*electronics*

*Article*

# A Feature-Based Robust Method for Abnormal Contracts Detection in Ethereum Blockchain

**Ali Aljofey [1,2], Abdur Rasool [1,2], Qingshan Jiang [1,*] and Qiang Qu [1]**

[1] Shenzhen Key Laboratory for High Performance Data Mining, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China
[2] Shenzhen College of Advanced Technology, University of Chinese Academy of Sciences, Shenzhen 518055, China
* Correspondence: qs.jiang@siat.ac.cn; Tel.: +86-186-6532-6469

**Abstract:** Blockchain technology has allowed many abnormal schemes to hide behind smart contracts. This causes serious financial losses, which adversely affects the blockchain. Machine learning technology has mainly been utilized to enable automatic detection of abnormal contract accounts in recent years. In spite of this, previous machine learning methods have suffered from a number of disadvantages: first, it is extremely difficult to identify features that enable accurate detection of abnormal contracts, and based on these features, statistical analysis is also ineffective. Second, they ignore the imbalances and repeatability of smart contract accounts, which often results in overfitting of the model. In this paper, we propose a data-driven robust method for detecting abnormal contract accounts over the Ethereum Blockchain. This method comprises hybrid features set by integrating opcode n-grams, transaction features, and term frequency-inverse document frequency source code features to train an ensemble classifier. The extra-trees and gradient boosting algorithms based on weighted soft voting are used to create an ensemble classifier that balances the weaknesses of individual classifiers in a given dataset. The abnormal and normal contract data are collected by analyzing the open source etherscan.io, and the problem of the imbalanced dataset is solved by performing the adaptive synthetic sampling. The empirical results demonstrate that the proposed individual feature sets are useful for detecting abnormal contract accounts. Meanwhile, combining all the features enhances the detection of abnormal contracts with significant accuracy. The experimental and comparative results show that the proposed method can distinguish abnormal contract accounts for the data-driven security of blockchain Ethereum with satisfactory performance metrics.

**Keywords:** fraud detection; blockchain security; abnormal contracts; feature transformation; ensemble learning model



**Citation:** Aljofey, A.; Rasool, A.; Jiang, Q.; Qu, Q. A Feature-Based Robust Method for Abnormal Contracts Detection in Ethereum Blockchain. *Electronics* **2022**, *11*, 2937. https://doi.org/10.3390/electronics11182937

Academic Editor: Cheng-Chi Lee

Received: 25 July 2022
Accepted: 13 September 2022
Published: 16 September 2022



## 1. Introduction

Fraud detection is a set of activities to prevent acquiring money or property through false claims. It is pragmatically applied in various online and offline services, such as internet services of banking or cryptocurrencies. Due to the development of artificial intelligence and data mining techniques, the need to share the most diverse data has increased; meanwhile, fraudulent activities have also emerged extremely in internet services. Unfortunately, the generation of this data belongs to several different security areas, so there is difficulty in sharing it, especially when there is information related to privacy and security. The inclusion of automation in the blockchain has led to the rapid adoption of technology in different sectors involving online finance, the internet of things, supply chain management, healthcare, insurance, etc. [1]. Blockchain is a shared and immutable ledger that facilitates the process of recording transactions between receiving and sending parties, and tracking assets in the business network [2,3]. Meanwhile, Ethereum is a decentralized blockchain platform that creates a peer-to-peer network that securely implements and verifies application code, named smart contract accounts. Smart contracts allow participants to

transact with each other without a trusted central authority. The blockchain is a globally shared transaction database [4–6]. A block includes many transactions, and a transaction in Ethereum is a message sent from one account address to another, which carries the import information, for example, a function parameter, a contract bytecode, etc. Ethereum has two categories of accounts typically involved in the transactions on the platform: externally owned accounts and smart contract accounts. The substantial variance is that the smart contract account includes executable code, while the externally owned account does not [7].

Ethereum, the most popular platform for smart contract accounts, has a market capitalization of over USD $21 billion [8,9]. When smart contract accounts are written in a high-level language such as Solidity, they must be compiled into bytecode, and then uploaded to Ethereum to be executed and validated by each Ethereum node. As a popular and open-source blockchain platform, Ethereum publishes many smart contract accounts to implement different businesses. At the same time, the technical characteristics come from a combination of many advanced technologies based on data-driven approaches, which increases the complexity of the blockchain and leads to increased technical limitations between smart contract accounts and investors. High technical limitations prevent investors from familiarizing themselves with the specific trading logic of smart contract accounts that run on the Ethereum network. These obstacles allow fraudsters to introduce abnormal contract accounts into the blockchain investment ecosystem.

The abnormal contract accounts such as High Yield Investment Programs (HYIP) or Ponzi schemes are a deceptive investment masquerading as a hopeful high rate of return for investors. Abnormal contracts are smart contract accounts with special characteristics. Due to their fraudulent nature, abnormal contracts differ from other contracts in several ways: (i) in most cases, abnormal contracts send Ether to their investors; (ii) it is possible for some accounts to receive more payments than their investment counts, such as the creator who frequently charges fees through contracts; and (iii) to maintain the image of fast and high returns, abnormal contracts might pay back investors once they have sufficient balances, resulting in a low balance [10]. On the other hand, a normal contract account on blockchain refers to a smart contract, which is a type of Ethereum account. Therefore, they can send transactions via the network since they have a balance. Although they are not controlled by users, they are deployed to the network and run according to their programming. Users can then submit transactions that execute the functions defined in a smart contract. The smart contract account can be used to define rules that are automatically enforced by the code, just like a regular contract. By default, smart contract accounts cannot be deleted, and interactions with them cannot be undone. The abnormal contract accounts create returns to early investors by getting new investors. They present automated peer-to-peer transactions while taking advantage of the decentralization benefits provided by blockchain technology. As stated by [10], from August 2015 to May 2017, 191 abnormal contract accounts alive on Ethereum raised almost $500,000 from over 2000 different users. Recently, online financing systems such as crowdfunding and peer-to-peer networking have become very popular. At the same time, abnormal contract accounts appeared in between, causing a very bad social impact [11]. Scientists note that abnormal contracts embedded in the blockchain are pervasive.

Data-driven security (DDS) effectively balances such impacts by an activity compelled with data derived from any security application instead of theoretical or personal experience. DDS is an evolving interdisciplinary area that concentrates on research and development by applying machine learning, data mining, data science, and deep learning approaches to tackle the high-security issues in the HYIP. In order to achieve success in this balance, there is a need to use effective security measures through DDS to minimize losses and prevent crimes for blockchain and cryptocurrency investors. However, there are quite a few studies on detecting abnormal contracts through DDS in Ethereum.

Existing abnormal contract detection techniques on the blockchain mostly include manual analysis or machine learning [10,12,13]. Manual analysis techniques employ etherscan.io, an analytics platform for observing transactions and wallet addresses on the

Ethereum blockchain, to retrieve contracts that have been verified with the source code and manually inspect the contract source code to confirm the type of smart contract account. However, these detection techniques are challenging to analyze the many new contracts being created daily on the blockchain system for DDS. Thus, it is impractical to detect smart contract accounts only through manual analysis. Recent detection methods [14–17] depend on opcodes or account data of contract accounts using machine learning methods to detect abnormal contract accounts automatically.

However, previous machine learning methods for identifying abnormal contract accounts suffer from the following problems: first, it is difficult to obtain features that can effectively detect abnormal schemes, and statistical work around the features is also unhelpful. Extracting effective features is an important task in a machine learning-based fraud detection solution. Some new features may be limited due to the technical enhancement of blockchain technologies. Overall, feature extraction is a difficult process and researchers continue to monitor any potential feature, which can improve the fraud detection performance of malicious contracts by conducting experiments on the standard dataset. Second, although some work has been done using machine learning methods to detect fraud in smart contract accounts, the accuracy is inadequate for data-driven analysis. For instance, the random forest method achieves a 95% precision rate and 69% recall rate [15] which is due to the smaller number of instances of abnormal contracts compared to normal contracts. It is obvious that abnormal contract accounts can be detected more accurately. A number of applications, including critical power system applications, have utilized ensemble learners to enhance performance. By compensating for the weaknesses of individual classifiers with the strengths of others, ensemble learning provides a superior performance [17]. Using Gradient Boost algorithms and Extra-Tree algorithms, we developed an ensemble classification model to detect abnormal contract accounts more accurately. Third, the problem of severe data imbalance may affect the performance of machine learning classification algorithms on large datasets. As indicated by research [18], 96% of Ethereum smart contract accounts are duplicates. As stated by a report on etherscan.io, the total number of normal accounts and transactions on Ethereum is over 500 million and 3.8 billion, respectively. In contrast, the total number of phishing accounts deployed on etherscan.io is only 2041 and searching for abnormal smart contracts is like searching for a needle in a haystack [19]. This issue will lead to overfitting and weak generalization of the detection model.

Therefore, this paper presents a data-driven robust solution to detect fraud-based abnormal contract accounts over the Ethereum blockchain, which extracts three-fold features from contract opcodes, transaction history, and contract source code. In particular, we propose a hybrid feature set including n-gram opcodes features of contract account without expert participation, different transaction features from contract account transaction data, and TF-IDF character level features from contract account source code. These features are integrated and fed to train the proposed method through an ensemble of Extra-Trees and Gradient Boosting Machine classifiers based on weighted soft voting to enhance the fraud detection efficiency for abnormal contract accounts. Extensive experiments show that the proposed detection method has competitively performed on real Ethereum blockchain datasets in various evaluation metrics. Accordingly, this paper presents the following main contributions.

- This paper proposes a robust fraud detection method with a supervised learning-based data-driven solution, which can automatically and accurately identify abnormal contracts using a hybrid feature set of a contract account.
- In terms of feature extraction, we combine various transaction features with opcode n-grams and source code characters features extracted by the TF-IDF model to get more comprehensive features and then compare various machine learning techniques using these features set to measure efficiency.
- We released a public dataset of addresses, codes, and their transaction history for Ethereum contract accounts, which can hopefully be used to further research and applications on this topic.

- We designed an ensemble classification model combining Extra-Trees and Gradient Boost algorithms to enhance the detection accuracy of abnormal contract accounts.

The rest of this paper is structured as follows. Section 2 presents a brief review of related work on the detection of abnormal contracts. A detailed explanation of the data collecting and prepossessing, extracted features, and classification model is provided in Section 3. Empirical results and analysis are summarized in Section 4. Finally, we conclude the paper in Section 5.

## 2. Literature Review

This section reviews some of the current relevant work related to the detection of abnormal contract accounts. Recently, many scams have appeared frequently on finial online models [20,21] and have caused huge losses to investors. Thus, the detection of abnormal contract accounts aroused the interest of researchers.

Bian et al. proposed a machine-learning-based Initial Coin Offerings (ICOs) rating system called ICORATING. They analyzed 2251 smart contract accounts for ICO and relative information such as white papers, founding team, GitHub repository, website, etc., to correlate the life span and the price change of a digital currency with various levels [22]. Our method employs the Ethereum network information and considers all abnormal contracts, thus not only deceptive ICOs. Vasek and Moore anatomized 1780 bitcoin-based Ponzi schemes from http://bitcointalks.org (accessed on 1 July 2022) and denoted that the social interaction between scammers and victims influences the lifespan of frauds [13]. Bartoletti et al. [10] provided a comprehensive review of Ponzi contracts on Ethereum. They analyzed the behavior of the Ponzi schemes for smart contract accounts using similarities between the contract account bytecodes to classify 184 of them. However, these methods cannot handle the large number of new contracts being constructed every day. Nerurkar et al. [23] proposed a supervised learning model to detect illegal activities in bitcoin. Nine features were designed to train the proposed model, and 1216 bitcoin users are categorized into 16 classes.

