Expose Your Mask: Smart Ponzi Schemes Detection on Blockchain

Shuhui Fan[1], Shaojing Fu[12*], Haoran Xu[1] and Chengzhang Zhu[3]
[1]College of Computer, National University of Defense Technology, Changsha, China
{fanshuhui18,xuhaoran12}@nudt.edu.cn
[2]Sate Key Laboratory of Cryptology, Beijing, China
shaojing1984@163.com
[3]Institute of War, Academy of Military Sciences, Beijing, China
kevin.zhu.china@gmail.com

***Abstract*—The anonymity of blockchain has caused Ponzi schemes to be transferred to smart contract platforms by scammers. These Ponzi schemes wearing the mask of smart contracts caused huge losses to people, which makes the detection of smart Ponzi schemes attract people's attention. Recent methods mainly focus on machine learning technology to enable automatic detection for smart Poniz schemes. However, there are some problems with their methods. Firstly, the gradient boosting algorithm in machine learning they used have the problem of prediction shift due to target leakage when processing category features and calculating gradient estimates. Secondly, they ignored the imbalance and repetitiveness of Ponzi schemes on smart contract platforms. These problems can directly lead to model overfitting and affect the generalization ability of trained models.**

This paper proposes a novel Ponzi schemes detection method on smart contract platform for blockchain. Our method addresses the above issues with the following strategies. Firstly, we leverage ordered target statistic (TS) to process the category features of smart contract. Secondly, we solve the imbalance of dataset through a data augmentation method. Thirdly, with the idea of ordered boosting algorithm, we train a PonziTect model to fight prediction shift caused by target leakage. Based on the above ideas, the experimental results fully manifest the effectiveness and reliability of our model in detecting smart Ponzi schemes on the blockchain. Specifically, our model achieves 98% F-score on the real-world dataset, which significantly outperforms the existing methods. Using our method, we estimate that there are about 532 Ponzi schemes on Ethereum.

***Index Terms*—blockchain, smart contract, ponzi schemes, ethereum, machine learning, data mining.**

I. Introduction

Blockchain has the characteristics of decentralization, anonymity, immutability, and security. Its appearance has provided a breeding ground for Ponzi schemes. Due to the lack of effective regulatory mechanisms on the blockchain and the general lack of awareness among many investors, some scammers see the potential profit prospects of operating frauds on the blockchain platform.

According to [1], the Ponzi schemes operating on Bitcoin from September 2013 to September 2014 has collected more than $7 million. M. Bartoletti et al. [2] found 191 Ponzi schemes on Ethereum, which have collected approximately $400,000 from August 2015 to May 2017. Ponzi schemes on Bitcoin are mainly posted through forums, e.g., bitcointalk.org, where scammers advertise Ponzi schemes as "high-yield investment programs" (HYIP), or as gambling games, while Ponzi schemes on Ethereum hidden under the mask of smart contracts. Due to the popularity of Ethereum, many smart contract platforms such as EOS, RSK and Echo have appeared. Their smart contracts are written in a high-level programming language, and they all adopt an Ethereum-like design that can implement any application. Once a smart contract is deployed on blockchain, it cannot be changed and automatically stopped. However, investors who have little knowledge of blockchain can hardly distinguish the true face of these smart contracts disguised as HYIP.

This work focuses on the problem of detecting Ponzi schemes on smart contract platform for blockchain. Early detection methods mainly through manual analysis [2] [3] which primarily rely on etherscan.io to retrieve verified contracts which are associated with a name, identifying Ponzi schemes by manually inspecting their source codes (including comments) and project website if available. Such detection poses the challenge of analyzing billions of non-open source smart contracts on the blockchain. According to [4], 77.3% of the smart contracts on Ethereum are non-open source, so it is impossible to judge the type of contract only by manual analysis. Another method is comparing contract similarities [2] by using the normalized Levenshtein distance (NLD) as a measure of contract's bytecode in an attempt to discover hidden Ponzi schemes from a large number of non-open source contracts. This method quantifies the difference between two strings by calculating the number of character edits that convert one string to another. However, this method is limited by the number of known smart Ponzi schemes and can only find the Ponzi schemes with the same code as the known Ponzi schemes. The recent methods [5] [6] base on the contract's bytecode and transaction data using machine learning technology to enable automatic detection for smart Poniz schemes. However, the gradient boosting algorithms they used have the problem of prediction shift (the conditional distribution for a training contract is shifted from the distribution for a test

This work is supported by the National Nature Science Foundation of China (NSFC).

