*International Journal of Advanced Technology and Engineering Exploration, Vol 10(109)*
*ISSN (Print): 2394-5443 ISSN (Online): 2394-7454*
*http://dx.doi.org/10.19101/IJATEE.2023.10102216*

Research Article

# Anomaly detection in smart contracts based on optimal relevance hybrid features analysis in the Ethereum blockchain employing ensemble learning

**Sabri Hisham*, Mokhairi Makhtar and Azwa Abdul Aziz**
Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, 22000 Besut, Terengganu, Malaysia

Received: 31-August-2023; Revised: 19-December-2023; Accepted: 21-December-2023


## Abstract
***Blockchain 2.0 has revolutionized the domain by introducing blockchain as a decentralized application (DApp) development platform, previously recognized mainly in the cryptocurrency sphere. Consequently, the rise of DApp development has inadvertently camouflaged fraudulent activities within smart contracts, leading to substantial losses for investors. Implementing machine learning (ML) approaches can significantly enhance the efficacy of anomaly detection. However, many studies still grapple with selecting the most pertinent features to optimize anomaly detection levels. This challenge intensifies when managing the high-dimensional raw data extracted directly from the Ethereum blockchain network, which falls under the category of big data. Smart contracts, the core of blockchain that governs DApp logic, have increasingly become a haven for fraud. This study focuses on analyzing three primary characteristic components based on contract source code (operation code (opcode), application binary interface (ABI) code, and contract transaction) to develop anomaly detection models in smart contracts using an ensemble hybrid feature strategy. The approach involves two key stages: firstly, reducing the initial feature size through constant, quasi-constant, and variant validation; and secondly, identifying the most relevant feature set using the searching for uncorrelated list of variables (SULOV) method, grounded in the minimum redundancy maximum relevance (MRMR) principle. The anomaly detection model employs a voting ensemble technique, harnessing a dataset of the most pertinent features. The model's effectiveness is gauged by comparing its performance with individual models, including random forest (RF), k-nearest neighbor (KNN), decision tree (DT), linear discriminant analysis (LDA), and stochastic gradient descent (SGD). The findings indicate that the proposed model achieves superior anomaly detection levels, with a determination value measurement rate of 92.99%, outperforming individual classifiers using the 44 most relevant features while minimizing classification time. The model's efficiency is further corroborated through comparative analysis with previous studies and alternative methodologies using the same contract dataset. The proposed ensemble-based model significantly improves anomaly detection in contract source code analysis, employing a minimal and relevant set of features refined through the SULOV method.***

## Keywords
***Ethereum, Blockchain, Smart contract, Features selection, Relevance features, Ensemble method, Anomaly detection.***

## 1.Introduction
Blockchain architecture is based on distributed and decentralised technology used to store transaction records in blocks [1]. These blocks are linked to each other based on the value of the hash address (previous hash) generated through a cryptographic mechanism [2]. Blockchain technology has developed as an open ledger to record transactions in a distributed manner. New blocks will be created after the mining process is complete through the protocol consensus that requires each peer to verify transactions [3–5].

The development of decentralised applications (DApp) has grown in popularity with the digitalisation of smart contracts, previously only utilised for cryptocurrency transactions, in the blockchain 2.0 Era. In the areas of health [6], decentralised voting [7], the internet of things (IoT) [8], and supply chain management [9], the use of DApp has expanded quickly and caught the attention of both industry and academics [10]. Nowadays, blockchain is a technological catalyst for the introduction of several new applications, such as non-fungible tokens (NFT), metaverse, decentralised autonomous organisation (DAO), and decentralised finance (DeFi) [11].

*Author for correspondence

Smart contracts are becoming more popular on the Ethereum network and have been introduced to solve the weaknesses discovered in Bitcoin [12]. In addition to the expansion of DApp development, Ethereum is the second-largest group in the world of cryptocurrencies through the Ether currency after Bitcoin [13], with a market capitalisation of around US$25 billion and had more than 67 million accounts in June 2019 [14]. Szabo [15] is credited with introducing the idea of smart contracts. The author described them as a set of digital agreement details that require members to abide by the terms of the agreement. Due to technological restrictions at the time, which demanded the deployment of security features through cryptographic protocols, this concept was abandoned. However, Satoshi Nakamoto originally created Bitcoin in 2008, and smart contracts re-emerged in popularity as a result of their adoption of Bitcoin technology. With the introduction of Ethereum and the increase in the usage of smart contracts, which apply programming script logic without requiring outside parties, Buterin [16, 17] has energised the development of DApp.

Basically, the architecture of Ethereum consists of two types of accounts, namely externally owned accounts (EOA) and contract accounts (CA). The main function of EOA, which is controlled by a private key, is to transfer Ether between accounts. Meanwhile, the contract code fully controls CA and is responsible for deploying smart contracts. Both types of accounts are in hexadecimal format [18]. Each account consists of four attributes (Hash, Storage, Ether Balance, and Nonce). Nonce provides an overview of the quality of transactions between accounts or contracts. At the same time, the Ether Balance reveals the account balance in Wei units, while the hash is a hash code generated by the ethereum virtual machine (EVM), and storage represents the 256-bit hash resulting from the Merkle Root mechanism [19]. The Ethereum network consists of two types of transactions: normal transactions and internal transactions [20]. Furthermore, the user accounts initiate normal transactions, and internal transactions refer to transactions initiated by smart contracts. Internal transactions are also managed off-chain (no cryptographic signatures). However, there are some managed on-chain (not a part of the blockchain) that require a small amount of gas (Ethereum Gas) for the transaction fee (affecting address balances). Therefore, internal transactions, also known as messages, do not have a transaction hash compared to normal transactions. Thus, each normal transaction has more than one internal transaction referring to it.

Smart contracts are exposed to threats of invasion and cybercrime to the point of causing huge losses to investors in particular. This is due to bugs in the contract, low quality of the source code, and no security assessment of the contract performed before production on the real network [21]. Among other causes are attitude as well as negligence by users, causing confidential information (private keys, for example) to be stolen, the nature of contracts that are automatic execution, and the lack of enforcement or regulatory mechanisms in managing blockchain applications. It is also due to the nature of blockchain users or accounts being anonymous (tracing behaviour becomes difficult) [22]. Smart contracts developed using the Solidity language are exposed to vulnerabilities like traditional programming languages (Java, C, and C++), and what differentiates them is that the contract's source code is unable to be modified after being produced into the blockchain network [23].

Other than that, weaknesses in smart contracts have provided hackers with the opportunity to manipulate the source code for the purpose of fraud or scams. User-oriented DApp developed through popular game applications such as Fomo three dimensions (3D) and Cryptokitties2 have provided wealth to early investors from the investment results of new investors [5]. This fraudulent activity is called a Ponzi scheme, and it also advertises its activities on bitcointalk.org as high-yield investment programmes (HYIP) or gambling games to trap new investors [24]. In addition, weaknesses in smart contracts caused the DAO attack in 2016 and Parity Wallet in 2017, causing an estimated loss of over $400 million [2]. The attackers have stolen 3.5 million Ether, equivalent to US$45 million, due to the vulnerability in the DAO contract. Hence, fixing the vulnerability requires launching a hard fork, which is risky, even if it involves a low cost [23].

Therefore, an initial anomaly detection system in the blockchain network is critical to protecting against cybercrime. Since smart contracts have been the focus of intruders committing fraudulent activities, an analysis of abnormalities in the source code needs to be conducted. Consequently, a preliminary study creates a manual review of the source code accessed from etherscan.io for the open source contract (source code available) to identify source code behaviour that reflects ponzi scams [25]. A manual

structural analysis review of the source code was performed in a study by [26], who used the etherscam.io platform to access verified contracts to detect Ponzi schemes hiding in contracts. Although this study succeeded in detecting Ponzi schemes, the analysis was limited to the structure of the programme code. However, analysing the billions of non-open source contracts in the Ethereum network is challenging since the source code is unavailable. Therefore, it is impossible for a manual approach to analyse anomalies, considering that 77.3% of smart contracts are open source [27]. Manual anomaly identification also requires machine specifications such as large memory, a central processing unit (CPU), storage, and many human resources. It is also very prone to errors as humans conduct this process. Since most blockchain accounts are anonymous, detecting fraudulent activities becomes increasingly difficult and challenging [28]. In addition, the size of large-capacity blockchain data (categorised as big data) challenges researchers to determine the most relevant features that can produce an optimal detection level. Thus, the machine learning (ML) approach that is able to extract features from large data sets and is scalable is very suitable to be adapted together with blockchain technology to detect fraudulent activities [29, 30]. Therefore, a study by [31–33] has analysed the smart contract source code based on the ML approach to detect Ponzi schemes. Most recent studies have begun to explore artificial intelligence (AI) [34, 35] and deep learning (DL) [36, 37] to detect anomalies in smart contracts.

However, previous studies still faced several issues and challenges in producing an optimal anomaly detection level based on smart contract source code analysis. First, determining the characteristic components of smart contract source code analysis is not comprehensive. Most analyses only focus on one or two characteristic components of the source code (operation code (Opcode), transaction) through an individual analysis approach (not hybrid features) [38]. Second, extracting characteristics for contract source code components is difficult as it involves semantic code-based data (textual) and needs to be monitored to produce a better level of anomaly detection [39]. Third, identifying the most relevant features from the large original feature dimension is a challenging task as it determines the performance of the final model [40]. Fourth, the performance of smart contract anomaly analysis in previous studies is still at a moderate level. For example, the 95% precision rate and 69% recall rate [32] can still be improved by adapting the ensemble approach to overcome the weaknesses of individual models [41–43]. Fifth, the problem of balancing the distribution of data labels on the data set interferes with the model's performance. For example, even though transactions on Ethereum exceed 3.8 billion, only 2041 phishing accounts were successfully detected. This contributes to the production of overfitting and weak models [4].

This scenario motivates researchers to investigate feature hybrid approaches to analyse anomaly detection for all feature components associated with source code, namely Opcode, application binary interface (ABI) code, and transaction. Nevertheless, the source code characteristics, namely Opcode and application binary interface code (ABI code), are presented in a textual format. Consequently, it becomes necessary to employ extraction techniques and feature vectorisation transformations. The utilisation of a hybrid feature method generates a substantial feature size and necessitates feature selection approaches to identify the most pertinent features while maintaining optimal model performance. Ultimately, implementing a prediction model that can enhance the performance of the existing model is necessary.

Hence, this study delineates fourth objective derived from the deficiencies and concerns identified in prior research. The primary aim of this study is to analyze the smart contract source code's behaviour for the purpose of anomaly detection. This analysis will be based on combining hybrid features, including Opcode, ABI code, and transaction data. The second purpose pertains to the execution of feature extraction and transformation methods on the source code, specifically the Opcode and ABI code, which encompasses textual information. The third purpose of this study is to implement feature filtering (reduction) and searching for an uncorrelated list of variables (SULOV) feature selection methods in order to identify and retain the most relevant characteristics. The fourth objective is to enhance the performance of the ultimate model by employing the voting ensemble technique.