To detect abnormal activity on the Ethereum blockchain, Farrugia et al. classified accounts based on their transaction data using the XGBoost algorithm. A total of 42 account features were obtained using the collected transactions. The time differences between the first and last transaction, the total Ether balance, and the minimum Ether value received by an account was identified as the most significant three features [24]. Depending on the two dominant account types—externally owned accounts (EOA) and smart contract accounts—Kumar et al. [25] identified malignant nodes using supervised machine learning-based fraud detection in the account's transactions data. Wuet et al. [18] developed a network embedding algorithm named trans2vec to obtain the features of Ethereum accounts and built a one-class SVM to classify the malicious nodes. Chen et al. [26] proposed a deep learning-based solution called MTCformer for detecting Ponzi schemes. The MTCformer first employs Structure-Based Traversal (SBT) method to create the token sequence to hold the structural information and then uses a TextCNN and a multichannel transformer to automatically obtain the structural and semantic features from the source-code and to find out the long-term dependencies between the tokens. Wang et al. [27] presented a Ponzi schemes detection approach based on oversampling-based Long Short-Term Memory (LSTM). The proposed approach integrates the features of transaction data with opcodes features of smart contract accounts to detect Ponzi schemes on Ethereum blockchain. In the work of Chen et al. [28], a semantic-aware detection model has been proposed to detect Ponzi contracts in the Ethereum. They provided a guiding symbolic implementation approach to first creating semantic information for each potential pathway in smart contracts and then defining investor-related conversion behaviors and approved distribution strategies through strict adherence to the definition of Ponzi schemes. Liang et al. [29] proposed a data driven security system for detecting a Ponzi scheme. The system employs a dynamic graph embedding technique to automatically recognize the representation of an account based on the account's opcode and transaction data.

In a bid to moderate illicit activity through the Ethereum blockchain via detecting Ponzi schemes deployed as contract accounts. Fan et al. [30] classified contracts based on their operation codes (opcodes). They proposed an anti-leakage smart Ponzi schemes detection (Al-SPSD) method based on the idea of ordered boosting. The method of [16] applied data-mining techniques to identify bitcoin addresses relative to Ponzi scheme contracts. Their method was able to classify 31 out of 32 Ponzi schemes with a 1% false positive. Rahouti et al. [31] reviewed data-mining-based techniques for detecting anomalies in bitcoin, including [16]. However, it inspired the approach of [15], which describes a similar method using transaction features but also added opcode features based on contract bytecode stored on the blockchain. They created three classification models, account, opcode, and account + opcode, using a random forest classifier, and the best outcomes come from opcode model, with an f1-score at 73% and recall at 82%. Jung et al. [32] extracted more transaction features based on the [15] method, but the data set simplification is probably causing the method to overfitting. We have taken four different steps in building our detection method. First, we extended the abnormal contracts dataset that has been verified by etherescan.io [10]. Second, we extracted new opcodes and transaction features that are more effective in detecting abnormal contracts; furthermore, the method and strategy of the observational with respect to the interpretation of these features [15] are implemented differently in our method. Third, we added source code features, which we call TF-IDF source code features. Fourth, we used a different classification model and achieved better performance. Table 1 shows a detailed comparison of the abnormal contract detection methods.

**Table 1.** Summary of the literature review.

| Author | Description | Dataset | Classifiers |
| --- | --- | --- | --- |
| Bian et al. [22] | - Propose deep-learning system to help investors detect scam initial Coin Offerings (ICO) projects.<br>- It considers just fraudulent ICOs, not all Ponzi schemes. | A private dataset of 2251 past ICO projects, which includes white papers, website information, and GitHub repositories from various providers. | Hierarchical LSTM, LDA |
| Bartoletti et al. [10] | - Provide a comprehensive review of abnormal contracts on Ethereum, analyze their behavior and impact from different views | A public dataset of 184 Ponzi schemes deployed on Ethereum (goo.gl/CvdxBp) | Manual analysis of the smart contract source code |
| Nerurkar et al. [23] | - Ensemble model of decision trees to detect illegal activities in the Bitcoin network.<br>- General understanding of machine learning algorithms to detect malicious users in the bitcoin network | A public dataset of 1216 Bitcoin addresses categorized into 16 categories | Ensemble tree, SVM, Logistic Regression, Random Forest, XGBoost |
| Farrugia et al. [24] | - A machine learning approach to detect illicit accounts on Ethereum based on their transaction history<br>- A total of 42 account features were extracted using the collected transactions. | A public data of 2179 illicit accounts coupled with 2502 normal accounts. | XGBoost classifier |
| Wuet et al. [19] | - Proposed a method to detect malicious frauds on the Ethereum network by mining its transaction data.<br>- Extract features automatically using network embedding algorithm regarding timestamp and amount. | A private dataset consisting of 500 million accounts, 3.8 billion transactions, and 1259 addresses were labelled as phishing addresses. | Network embedding algorithm called trans2vec, one-class SVM |
| Fan et al. [30] | - Introduced method for detecting smart Ponzi contracts in Ethereum blockchain-based opcode features.<br>- Predict the shift caused by the targeted leakage of smart contract category features using the idea of the ordered boosting algorithm. | A private dataset includes 386 smart Ponzi contracts and 3239 non-Ponzi contracts. | Ordered boosting algorithm |

**Table 1.** *Cont.*

| Author | Description | Dataset | Classifiers |
| --- | --- | --- | --- |
| Bartoletti et al. [16] | - Applied data-mining algorithms to identify bitcoin addresses relative to abnormal contract accounts.<br>- An open-source tool that extracts the dataset from the bitcoin blockchain | A public data set of addresses and features of bitcoin abnormal contracts (32 instances) with 6400 randomly chosen non-Ponzi instances. | RIPPER, Bayes Network, Random Forest. |
| Chen et al. [15] | - Proposed a machine learning model to identify smart Ponzi contracts on Ethereum based on opcodes and transaction data.<br>- Raise the number of Ponzi scheme contracts and the account features from their previous work (Chen et al. [12]) | A public dataset of 200 Ponzi contracts and 3580 non-Ponzi contracts | Decision trees, SVM, XGBoost |
| Jung et al. [32] | - Use data-mining techniques to present a identification model for Ponzi contracts on Ethereum, improving Chen et al. [13] work.<br>- Added more account features | 3203 non-Ponzi schemes and 172 Ponzi schemes based on [10] work | J48, Random Forest, Stochastic Gradient Descent |
| Chen et al. [26] | - Employ Multi-Channel TextCNN and Transformer to detect Ponzi Scheme Contracts. | They conducted experiments on [15] dataset, which contains 200 Ponzi contracts and 3588 non-Ponzi contracts. | Combines multi-channel TextCNN and Transformer |
| Liang et al. [29] | - A data-driven system that can automatically recognize a representation of the account based on the input data.<br>- The system generates a feature vector based on the structural information of the transaction network, the account control logic, and the dynamic changes of the transaction network. | 1251 normal contract accounts and 131 abnormal contract accounts based on [14] work. | MultiLayer Perceptron (MLP) |
| Chen et al. [28] | - Proposed a Ponzi detection approach based on semantic information extracted from contracts opcodes | A public dataset which consists of 133 Ponzi contracts and 1395 non-Ponzi schemes based on [14,15] work | A prototype system that compares semantics extracted from opcodes with summarized Ponzi scheme patterns |
| Our proposed method | - It uses machine learning techniques to provide abnormal contract detection model for the security of the Ethereum network.<br>- It extracts three-fold features from opcodes, transaction history, and contract account source-code. | A public dataset of 1596 normal contracts and 308 abnormal contracts | Ensemble of GMB and Extra-Trees algorithms |

The features extracted in some existing works are based on manual analysis and need extra labor because these features require resetting according to the dataset, which may affect the generalization of abnormal contract detection methods. We proposed three new feature sets to improve the accuracy of detecting abnormal contract accounts. We got the impetus from the above works and proposed our method in which the current work extracts n-gram features from opcodes without expert intervention, various transaction features from transaction data, and TF-IDF characters level features from contract account source code. Moreover, our method integrates and trains these features through a combination of bagging and boosting tree algorithms to detect new abnormal contract accounts. The further details of these feature sets and the proposed detection methods are comprehensively presented in Section 3.

## 3. Proposed Methodology

In this section, the proposed work for detecting abnormal contracts in the Ethereum network is presented, including workflow, data acquisition and preprocessing of extracted features, and classification model. Figure 1 illustrates the general workflow of our method in four phases.

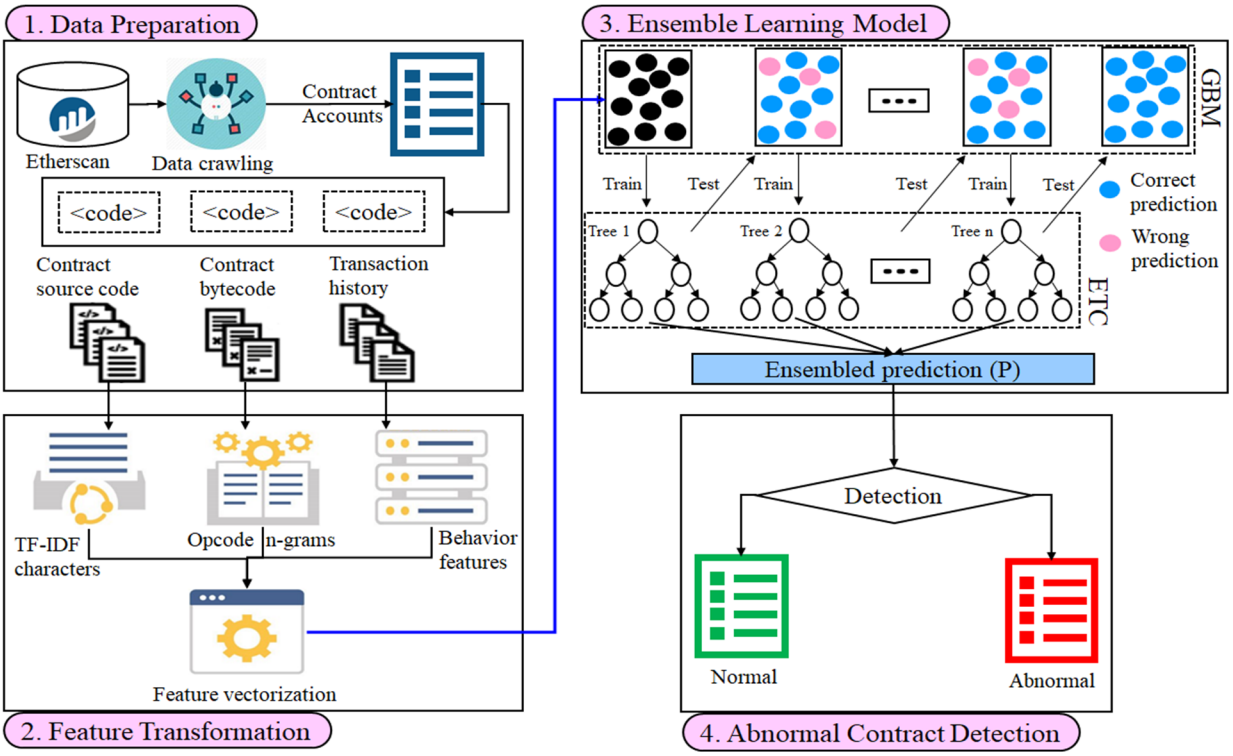



**Figure 1.** The schematic workflow of our proposed methodology for the abnormal contract detection.

The proposed method consists of four primary modules to create an efficient method for detecting abnormal contracts on Ethereum, which are explained below.