*Corresponding Author

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

contract) due to target leakage (gradients used at each step are estimated using the target values of the same data points the current model was built on) when processing category features and calculating gradient estimates.

Also, they ignored the fact that 96% of the smart contracts on Ethereum were duplicated according to [7], leaving the existing Ponzi scheme's dataset small and unbalanced. These problems will cause overfitting and weak generalization ability of their models. A potential solution to the above problems is to process the category features of smart contract by ordered TS and calculate gradient estimation by ordered boosting algorithm [8], since we need unbiased residuals for all training contracts that we maintain a set of models differing by contracts used for their training. Then, for calculating the residual on an contract, we use a model trained without it, which ensures that the target of the contract is not used for training the model (neither for TS calculations nor for gradient estimation). In addition, the number of Ponzi schemes can be increased by using a data augmentation methods.

In this paper, we propose a novel Ponzi schemes detection method on smart contract platform to solve the above problems. Our method leverages a few labeled contracts to train a model. That is, with the bytecodes of the contracts as inputs, we use data mining and machine learning technology to automatically detect smart Ponzi schemes as soon as they are deployed on blockchain. Specifically, we process the category features of smart contract by ordered TS, which relies on ordering principle that the values of TS for each feature base only on the prior history, and then increase the number of smart Ponzi schemes by using a data augmentation method. Lastly, we train a PonziTect model with the idea of ordered boosting to fight prediction shift caused by target leakage when calculating gradient estimates.

Accordingly, this paper makes the following major contributions.

- We propose a novel Ponzi schemes detection method on smart contract platform for blockchain. In contrast to the previous methods of manual analysis or other methods based on gradient boosting algorithm, we use the idea of ordered boosting to train the PonziTect model, in which the ordered target statistics approach can directly deal with category features and avoid prediction shift caused by target leakage. Therefore, smart Ponzi schemes can be detected as soon as they are deployed on the blockchain, based only on the bytecode of smart contract.
- We use a data augmentation method to solve the problem of imbalanced data set. We prefer to strengthen the proportion of smart Ponzi schemes at the boundary, and increase the randomness of artificially synthesized contracts to generate dataset closer to the real contract. This indirectly improves the quality and performance of our model.
- We show that our method is suitable for detecting Ponzi schemes in smart contract platforms for blockchain. Our model is generally superior to four competing models by greatly improving the recall rate while ensuring the precision rate. Using our model, we estimate that there are about 532 Ponzi schemes on Ethereum.

In the rest of this paper, we discuss the related work in Section II. Our method is detailed in Section III. Experiments are introduced in Section IV, followed by an evaluation in Section V. We conclude this work in Section VI.

II. Related Work

Existing smart Ponzi schemes detection methods on blockchain mainly include manual analysis, comparison of contract similarity and machine learning.

A. Manual Analysis-based Method

These manual analysis methods include following online forums [1] [2] [3], retrieving contracts and human analysis [2]. They normally search for ads claiming high return on investment manually on Reddit and bitcointalk.org, and then hunt for their Bitcoin addresses. Or they use etherchain.org to retrieve keywords like e.g. "Ponzi", "HYIP", "pyramid", "scam", "fraud", etc., and examine the declared interest rates. After that, they use etherscan.io to retrieve validated contracts with source code, and manually analyze the code, comments, and project website to confirm the type of contract. This method provides us with the most original labeled dataset. However, facing with hundreds of millions of contracts, we can no longer analyze them through time-consuming and inefficient manual methods.

B. Contract Similarity Comparison-based Method

In contrast to manual analysis methods, contract similarity comparison method [2] works on the bytecode of contract. This method generally has substantially better efficiency than manual analysis methods, since it use the normalized Levenshtein distance (NLD) to calculate the number of character edits that convert one bytecode to another as a measure of contract similarity. Some researchers also use this method to discriminate smart honeypots [9] on Ethereum. However, this method is limited by the number of known smart Ponzi schemes and can only find the Ponzi schemes with the same code as the known Ponzi schemes. What's more, when the amount of contract code is large, the speed of calculating the number of character edits in string conversion decreases.

C. Machine Learning-based Method

The very recent efforts [5] [6] to learn account features and code features of smart Ponzi schemes are based on gradient boosting algorithm and decision trees in machine learning. However, the gradient boosting algorithm XGBoost they used has the problem of target leakage that the gradients used at each step are estimated using the target values of the same data points the current model was built on, which results in prediction shift that the conditional distribution for a training contract is shifted from the distribution for a test contract. In addition, neither of their methods takes into account imbalance in the dataset, nor removes duplicate contracts. All of these can lead to overfitting and weak generalization ability of their models.