Generally, processes start with the initial feature reduction, which checks and verifies the quasi-constant and variance thresholds. Consequently, the most relevant features (screening uncorrelated features) are determined from a large number of features in order to contribute to improving the performance of the final model using the SULOV technique (based on the minimum redundancy

maximum relevance (MRMR) approach). The final model is built based on the parameter input of the most relevant feature set to be trained and tested using the approach ensemble. As it is known, the ensemble approach can overcome the performance of weak classifier models [44] and avoid the occurrence of overfitting [45].

Therefore, the following procedures must be conducted in order to accomplish the study's objective and paper contribution: 1) Using a dataset of 1,904 Ponzi contracts derived from research [38]. 2) Using the Etherscan.io application programming interface (API), create three different types of contract source code datasets (Opcode, ABI code, and contract-account transaction). 3) Collecting and converting text semantic data sets using vectorisation transformation techniques (N-Gram and term frequency-inverse document frequency (TF-IDF)). 4) Combining in an ensemble to create a hybrid feature composed of contract-account transactions, ABI code, and Opcode.5) Apply quasi-constant and variance analysis approaches for prefix feature reduction. 6) Employ the MRMR-based SULOV approach to identify the most pertinent features. 7) The ensemble voting method is used to develop the final model. 8) Analysing the findings by contrasting the ensemble model's performance with that of the individual model and the findings of other studies that used the same data set. The study's findings demonstrate that the voting ensemble model, which performed better than the individual classifier model, obtained an accuracy value of 92.99%.

This study's structure is broken down into sections. The evaluation of earlier study findings is described in section 2. The proposed research methodology is thoroughly explained in section 3. Correspondingly, the analysis of the experimental study's outcomes is presented in section 4. The findings are explained and further discussed in section 5. The study recommendations, findings, and potential future studies are summarised in section 6.

## 2.Literature review

This section describes previous studies related to anomaly detection based on smart contract source code analysis. It is crucial to analyse prior research to expand our understanding of a field. A comparison of earlier studies on anomaly detection in smart contracts is provided in *Table 1*.

**Table 1** Previous study for anomaly detection in smart contracts

| References | Description | Dataset | Features | Model | Limitations |
|---|---|---|---|---|---|
| [46] | Long short-term memory (LSTM)-based oversampling is used in Ponzi schemes detection approach based on oversampling-based (PSD-OL), a method for smart contracts that detects Ponzi schemes. | Ponzi Dataset from XBlock dataset (3,019 contracts- 2,851 normal contracts, 168 Ponzi contracts) | Transaction, Opcode | LSTM | Did not use relevant feature selection techniques. |
| [38] | Detection of normal and abnormal behaviour in a smart contract-based Ponzi scheme dataset. | 1,904 contracts (Ponzi label) from Etherscan.io | Transaction Opcode, Source code | Soft Voting Ensemble | Did not use relevant feature selection techniques. Did not use the ABI Code format since the original source code is too complex and difficult to read and understand by humans. |
| [5] | Develop a Multi-view Cascade Ensemble model (MulCas) using the ML approach for Ponzi detection in a smart contract. Extract three features (bytecode, semantic, and developer) from two data sources (opcode, transaction) | 6,498 contracts (314 Ponzi contracts,6,184 normal contracts) from Etherscan.io | Opcode, Transaction | Ensemble | Did not use relevant feature selection techniques. The source code needs to be compiled on a different version of the solidity compiler. |
| [47] | Introduce SourceP, a technique that uses data flow and pre-trained models to find smart Ponzi schemes on the Ethereum platform. | Ponzi Dataset from XBlock dataset and collect 6,498 (318 Ponzi contracts,6,180 normal contracts) from Etherscan.io | Opcode | GraphCodeBert | Did not use relevant feature selection techniques and ensemble learning approach. |

| References | Description | Dataset | Features | Model | Limitations |
|---|---|---|---|---|---|
| [23] | Create sFuzz tools, an adaptive fuzzer for smart contracts on the Ethereum platform that targets those hard-to-cover branches using an American fuzzy lop (AFL) fuzzer and an effective, lightweight multi-objective adaptive strategy. | Smart contract code coverage analysis for the test suite | Solidity code (sol files) | AFL-based | Did not use the feature filtering method and relevant feature selection techniques. |
| [48] | Present the convolutional-based bidirectional gated recurrent Unit (CBGRU) model, a novel hybrid DL approach that carefully incorporates various word embeddings (Word2Vec, FastText) and DL techniques (LSTM, gated recurrent unit (GRU), bidirectional LSTM(BiLSTM), convolutional neural network(CNN), and BiGRU). | SmartBugs Dataset-Wild from [49] contained 47,587 real sol files | Source code files (solidity) | DL | Did not use the feature filtering method and relevant feature selection techniques after feature extraction. |
| [50] | Purpose the Echidna smart contract fuzzer tools, an open-source tool that enables it to automatically generate tests to find assertion and custom property violations. | VeriSmart benchmark, TetherToken | Solidity code (sol files) | property-based fuzzing (QuickCheck) | Did not use the feature filtering method and relevant feature selection techniques. |
| [51] | Contained two parts (a sharing layer and a task-specific layer)<br>sharing layer: text-to-vector transformation<br>task-specific layer: construct a classification model using CNN | XBlock platform (149,363 smart contracts) | Opcode | Classical CNN | Did not use the feature filtering method and relevant feature selection techniques. |
| [52] | Propose a tool called SmartCheck that significantly enhances the detection of vulnerabilities linked to the DASP10 categories of faulty randomisation, temporal manipulation, and access control. | Two datasets of Solidity contracts with 208 tagged vulnerabilities and 47,518 unique contracts were collected through Etherscan | Solidity code (sol files) | Based on DASP10 categories | Did not use the feature filtering method and relevant feature selection techniques. |
| [2] | Propose an ML model based on LightGBM and N-gram characteristics to identify honeypot contracts based on frequency opcodes. | 218,250 negative samples (non-honeypot contracts) and 616 positive samples (honeypot contracts) were obtained | Opcode | LightGBM | Did not use the feature filtering method or ensemble learning approach and focused on opcode source code analysis without examining the behaviour contract transaction. |
| [39] | Propose a semantic-aware detection method for ML-based detection of Ponzi schemes in Ethereum smart contracts. Semantic-aware detection approach for Ponzi (SADPonzi) using symbolic method | 1,395 well-labelled sample | Opcode | eXtreme gradient boosting (XGB), RF | Did not use the feature filtering method, relevant feature selection techniques, and ensemble learning approach. |

| References | Description | Dataset | Features | Model | Limitations |
|---|---|---|---|---|---|
| [53] | Propose an ordered boosting-based anti-leakage smart Ponzi scheme detection (Al-SPSD) model. Optimise using synthetic minority oversampling technique (SMOTE) and Optuna framework. | Extract the contract from Google Big Query. (81 Ponzi,644 non Ponzi) | Opcode | Gradient boosting method (GBDT) | Did not use the feature filtering method, relevant feature selection techniques, and ensemble learning approach. |

The present investigation, carried out by [46], has analysed the Ponzi scheme phenomenon within the context of smart contracts. The source code analysis relies on utilising CA characteristics and bytecode Opcode. This is achieved by introducing PSD-OL, an oversampling-based LSTM technique. Note that the LSTM model was trained using datasets provided by the XBlock public repository (3019 contracts). The study's findings indicate that the proposed model achieved a high accuracy value of 0.96 using the SMOTE technique. Nevertheless, the present study does not employ the feature selection technique to ascertain the most pertinent characteristics from the pool of 83 features (comprising opcode and transaction features) prior to training them using an LSTM model, which can potentially enhance the final model's overall performance.

The aforementioned study by [38] examines anomalies in smart contracts by analysing three components of contract source code: Opcode, account features, and source code features. This analysis was performed using a hybrid approach that combines various features. A comprehensive dataset of 1,904 CAs was utilised to gather the necessary opcode data sets, account features, and source code. Moreover, the present study employs a soft ensemble voting methodology for the purpose of training models for contract anomaly detection. The investigation yielded an accuracy value of 0.88%, surpassing the previous study conducted by [32], which achieved an accuracy of 0.79% using the identical dataset. However, this study does not employ precise feature selection strategies to generate the most pertinent features that enhance the efficacy of the ultimate model. This study also utilises the original source code dataset, which exhibits a high level of complexity, semantic intricacy, and logical structure and presents challenges in terms of human readability and comprehension. The utilisation of ABI code is perceived as more appropriate for studying source code due to its inherent comprehensibility for human interpretation, facilitating the enhancement of anomaly prediction capabilities.

Insufficient consideration was provided in the aforementioned study conducted by [5] to the sampling methodology employed for the data sets utilised, as well as an undue emphasis on the analysis of CA transactions. Consequently, the researcher introduced a theoretical framework known as the MulCas by analysing the source code (opcode and transaction) and expanding the existing dataset from the prior investigation, encompassing 6,498 CAs. This study extracts three new features (Opcode, semantic, and developer) from two data set sources (Opcode and transaction). Other than that, this study has contributed the largest contract data set (6,498 contracts) from [31]. The study's findings indicate that the MulCas model, as described, achieves a recall value of 0.674, a precision of 0.951, and an F1-score of 0.789. Note that these metrics demonstrate superior performance compared to the SadPonzi [39]. Analysis of contract source code in a hybrid way (opcode, transaction) through feature extraction has produced a large feature size or dimension. Thus, no relevant feature selection techniques are specified in this study to help improve model performance.

The implementation of graph analysis utilising flow graph data to analyse Ponzi schemes within a contract has been carried out [47]. This approach involves constructing a classification model based on the aforementioned data. This methodology diverges from prior research endeavours focused on feature extraction when analysing the source code. The model, referred to as SourceP, has analysed the source code in opcode format. Moreover, the experimental findings have yielded an F1-score of 90.7% and a recall of 87.2%. The findings from the comparative analysis with previous studies indicate that SourceP outperforms MulCas [5] and SadPonzi [39] in terms of recall, f1-score, and precision. The primary objective of this study is to examine the method of source code analysis based on individual characteristics, specifically opcode. The paper argues that the hybrid characteristic approach, which combines opcode with other characteristics such as source code, transaction, or account, has some limitations. Nevertheless, this study does not perceive the identification of pertinent attributes as a means to

enhance the performance of models, as its scope is limited to pre-training model creators.

The researchers in [23] conducted a study wherein they devised a contract fraud detection tool named sFuzz, which was built around the multi-objective AFL fuzzer method. The efficacy of this tool has been evaluated by the development of a contract simulation encompassing a sample size exceeding 4,000 contracts. The primary aim of Sfuzz is to do source code testing on contracts prior to their deployment on the Ethereum network to identify any potential flaws or vulnerabilities. Consequently, the present study employs Opcode source code to conduct an analysis aimed at achieving high code coverage. The experimental findings indicate that sFuzz exhibits notable reductions in processing times, superior effectiveness, and enhanced reliability compared to the current fuzzer.