1. Data Preparation: The bytecodes, transactions, and source code of all the contract accounts in our data are obtained from etherescan.io. Then, the duplicate smart contracts bytecodes are removed and decoded into opcodes, and their transactions are preprocessed and saved into one table.
2. Feature Transformation: Three types of features such as n-grams, behavior, and TF-IDF characters are extracted from the opcodes, transaction data, and source code of smart contract accounts.
3. Ensemble Learning Model: The ensemble classifier combines extra-trees classifier (ETC) and gradient boosting machine (GBM) to train the extracted features. To synthesize more abnormal account contracts, the adaptive synthetic sampling (ADASYN) approach is used to sample density in the abnormal contracts to match the number of normal contracts.
4. Abnormal Contracts Detection: The process to differentiate whether the unknown contract accounts are abnormal or not.

### *3.1. Data Preparation*

A training dataset is required for both normal and abnormal contract accounts in Ethereum. Obtaining data is often the most difficult for fraud detection projects, which require manual analysis and careful research to classify abnormal from normal contracts. To this end, we used a web crawler to obtain unique bytecodes for 1904 contracts verified by etherscan.io, engaged in at least ten transactions, which were labeled as abnormal contracts (308) and normal contracts (1596). The abnormal addresses are obtained from [33], while the normal ones are obtained from [34].

We converted these bytecodes into opcodes using a disassembler called the pyevmasm library (https://github.com/crytic/pyevmasm, accessed on 1 July 2022), and removed the operands after some opcodes, for example, PUSH5 and its operands (such as 0x174876e800). We have also crawled the source codes of addresses collected using Etherscan API [35].

Moreover, we obtained all the corresponding transactions that reacted with these contract accounts using the Etherscan API [36]. The transactions include an 'isError' field, which indicates whether the transaction is successful or not. We use all the successful and unsuccessful transactions in our method. We used the DBMS (i.e., pgAdmin) with Python to import and setup the dataset. Our data consists of three files: the first file contains the bytecode of normal and abnormal smart contract accounts, while the second file contains the transaction records for these contracts. This file is extremely large, including 3,575,183 transaction records, making it necessary to include a varying number of transactions for each contract. Finally, the source code for these contract accounts is included in a last file as well. The data set was randomly divided into 80:20 ratios for training and testing, respectively. Our dataset of contract accounts and all the transaction data are available at this link: https://github.com/abdul-rasool/Abnormal-Contracts-Detection (accessed on 1 July 2022).

Figure 2 shows bytecode length distributions in our dataset for both abnormal and normal contract accounts. As shown in the figure, most normal contracts have a length of between 5000 and 8000 bytes, while most abnormal contracts contain lower than 5000 bytes. The highest length of the normal contract accounts is 45,000, and the highest length of abnormal contract accounts is 35,000.

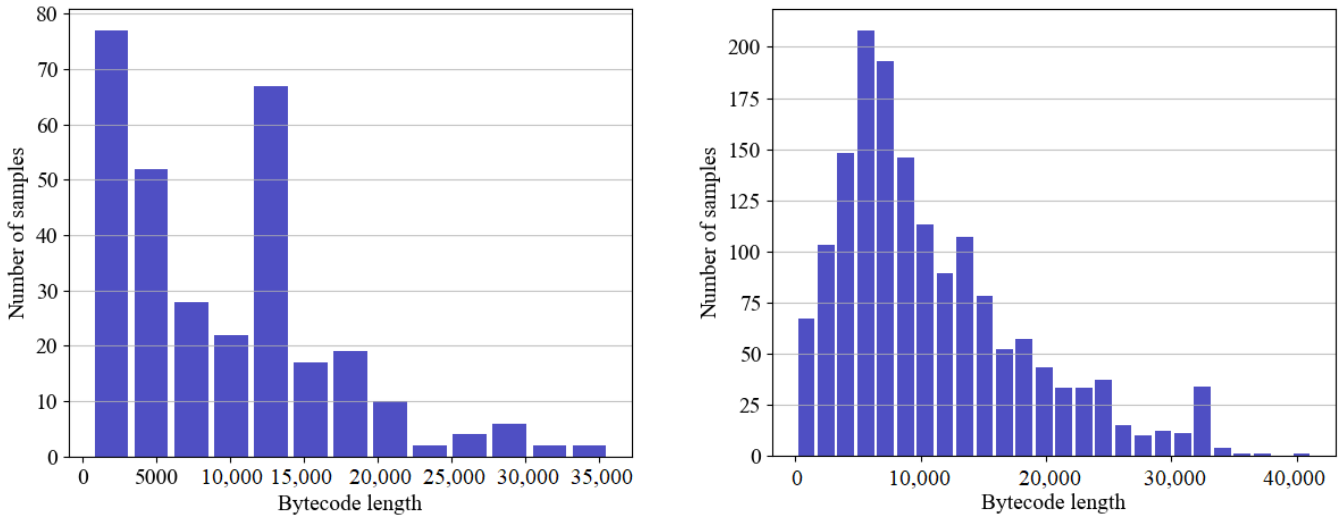
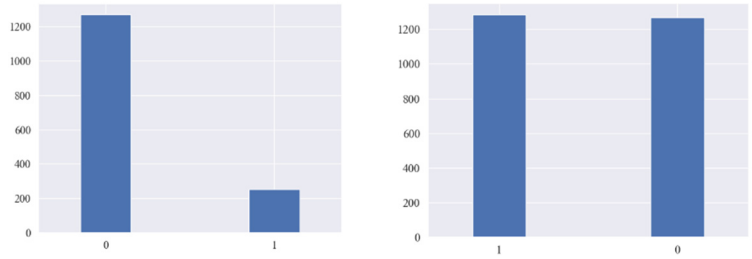



**Figure 2.** Distribution of contracts' bytecodes size in our dataset in (**a**) abnormal contracts and (**b**) normal contracts.

Data Resampling Procedure

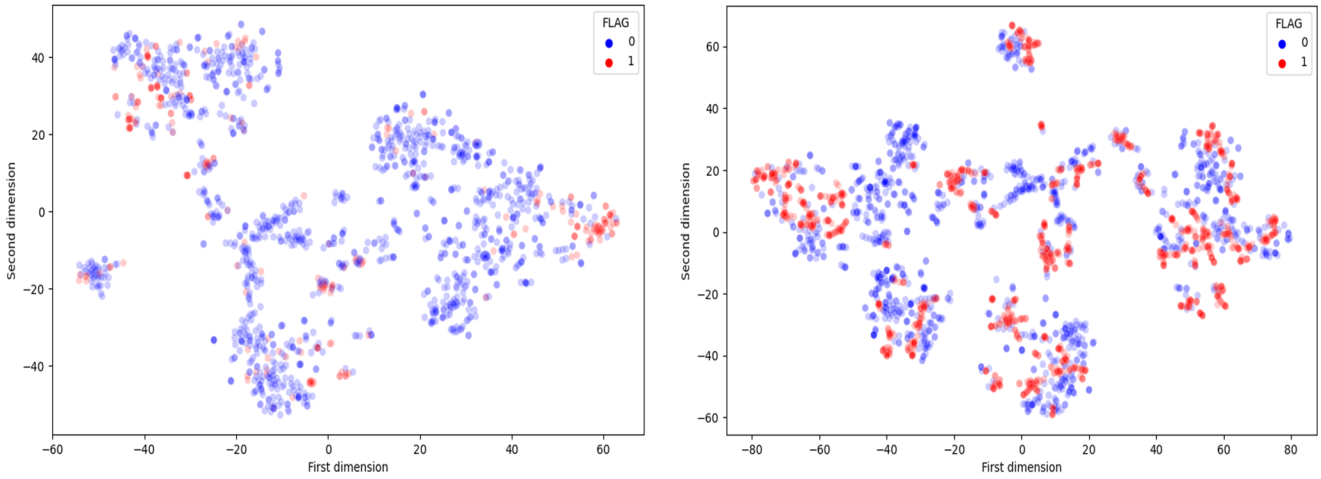
The defiance of working with imbalanced-class datasets is that most machine learning techniques will ignore and therefore perform incompletely in the rare class, although their performance in the rare class is the most important. One method to deal with imbalanced-class datasets is to oversample the rare class. The simplest method involves iterating tuples into the classes of interest class until there are an equal number of positive and negative tuples. Alternatively, new tuples can be made from existing tuples. In our training data, the ratio of abnormal contract accounts to normal contracts is 1:5. We employ Adaptive Synthetic Sampling (ADASYN) algorithm to generate more abnormal contract accounts to match the number of normal contract accounts to improve efficiency. The ADASYN algorithm uses the density distribution as a criterion to indicate the number of synthetic tuples that must be automatically created for each rare data tuple [37]. The purpose of adaptive synthetic sampling is to generate fewer synthetic examples in regions of the feature space where minority examples are low, and more or none in regions where minority examples are dense. In ADASYN, data samples of minority classes are adaptively generated according to their distributions, with a greater amount of synthetic data being

produced for minority class samples that are difficult to learn than those minority samples that are easier to learn. In this way, biased training will be prevented against a particular group in the data set. Figure 3 shows the distribution of normal and abnormal contracts before and after balancing our data. As part of the training stage, we used only ADASYN to analyze the training data, but the test data was not equally distributed or visible. The t-Distributed Stochastic Neighbor Embedding (t-SNE) is a nonlinear dimensionality reducing approach in which interrelated high dimensional data is mapped into low-dimensional data while preserving the significant structure of original high dimensional data [38]. Figure 4 illustrates the t-SNE transformation implemented to the original feature space concerning the smart contract class label (normal or abnormal).




(**a**) (**b**)

**Figure 3.** Distribution of normal (0) and abnormal (1) contracts before and after balancing our dataset. (**a**) Original training class labels. (**b**) Oversampled training class labels.




(**a**) (**b**)

**Figure 4.** Two-dimensional scatter plot following t-SNE transformation concerning contract label. Red and blue data points clarify the abnormal and normal contracts, respectively, in (**a**) original and (**b**) oversampled training data.

### *3.2. Feature Extraction and Transformation*

In Python, a custom code has been implemented that takes the account contract as an input and obtains information based on opcodes, transaction history, and source code for that account. To determine if a particular contract account is an abnormal contract or not, these extracted features are fed into a machine learning model. The extracted contract account-based features are categorized into three sets:

- Opcode n-grams;
- Transaction data;
- Source code characters.

#### 3.2.1. Opcode N-Gram Features

The Ethereum contract account is composed of a chain of hexadecimal representations of contracts on the Ethereum blockchain. By converting the bytecodes to opcodes, which are similar to natural language, the opcodes become readable to humans. As shown in Table 2, the opcodes comprise instructions mnemonics (i.e., ADD, SSTORE JUMP, LT, MOD, etc.) as well as their operands.

**Table 2.** Contract address, bytecode, opcode, and label.

| Address | Bytecode | Opcode | Label |
|---|---|---|---|
| 0xff7c3b7f4e426009 . . . | 60806040526040516 . . . | DUP1 PUSH1 0x40 MSTORE PUSH1 . . . | 1 |
| 0x8d790f3989b6d24 . . . | 60606040523415620 . . . | PUSH1 0x60 BLOCKHASH MSTORE . . . | 0 |
| 0xe029d93dbf50331 . . . | 61005857508251601 . . . | STOP PC JUMPI POP DUP3 MLOAD . . . | 0 |
| 0x7384c268c27c216 . . . | 010155b1561006257 . . . | ADD SSTORE JUMP LT MOD . . . | 1 |
| 0x8f4e3d448a318c1 . . . | 0160a060020a0316ff . . . | PUSH1 0xa0 PUSH1 0x02 EXP SUB . . . | 1 |

The use of operation codes has been successfully applied to various fundamental issues of contract accounts in previous studies [18,39]. Therefore, we extract the n-gram features from the opcode sequences of the contract accounts to detect abnormal schemes. The n-grams are used frequently in the field of natural language processing (NLP). However, it is also very common in malware detection tasks. In this section, opcodes for contract accounts are analyzed using n-grams without relying on any other interventions from experts to characterize them. N-grams are set between 2 and 3 bytes in length, making them language independent. However, they can be large for large documents, resulting in great computational complexity [40]. Figure 5 shows a comparison of some normal and abnormal opcodes based on average frequency values for each opcode in our data. The occurrence of each contract's opcode is calculated and saved in the database. Opcodes that frequently occur, such as PUSH, DUP, and SWAP, and those with a frequency less than zero, are filtered out. Fifty-nine different opcodes were extracted from 1904 contract accounts. The Ethereum yellow paper appendix [41] includes a complete list of EVM opcodes.