III. METHODS

In this section, we introduce the methods used in this work, including the overall workflow, the dataset used, data preprocessing, how to obtain features from the smart contracts, how to augment data, the proposed PonziTect model, and data preparation for models and classification.

A. Workflow

The overall workflow of this work is illustrated in Fig. 1. To build an effective model for detecting smart Ponzi schemes on blockchain, we first obtain the bytecodes of all the smart contracts in the dataset from Google BigQuery [10], a RESTful web service that supports interactive analysis of large dataset on the Ethereum blockchain. After removing the duplicate contracts, the data is then pre-processed by decompiling the bytecode of the smart contract into opcode. Later, we capture the category features of the opcode based on Bag of n-grams. In order to balance the dataset, we use a data augmentation method to synthesize more smart Ponzi schemes to match the number of non-Ponzi schemes. After that, a better model named PonziTect is proposed compared with other methods. Finally, the model is applied to detect potential smart Ponzi schemes on Ethereum.

B. Data Acquisition and Pre-processing

We download 3,660 verified non-Ponzi scheme contracts from the website[1] provided by W. Chen et al. [5], and 184 verified Ponzi scheme contracts from the website[2] provided by M. Bartoletti et al. [2][3], making a total of 3,844 contracts. However, the published dataset does not contain any bytecode, and part of the contracts have been self-destructed and their bytecodes are not available. Thus we get timestamp as input to execute SQL statements for interactive analysis with the Google BigQuery and download the bytecodes for 3,647 contracts, of which 3,394 are unique in terms of exact bytecode match. Eventually, we get 155 Ponzi schemes and 3239 non-Ponzi schemes. In the end, we do the conversion of these bytecodes into opcodes using disassembly tool called pyevmasm library[4], and remove the operands from the data.

C. Feature Extraction and Data Augmentation

During feature processing, we transform the opcode of every contract into eigenvectors by using Bag of Words (BOW) [11] on n-gram. To operate feature extraction, we create a Python dictionary of n-grams from the opcode sequence, which applies information metrics of the opcode context to the n-gram list. Our feature selection method filters out those frequently repeated opcodes such as PUSH, DUP and SWAP by specifying stop words in BOW.

Usually, these stop words have no effect on understanding the semantics of the entire logic of smart contract. For other opcode features, we choose those in our data that contribute most to the target variable by computing ANOVA F-value, which is used for classification task, for the provided sample to remove the less important features in the dataset and achieve the purpose of feature selection. As a final step, our feature construction method generate a feature vector for each contract, and all the vectors are combined into a word frequency matrix. These category features will be processed without target leakage by ordered target statistics.

Since the contracts on the borderline and the ones nearby are more apt to be misclassified than the ones far from the borderline, we use Borderline-SMOTE 2 [12] oversampling technology as data augmentation method to synthesize more smart Ponzi schemes to match the number of non-Ponzi schemes in order to achieve better prediction. It only generates new contracts for the Ponzi schemes that have more than half of the non-Ponzi schemes in the K nearest neighbors, which can strengthen the presence of smart Ponzi schemes at the boundary and make the new contracts closer to the true contract.

D. PonziTect Training

To reduce overfitting, we use all the data in the training set for training. Suppose there are n smart contracts in the dataset $\mathcal{D} = \{(\mathbf{x}_k, y_k)\}_{k=1,\ldots,n}$. According to some unknown distribution $P(\cdot,\cdot)$, contracts $(\mathbf{x}_k, y_k)$ are independently and identically distributed. $\mathbf{x}_k = \left(x_k^1, \ldots, x_k^m\right)$ is a random vector of m category features extracted from the opcode of the contract. $y_k \in \mathbb{R}$ is a label value. When $y_k = 1$, the smart contract is marked as a Ponzi scheme. Let arrangement $\sigma = (\sigma_1, \ldots, \sigma_n)$ be used to randomly arrange the dataset. For each contract we compute average label value for the contract with the same category value placed before the given one in the permutation. $x_{\sigma_p,k}$ is expressed by ordered target statistics:

$$\frac{\sum_{j=1}^{\mathrm{p}-1}\left[x_{\sigma_j,k} = x_{\sigma_p,k}\right] Y_{\sigma_j} + a \cdot p}{\sum_{j=1}^{\mathrm{p}-1}\left[x_{\sigma_j,k} = x_{\sigma_p,k}\right] + a} \tag{1}$$

Where $[\cdot]$ indicates Iverson brackets, i.e., $\left[x_{\sigma_j,k} = x_{\sigma_p,k}\right]$ equal to 1 if $x_{\sigma_j,k} = x_{\sigma_p,k}$, otherwise equal to 0. p is a priori value, the parameter $\mathrm{a} > 0$, which is the weight of p.