A study using a hybrid DL CBGRU model was discussed [48]. The study has analysed various word embedding techniques, including Word2Vec and FastText, and employed a range of DL methodologies, including LSTM, GRU, BiLSTM, CNN, and bidirectional gated recurrent unit (BiGRU). Moreover, the proposed hybrid technique utilises dataset-wild data sets, specifically SmartBugs, which are in the solidity format (sol files), to identify vulnerabilities in smart contracts. The experimental findings demonstrate that the CBGRU model effectively achieves an average accuracy rate of 93% when analysing vulnerabilities such as timestamp manipulation, infinite loop, and reentry. This study utilises an extraction methodology that combines two features without employing the feature filtering method or selecting the most relevant features to enhance the model's performance.

Research conducted by [50] has devised a set of testing tools named Echidna, specifically designed for smart contracts. These tools are built upon the foundation of smart contract fuzzers. The efficacy of this tool has been evaluated through the use of ten commercially available security solutions in order to solicit comments pertaining to security, usability, and user experience. In addition, the primary objective of this work is to identify and address software defects in smart contracts during their first stages while ensuring that the processing performance remains at an appropriate level. The experimental findings indicate that Echidna demonstrates a detection capability for bugs within a time frame of less than 2 minutes. However, Solfuzz, when utilising real tokens (namely Tether), requires a minimum of 15 minutes or maybe longer.

A study was undertaken by [51] to perform a semantic analysis of the source code of the contract. The study employed a text processing strategy known as word embedding in the bottom sharing layer, and DL techniques were utilised to create the model in the task-specific layer. Subsequently, the multi-task model, alternatively referred to as this technique, has demonstrated superior effectiveness in detecting fraudulent activities. The multi-task model is characterised by its cost-effectiveness, efficiency in terms of time, utilisation of human resources, and storage requirements compared to a single-task model. This study uses extracting Opcode data sets and labelling normal and abnormal using vulnerability tools. However, this study lacks the implementation of feature screening techniques and the selection of the most relevant features after going through the Opcode feature extraction process on the bottom screen using the word embedding method before the model is developed using CNN.

The analytical tool Smartbug, developed by [52], tests and debug smart contracts. An analysis informed the development of this tool of Solidity source code (sol files), and it offers compatibility with a total of ten additional tools for integration purposes. The Smartbug detection repository contains a comprehensive collection of 143 annotated vulnerabilities, specifically focusing on 208 distinct vulnerabilities. The percentage range of 11% to 24% is determined by the decentralized application security project (DASP) 10 category, which encompasses time manipulation, access control, and faulty randomness.

The investigation of honeypot identification in smart contracts was conducted by [2] with the examination of honeypot scheme detection. The present study examined the symbolic and contractual behaviour within the Opcode through analysis. Hence, N-gram (a text processing technique) and the light gradient-boosting machine (LightGBM) algorithm were employed in constructing the detection model. The study's findings indicate that the utilisation of features, specifically unigram and bigram, effectively yields the F1-score (0.93) and area under the roc curve (AUC) (0.99) values in the context of detecting honeypots in smart contracts. One notable advantage of this study is its systematic execution of feature selection, extraction, undersampling, and feature significance procedures prior to the building of the

LightGBM model. The study could be enhanced by incorporating a feature filtering procedure with suitable methodologies and examining it within the framework of transaction behaviour.

The heuristic-guided symbolic technique has been used by [39] to develop SADPonzi. This study uses the solidity dataset to analyse investor behaviour, such as money transfer transactions (semantic approach) and its relationship with other investors (other users). The effectiveness of SADPonzi was analysed with 3.4 million contracts and successfully detected ponzi schemes (835). The evaluation result for the SADPonzi approach produced 100% precision, recall, and f1-score compared to the TxML and OpcodeML approaches. This study focuses on the analysis of semantic information generated from the process of symbolic execution, Opcode and does not use feature filtering or the selection of the most relevant features.

Correspondingly, a research investigation was conducted by [53] to examine the disparity in data sets pertaining to target classes through the analysis of contract source code (Opcode). The presence of imbalanced class distribution within the data has led to data leakage, resulting in a model exhibiting inferior performance. Therefore, this study has put out measures to mitigate data leaking using Al-SPSD methodology, which stands for Ponzi scheme detection using Ordered Boosting, has been designed to identify Ponzi scams within contracts by analysing the opcode source code. The study's findings indicate that Al-SPSD achieved a notable F1-score of 96%. The N-Gram approach is employed to extract opcode characteristics. In this work, an experiment was undertaken to ascertain the ideal value of n-gram (specifically, n = 1, 2, 3, or 4) that yields the maximum performance of the model. Nevertheless, the present study does not investigate the technique of feature filtering, the selection of the most pertinent features, or the utilisation of ensemble learning models.

Based on the analysis of previous studies related to the detection of anomalies through the analysis of smart contract source code, it has been proven that most studies analyse one or two source code features using an individual or hybrid analysis approach. The observation also discovered that a hybrid analysis for the Opcode source code, ABI code, and transaction features has not yet been implemented. The source code feature of ABI code is easier to understand and read by humans compared to the original source code (solidity). Most studies also focus on the feature extraction and model development process, compared to the adaptation of methods for model optimisation, such as feature screening techniques and the selection of the most relevant features.

## 3.Methods

The proposed smart contract-based anomaly detection framework through the analysis of the three components of the contract source code (opcode, ABI code, and contract transaction) is explained in this section (*Figure 1*). This process started with data collection for three source code components (opcode, ABI code, and contract transaction) based on the ponzi dataset containing 1904 CA shared publicly by [38]. However, this data set does not provide data for source code (ABI code and opcode) as well as CA transactions. Thus, these three datasets are obtained through the Etherscan API in JavaScript Object Notation (JSON) format. These three raw data are read directly from Etherscan and stored in the My's Structured Query Language (MySQL) database through their respective tables to facilitate the feature extraction process in the pre-processing phase. However, the source code data (ABI code and Opcode) is textual-based, and the ML approach only operates optimally based on numerical data. Therefore, Opcode and ABI code must undergo a feature transformation process to produce numeric vector values through the N-Gram and TF-IDF. In general, these features are classified into two types of feature categories: code features (Opcode and ABI code) and account features (contract transactions).

These three data sets are combined as an ensemble to form a main data set through a hybrid feature combination approach. Subsequently, this main data set, which has a large feature dimension, will reduce the number of features through the quasi-constant and variance validation methods. This feature set then goes through the process of selecting the most relevant features using the SULOV method by filtering uncorrelated list variables. This data set, which is the most revealing, is balanced using the SMOTE, as the distribution of data labelled Ponzi and non-Ponzi is unbalanced. The last phase is to train an anomaly detection model based on the ensemble approach using the data set parameter input resulting from the previous process. The anomaly prediction results in the output that determines whether the contract is normal (non-Ponzi) or abnormal (Ponzi). This research experiment was conducted using the Python (Jupyter) programming

languages, from the data collection phase to the development of the final model.

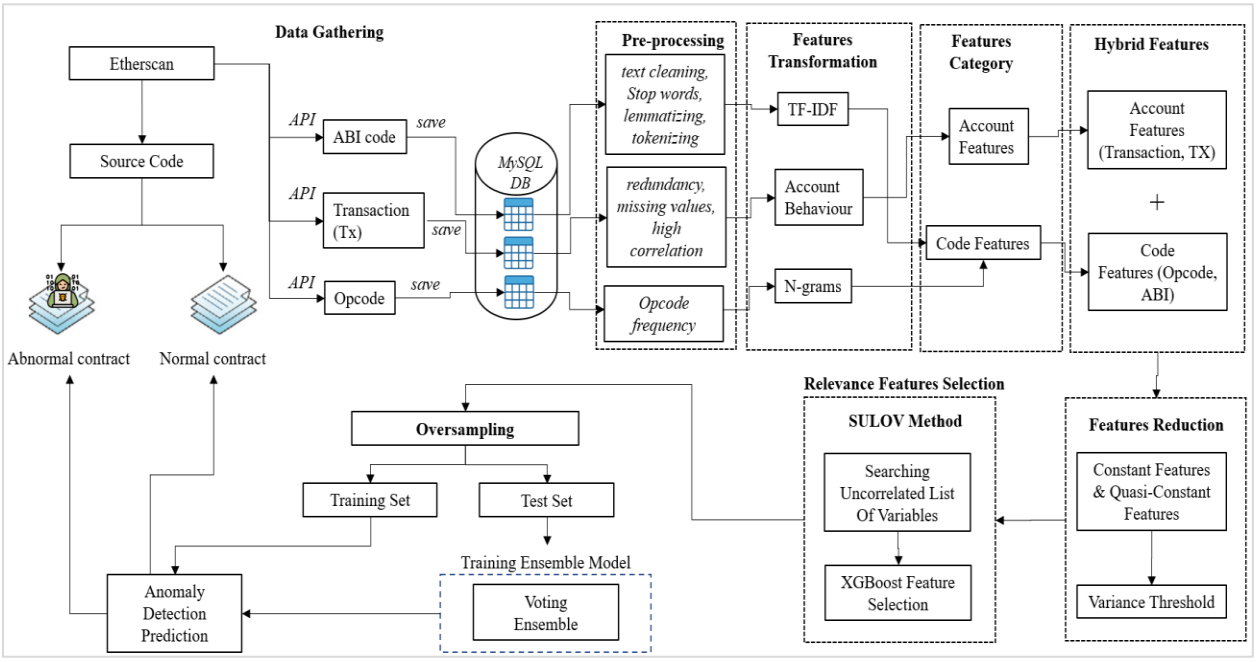


**Figure 1** Smart contract anomaly detection framework

### 3.1Data gathering

The source of the data set is a crucial aspect and requirement of the initial research phase. This study analyses the smart contract source code based on three components of the source code (Opcode, ABI code, and contract transaction) extracted directly from etherscan.io using a Ponzi dataset of 1,904 labelled CA shared by [38]. This dataset containing 1,904 CA was labelled with 1,599 as '0' (non-Ponzi) and 305 as '1' (Ponzi), as indicated in *Table 2*.

**Table 2** Ponzi contract dataset

| Contract account | Target distribution | Size |
|---|---|---|
| 1,904 | non-Ponzi | 1,599 |
| | Ponzi | 305 |

The provided dataset exclusively consists of categorised CA, specifically distinguishing between Ponzi and non-Ponzi accounts. It does not encompass datasets pertaining to source code components such as opcode, ABI code, and transaction information. Thus, the three characteristic components of the contract source code (Opcode, ABI code, and contract transaction) are extracted directly from Etherscan.io (blockchain explorer platform) using the API endpoint (using the API key obtained after registration at Etherscan.io) based on 1,904 CA labelled Ponzi and non-Ponzi (refer to *Table 3*). Note that this data crawling process has produced 1,904 Opcodes, ABI codes, and normal transactions (external transactions). Normally, a Ponzi has a lifespan with a median rate of only 2.5 days before being blocked by Etherscan.io. Thus, the transaction for this CA is too minimal or zero [5]. Therefore, the data set containing the normal contract (labelled '0') always has new transactions and is constantly growing up to now. An example of transaction details for a normal contract address is '0x00000000219ab540356cBB839Cbe05303d7705Fa' as in *Figure 2*.