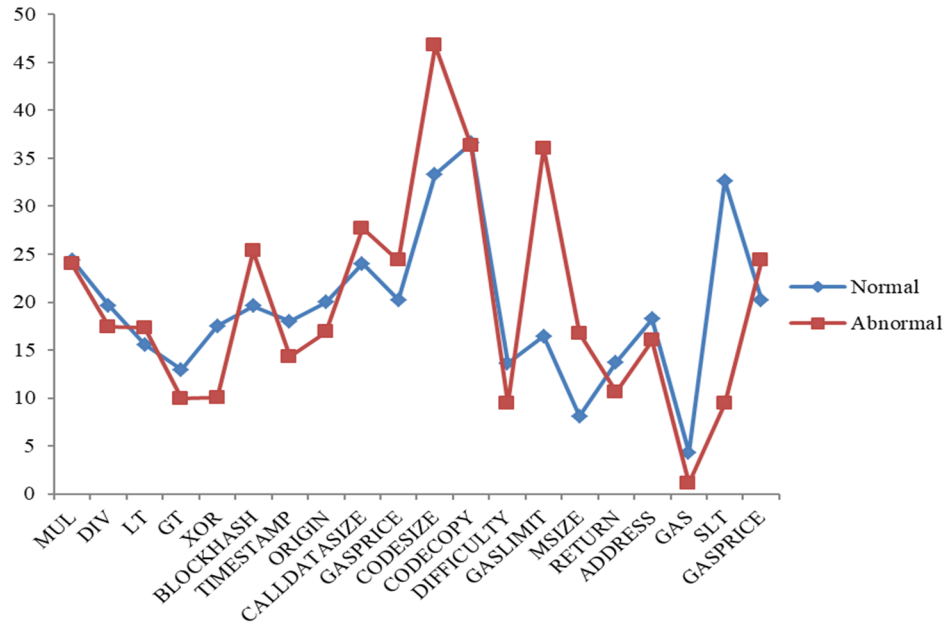



**Figure 5.** Distribution of some opcodes for both abnormal and normal contracts in our dataset.

#### 3.2.2. Transaction Features

Investors who join abnormal contracts receive a huge payout when they invest early, but later investors receive only a small payout. Many investors never get their money

back from abnormal contracts due to the fact that the sooner one invests, the bigger the reward. Behavior-based features use contract's intelligent interactions with its users, i.e., transactions. These features catch logical and relevant insights that reflect the behavior of contract accounts. Smart abnormal contracts are characterized by their intrinsic characteristics, which can be used to determine if they are smart abnormal schemes. Using the Ethereum blockchain data, we are able to obtain all transactions made by normal and abnormal contracts that reflect the main behavior of a contract. All transactions are stored in a JSON file format. We use the JSON library in Python to load the file, analyze it, and extract portions of information from stored transactions. For these transactions, we get the respective fields defined in Table 3 to perform feature extraction. The respective fields include information such as which account paid the amount of Ether to the respective contract account and when. This information is then used to observe the behavior of the contract account and convert them into features for classification. In our feature extraction stage, we created 57 explanatory variables concerning transaction data to analyze smart contract accounts. Explanatory variables are described in Table 4. A small subset of the features extracted was comparable to those gained in similar works [15,32,42]. Although all extracted transaction features are not necessary for classifier training, some of them do not participate in improving the efficiency of the classification model. Thus, we use the XGBboost feature importance algorithm as a feature-reducing method to decrease the dimensions of the feature vector component. We choose the top 32, with the highest score in information gain, as shown in Figure 6.

**Table 3.** Transaction fields of interest.

| Field of Interest | Description | Data Type |
|---|---|---|
| Timestamp | The date and time at which a transaction is mined | UNIX time |
| From | The sending address of the transaction | 20-byte value |
| Interacted with | The receiving address of the transaction | 20-byte value |
| Value | The amount of Ether to be transferred to the recipient with the transaction | Long Integer |
| Transaction Fee | The amount of Ether paid to the miner for processing the transaction, which is computed by multiplying the amount of gas used by the gas price | Long Integer |
| Gas Price | The amount of Ether (in a small unit called gwei) that must be paid to miners for processing transactions | Long Integer |
| Gas Limit | The maximum amount of gas provided for a transaction to happen | Long Integer |
| Gas Used | The exact units of gas that was used for the transaction | Long Integer |
| iserror | A value that determines whether a transaction is successful or not | Boolean |

**Table 4.** List of transaction features.

| S.N | Extracted Feature | Rank | Description |
|---|---|---|---|
| 1 | Min_Eth_Sent | 4 | The minimum Ether value sent |
| 2 | Avg_Eth_Sent | 23 | The average Ether value sent |
| 3 | Max_Eth_Received | 30 | The maximum Ether value received |
| 4 | Avg_Eth_Received | 11 | The average Ether value received |
| 5 | Txn_Fee_Received | 8 | The total of transaction fees spent in received transactions |
| 6 | Total_Txn_Fee | 26 | The total of transaction fee spent on all received and sent transactions |
| 7 | GasUsed_Received | 27 | The total of gas used value in received transactions |
| 8 | Failed_Txn_Received | 10 | The total number of failed received transactions |
| 9 | Failed_Txn_Sent | 3 | The total number of failed sent transactions |
| 10 | Total_Failed_Txn | 31 | Total number of failed transactions |
| 11 | Success_Txn_Received | 25 | The total number of successfully received transactions |
| 12 | Total_Success_Txn | 17 | The total number of successful transactions |
| 13 | Std_Eth_Received | 20 | The standard deviation of ether value received |
| 14 | Std_GasPrice_Received | 19 | The standard deviation of gas price value used in received transactions |
| 15 | Avg_Gas_Used_Received | 29 | The average gas value used in received transactions |
| 16 | GasPrice_Received | 16 | The total number of gas price values used in received transactions |

**Table 4.** *Cont.*

| S.N | Extracted Feature | Rank | Description |
|---|---|---|---|
| 17 | GasPrice_Sent | 6 | The total number of gas price values used in sent transactions |
| 18 | Gas_Sent | 5 | Total of gas value used in sent transactions |
| 19 | Gas_Received | 32 | Total of gas value used in received transactions |
| 20 | Avg_GasPrice_Received | 18 | The average gas price value used in received transactions |
| 21 | Avg_GasPrice_Sent | 2 | The average gas price value used in sent transactions |
| 22 | Txn_Received | 21 | The total number of received transactions |
| 23 | Total_Txn | 1 | The total number of transactions sent and received |
| 24 | Total_Eth_Received | 9 | The total ether value received |
| 25 | Unique_Txn_Received | 13 | The total number of transactions received from unique addresses |
| 26 | Unique_Address_Received | 24 | The total distinctive addresses from which contract received transactions |
| 27 | Time_Diff_Between_First_and_Last | 22 | The time difference between the first and last transaction in minutes |
| 28 | Txn_Fee_Contract_Create | 15 | The transaction fee paid in contract generation |
| 29 | GasUsed_Contract_Create | 7 | Gas spent during contract creation |
| 30 | GasPrice_Contract_Create | 12 | Gas price used in contract creation |
| 31 | Gini_Eth_Sent | 14 | Gini coefficient calculated over the Ether value amount of sent transactions |
| 32 | Gini_Eth_Received | 28 | Gini coefficient calculated over the Ether value amount of received transactions |

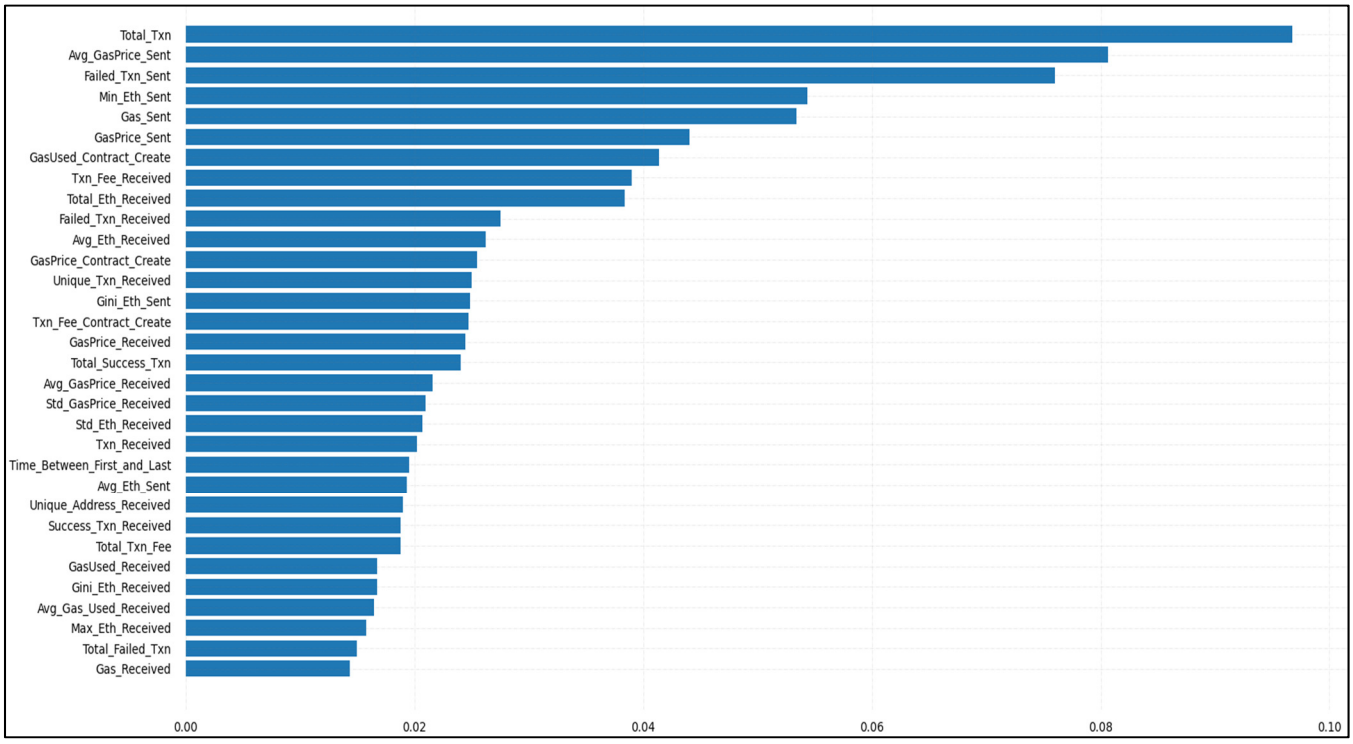



**Figure 6.** The ranking of the transaction features as determined by the XGBoost classifier.

Based on our dataset, Table 5 shows the frequency of the top 10 transaction-based features among abnormal and normal contract accounts. It is evident from the table that abnormal and normal contracts have very different statistics. As a first point of observation, normal contracts have larger standard deviations than abnormal contracts for most features. It implies that smart abnormal contracts may behave similarly, leading to lower standard deviations. It is apparent that abnormal contracts are a special type of smart contract account based on the significant differences in statistics of some features. In normal contracts, for example, the total number of transactions sent and received, the total number of failed received transactions, and the total number of failed sent transactions are greater

than those in abnormal contracts. Normal contract accounts use a greater number of gas price values and gas values in their sent transactions than abnormal contract accounts. Using the MinMaxscaler technique, we scaled each input feature in the 0–1 range separately before modeling in order to achieve maximum precision.