We convert every category feature to numerical features according to (1). Then we train the function $F : \mathbb{R}^m \to \mathbb{R}$ to minimize the expected loss:

$$\mathcal{L}(F) := \mathbb{E}L(y, F(\mathbf{x})) \tag{2}$$

Where $L(\cdot,\cdot)$ is a smooth loss function and $(\mathbf{x}, y)$ is a test contract sampled from P, independent of training set $\mathcal{D}$.

Further, we use the gradient boosting program to iteratively construct a series of approximations $F^t : \mathbb{R}^m \to \mathbb{R}, t = 0, 1, \ldots$ in a greedy way:

$$F^t = F^{t-1} + \alpha h^t \tag{3}$$

[1] ibase.site/scamedb

[2] goo.gl/CvdxBp

[3] The original dataset collected in [2] contained 137 Ponzi schemes, and the authors added more to the dataset in 2019.

[4] https://github.com/crytic/pyevmasm

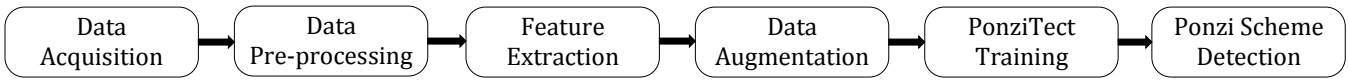

Fig. 1. The overall workflow of this work.

Where α is the step size and $h^t : \mathbb{R}^m \rightarrow \mathbb{R}$ (basic predictor) is selected from the function family H to minimize the expected loss:

$$\begin{aligned} h^t &= \arg\min_{h \in H} \mathcal{L}(F^{t-1} + \alpha h^t) \\ &= \arg\min_{h \in H} \mathbb{E}\mathcal{L}(y, F^{t-1}(\mathbf{x}) + h(\mathbf{x})) \end{aligned} \tag{4}$$

We approximate the minimization problem with a negative gradient step by using least-squares approximation:

$$h^t = \arg\min_{h \in H} \mathbb{E}\big(-g^t(\mathbf{x}, y) - h(\mathbf{x})\big)^2 \tag{5}$$

Where the gradient step $h^t(\mathbf{x})$ approximates $-g^t(\mathbf{x}, y)$:

$$g^t(\mathbf{x}, y) := \left.\frac{\partial L(y, s)}{\partial s}\right|_{s=F^{t-1}(\mathbf{x})} \tag{6}$$

We use a binary decision tree as the base predictor. The decision tree recursively divides the category feature space $\mathbb{R}^m$ into a number of disjoint regions R_j(tree nodes) according to the segmentation attribute a, and b_j is the leaf value. $a = \mathbb{1}_{\{x^k > t\}}$ is used to determine whether the category feature x^k exceeds the threshold t. The decision tree h can be written as:

$$h(\mathbf{x}) = \sum_{j=1}^{J} b_j \mathbb{1}_{\{\mathbf{x} \in R_j\}} \tag{7}$$

Finally, we assign a value to each final region (leaf of the tree), which is an estimate of the predicted class label in Ponzi schemes detection.

The pseudocode of PonziTect training algorithm is shown in Algorithm 1. In the $BuildTree$ function, we train a separate model M_k for each contract $\mathbf{x}_k$ and the model M_k is never updated using a gradient estimate for this contract, so as to use unbiased estimates of the gradient step. With M_k, we use $CalcGradient$ function to calculate the gradient on $\mathbf{x}_k$ and use this estimate to score the resulting tree. The function $GetLeaf(\mathbf{x}, T, \sigma_r)$ is used to calculate the leaf $leaf_r(\mathbf{x})$ that matches the contract $\mathbf{x}$, and σ_0 serves for choosing the leaf values b_j of the obtained trees. In line 20, $ApplyMode$ is used to replace a permutation σ in $GetLeaf$, which makes it practical to apply the training model to the new contract using all training data.

IV. Experiments and Evaluation

In this section, we validate our method on Ethereum and measure the overall prevalence of smart Ponzi schemes currently active on the Ethereum. We first give the experimental environment configuration parameters and experimental results. Then, we evaluate the correctness and validity of our model.