An instance of a Ponzi CA address, namely '0x1ce7986760ADe2BF0F322f5EF39Ce0DE3bd0C82B', has been in existence for a duration of 1,299 days and has been subjected to a blocking action by Etherscan (see *Figure 3*).

**Table 3** Etherscan.io API endpoint for data crawling

| Features category | Etherscan API endpoint |
|---|---|
| Opcode | http://etherscan.io/api?module=opcode&action=getopcode |
| ABI code | https://api.etherscan.io/api?module=contract&action=getsourcecode |
| Contract transaction | https://api.etherscan.io/api?module=account&action=txlist&address=xxx |

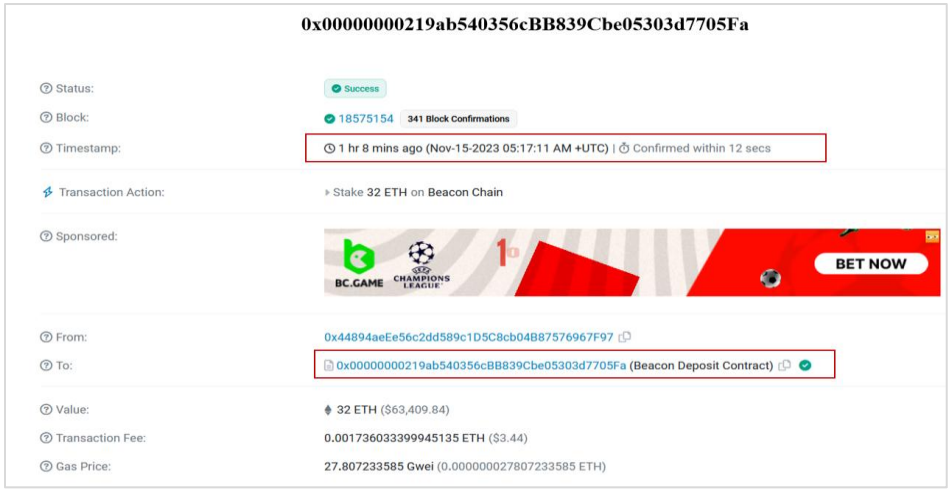


**Figure 2** The latest transaction of a normal contract

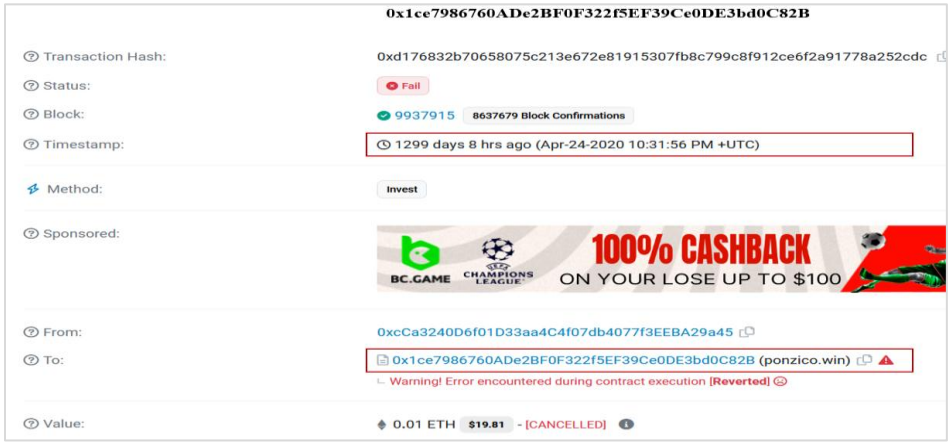


**Figure 3** The detailed transaction of the Ponzi contract

Many prior research investigations, such as those conducted by [46, 54, 55], have employed the Python library (disassembler) to produce opcodes from bytecodes. Nevertheless, this approach is vulnerable since it introduces the possibility of encountering failure while executing the conversion from bytecode to opcode. Therefore, the approach of acquiring the opcode source code straight via the API channel proves to be more efficient as it leverages Etherscan's management of the bytecode-to-opcode conversion process.

## 3.2Data Pre-processing

In section 3.1 of the data collection, the researchers extracted three components of the source code dataset: Opcode, ABI code, and CA transaction history. These components were obtained from labelled CA specifically those categorised as either Ponzi or non-Ponzi. The retrieved data was then placed in separate tables within the MySQL database. The pre-processing phase serves as an initial stage in which the data set is extracted prior to its utilisation in subsequent processes. The determination of whether the CA is classified as normal (non-Ponzi) or abnormal (Ponzi) relies on an analysis of a dataset that has undergone multiple pre-processing stages.

### 3.2.1Opcode features

As explained in the previous section, the proposed framework is based on access through the Etherscan API to obtain opcodes (refer to *Figure 4*). This method is easier since the Etherscan engine automatically converts bytecode to Opcode. The conversion of bytecode to Opcode is crucial for anomaly analysis since the content of the bytecode format is incomprehensible to humans [38]. However, most previous studies performed the process of converting bytecode to Opcode through disassembler software developed using the Python library. Yellow Paper (Ethereum) provides tabular references for binary bytecode instructions in

mnemonic Opcode form. This conversion process starts with making the bytecode in the form of a token, and this token is converted to an opcode through a set of instructions. In general, an opcode consists of two main components: the mnemonic (opcode) and the operand (hexadecimal number type). Examples of mnemonic opcodes are "PUSH2", "MSTORE," and "CALDATASIZE." While hexadecimal operands are "0x00f7," "0x00," and "0x80".

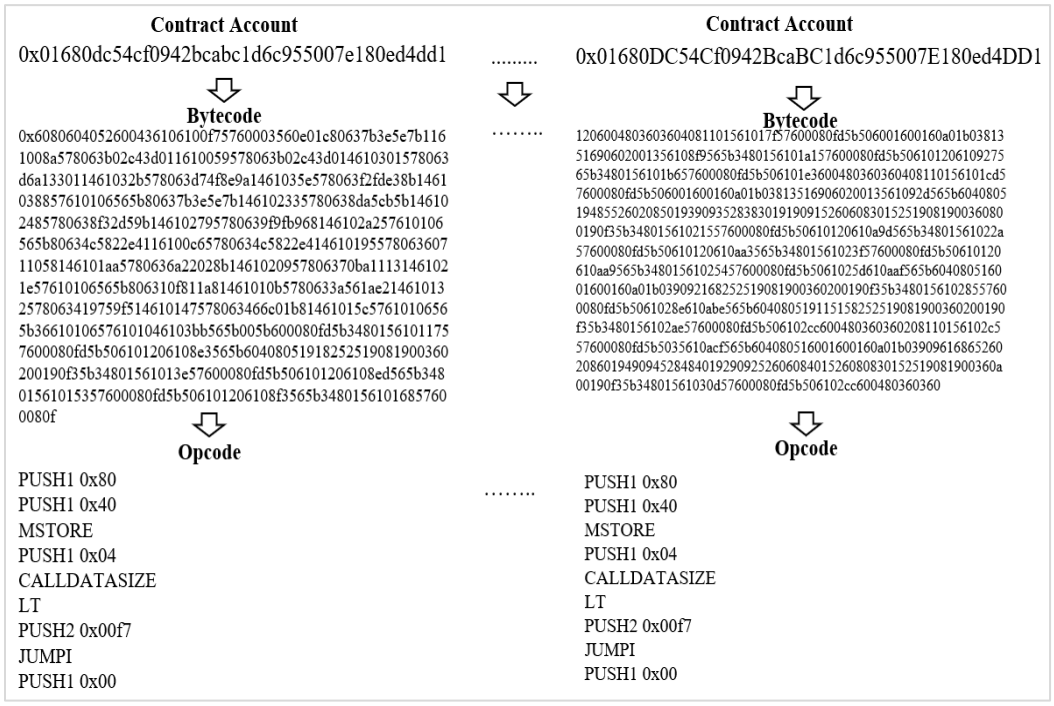


**Figure 4** Bytecode to Opcode conversion

In this section, the extraction of Opcode features consists of word mnemonics using feature extraction methods based on natural language processing (NLP). Therefore, within the framework of this proposal, a text analytical classification method has been used using the N-Gram (2, 3) method, which combines bigrams and trigrams to identify relevant word combinations. The latest Opcode feature set is produced after the removal process of words exhibiting high word frequency values (see *Figure 5*).

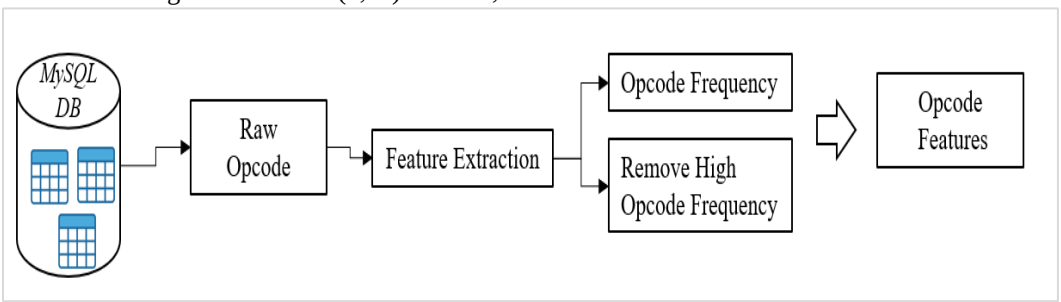


**Figure 5** Pre-processing of opcode

### 3.2.2ABI code features

This study focuses on open-source contracts that have source code. According to [56], over 2 million Ethereum contracts are deployed on the Ethereum network; only about 1% have source code, and the rest are hidden source codes (contracts). Technically, the contract owner can decide whether to publish the contract source code so that the public can interact through the contract address or not publish. However, to increase the level of reliability, the contract owner

can publish the contract using the compiler (setting the version and flag) so that public users can independently verify it through the contract address. Through the user's transparent verification mechanism, using compiler information (version and flag), the source code can be generated and viewed publicly. Therefore, in this study, the ABI code is obtained through the API for contracts that have been verified only so that the source code can be obtained. *Figure 6* displays the ABI code after the user verifies the contract. Meanwhile, *Figure 5* illustrates that the ABI code is unable to be obtained since the user has not verified this contract. However, the real source code is not used in this study as it contains sentences or words that do not contribute to improving the model's performance and yield insignificant results. Comment statements like those in *Figure 7* on the programme code discovered in the original source code do not provide a clear meaning in the model classification. As a result, the ABI code's content is more meaningful and clear than the original source code [57]. Nevertheless, a study by [38] used the original source code to be analysed in a hybrid way along with other source code components such as opcode and transaction contract.

ABI code contains source code based on textual, semantic, and programme logic. Therefore, the ABI code needs to go through a text-cleaning process to remove symbols, blank spaces, numbers, bad characters, and non-English characters. The next step is the pre-processing process, which includes tokenising, stop words, and word reduction based on the dictionary (lemmatising) (see *Figure 8*).