**Table 5.** Statistics of top 10 transaction-based feature frequency.

| Feature | Abnormal | | Normal | |
|---|---|---|---|---|
| | **Avg** | **Stdev** | **Avg** | **Stdev** |
| Total_Txn | 351.35 | 1126.15 | 2628.41 | 4714.27 |
| Avg_GasPrice_Sent | 998,606,557.39 | 4,718,588,870.74 | 6,857,603,992.97 | 35,865,147,124.55 |
| Failed_Txn_Sent | 3.13 | 54.07 | 44.75 | 382.46 |
| Min_Eth_Sent | 0.004 | 0.057 | 6.91 | 0.00025 |
| Gas_Sent | 2,574,176.26 | 40,909,530.65 | 89,496,313.95 | 388,481,804.54 |
| GasPrice_Sent | 381,970,128,912.006 | 6,252,354,182,317.71 | 15,696,757,925,289.8 | 109,755,731,881,321 |
| GasUsed_Contract_Create | 0.84 | 0.22 | 0.75 | 0.29 |
| Txn_Fee_Received | 2.87 | 1.04 | 2.54 | 1.83 |
| Total_Eth_Received | 3025.48 | 36,738.49 | 44,868.01 | 541,176.68 |
| Failed_Txn_Received | 21.72 | 97.88 | 179.71 | 667.36 |

#### 3.2.3. Source Code Features

TF-IDF stands for Term Frequency-Inverse Document Frequency. A term's significance is directly proportional to how many times it appears in a set of documents. The weight of the TF IDF is a statistical measure that represents the significance of a term in a set of documents. The weight of a TF-IDF is calculated by multiplying two different scales:

1. Term frequency (TF): There are several ways to calculate this frequency, the simplest being the number of instances where a term appears in a document. Then, there are ways to adapt the frequency, depending on the length of the document or the raw frequency of the most common term in the document.
2. Inverse document frequency (IDF): This means the importance of a term in the entire corpus. The closer it is to 0, the more common a term is. This metric can be computed by taking the total number of documents, dividing it by the number of documents containing a term, and computing the logarithm.

The TF-IDF score for the term t in document $d$ from the collection of documents $D$ is calculated as follows [40]:

$$TF - IDF(t, d, D) = TF(t, d) * IDF(t, D) \quad (1)$$

where

$$TF(t, d) = log(1 + freq(t, d)) \quad (2)$$

$$IDF(t, D) = log\left(\frac{N}{count(d \in D : t \in d)}\right) \quad (3)$$

and $N$ is documents count in $D$.

There are different levels of input tokens that can be generated using the TF-IDF technique (words, letters, and n-grams). One limitation is that the TF-IDF technique may fail if the extracted keywords are irrelevant, misspelled, or skipped. The source code for the contract account is written in high-level language programming (i.e., solidity [5]). Since solidity source code in our method is extracted from the selected contract account using JSON parser, the TF-IDF character level technique is employed with max features as 10,000. To get appropriate text information of the contract account source code, the extra parts are removed by regular expressions, including punctuation symbols, numbers, spaces, newline, etc., as shown in Figure 7. Finally, the TF-IDF object is used in order to transform the text of the smart contract source code.

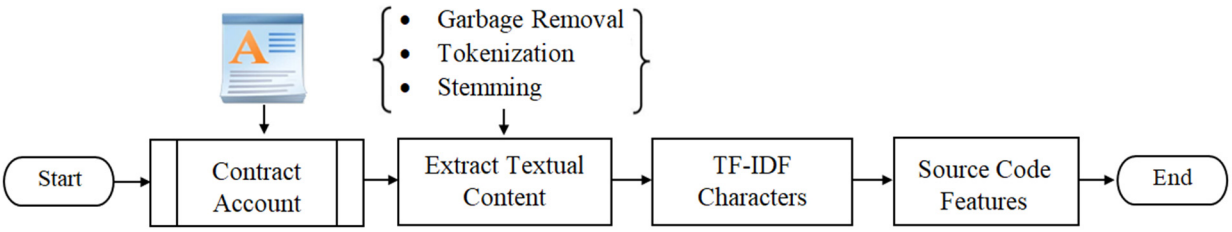



**Figure 7.** The process of generating source code features.

### *3.3. Ensemble Learning Model*

In order to construct a labeled dataset, we apply feature vectorization to extract the features for each smart contract. We combine opcode n-grams with source code characters and transaction features to generate a feature vector needed for training the proposed method through an ensemble classifier of Extra-Trees and grading boosting machine algorithms. Our goal in combining bagging with boosted models is to create a strong classifier that balances the weaknesses of individual classifiers in a given data set. There are different approaches to create an ensemble of classifiers [43]. A soft voting ensemble learning model is applied using the scikit-learn package (http://scikit-learn.org, accessed on 1 July 2022), and features are extracted using Python. Extremely Randomized Trees (Extra-Trees Classifier, ETC) is a type of bagging learning model which builds randomized trees whose structures are independent of the output values of the learning sample. Bagging, also known as (bootstrap aggregation), is an ensemble learning method commonly used to reduce variance within a noisy data set. In bagging, random subsets $D$ of data in the training set $N$ is chosen with replacement. In this case, the average or majority of votes is used for the prediction of all trees and is therefore more powerful than a single decision tree. A base learner $c_j(x)$ is trained based on the random subsets $D_{subset_j}$, and the prediction of the ETC classifier is voted as follows [44]:

$$etc(x) = argmin_{i \in \{0,1\}} \sum_{j=1}^{m} \varnothing\left(c_j(x) = i\right) \tag{4}$$

where $\varnothing$ is the characteristic function to finding the best parameters that best fit the training data $x_i$ and labels, $i$ is the set of unique class labels, and $j$ is the number of a base learner.

Meanwhile, another classifier, Gradient Boosting Machine (GBM), is a machine learning classifier for tree boosting used in classification and regression tasks. It provides a strong predictive model in the form of a group of weak learning models, which are usually decision trees [45]. Suppose there are $N$ contract accounts in the dataset $\{(x_i, y_i) | \, i = 1, 2, \ldots, N\}$, where $x_i \in R^d$ are the extracted features associated with the $i$-th contract account $y_i \in \{0, 1\}$ are the class label, such that $y_i = 1$ if and only if the contract account is a labeled abnormal contract. The GBM introduces additive modeling, which means that first builds a model, find its residual, and build another model $G_k(x)$ on the residual. Thus, the generalized equation for a boosting algorithm at iteration $k$ can be calculated by following the mathematically model [45]:

$$f_k(x) = f_{k-1}(x) + \alpha G_k(x) \tag{5}$$

The term gradient boost comes from incorporating gradient descent into boosting. A method based on gradient descent is used to determine the alpha $\alpha$ or step size that minimizes the average value of the loss function on the training set. For alpha computation, at step $k$, first pseudo-residual $(r_{ik})$ or negative gradient is calculated, and the new model $G_k(x)$ (or base learner, e.g., tree) is generated on the training set $\{x_i, r_{ik}\}$. The pseudo-residual is calculated by [45]:

$$r_{ik} = -\left[\frac{\partial Loss(y_i, f_{k-1}(x_i))}{\partial f_{k-1}(x_i)}\right], \; i = 1, \ldots, N \tag{6}$$

Now, compute $\alpha$ so that the Loss function is minimized [45].

$$\alpha = argmin_{\alpha} \sum_{i=1}^{N} Loss(y_i, f_{k-1}(x_i) + \alpha G_k(x_i)) \quad (7)$$

The rate calculation in the GBM classifier requires two steps: (i) calculate pseudo-residual and (ii) calculate step size $\alpha$. So, we can plug in those values of $\alpha$ and $G_k(x)$ to update the model and get $f_k(x)$. Finally, we are ready to get new predictions by adding our base GBM model with the new tree we made on residuals $r_{ik}$. The contract accounts are classified into two possible categories: abnormal and normal using a binary classifier. When the contract account is uploaded to the Ethereum Blockchain, the trained classifier determines the prediction of a particular account contract from the created feature vector.

Ensemble methods can be used to increase overall accuracy by learning and integrating several base classification models. The ensemble method can be generated from different classification algorithms, for example, Extra-Trees, GBM, etc. Alternatively, the same base classification algorithm can also be used, imposing different subsets of the training data set. Given a new data set to classify, each classifier votes to name that set's class. The ensemble method collects votes to return the final class prediction $P$. Figure 8 illustrates the concept of our ensemble method using the weighted soft voting strategy.

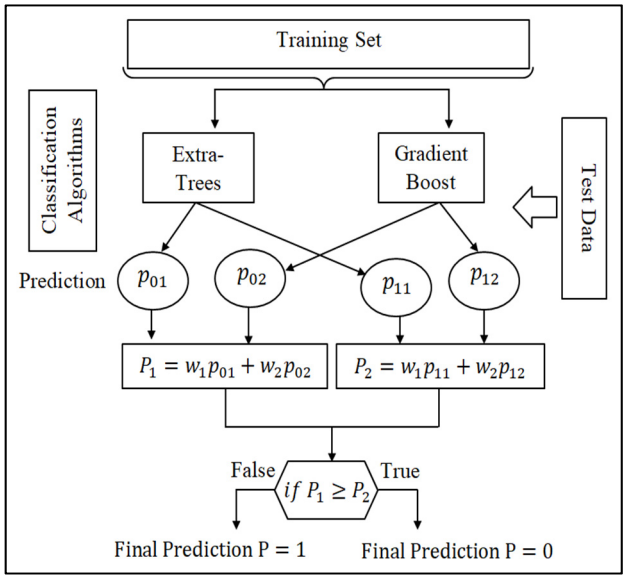



**Figure 8.** Ensemble classification model based on weighted soft voting for abnormal contracts detection.

The ensemble method includes a sequential four-step process: (i) split the dataset into a training and test set; in the training process, the training data is passed after balanced to the ensemble model; the dataset was randomly divided into 80:20 ratios for training and testing, respectively; (ii) training base classifiers (Extra-Trees, GBM) on the training set; (iii) use the test set and its predictions to create an ensemble-classifier; and (iv) make final predictions using this classifier. During the testing process, the ensemble model determines whether or not the unknown contract account is abnormal. The hybrid model, after training, outputs predictive values between zero and one for each record in the examination data.