Algorithm 1: PonziTect Training

Input: A set of contracts $\mathcal{D} = \{(\mathbf{x}_k, y_k)\}_{k=1}^n$, the number of trees I, loss function L, step size α, s, boosting mode $Mode = Ordered$

Output: $F(\mathbf{x})$

1 Generate σ_r for random arrangement of dataset $\mathcal{D} = \{(\mathbf{x}_k, y_k)\}_{k=1}^n$ for $r = 0..s$;
2 Initialize the model $M_0(\mathbf{x}_k) = 0$ for $k = 1..n$;
3 **for** $j = 1 \rightarrow n$ **do**
4 $M_{r,j}(\mathbf{x}_k) = 0$ for $r = 1..s$, $k = 1..2^{j+1}$;
5 **for** $t = 1 \rightarrow I$ **do**
6 $T_t, M_r = BuildTree(\{M_r\}_{r=1}^s, \{(\mathbf{x}_k, y_k)\}_{k=1}^n, \alpha, L, \{\sigma_i\}_{i=1}^s)$;
7 $leaf_0(\mathbf{x}_k) = GetLeaf(\mathbf{x}_k, T_t, \sigma_0)$ for $k = 1..n$;
8 $g_0 = CalcGradient(L, M_0, y)$;
9 **foreach** $leaf\ j\ in\ T_t$ **do**
10 $b_j^t = -avg(g_0(\mathbf{x}_k)$ for $\mathbf{x}_k : leaf_0(\mathbf{x}_k) = j)$;
11 $M_0(\mathbf{x}_k) = M_0(\mathbf{x}_k) + \alpha b^t_{leaf_0(\mathbf{x}_k)}$ for $k = 1, ..n$;
12 return $F(\mathbf{x}) = \sum_{t=1}^{I} \sum_{j} \alpha b_j^t \mathbb{1}_{\{GetLeaf(\mathbf{x}, T_t, ApplyMode) = j\}}$

A. Experimental Settings

After augmenting the minority data, the number of Ponzi schemes is increased to 3239, which is equal to the number of non-Ponzi schemes. To leverage the ability of PonziTect for detecting smart Ponzi schemes effectively, we make cross validation on the data to avoid overfitting and underfitting. We also find the best parameters through automatic hyperparameter tuning. Firstly, we randomly select 80% of the data as a training set and 20% of the data as a test set. Secondly, we adopt 5-fold cross-validation on the training set and use the average value of F-score to characterize the performance of PonziTect model. Finally, we take $1-F-score$ as the objective function that needs to be minimized by using Optuna [13], an open source automatic superparameter optimization frameworks. All experiments are conducted on a 64-bit Ubuntu 16.04 with kernel version 4.13.0-36.

For the PonziTect model, the combination optimization of five important parameters includes iterations, depth, learning_rate, l2_leaf_reg and bagging_temperature, where l2_leaf_reg means the coefficient at the L2 regularization term of the cost function and bagging_temperature means using the Bayesian bootstrap to assign random weights to objects. The best combination with 1-gram opcodes is iterations=553, depth=7, learning_rate=0.1809, l2_leaf_reg=183, and bagging_temperature=0.0102, with other parameters are

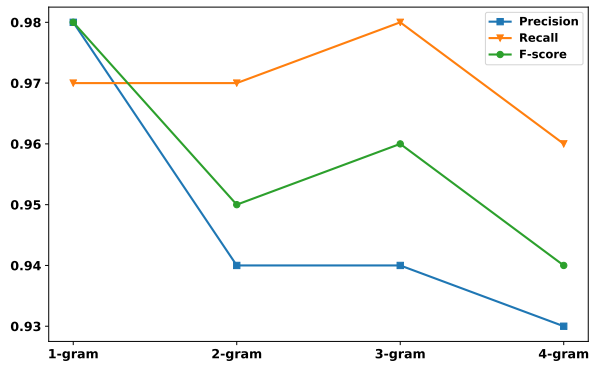

Fig. 2. The performance of PonziTect model with n-gram opcode for smart Ponzi schemes detection.

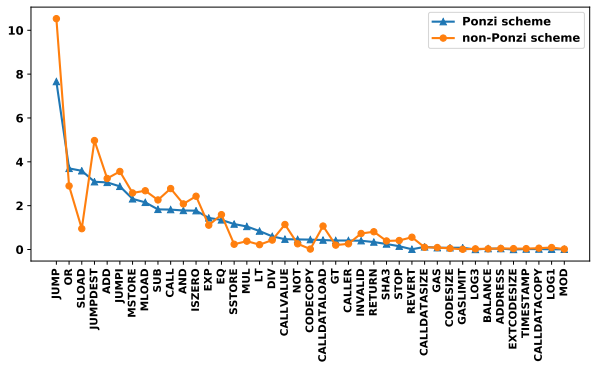

Fig. 3. Stacked bar chart with opcode ratios greater than zero for both the Ponzi schemes and the non-Ponzi schemes in our dataset.

default values. We evaluate the performance of our model on n-gram opcodes for smart Ponzi schemes detection with different n. In addition, the evaluation indicators frequently used in the industry include Precision, Recall and F-score, etc. Our model will also be evaluated using the above three metrics. The calculation formula is as follows:

- **precision**: the ratio of actual Ponzi schemes to those classified as Ponzi.