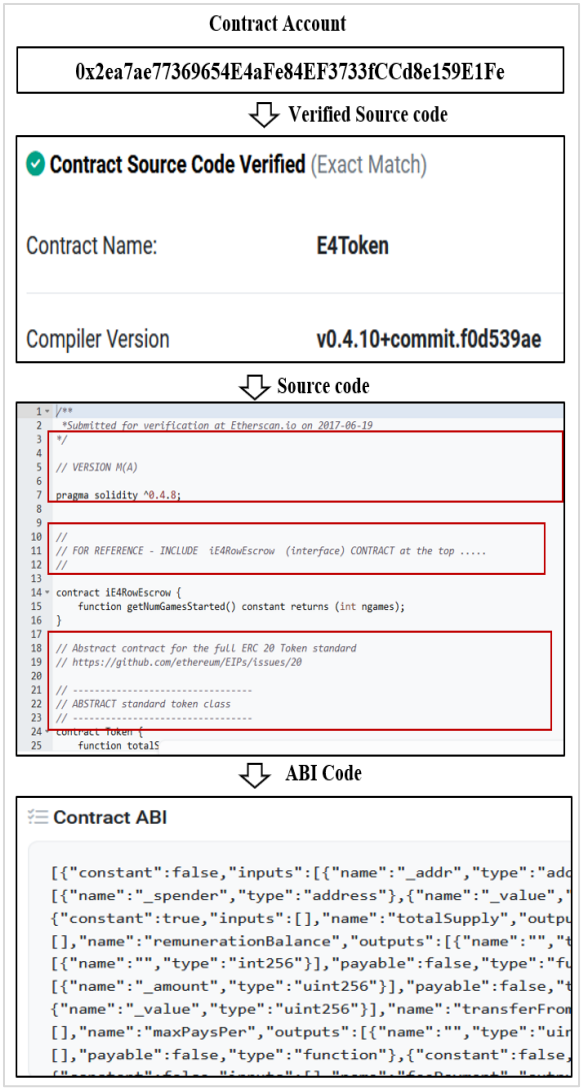


**Figure 6** Contract source code verified

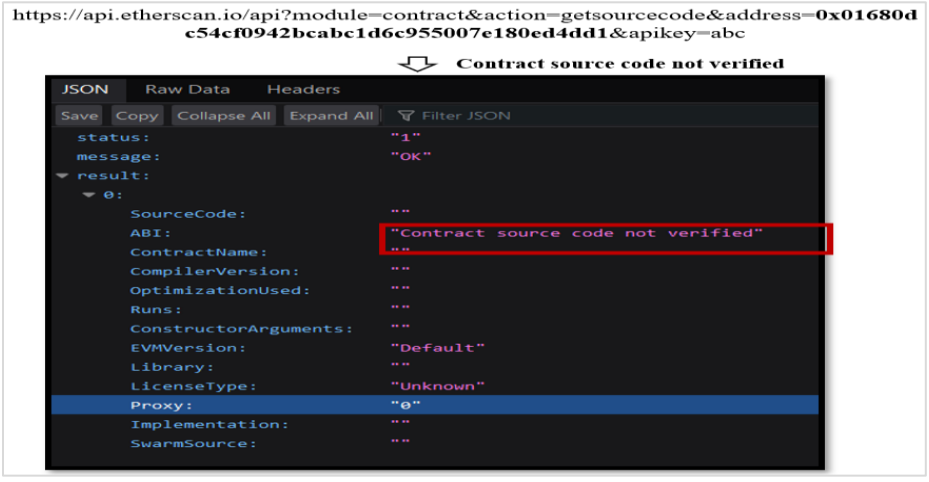


**Figure 7** Contract source code not verified

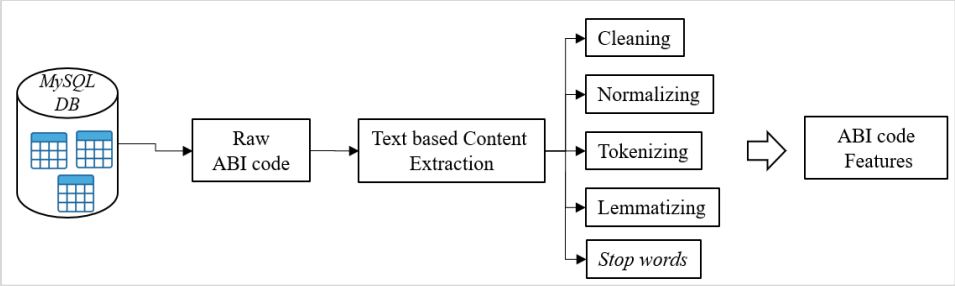


**Figure 8** Pre-processing of ABI code

### 3.2.3Account features

Transaction recording in Ethereum occurs when a user transfers ether to another user's account through a contract, creates a new smart contract, and invokes a smart contract (function). Basically, Ethereum consists of three categories of transactions, namely external, internal, and token transactions (ethereum request for comment (ERC)-20 and ERC-721) (see *Figure 9*). These two main categories are external transactions (also called normal transactions) and internal transactions. The normal transaction is referred to as the recording of Ether (ETH) transfer transactions between external users or EOAs differently via wallet addresses. Note that internal transactions are off-chain (no transaction recording in Ethereum and no cryptographic signature). They affect the ETH balance after the normal transaction is recorded. Transactions related to symbolic digital assets are related to ERC-20 tokens (fungible tokens) and ERC-721 (non-fungible). These three transaction categories are accessed through the transaction tab in Etherscan.io, as portrayed in *Figure 10*. However, this study only focuses on anomaly analysis on external transactions since it involves interaction from EOA, including scammers, hackers, and intruders through verified contracts.

The account feature refers to transaction analysis based on CA addresses labelled normal (Ponzi) or abnormal non-Ponzi). The raw data extracted directly from Etherscan through the API is stored in MySQL before it goes through the pre-processing phase. This phase filters features that contain null values (missing values) by replacing them with the median, removing features that have a zero variant value, removing features that are not applicable (ERC20_most_rec_token_type and ERC20_most sent token type), and removing duplicates on records (see *Figure 11*).

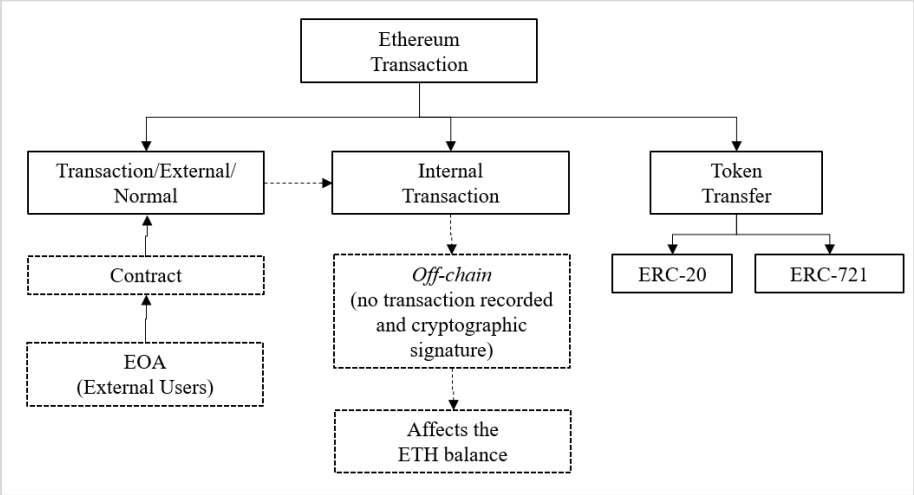


**Figure 9** Ethereum transaction category

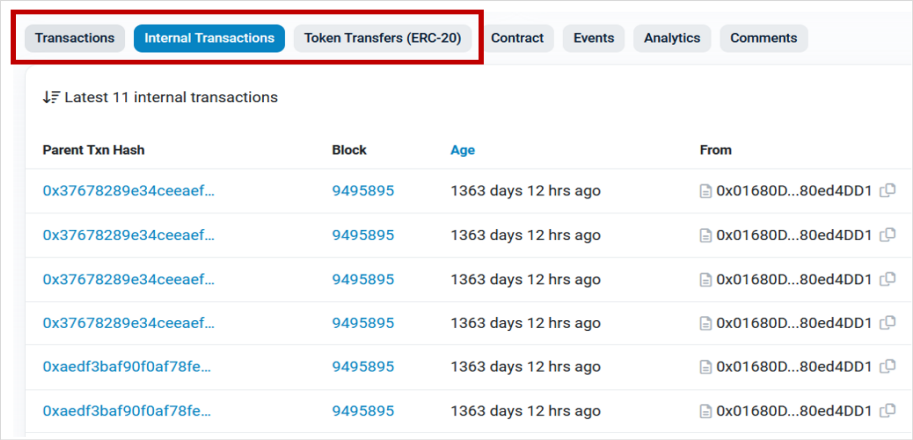


**Figure 10** Transaction tab in Etherscan blockchain explorer

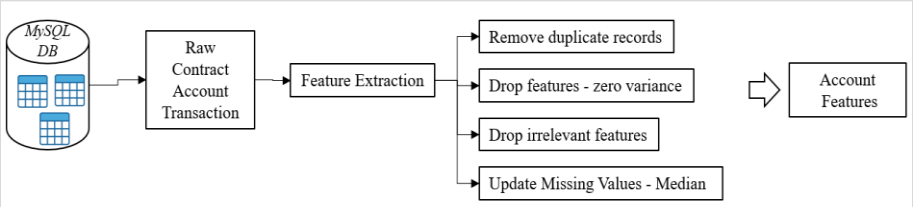


**Figure 11** Pre-processing of account features

### 3.3Features transformation

The dataset (opcode and ABI code) obtained from Etherscan.io comprises programming code that exhibits semantic, code-based, and text-based characteristics. However, in order to train the model using the ML approach according to established best practices, it is necessary to accept input parameters of the numerical type. Consequently, it is necessary to subject two datasets of source code (opcode and ABI code) to the process of feature extraction using vectorisation techniques such as N-Gram and TF-IDF. The TF-IDF method is a statistical method for determining the relevance of word groupings in a document [58]. The Python TfidfVectoriser function with token settings (char, word, ngram_range, max_features) is used to convert textual material to vector numbers. This technique is commonly used to analyse text-based data sets in order to discover phishing activities, abnormalities, fraud, and so on [59]. The TF-IDF approach, however, contains flaws such as extracted keywords with unclear meanings, misspellings, and so on. As a result, the pre-processing phase is critical for reducing feature extraction mistakes and increasing the accuracy of the final model produced by the processes of text cleaning, normalisation, tokenising, lemmatising, and stop words described in the preceding section.

The results of vectorisation in the generation of a vector value of numerical type lead to an expansion in the size of the feature dimension. Correspondingly, the feature vectorisation process has resulted in the data set dimensions of Opcode being (1904, 17309), ABI code being (1904, 10838), and contract transaction being (11722, 49). The research methodology employed involves examining the amalgamation of three distinct feature components inside a hybrid ensemble, resulting in the measurement of data dimensions (15530, 17356). *Table 4* summarises the dimensions of rows and features for the three feature components.