The ensemble vote classifier is a set of classifiers whose individual decisions are integrated in some way (usually by hard or soft voting) to identify new examples. In hard voting, data is classified based on class labels and the weights associated with each classifier. On the other hand, a soft voting classifier classifies the data based on the probabilities and weights associated with each classifier. We employ Extra-Trees and GBM algorithms to design an ensemble meta-classifier with a soft voting strategy. Meanwhile, weights are the learnable parameters of some machine learning models, including ensemble models, and are commonly referred to as $w$. Since we have a binary classification problem with class labels $i \in \{0, 1\}$ and an ensemble of two base classifiers (ETC and GBM), let us assume that the two classifiers return the following class membership predications for a particular

sample $x$: $p_{01}$ and $p_{11}$ for normal and abnormal contracts of ETC classifier, $p_{02}$ and $p_{12}$ predictions for normal and abnormal contracts of GBM classifier. We can then calculate the individual probability of first-class label $P_1$ using a weighted soft vote as follows:

$$P_1 = w_1 p_{01} + w_2 p_{02} \tag{8}$$

where $p_{01}$ and $p_{02}$ refer to the predicted probabilities of the ETC and GBM classifiers, respectively, for class label 0. It has been trained on our dataset and applied to test data when forecasting the likelihood of a particular contract, such as whether the contract is normal or not. Similarly, we can write the weighted soft vote of the predicted second-class label $P_2$ as follows:

$$P_2 = w_1 p_{11} + w_2 p_{12} \tag{9}$$

where $p_{11}$ and $p_{12}$ refer to the predicted probabilities of the ETC and GBM classifiers for class label 1. While $w_1$ and $w_2$ are the weights associated with the two classifiers of Equations (8) and (9), weights are usually randomly generated as default values between 0 and 1 to speed up the training process of machine learning algorithms [43]. When $P_1 \geq P_2$, the contract account to be identified is normal; otherwise, it is abnormal contract.

$$P = \begin{cases} 0, & if\ P_1 \geq P_2 \\ 1, & Otherwise \end{cases} \tag{10}$$

where $P_1$ and $P_2$ are the weighted soft vote of the predicted first- and second-class labels according to Equations (8) and (9).

### *3.4. Abnormal Contracts Detection*

The prediction stage includes building a robust model using the ensemble classifier, bagging, and boosted models. The ensemble method includes using many base learners to enhance the efficiency of any single one of them individually. This method can be represented as techniques that use a group of weak learners together to generate a robust, aggregated one. Here, an ensemble model is employed on a combined feature set to build a robust classifier for abnormal contract account detection. Using a binary classifier, contract accounts are categorized into two possible values: abnormal contracts and normal contracts. The ensemble classifier is trained using the feature vectors collected from each contract account in the training set. Once trained, the classifier establishes whether the contract account is abnormal. The transaction features component generates 32-dimensional feature vector as:

$$F_B = \langle b_1, b_2,\ b_3,\ \ldots,\ b_{32} \rangle \tag{11}$$

and the opcode n-grams features component generates a D-dimensional feature vector as:

$$F_N = \langle n_1, n_2,\ n_3,\ \ldots,\ n_D \rangle \tag{12}$$

whereas source code characters component generates K-dimensional feature vector as:

$$F_T = \langle t_1, t_2,\ t_3,\ \ldots,\ t_K \rangle \tag{13}$$

where $D$ and $K$ are the sizes of dictionaries calculated from the textual content corpus; the source code vector contains $K$ features, the opcodes n-gram contains $D$ features, and a transaction feature contains 32 features. From our experiments, we note that $D$ and $K$ each contain 17,047 and 10,303 elements, respectively. The above three feature vectors are integrated to create the final feature vector, fed as an input parameter to the ensemble learning model to identify the probability of anomaly $P$ of contract account $S$ as shown in Algorithm 1.

$$F_V = F_B + F_N + F_T = \langle b_1, b_2, b_3, \ldots,\ b_{32}, n_1, n_2, n_3, \ldots n_D,\ t_1, t_2, t_3, \ldots, t_K \rangle \tag{14}$$

**Algorithm 1:** Abnormal Contract Detection Algorithm

**Input:** The contract account addresses $S$, $S = \{s_1, s_2, \ldots, s_n\}$
**Output:** The probability of anomaly $P \in \{0, 1\}$, 0—normal, 1—abnormal

1. Initialize the contract addresses set $N = |S|$
2. H = $\varnothing$ *//represents the validation feature set*
3. **For each** contract $(s_n \in N)$ **do**
4. Initialize feature vectors $F_B = \varnothing,\ F_N = \varnothing,\ F_T = \varnothing,\ F_V = \varnothing$
5. Compute transaction features component $F_B$ using Equation (11)
6. Generate opcodes n-grams vector $F_N$ using Equation (12)
7. Create source code chars features $F_T$ using Equation (13)
8. Concatenate $F_B + F_N + F_T$ to form final feature vector $F_V$ using Equation (14)
9. Append the final feature vector $F_V$ to the validation feature set $H = H + F_V$
10. **End for**
11. Fetch the predictions of first-class label $P_1$ for validation feature set $H$ using Equation (8)
12. Compute the predictions of second class $P_2$ for validation set according to Equation (9)
13. Predict the probabilities of anomaly $P$ of validation set based on predictions $P_1$ and $P_2$ using Equation (10)
14. **Return:** $P$

## 4. Implementation and Evaluation

A desktop machine with a Core™ i7 processor with a clock speed of 3.4 GHz and 16 GB of RAM is used to implement the proposed detection method. The proposed method is implemented using Python, since it provides substantial support for its libraries and is relatively simple to compile. The JSON library is used to parse the transaction data and source code of the selected contract account. The proposed method takes suspicious contract account as input and outputs the type of the contract account scheme as normal or abnormal. The prior extracted three features (Opcode n-gram features, Transaction features, and Source code characters features) are integrated into hybrid features.

We used a variety of performance statistics to measure the efficiency of our proposed method, including true positives, true negatives, false positives, false negatives, accuracy, precision, and F1 scores. Each is defined below.

$$TPR = \frac{\#\,of\ abnormal\ contracts\ classified\ as\ abnormal}{Total\ \#\,of\ abnormal\ contracts} \tag{15}$$

$$TNR = \frac{\#\,of\ normal\ contracts\ classified\ as\ normal}{Total\ \#\,of\ normal\ contracts} \tag{16}$$

$$FPR = \frac{\#\,of\ normal\ contracts\ classified\ as\ abnormal}{Total\ \#\,of\ normal\ contracts} \tag{17}$$

$$FNR = \frac{\#\,of\ abnormal\ contracts\ classified\ as\ normal}{Total\ \#\,of\ abnormal\ contracts} \tag{18}$$

$$Accuracy = \frac{\#\,of\ correctly\ classified\ abnormal,\ normal\ contracts}{Total\ \#\,of\ contract\ accounts} \tag{19}$$

$$Precision = \frac{\#\,of\ abnormal\ contracts\ classified\ as\ abnormal}{Total\ \#\,of\ contract\ accounts\ classified\ as\ abnormal} \tag{20}$$

$$\mathrm{F1-score} = 2 * \frac{\mathrm{Precision} * \mathrm{TPR}}{\mathrm{Precision} + \mathrm{TPR}} \tag{21}$$

### *4.1. Evaluation of Features*

We have evaluated the performance of our proposed features (opcode n-grams, source code characters, and transaction features). The results shown in Table 6 demonstrate that the proposed individual feature sets are useful in abnormal contract detection. However, one type of feature is not appropriate to detect all types of abnormal contract accounts and

does not result in high accuracy. Therefore, we have combined all features (hybrid features) to get more comprehensive features and better accuracy of the proposed method.

**Table 6.** Evaluation of the proposed feature set using the ensemble classifier.

| Features | Precision | Sensitivity | F1-Score | Accuracy |
|---|---|---|---|---|
| Opcode n-grams | 95.61 | 66.66 | 78.55 | 81.80 |
| Transaction features | 80.64 | 66.03 | 72.61 | 75.09 |
| Source code characters | 92.91 | 72.22 | 81.27 | 83.35 |
| Hybrid features | **97.44** | **81.48** | **88.74** | **89.67** |

In Figure 9, we compare the four feature sets in terms of accuracy, i.e., TNR, FPR, FNR, and TPR. From Table 7, we construct a confusion matrix for the proposed feature sets by using the ensemble model to detect 381 contracts in our test data. The hybrid feature set accurately identifies 320 normal contracts and 44 abnormal contracts and includes 10 FNs and 7 FPs, which is much better than other features. Opcodes n-grams and transaction features have 10 and 51 FPs, but FNs reach 18 and 19, respectively. The number of FNs and FPs for the source code features is 15 and 18, respectively, which is more impressive than n-gram opcodes and transaction features. A receiver operating characteristic (ROC) curve represents TPR on the *Y* axis and FPR on the *X* axis (Figure 10). The large area under the curve (AUC) indicates that the classifier is performing perfectly when applied to the data.

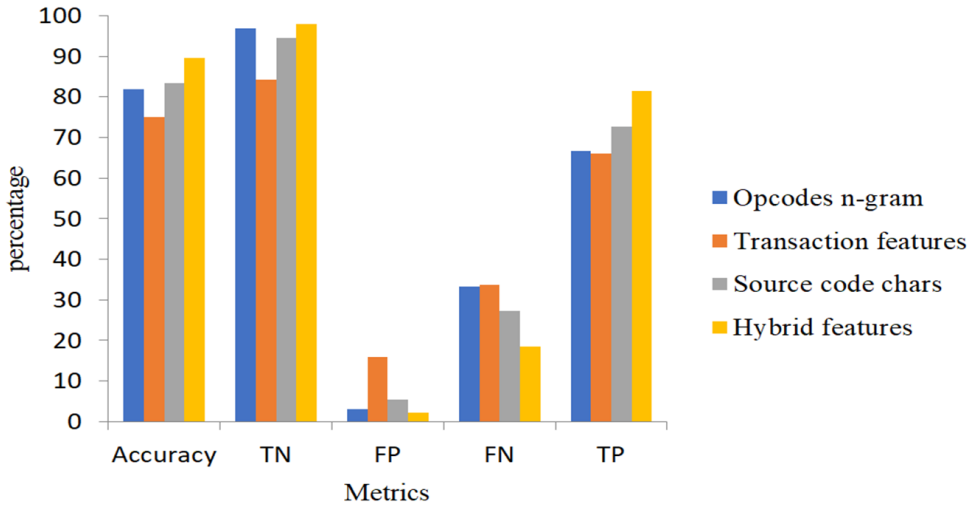



**Figure 9.** Performance of the proposed feature sets.

**Table 7.** Confusion matrix of the proposed method on our dataset.

| Confusion Matrix | | | Predicted | |
|---|---|---|---|---|
| | | | **P** | **N** |
| Opcode n-gram features | Actual | P | 36 | 18 |
| | | N | 10 | 317 |
| Transaction features | Actual | P | 35 | 19 |
| | | N | 51 | 276 |
| Source code features | Actual | P | 39 | 15 |
| | | N | 18 | 309 |
| Hybrid features | Actual | P | 44 | 10 |
| | | N | 7 | 320 |

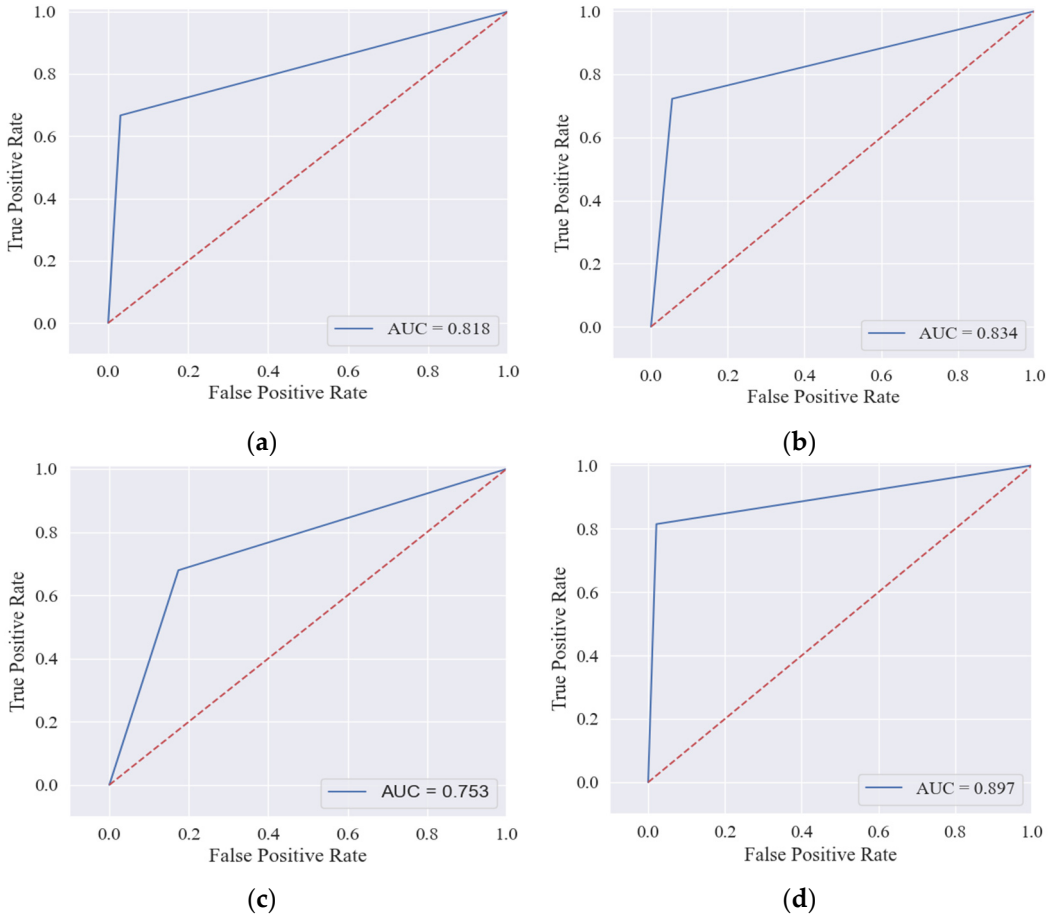



**Figure 10.** Receiver operating characteristic of the proposed feature sets in (**a**) opcodes n-gram, (**b**) source code, (**c**) transactions, and (**d**) hybrid features.