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

- **recall**: the ratio of correctly classified Ponzi schemes to all smart Ponzi schemes.

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

- **F-score**: the weighted harmonic average of precision and recall.

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

B. Classification Results

We apply the proposed method with the experimental configurations defined before to test 679 contracts, including 642 non-Ponzi schemes and 37 Ponzi schemes. We first show the performance of PonziTect model with n-gram opcodes for Ponzi schemes detection, and then reproduce other methods for comparative experiments on the same data set.

In many task of malware detection [14], [15], n-gram opcode-based methods have proven to be more advantageous than traditional malicious detection methods. Therefore, opcode features can also play a role in the detection of Ponzi scheme contracts. Fig. 2 depicts the overall performance of the PonziTect model with n-gram opcodes for Ponzi scheme detection. With the value of n ranging from 1 to 4, we repeatedly tune the parameters of the PonziTect model in order to obtain a good performance. However, one interesting observation is that PonziTect shows a best performance with F-score as high as 98% when n is 1 compared to the others. The recall rate increases as n is increased but declines when n is greater than 3. At the same time, the precision rate goes down after n is greater than 1. This indicates that with the increase of n, the false positive rate of the model also increases, resulting in the decrease of precision rate. Affected by recall and precision at the same time, the overall value of F-score has a downward trend.

In order to further confirm that our model's performance is reliable when n equals to 1, we conduct detailed statistics on the frequency of each opcode appearing in the dataset. The statistical results are shown in Fig. 3. Based on the statistical results, it is hard to tell the type of a smart contract. However, we observe that smart Ponzi schemes involve more SLOAD and SSTORE than non-Ponzi schemes. Actually, SSTORE is used to save all contract's fields and mappings in storage to create persistent associative maps and SLOAD is used to read data from storage. Smart Ponzi schemes also contain more logical operation codes such as EXP, NOT, DIV, LT and MUL, while non-ponzi schemes contain more jump-related instructions such as JUMP, JUMPI and JUMPDEST.

With the same dataset and opcode features, we compare our method with other competing methods. The important parameters used by the competing methods have also been tuned by Optuna. A brief description of the competitive method is given below:

- **M1**: a method proposed by Chen et al. [6] that uses Random Forest algorithm [16] to create independent decision trees, each of which spits out a class prediction and the class with the most votes becomes the model's prediction. We reproduce this method for comparative experiments on the same data and features, because this method is also based on decision trees and has been applied to detect smart Ponzi schemes.
- **M2**: a method proposed by Chen et al. [5] that uses a gradient boosting algorithm called XGBoost [17] to create gradient boosted decision trees in sequential form, and then ensemble these individual classifiers to provide a strong model. We reproduce this method using the same data and features and regard this method as a baseline because it has been applied to detect smart Ponzi schemes but has a problem of prediction shift due to target leakage.
- **M3**: a method leverages gradient boosting framework

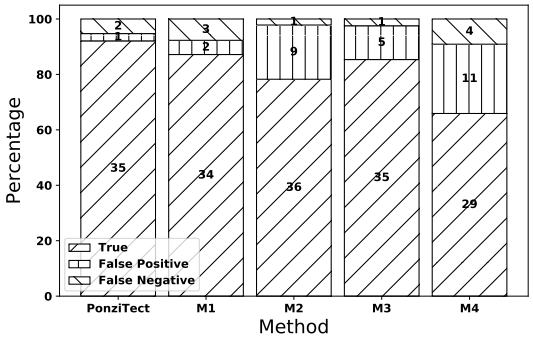

Fig. 4. The detection number of true, false positive, and false negative Ponzi schemes per method.