**Table 4** Feature vectorisation transformation

| Features category | Features | Feature extraction method | Rows size | Features size |
|---|---|---|---|---|
| Code features | Opcode | TF-IDF with input (word, ngram_range=(2,3) and max_features=30000) | 1904 | 17309 |
| | ABI code | TF-IDF with input (character, | 1904 | 10838 |

| Features category | Features | Feature extraction method | Rows size | Features size |
|---|---|---|---|---|
| | | ngram_range=(2,3) and max_features=30000) | | |
| Account features | Transaction | Account Behaviour | 11722 | 49 |
| **Hybrid features** | **Opcode, ABI code and transaction** | **Hybrid features** | **15530** | **17356** |

This scenario is based on the opcode data set (op), which has dimension J as follows:
$O_p = \{O_1, O_2, \ldots\ldots O_J\}$.
Meanwhile, the ABI code data set (ab) produces a feature size L after the vectorisation transformation process.
$A_b = \{A_1, A_2, \ldots\ldots A_L\}$.
The normal transaction dataset (Tx) for the contract yielded 49 features generated from the extraction process from etherscan.io.
$T_r = \{T_1, T_2, \ldots\ldots T_{49}\}$.
Therefore, the combination of the three components of the source code of this contract will produce a new data set (Co) as follows:
$C_o = O_p \cup A_b \cup T_r$
$C_o = \{O_1, O_2, \ldots\ldots O_J \ldots A_1, A_2, \ldots\ldots A_L \ldots T_1, T_2, \ldots\ldots T_{49}\}$.

### 3.4Features reduction

This study encompasses two primary phases of feature reduction for hybrid features, specifically opcode, ABI code, and transaction. These stages involve feature cleaning and feature filtering, including identifying and removing constant features and quasi-constant features (see *Table 5*). This strategy serves as a first way for screening features to reduce their number before proceeding with selecting the most relevant features using the proposed method and then creating the final anomaly detection model. The feature cleaning procedure is determined by the size of the feature dimension (15530, 17356), which is obtained by combining three feature categories: Opcode, ABI code, and transaction. The aforementioned procedure has effectively diminished the number of features by 10. This was achieved through several steps, including the replacement of missing data with median values, the elimination of features with zero variance, the removal of irrelevant features (namely, ERC20_most_rec_token_type and ERC20_most_sent token type), and the elimination of duplicate records.

The feature filter step employs the constant features approach in order to detect values that remain consistent across the entire row of the dataset. These features are deemed unnecessary and are subsequently removed, as they do not contribute to the predictive performance of the model for the target variable. Consequently, by eliminating the constant features, a total of 875 features have been decreased from the initial count of 17,346 to 16,471. The subsequent filtering technique pertains to quasi-constant features, which serve the purpose of identifying the predominant features within the dataset that do not contribute significantly to enhancing prediction accuracy. By employing this approach, a grand total of 201 features were effectively eliminated from the initial feature set, reducing its size from 16,471 to 16,270.

**Table 5** Basic features reduction for hybrid features

| Method | Before Features Reduction | After Features Reduction | Total Features Reduction |
|---|---|---|---|
| Features cleaning | 17356 | 17346 | **10** |
| Constant features | 17346 | 16471 | **875** |
| Quasi-constant features | 16471 | 16270 | **201** |

### 3.5Relevance features selection using SULOV

The feature size (201 features) produced from the feature reduction step explained in the previous section is still too large, and the selection of the most relevant features is necessary to develop a more optimal detection model. There are some common questions faced by data scientists about identifying the most important or irrelevant features, and this step becomes more challenging if the feature size is too large. Another question is to identify features that are highly correlated to the point of causing redundancy and to be sure of the results of features that are truly performing or overfitting.

Therefore, this study uses the SULOV for the selection of features of minimal size relevant to

maintaining optimal model performance. SULOV is a technique that inspires algorithms. MRMR became more popular after an article related to this algorithm was published by an Uber engineer in 2019 [60]. The MRMR algorithm targets minimal relevant features (not all relevant features) and optimal performance, while Boruta (also one of the best feature selection techniques) is capable of determining all relevant features. Therefore, usually, the size of features produced by SULOV is less than that of Boruta without removing the performance level at an optimal level. The SULOV technique consists of two distinct steps, referred to as phase 1 (SULOV) and phase 2 (recursive XGBoost (XGB)), as depicted in *Figure 12*. Phase 1 encompasses a series of procedures aimed at identifying pairs of features that exhibit a lack of correlation, as demonstrated by a high mutual information (MI) score. Pairs of characteristics exhibiting strong correlations were excluded due to their poor MI scores.

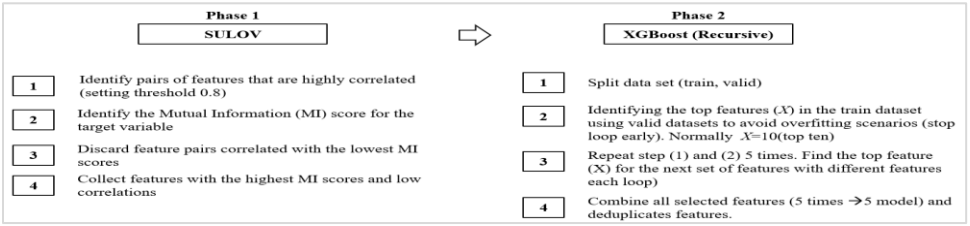


**Figure 12** SULOV method

In phase 2, the uncorrelated feature set obtained from phase 1 is utilised to partition the data set (train, valid) by selecting the ten most significant feature sets (i.e., those with the highest level of relevance). This process is iterated five times. The iterative procedure will generate a set of five XGB models, each exhibiting a distinct assortment of characteristics based on their respective levels of relevance. Therefore, the generation of the most minimum and optimal set of features is achieved by combining the top 10 characteristics from the 5 XGB models and subsequently conducting a de-duplication process (see *Figure 13*). The present study has undertaken an experimental investigation with the most recent feature set comprising 201 features. The SULOV technique was employed to generate the final feature set that is most minimal and optimal to the research objectives. The experimental findings demonstrate the efficacy of the SULOV technique in reducing the number of features from 201 to 44, resulting in a final set of features deemed the most optimal (refer to *Table 6*).

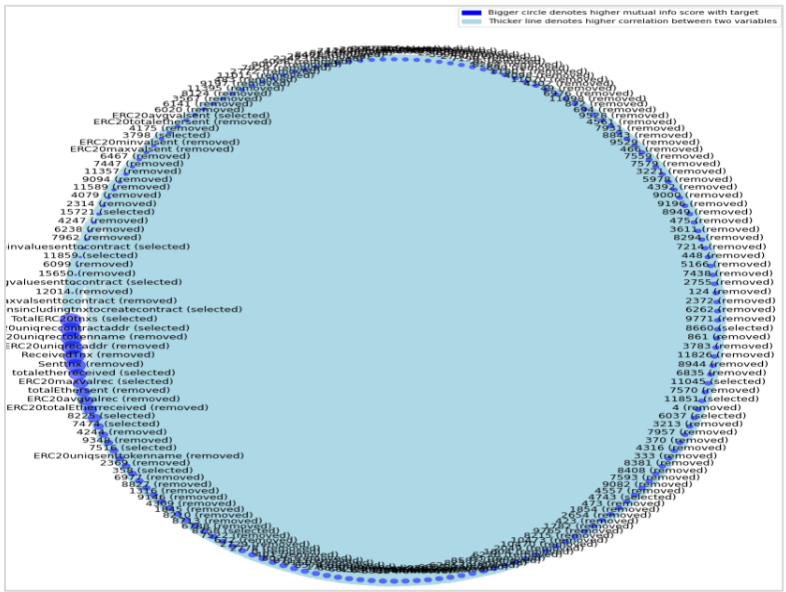


**Figure 13** Filtering uncorrelated features using the SULOV method

**Table 6** Relevance feature selection using the SULOV method

| Method | Before SULOV method | Feature reduction size | Relevance features |
|---|---|---|---|
| SULOV Method | 201 | 157 | 44 |

The aforementioned approach has yielded a total of 44 sets of very pertinent features while simultaneously eliminating 157 features that exhibit a significant association out of the initial pool of 201 features (refer to *Table 7*).

**Table 7** Final relevance features

| S. No. | Relevance features | S. No. | Relevance features |
|---|---|---|---|
| 1 | Total ERC20 tnxs | 23 | ERC20 uniq rec contract addr |
| 2 | Time Diff between first and last (Mins) | 24 | ERC20 min val rec |
| 3 | Number of Created Contracts | 25 | 12604 |
| 4 | Unique Received From Addresses | 26 | 11312 |
| 5 | Total Transactions (including tnx to create the contract) | 27 | ERC20 uniq sent addr |
| 6 | Total Ether Balance | 28 | max value received |
| 7 | Min val sent | 29 | 7516 |
| 8 | 7474 | 30 | 8136 |
| 9 | 8225 | 31 | 11851 |
| 10 | 358 | 32 | Avg min between sent tnx |
| 11 | Avg val received | 33 | min value received |
| 12 | Avg min between received tnx | 34 | 11790 |
| 13 | 8758 | 35 | ERC20 avg val sent |
| 14 | 3798 | 36 | 15721 |
| 15 | ERC20 max val rec | 37 | max val sent |
| 16 | 11045 | 38 | 11859 |
| 17 | 4743 | 39 | Unique Sent To Addresses |
| 18 | Total Ether Received | 40 | avg val sent |
| 19 | 8660 | 41 | min value sent to contract |
| 20 | 16920 | 42 | ERC20 uniq sent addr.1 |
| 21 | 6037 | 43 | ERC20 total Ether sent contract |
| 22 | 16295 | 44 | avg value sent to contract |

### 3.6Data preparation

The subsequent step is partitioning the dataset in accordance with a 70:30 ratio. The divide resulted in a training data set containing 10,871 rows, while the remaining 4,656 rows were allocated to the testing data set (see *Table 8*).

**Table 8** Training and testing set data

| Training set data (70%) | Testing set data (30%) |
|---|---|
| (10871, 17346) | (4659, 17346) |

The observation on class balance reveals an imbalance in the number of target classes (0,1), which, if left unaddressed, may lead to the development of a weak model characterised by overfitting. Consequently, the present study has implemented the SMOTE oversampling technique in order to generate synthetic data, achieving a balanced distribution of target classes. The aforementioned procedure results in an equal distribution of rows for both the training dataset, with 8,716 non-Ponzi and 8,717 Ponzi instances, as well as the test dataset, with 3,747 non-Ponzi and 3,748 Ponzi instances, as illustrated in *Table 9.*

**Table 9** SMOTE oversampling

| Set data | Before SMOTE | After SMOTE |
|---|---|---|
| Training | non-Ponzi: 8716<br>Ponzi: 2155 | non-Ponzi: 8716<br>Ponzi: 8717 |
| Testing | non-Ponzi: 3747<br>Ponzi: 912 | non-Ponzi: 3747<br>Ponzi: 3748 |

### 3.7Ensemble learning

ML classifiers exhibit diverse performance characteristics and limitations. The ensemble strategy is implemented to enhance the limitations of individual classifiers, resulting in a composite of more robust classifiers. Consequently, this study employs the most pertinent collection of attributes derived from the preceding procedure. An anomaly detection model was built using an ensemble model based on soft voting. Note that the model was trained on a labelled dataset consisting of two classes: 0 for non-Ponzi and 1 for Ponzi. The dataset included 44 sets of important features. The study of the contract source code, as depicted in *Figure 14*, served as the basis for this model. Consequently, this study has introduced an ensemble soft voting methodology, where each classifier engages in a class voting procedure with one another. The ultimate forecast generated is derived from a methodology known as a weighted voting technique. The ensemble model that has been developed utilises a selection strategy wherein two classifiers are chosen from a pool of five classifiers, namely XGB, extra-tree classifier (ETC), bagging classifier, gradient boosting (GB) classifier, and random forest (RF). This selection process results in six different combinations of classifiers, which are then used as the estimator parameters for the voting classifier. Hence, the accuracy values obtained from the performance evaluation of the six combinations of classifiers serving as estimators are organised in a ranked manner. The technique outlined in this study suggests that the final model selection is determined by identifying the model with the highest ranking.