### *4.2. Evaluation of Ensemble Model*

In this section, we have applied various bagging algorithms (i.e., random forest, extra trees, bagging tree, etc.) and boosting models (i.e., AdaBoost, gradient boosting machine, XGBoost, light gradient boosting, etc.) to train our proposed hybrid features. This experiment aims to create a robust ensemble classifier by combining bagging and boosted techniques. The experimental results are shown in Table 8. Random forest, extra-trees, and bagging-tree are bagging algorithms and outperformed the other classifiers with adequate accuracy. On the other hand, AdaBoost, XGBoost, gradient boosting, and light gradient boost are a type of boosting tree algorithms that convert weak learners into strong learners; thus, they have high precision, accuracy, sensitivity, and F1-score. We noticed that Extra-Trees and gradient boosting machine performed well from bagging and boosting models, respectively, so we chose these two algorithms to generate an ensemble classifier with a soft voting approach. This classifier outperformed other methods in terms of accuracy, precision, F1-score, and sensitivity, and achieved a high-performance rate. The results of various classifiers of the hybrid features set are also shown in Figure 11.

To illustrate the learning capability of the proposed method, as shown in Figure 12, we include both classification errors and log losses in our training and test data set with respect to how many iterations the ensemble model executes for each epoch. Log loss, short for logarithmic loss, refers to the price paid for incorrect predictions in classification. According to the figure, the learning model reaches convergence after approximately 30 iterations.

**Table 8.** Test results of the classification algorithms with respect to hybrid features.

| Algorithm | Precision | Sensitivity | F1-Score | Accuracy |
|---|---|---|---|---|
| DT | 93.28 | 72.22 | 81.41 | 83.51 |
| RF | 98.37 | 74.07 | 84.51 | 86.42 |
| BT | 97.58 | 74.07 | 84.21 | 86.11 |
| XGB | 97.87 | 70.37 | 81.87 | 84.42 |
| AB | 96.25 | 62.96 | 76.12 | 80.25 |
| LGBM | 96.80 | 74.07 | 83.92 | 85.82 |
| ETC | 97.32 | 77.77 | 86.45 | 87.81 |
| GBM | 95.38 | 75.92 | 84.55 | 86.12 |
| Ensemble model | **97.44** | **81.48** | **88.74** | **89.67** |

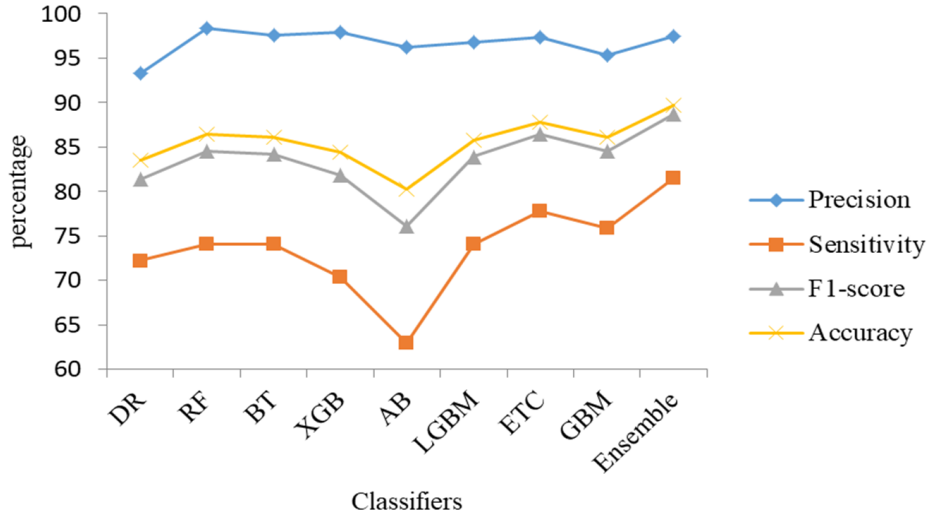



**Figure 11.** Test results of various classifiers with respect to hybrid features.

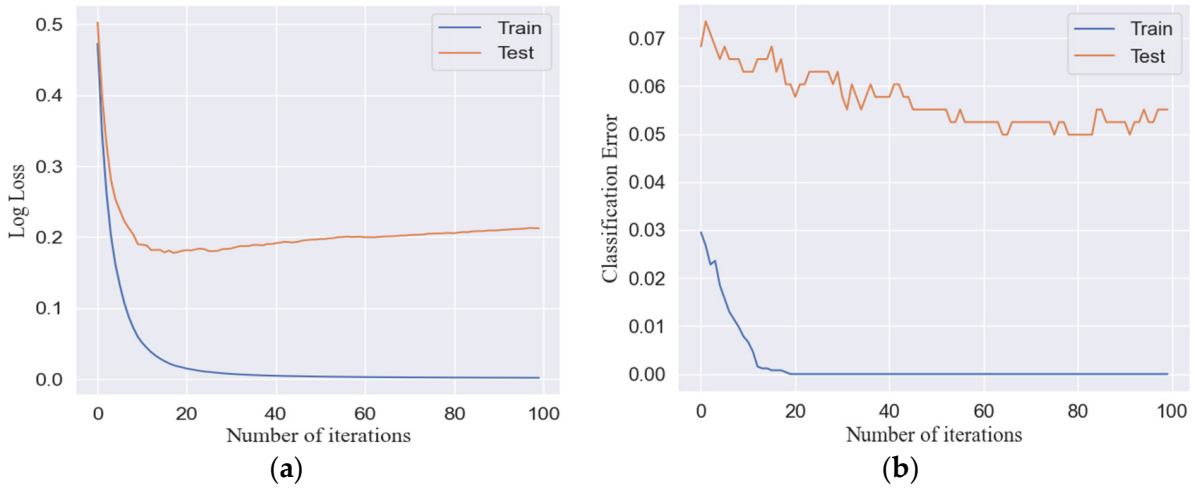



**Figure 12.** Learning curve for logarithmic loss and classification error on our data with respect to the ensemble classifier in (**a**) log loss and (**b**) classification error.

### *4.3. Comparison with Existing Works*

A comparison between the proposed method and other competing methods was conducted. We applied Chen et al. [14] and Chen et al. [15] methods on our dataset D1 to evaluate the performance of our proposed method. The main purpose of choosing these methods is due to the likewise in the features extracted from both transaction history and opcodes of contract accounts. Furthermore, we evaluated our method against the data (D2)

presented by Chen et al. [15] in regard to the three evaluation metrics used in the paper. The results of this comparison are presented in Table 9. As a result of our method's superior performance over other methods, our method is more effective at detecting abnormal contracts.

**Table 9.** Comparison of our method with existing methods.

| Metrics (%) | Chen et al. [15] (D2) | Our Method (D2) | Chen et al. [14] (D1) | Chen et al. [15] (D1) | Our Method (D1) |
|---|---|---|---|---|---|
| Precision | 0.94 | 0.99 | 0.97 | 0.95 | **0.97** |
| Sensitivity | 0.73 | 0.82 | 0.64 | 0.68 | **0.81** |
| F1-score | 0.82 | 0.90 | 0.77 | 0.79 | **0.88** |

As part of the analysis, we also attempted to determine the entire training time and testing time for the proposed method in light of a hybrid feature set for detecting abnormal contracts based on datasets D1 and D2. The results are shown in Table 10.

**Table 10.** Results of the proposed method on datasets D1 and D2.

| Evaluation Metrics | Dataset D1 | Dataset D2 |
|---|---|---|
| TPR (%) | 81.48 | 82.14 |
| TNR (%) | 97.85 | 99.81 |
| FNR (%) | 18.51 | 17. 85 |
| FPR (%) | 2.14 | 0.18 |
| Precision (%) | 97.44 | 99.77 |
| F1-score (%) | 88.74 | 90.10 |
| Accuracy (%) | 89.67 | 90.97 |
| Training Time (s) | 455 | 1056 |
| Test Time (s) | 0.06 | 0.11 |

### *4.4. Complexity of the Proposed Method*

In the proposed method, the computational complexity depends on how the proposed features are extracted and computed. The opcode n-grams features and source code features are extracted using TF-IDF algorithm. The TF-IDF is based on the frequency of each term and indexing a document of $b$ tokens, which required linear time $O(b)$ and space complexity. The transaction-based features are extracted using python and SQL queries. SQL queries are executed based on the square of the database size and the manner in which the query will be processed by the SQL engine. Therefore, many of the transaction features in our method (i.e., Total_Txn, Avg_Eth_Sent, Total_Txn_Fee, Failed_Txn_Sent, Gas_Sent, Success_Txn_Received, etc.) require logarithmic time complexity $O(log(n))$, while others (i.e., Unique_Txn_Received, Unique_Address_Received, Avg_Time_Diff, etc.) require linearithmic time complexity $O(n\ log(n))$. The learning model of our method is an ensemble of extra-trees and gradient boost machine based on weighted soft voting. In case the training data contains $n$ points with $d$ dimensions, the extra-trees algorithm has the following train time complexity $O(n * log(n) * d * m)$, where $m$ is the number of decision trees in the tree. The space complexity consists of $O(p * m)$, where $p$ represents the number of nodes within the tree, and the run time complexity consists of $O(k * m)$, where $k$ represents the depth of the tree. Accordingly, gradient boost algorithms have a train time complexity of $O(n * log(n) * d * m)$, a space complexity of $O(p * m + gamma)$ as it is multiplied with $m$ models, and a runtime complexity of $O(k * m)$.

## 5. Conclusions

With the rapid growth of online finance, abnormal and fraudulent contract accounts are also growing. This results in substantial economic losses for users of blockchain technology, i.e., Ethereum. Therefore, an effective solution is demanded to detect fraud-based abnormal contracts on Ethereum to avoid such malware and maintain the data-driven security of

the blockchain system. The abnormal contract accounts look similar to normal contracts, and the challenge is how to distinguish them. One of the biggest obstacles to detecting abnormal contract accounts on Ethereum is the severe data imbalance. This issue will lead to overfitting and poor generalization of the detection method in the absence of over-sampling of training data.

