. 1409–1418, Lyon France, April 2018.

[18] N. Szabo, "Formalizing and securing relationships on public networks," *First Monday*, vol. 2, no. 9, 1997.

[19] V. Buterin, "Ethereum white paper," *GitHub repository*, vol. 1, pp. 22-23, 2013.

[20] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[21] "Authoritative definition of ponzi scheme," https://www.sec.gov/spotlight/enf-actions-ponzi.shtml.

[22] E. Etherscan, "Ponzi contract," https://etherscan.io/accounts/label/ponzi.

[23] 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.

[24] L. Kiffer, D. Levin, and A. Mislove, "Analyzing ethereum's contract topology," in *Proceedings of the Internet Measurement Conference 2018*, pp. 494–499, Boston MA USA, October 2018.

[25] L. Han, Z. Yang, Yu Jiang, W. Zhao, and J. Sun, "Enabling clone detection for ethereum via smart contract birthmarks," in *Proceedings of the 2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*, pp. 105–115, IEEE, Montreal, QC, Canada, May 2019.

[26] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: a statistical framework," *International Journal of Machine Learning and Cybernetics*, vol. 1, no. 1-4, pp. 43–52, 2010.

[27] J. HanLau and T. Baldwin, "An empirical evaluation of doc2vec with practical insights into document embedding generation," 2016, https://arxiv.org/abs/1607.05368.

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