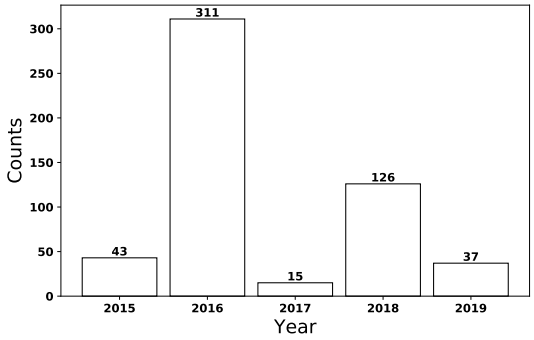

Fig. 5. The estimated number of surviving smart Ponzi schemes on Ethereum.

named LightGBM [18] based on decision tree algorithm. We choose this method for comparative experiments on the same data and features, because this method also has the same problem in M2.

- **M4**: a method using SVM algorithm [19] aims to find a hyperplane in an N-dimensional features space that distinctly classifies the data points. We choose this method for comparative experiments on the same data and features, because this method has been early applied in binary classification problems.

Fig. 4 depicts the detection results of smart Ponzi schemes under the optimal combination parameters of the models trained by each method. Out of the 679 analysed contracts, PonziTect has correctly labeled 641 as non-Ponzi schemes and 35 as Ponzi schemes, with two false negatives and one false positive. M2 has labeled 36 as Ponzi schemes with only one false negative. However, it has a very high number of false positive, with a total of 9.

Similarly, although M3 only has one false negative, the number of false positive is 5, which is higher than PonziTect and M1. For M4, the number of false positive and false negative is 4 and 11, respectively, and it has the lowest accuracy compared with other methods. M1 performs better, but its performance is still a bit poor compared to PonziTect. Table I lists the evaluation metrics per method.

Several conclusions can be made from the table. First, PonziTect has the highest F-score and shows the best performance. Because it guarantees a high precision rate with only a small loss of recall, which shows that it can effectively distinguish Ponzi schemes and non-Ponzi schemes. However, the performance of M1 is slightly weaker then PonziTect. Second, M2 and M3 both have higher recall rates but lower precision, indicating that both of them have a high false positive rate. Third, M4 has the lowest F-score, which implies that its performance is the worst, not only the precision rate is low, but also the false positive rate is high.

TABLE I
A PERFORMANCE COMPARISON.

Method	Precision	Recall	F-score
PonziTect	0.98	0.97	0.98
M1	0.97	0.96	0.96
M2	0.90	0.98	0.94
M3	0.94	0.98	0.96
M4	0.87	0.94	0.90

C. Validation and Application

In order to confirm the correctness of our model, we perform a manual inspection of the source code of the contracts that have been misclassified. We verify the contracts by contract name, source code, code comments and transactions data. For example, the false positive contract named EthOne has no code comments but from the source code it defines a tree-based structure to record the address, amount and the order of investors. When the investment amount is less than a certain value, the investor's information will not be recorded and the investor has no return. If the investment amount is greater than that the contract will pay different returns to each investor depending on their order, and the earlier the order, the greater the payoff. We consider this contract to be indeed a smart Ponzi scheme since the code clearly shows the logic of Ponzi scheme. Therefore, the classification result of this contract is correct and it's not a false positive. We also review the two false negative contracts in the same way and confirm that they are indeed false negatives.

To further apply our model to the actual smart contract platform, we download over 9.26 million contracts from Ethereum from 1/1/2015 to 11/30/2019. After removing the contracts with duplicate bytecode, we get 245,346 unique contracts and apply our model on them to detect smart Ponzi schemes.

Fig. 5 shows the quantity of surviving smart Ponzi schemes on Ethereum in each of the past five years. Our results show that a total of 532 surviving Ponzi schemes and 244,814 non-Ponzi schemes were detected. In fact, the number of smart Ponzi schemes should be more than that, because some of the Ponzi scheme contracts have been self-destructed and their bytecodes are not available for detecting. In terms of quantity, there have been a small number of smart Ponzi schemes since the birth of Ethereum in 2015. However, since 2016, the number of smart Ponzi schemes has reached a peak,

with a total of 311 active contracts on Ethereum, accounting for more than half of all the detected smart Ponzi schemes. After entering 2017, the number of smart Ponzi schemes has dropped sharply and gradually increased again in 2018. However, the trend is down again in 2019.