This paper uses data mining techniques to provide a robust method for abnormal contract detection on the Ethereum network. First, we collected abnormal and normal contracts data from Ethereum and solved the problem of imbalanced data by performing adaptive synthetic sampling. Next, we defined three types of features set based on the operation code n-grams, transaction data, and TF-IDF source code characters of contract accounts and combined them to get more comprehensive features. Finally, we designed an ensemble classification model combining Extra-Trees and gradient boosting classifiers to improve the classification accuracy of abnormal contracts compared to other methods.

There are abnormal scheme contract accounts that do not follow the patterns of existing abnormal contracts. This may also lead to misclassifying such abnormal contract accounts as normal contracts. For future work, we plan to build a neural network model against the latest abnormal contract accounts based on contract account bytecodes. Furthermore, we would like to investigate the applicability of our method to permissioned blockchains.

**Author Contributions:** Data curation, A.A. and Q.J.; funding acquisition, Q.J. and Q.Q.; investigation, Q.J. and Q.Q.; methodology, A.A., Q.J. and A.R.; project administration, Q.J.; software, A.A.; supervision, Q.J.; validation, Q.Q. and A.R.; writing—original draft, A.A.; writing—review & editing, A.R. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research work is supported by the National Key Research and Development Program of China Grant No. 2021YFF1200100 and 2021YFF1200104.

**Data Availability Statement:** The dataset generated during the current study are available inthe Google Drive and GitHub repositories: https://drive.google.com/u/1/uc?export=download&confirm=hPyP&id=1izK9Sm5yfwq3Ck9f71SUXAL41GVKQFBS and https://github.com/abdul-rasool/Abnormal-Contracts-Detection, (accessed on 14 September 2020).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Kaspars, Z.; Renate, S.A. Blockchain Use Cases and Their Feasibility. *Appl. Comput. Syst.* **2018**, *23*, 12–20.
2. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 1 July 2022).
3. Hasan, A.S.M.T.; Sabah, S.; Haque, R.U.; Daria, A.; Rasool, A.; Jiang, Q. Towards Convergence of IoT and Blockchain for Secure Supply Chain Transaction. *Symmetry* **2022**, *14*, 64. [CrossRef]
4. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, X.; Wang, H. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2018**, *14*, 352. [CrossRef]
5. Solidity. Solidity Documentation. 2019. Available online: https://solidity.readthedocs.io/en/v0.5.11/index.html (accessed on 1 July 2022).
6. Muzammal, M.; Qu, Q.; Nasrulin, B. Renovating blockchain with distributed databases. An open source system. *Future Gener. Comput. Syst.* **2019**, *90*, 105–117. [CrossRef]
7. Hu, T.; Liu, X.; Chen, T.; Zhang, X.; Huang, X.; Niu, W.; Lu, J.; Zhou, K.; Liu, Y. Transaction-based classification and detection approach for Ethereum smart contract. *Inf. Process. Manag.* **2021**, *58*, 102462. [CrossRef]
8. Ethereum (ETH) Market Cap. 2016. Available online: https://coinmarketcap.com/currencies/ethereum/ (accessed on 20 September 2021).
9. Higgins, S. SEC Seizes Assets from Alleged Altcoin Pyramid Scheme. 2015. Available online: https://www.coindesk.com/markets/2015/10/01/sec-seizes-assets-from-alleged-altcoin-pyramid-scheme/ (accessed on 1 July 2022).
10. Bartoletti, M.; Carta, S.; Cimoli, T.; Saia, R. Dissecting Ponzi schemes on ethereum: Identification, analysis, and impact. *Future Gener. Comput. Syst.* **2020**, *102*, 259–277. [CrossRef]
11. Morris, D. The Rise of Cryptocurrency Ponzi Schemes. 2017. Available online: https://www.theatlantic.com/technology/archive/2017/05/cryptocurrency-ponzi-schemes/5286 (accessed on 1 July 2022).

12. Zhou, Y.; Kumar, D.; Bakshi, S.; Mason, J.; Miller, A.; Bailey, M. Erays: Reverse engineering ethereum's opaque smart contracts. In Proceedings of the 27th USENIX Security Symposium (USENIX Security'18), USENIX Association, Baltimore, MD, USA, 15–18 August 2018; pp. 1371–1385. Available online: https://www.usenix.org/conference/usenixsecurity18/presentation/zhou (accessed on 1 July 2022).
13. Vasek, M.; Moore, T. Analyzing the Bitcoin Ponzi Scheme Ecosystem. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany; pp. 101–112.
14. Chen, W.; Zheng, Z.; Cui, J.; Ngai, E.; Zheng, P.; Zhou, Y. Detecting Ponzi Schemes on Ethereum: Towards Healthier Blockchain Technology. In *World Wide Web Conference*; International World Wide Web Conferences Steering Committee: Geneva, Switzerland, 2018; pp. 1409–1418.
15. Chen, W.; Zheng, Z.; Ngai, E.C.; Zheng, P.; Zhou, Y. Exploiting blockchain data to detect smart Ponzi schemes on Ethereum. *IEEE Access* **2019**, *7*, 37575–37586. [CrossRef]
16. Bartoletti, M.; Pes, B.; Serusi, S. Data Mining for Detecting Bitcoin Ponzi Schemes. 2018. Available online: http://arxiv.org/abs/1803.00646 (accessed on 1 July 2022).
17. Aljofey, A.; Jiang, Q.; Qu, Q. A Supervised Learning Model for Detecting Ponzi Contracts in Ethereum Blockchain. In Proceedings of the 3rd International Conference on Big Data and Security, ICBDS 2021, Shenzhen, China, 26–28 November 2021; pp. 1–16, *in press*. [CrossRef]
18. He, N.; Wu, L.; Wang, H.; Guo, Y.; Jiang, X. Characterizing code clones in the ethereum smart contract ecosystem. *arXiv* **2019**, arXiv:1905.00272. [CrossRef]
19. Wu, J.; Yuan, Q.; Lin, D.; You, W.; Chen, W.; Chen, C.; Zheng, Z. Who are the phishers? Phishing scam detection on ethereum via network embedding. *arXiv* **2019**, arXiv:1911.09259. [CrossRef]
20. Lin, L.; Wei-Tek, T.; Zakirul, A.B.M.; Hao, P.; Mingsheng, L. Blockchain-enabled fraud discovery through abnormal smart contract detection on Ethereum. *Future Gener. Comput. Syst.* **2022**, *128*, 158–1669. [CrossRef]
21. Aljofey, A.; Jiang, Q.; Rasool, A.; Chen, H.; Liu, W.; Qu, Q.; Wang, Y. An effective detection approach for phishing websites using URL and HTML features. *Sci. Rep.* **2022**, *12*, 8842. [CrossRef]
22. Bian, S.; Deng, Z.; Li, F.; Monroe, W.; Shi, P.; Sun, Z.; Wu, W.; Wang, S.; Wang, W.Y.; Yuan, A.; et al. Icorating: A deep-learning system for scam ICO identification. *arXiv* **2018**, arXiv:1803.03670. [CrossRef]
23. Nerurkar, P.; Bhirud, S.; Patel, D.; Ludinard, R.; Busnel, Y.; Kumari, S. Supervised learning model for identifying illegal activities in Bitcoin. *Appl. Intell.* **2020**, *5*, 3824–3843. [CrossRef]
24. Farrugia, S.; Ellul, J.; Azzopardi, G. Detection of illicit accounts over the Ethereum blockchain. *Expert Syst. Appl.* **2020**, *150*, 113318. [CrossRef]
25. Kumar, N.; Singh, A.; Handa, A.; Shukla, S.K. Detecting Malicious Accounts on the Ethereum Blockchain with Supervised Learning. *Cyber Secur. Cryptogr. Mach. Learn.* **2020**, *12161*, 94–109. [CrossRef]
26. Chen, Y.; Dai, H.; Yu, X.; Hu, W.; Xie, Z.; Tan, C. Improving Ponzi Scheme Contract Detection Using Multi-Channel Text CNN and Transformer. *Sensors* **2021**, *21*, 6417. [CrossRef]
27. Wang, L.; Cheng, H.; Zheng, Z.; Yang, A.; Zhu, X. Ponzi scheme detection via oversampling-based Long Short-Term Memory for smart contracts. *Knowl.-Based Syst.* **2021**, *228*, 107312. [CrossRef]
28. Chen, W.; Li, X.; Sui, Y.; He, N.; Wang, H.; Wu, L.; Luo, X. SADPonzi: Detecting and Characterizing Ponzi Schemes in Ethereum Smart Contracts. *Proc. ACM Meas. Anal. Comput. Syst.* **2021**, *5*, 2. [CrossRef]
29. Liang, Y.; Wu, W.; Lei, K.; Wang, F. Data-driven Smart Ponzi Scheme Detection. 2021. Available online: https://arxiv.org/abs/2108.09305v1 (accessed on 1 July 2022). [CrossRef]
30. Fan, S.; Fu, S.; Xu, H.; Cheng, X. Al-SPSD. Anti-leakage smart Ponzi schemes detection in blockchain. *Inf. Process. Manag.* **2021**, *58*, 102587. [CrossRef]
31. Rahouti, M.; Xiong, K.; Ghani, N. Bitcoin concepts, threats, and machine-learning security solutions. *IEEE Access* **2018**, *6*, 67189–67205. [CrossRef]
32. Jung, E.; Le Tilly, M.; Gehani, A.; Ge, Y. Data mining-based ethereum fraud detection. In Proceedings of the 2019 IEEE International Conference on Blockchain, Seoul, Korea, 14–17 July 2019; pp. 266–273.
33. Available online: https://etherscan.io/accounts/label/phish-hack (accessed on 4 September 2021).
34. Available online: https://etherscan.io/accounts (accessed on 4 September 2021).
35. Available online: https://api.etherscan.io/api?module=contract&action=getsourcecode&address=0xBB9bc244D798123fDe783fCc1C72d3Bb8C189413&apikey=YourApiKeyToken (accessed on 1 July 2022).
36. Available online: https://api.etherscan.io/api?module=account&action=txlist&address=0xc5102fE9359FD9a28f877a67E36B0F050d81a3CC&startblock=0&endblock=99999999&page=1&offset=10&sort=asc&apikey=YourApiKeyToken (accessed on 1 July 2022).
37. Haibo, H.; Yang Bai, E.; Garcia, A.; Shutao, L. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. (2008). In Proceedings of the IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Padua, Italy, 18–23 July 2008; pp. 1322–1328. [CrossRef]
38. Maaten, L.V.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
39. Bistarelli, S.; Mazzante, G.; Micheletti, M.; Mostarda, L.; Tiezzi, F. Analysis of Ethereum Smart Contracts and Opcodes. In *Primate Life Histories, Sex Roles, and Adaptability*; Springer: Berlin/Heidelberg, Germany, 2020.
40. Alruily, M. Classification of Arabic Tweets: A Review. *Electronics* **2021**, *10*, 1143. [CrossRef]

41. Wood, G. Ethereum: A Secure Decentralized Generalized Transaction Ledger. Available online: http://gavwood.com/paper.pdf (accessed on 1 July 2022).
42. Hirshman, J.; Huang, Y.; Macke, S. *Unsupervised Approaches to Detecting Anomalous Behavior in the Bitcoin Transaction Network*; Technical Report; Stanford University: Stanford, CA, USA, 2013.
43. Raschka, S. *Python Machine Learning*; Packt Publishing Ltd.: Birmingham, UK, 2015.
44. Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *24*, 123–140. [CrossRef]
45. Hastie, T.; Tibshirani, R.; Friedman, J.H. *Boosting and Additive Trees. The Elements of Statistical Learning*, 2nd ed.; Springer: New York, NY, USA, 2009; pp. 337–384, ISBN 978-0-387-84857-0.