The bagging classifier is a type of ensemble learning technique that is utilised for both classification and regression tasks. The classifier, known as Bootstrap Aggregating or Bagging, was initially proposed by Breiman in 1996 [61]. It effectively addresses the problem of overfitting in the model. The primary objective of this study is to examine the soft vote technique, as depicted in Equation 1, with k representing the total number of classes and $\check{f}_{bag}(x)$ denoting the prediction function.

$$\check{G}_{bag}(X)=argmax_k \check{f}_{bag}(X). \quad (1)$$

Meanwhile, the GB model generates a new tree based on the updated $\alpha$ and $G_k(x)$ values in the model (see Equation 2).

$$\alpha = argmax_{\alpha} \sum_{i-1}^{N} Loss(y_i, f_{k-1}(X_i) + \alpha G_k(X_i)) \quad (2)$$

XGB is derived from the most recent iteration of gradient boosting, as seen in Equation 3.

$$f_k(x)=f_{k-1}(x)+\propto G_k(x) \quad (3)$$

The ETC is an ML model that falls under the category of bagging, specifically bootstrap aggregation. It is built using RF and has the ability to mitigate changes within the dataset. Note that the subsequent explanation delineates the formulation of the ETC, which is derived from the base learner ($C_j(x)$) as depicted in Equation 4.

$$extraT(x)=argmin_{i\varepsilon\{0,1\}} \sum_{j=1}^{m} \emptyset\left(c_j(x) = i\right) \quad (4)$$

Therefore, the anomaly detection model employs the soft ensemble voting classifier technique to determine the ultimate classification as either normal (non-Ponzi) or abnormal (Ponzi).

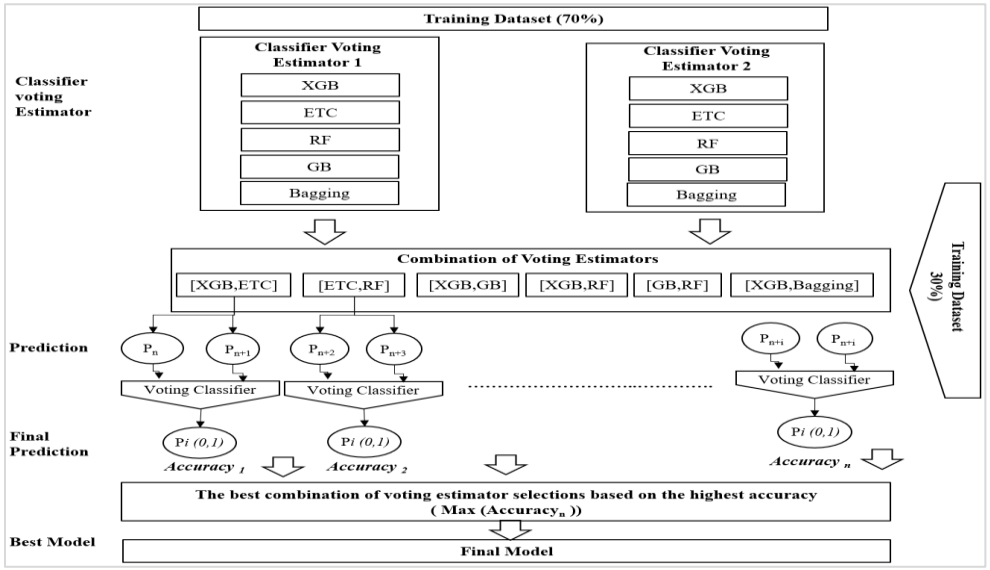


**Figure 14** Voting ensemble prediction strategy

## 4.Results

The experimental procedure employed in this investigation utilised a laptop computer equipped with particular technical parameters, including 32 gigabytes of random access memory (RAM) powered by an Intel i7-7700 processor operating at a frequency of 2.8 gigahertz and requiring a minimum of 10 gigabytes of available disc space. In order to facilitate uninterrupted data crawling from etherscan.io using the API, it is imperative to have a reliable and high-speed internet connection. The dataset preparation for the three source code components, namely opcode, ABI code, and account transaction, is conducted using data obtained from the Etherscan API. This dataset is derived from the Ponzi dataset (account contract) as described in the referenced study [38]. Moreover, the present work commences with the initial stage of dataset preparation and proceeds towards constructing an anomaly detection model utilising the Python and Jupyter programming languages. The evaluation of the suggested model is conducted thoroughly, utilising eight assessment metrics: precision, recall, F1-score, accuracy, true positive (TP), false positive (FP), true negative (TN), and false negative (FN). The specifics of this measurement metric are displayed in *Table 10.*

**Table 10** Metric measurement for performance analysis

| Metric measurement | Definition | Remarks |
|---|---|---|
| **True positive (TP)** | The observed class is "Ponzi," and the anticipated class is also "Ponzi." | A high value that serves as an indicator of strong performance. |
| **False positive (FP)** | Actual is "non-Ponzi" but predicted "Ponzi" class | Lower values are indicative of excellence in performance. |
| **True negative (TN)** | Actual is "non-Ponzi" and predicted 'non-Ponzi' class | A high value that serves as an indicator of strong performance. |
| **False negative (FN)** | Actual 'Ponzi' but predicted 'non-Ponzi' class | Lower values are indicative of excellence in performance. |
| **Precision (P)** | TP /(TP+FP) | A high level of precision is indicative of a model that generates a low number of FP. |
| **Recall (R)** | TP /(TP+FN) | A low recall value suggests that the model produces a significant number of FN. |
| **F1-Score** | 2.P.R /(P+R) | A high score is indicative of a strong performance. |
| **Accuracy** | (TP+TN) /(TP+TN+FP+FN) | A high value that serves as an indicator of a strong performance. |

### 4.1Evaluation of proposed features

This section provides an examination of the efficacy of the planned study in generating the most pertinent features. The performance of the anomaly detection level, as determined by the final model developed, is contingent upon the careful selection of optimal features. Hence, an evaluation is conducted to assess the efficacy of using the most pertinent features by comparing them to the complete set of features, as exhibited in *Table 11*. This evaluation is based on their accuracy and classification time performance. The findings of this experiment indicate that the utilisation of pertinent features, consisting of 44 features, resulted in an accuracy rate of 92.99%. In comparison, including all features, totalling 17,346, yielded a slightly higher accuracy rate of 93.07%. The utilisation of pertinent features has exhibited a marginal decline, namely a loss of 0.08% compared to the achievement of 93.07% accuracy using all available features. This finding demonstrates that employing a set of 44 pertinent features yields nearly equivalent performance to utilising the complete set of features for anomaly identification. Other than that, the proposal to employ a subset of pertinent features holds greater importance in comparison to utilising the full feature set to analyse the classification time, encompassing both training and testing phases.

Hence, it was observed that using pertinent features significantly reduces the classification time (comprising training and testing time) compared to the classification time required when using the complete feature set, as measured in milliseconds (ms). The use of the pertinent feature set necessitates a training duration of 3.87 ms, significantly smaller than the 70.04 ms required when using the complete feature set. The aforementioned situation is comparable to the testing time of 0.26 ms that arises from utilising a specific set of pertinent features. Note that this testing time is notably reduced compared to the 1.83 ms that results from employing

the complete set of features. The experiment's findings indicate that the selected set of features effectively detects anomalies by achieving high levels of accuracy. Additionally, this feature set optimises the time required for classification, resulting in time savings, reduced human resource utilisation, and lower processing costs during the training and testing phases.

**Table 11** Performance comparison: full features and proposed features

| Features approach | Features size | Accuracy (%) | Training time (ms) | Testing time (ms) |
|---|---|---|---|---|
| Full Features | 17346 | 93.07% | 70.04 | 1.83 |
| Proposed Features | 44 | 92.99% | 3.87 | 0.26 |

### 4.2Evaluation of proposed ensemble model

The data set generated using the SULOV method serves as the input parameter for training the anomaly detection model employing the ensemble voting strategy. The building of the final model is determined by picking the optimal estimator voting combination from a set of six combinations: (XGB, ETC), (XGB, Bagging), (ETC, RF), (XGB, GB), (XGB, RF), and (GB, RF). Consequently, these combinations are rated based on their accuracy values, as provided in *Table 12*.

**Table 12** Performance analysis of ensemble voting estimator combinations

| S. No. | Voting estimator | Precision | F1-score | Recall | Accuracy | TPR | FPR | False negative rate (FNR) | TNR | Ranking |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | XGB, ETC | 97.35 | 92.66 | 88.39 | 92.99 | 0.88 | 0.02 | 0.12 | 0.98 | 1 |
| 2 | XGB, Bagging | 96.78 | 92.31 | 88.23 | 92.65 | 0.88 | 0.03 | 0.12 | 0.97 | 2 |
| 3 | ETC, RF | 97.32 | 91.36 | 86.1 | 91.86 | 0.86 | 0.02 | 0.14 | 0.98 | 4 |
| 4 | XGB, GB | 96.19 | 91.99 | 88.15 | 92.33 | 0.88 | 0.03 | 0.12 | 0.97 | 3 |
| 5 | XGB, RF | 92.98 | 89.27 | 85.86 | 89.69 | 0.86 | 0.06 | 0.14 | 0.94 | 5 |
| 6 | GB, RF | 96.01 | 91.48 | 87.35 | 91.86 | 0.87 | 0.04 | 0.13 | 0.96 | 4 |

The analysis of the estimator voting combination reveals that the combination of XGB and ETC exhibits the most favourable performance, achieving the greatest accuracy score of 92.99%. The combination of (XGB, Bagging) holds the second position, followed by (XGB, GB) in third place. Meanwhile, (ETC, RF) and (GB, RF) are tied for fourth place, while the combination of (XGB, RF) occupies the last spot. Based on the evaluation results, it can be concluded that the combination of XGB and ETC exhibits the highest performance among all the estimator voting combinations. This combination is ranked first and is recommended to be chosen as the final model. Moreover, the efficacy of the proposed model in detecting anomalies is evaluated by assessing its performance against different individual classifiers. This evaluation is conducted utilising eight key metrics for assessment, namely precision, recall, f1-score, accuracy, true positive rate (TPR), false positive rate (FPR), FN rate (FNR), and true negative rate (TNR). These metrics are presented in *Table 13*.