V. Conclusion

In this work, we present a method that combines data mining and machine learning technology for automated detection of Ponzi schemes on smart contract platform for blockchain. In contrast to the previous methods based on gradient boosting algorithm in machine learning, we use the idea of ordered boosting to train the PonziTect model, in which the ordered target statistics approach can directly deal with category features and avoid prediction shift caused by target leakage. We also use a data augmentation method to solve the problem of imbalanced dataset in order to improve the quality and performance of our model. Therefore, smart Ponzi schemes can be effectively detected as soon as they are deployed on the blockchain, based only on the bytecode of smart contract. By comparing the performance of multiple competing methods, we prove that PonziTect not only achieves high precision, but also has a low rate of false positive. In a large-scale analysis of more than 17 million smart contracts on Ethereum, our model detects 532 surviving contracts to be smart Ponzi schemes. At the same time, our method can also be used to detect Ponzi schemes on other smart contract platforms or even be used to detect cryptocurrency-based frauds hidden under the mask of smart contract like phishing attacks [20] as long as enough data is collected. Also, other types of fraud based on smart contract may also be tracked and studied in a similar way.

In addition, Ponzi schemes have also been discovered in decentralized applications (dapp) [21] as well as Bitcoin [22], and new types of fraud have also emerged, such as "honeypot" smart contracts [9], which pretend to leak funds to victims through loopholes and eventually capture the victim's funds. There are also famous "pump and dump" (P&D) frauds in the stock market that have been transferred to the Bitcoin by scammers [23]. The above frauds may be just a tip of the iceberg on the blockchain platform. More types of fraud are waiting to be discovered. All of these studies have highlighted the urgent need for automated techniques to detect illegal cryptocurrency activity.

Acknowledgment

This work is supported by the National Nature Science Foundation of China (NSFC) under grant 61572026, Open Foundation of State Key Laboratory of Cryptology (No:MMKFKT201617), and National Key Research and Development Program of China (No. 2018YFB0204301).

References

[1] M. Vasek and T. Moore, "There's no free lunch, even using bitcoin: Tracking the popularity and profits of virtual vurrency scams," in *Financial Cryptography and Data Security*. Springer, 2015, pp. 44–61.

[2] 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.

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

[4] Y. Zhou, D. Kumar, S. Bakshi, J. Mason, A. Miller, and M. Bailey, "Erays: reverse engineering ethereum's opaque smart contracts," in *27th USENIX Security Symposium (USENIX Security'18)*. USENIX Association, 2018, pp. 1371–1385. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/zhou

[5] 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*. International World Wide Web Conferences Steering Committee, 2018, pp. 1409–1418.

[6] W. Chen, Z. Zheng, E. C. . Ngai, P. Zheng, and Y. Zhou, "Exploiting blockchain data to detect smart ponzi schemes on ethereum," *IEEE Access*, vol. 7, pp. 37 575–37 586, 2019.

[7] N. He, L. Wu, H. Wang, Y. Guo, and X. Jiang, "Characterizing code clones in the ethereum smart contract ecosystem," arXiv preprint arXiv:1905.00272, 2019.

[8] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2018, pp. 6638–6648.

[9] C. F. Torres and M. Steichen, "The art of the scam: Demystifying honeypots in ethereum smart contracts," arXiv preprint arXiv:1902.06976, 2019.

[10] Google, "Google bigquery - ethereum," https://bigquery.cloud.google.com/dataset/bigquery-public-data:ethereum_blockchain, 2018.

[11] 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.

[12] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *International conference on intelligent computing*. Springer, 2005, pp. 878–887.

[13] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 2623–2631.

[14] I. Santos, F. Brezo, J. Nieves, Y. K. Penya, B. Sanz, C. Laorden, and P. G. Bringas, "Idea: Opcode-sequence-based malware detection," in *International Symposium on Engineering Secure Software and Systems*. Springer, 2010, pp. 35–43.

[15] B. Kang, S. Y. Yerima, S. Sezer, and K. McLaughlin, "N-gram opcode analysis for android malware detection," *International Journal on Cyber Situational Awareness*, vol. 1, no. 1, pp. 231–255, 2016.

[16] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston, "Random forest: a classification and regression tool for compound classification and qsar modeling," *Journal of chemical information and computer sciences*, vol. 43, no. 6, pp. 1947–1958, 2003.

[17] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.

[18] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 3146–3154. [Online]. Available: http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf

[19] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[20] A. A. Andryukhin, "Phishing attacks and preventions in blockchain based projects," in *2019 International Conference on Engineering Technologies and Computer Science (EnT)*. IEEE, 2019, pp. 15–19.

[21] K. Wu, "An empirical study of blockchain-based decentralized applications," arXiv preprint arXiv:1902.04969, 2019.

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

[23] W. Chen, Y. Xu, Z. Zheng, Y. Zhou, J. E. Yang, and J. Bian, "Detecting "pump & dump schemes" on cryptocurrency market using an improved apriori algorithm," in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 2019, pp. 293–2935.