**Table 13** Comparative analysis of ensemble model performance with individual classifiers

| S. No. | Approach | Precision | F1-Score | Recall | Accuracy | TPR | FPR | FNR | TNR |
|---|---|---|---|---|---|---|---|---|---|
| 1 | RF | 96.99 | 91.62 | 86.82 | 92.06 | 0.87 | 0.03 | 0.13 | 0.97 |
| 2 | k-nearest neighbour algorithm (KNN) | 81.45 | 78.96 | 76.62 | 79.58 | 0.77 | 0.17 | 0.23 | 0.83 |
| 3 | DT | 90.7 | 71.38 | 58.85 | 76.41 | 0.59 | 0.06 | 0.41 | 0.94 |
| 4 | LDA | 61.58 | 70.82 | 83.32 | 65.67 | 0.83 | 0.52 | 0.17 | 0.48 |
| 5 | SGD | 59.96 | 23.93 | 14.95 | 52.48 | 0.15 | 0.1 | 0.85 | 0.9 |
| 6 | **Proposed Model** | **97.35** | **92.66** | **88.39** | **92.99** | **0.88** | **0.02** | **0.12** | **0.98** |

The observation results indicate that the proposed model has demonstrated superior performance compared to the RF classifier. This conclusion is based on the evaluation of eight measurement metrics, where the proposed model achieved an accuracy value of 92.99%, surpassing the RF classifier's accuracy of 92.06%. These findings are visually represented in *Figure 15*, highlighting the

suggested model's position as the highest-performing model. Four classifiers, namely the KNN, decision tree (DT), linear discriminant analysis (LDA), and stochastic gradient descent (SGD) exhibited an accuracy value below 80%. The analytical findings demonstrate that the suggested model effectively achieves an optimal level of anomaly detection.

The efficacy of the proposed model is further assessed by considering the rate of misclassification errors, as depicted in *Figure 16*. The analysis findings indicate that the suggested model has yielded the lowest FPR of 0.02 compared to alternative classifiers. In comparison to other classifiers, the proposed model demonstrates the lowest FNR value of 0.12. The misclassification error rate, namely the FPR and FNR, indicates the model's strength and ability to effectively detect anomalies. Hence, a lower misclassification error rate suggests that the created model is stronger and more optimal in its anomaly detection capabilities.

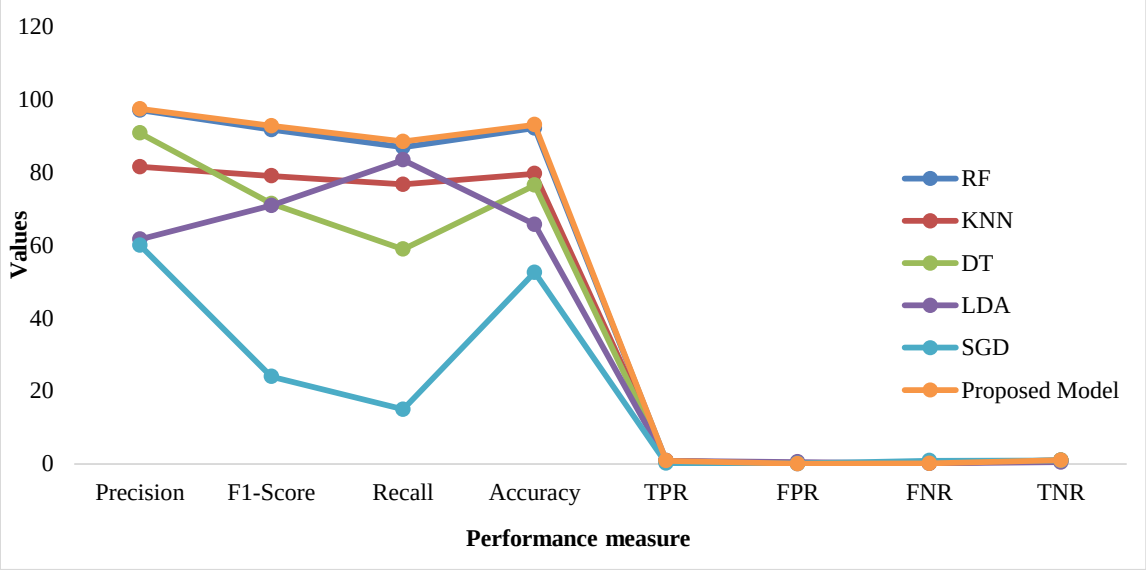


**Figure 15** Analysis of model performance based on metric measurement

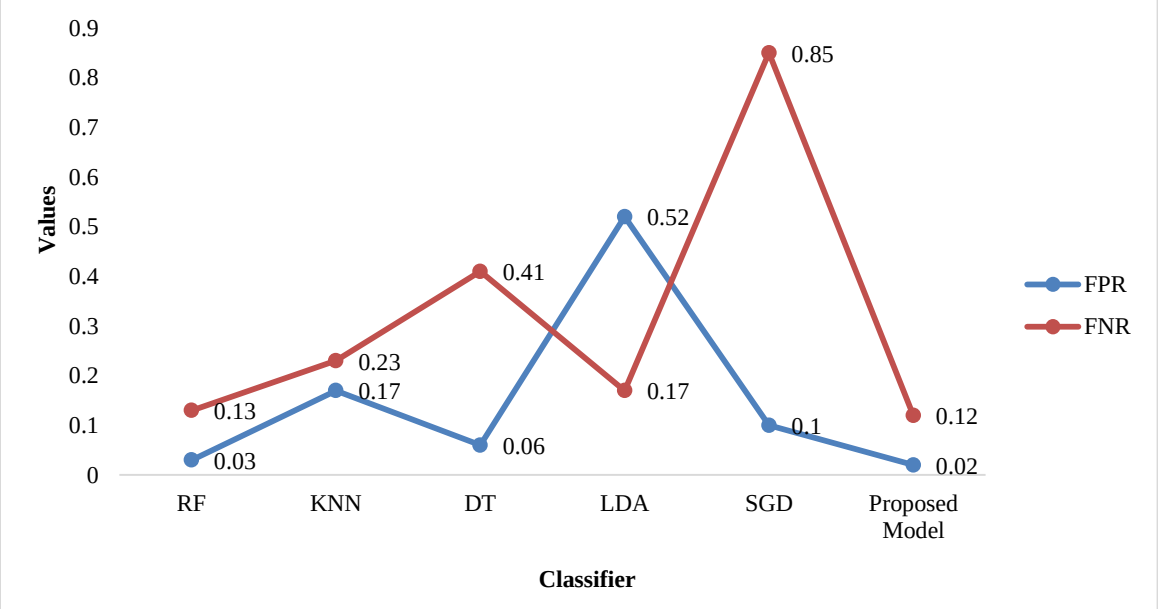


**Figure 16** Misclassified error rate analyst (FPR, FNR)

### 4.3Comparison with existing works

The efficacy of the suggested model is assessed by conducting a comparative analysis of the study's findings with those of prior studies utilising the most relevant features in the dataset, as depicted in *Table 14*. The present comparative analysis reveals that the proposed model has achieved an accuracy rate of 92.99%, exhibiting a notable improvement of 3.32% compared to the 89.67% reported in the prior study. Furthermore, the suggested model exhibits the highest recall value, achieving a rate of 92.66%, surpassing the 81.48% achieved by the prior model. However, a marginal decline was observed in the proposed model, achieving an accuracy of 0.09% and an F1-score of 0.35% compared to the preceding investigation. Nevertheless, in a comprehensive analysis, the proposed model demonstrated superior performance compared to the prior work, utilising the most relevant features in the dataset and demonstrating the potential for enhanced anomaly detection capabilities.

**Table 14** Comparative analysis of the performance of the proposed model with previous work

| References | Feature Reduction Approach | Algorithm | Dataset | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|---|---|---|
| [38] | Feature Importance XGB | Ensemble | Ponzi (1904) | 97.44% | 81.48% | 88.74% | 89.67% |
| **Proposed Model** | **Basic features reduction and relevance features using the SULOV Method** | **Ensemble** | **Ponzi (1904)** | **97.35%** | **92.66%** | **88.39%** | **92.99%** |

### 4.4Comparison with the boruta method

The competitiveness aspect of the proposed study was tested using the Boruta feature selection technique. The Boruta technique was introduced by two researchers from the University of Warsaw (Witold and Miron) based on RF [62]. Therefore, an experiment was conducted by replacing the SULOV technique with Boruta, as displayed in *Figure 17*.

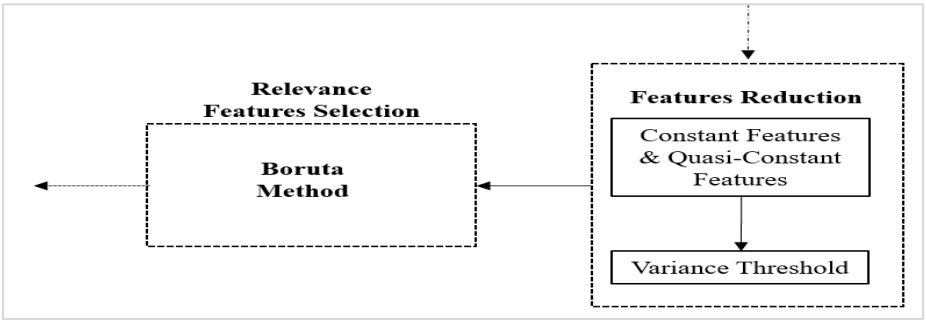


**Figure 17** Relevance features selection with Boruta

The experimental results suggest that the Boruta technique used together with the proposed model produces an accuracy value of 92.89%, which is slightly lower compared to the 92.99% produced by the SULOV technique (see *Table 15*). The number of relevant features produced by Boruta is much higher (175 features) compared to SULOV (44 features). The size of 175 features generated by Boruta has caused the classification time (training time) to take a long time (21.16 ms) compared to SULOV (3.87 ms). This scenario demonstrates that the SULOV technique produces minimal and optimal features more and requires a shorter classification time than the Boruta technique. The result is a positive impact from the point of view of blockchain network anomaly detection performance through contract source code analysis.

**Table 15** Comparative analysis of the performance of the proposed model with Boruta Method

| Feature reduction approach | Algorithm | Relevance features | Total features reduction | Precision | Recall | F1-Score | Accuracy | Training Time (ms) | Testing Time (ms) |
|---|---|---|---|---|---|---|---|---|---|
| Proposed model with Boruta Method | Ensemble | 175 | 26 | 97.04% | 88.47 | 92.56 % | 92.89% | 21.16 | 0.15 |
| **Proposed model with SULOV Method** | **Ensemble** | **44** | **157** | **97.35%** | **92.66 %** | **88.39 %** | **92.99%** | **3.87** | **0.26** |

## 5. Discussion

This analysis utilises a dataset generated by [38], comprising 1904 account contracts categorised as either ponzi (305) or non-ponzi (1,599). Most prior research endeavours continue to encounter the obstacle of ascertaining the most pertinent attributes for constructing anomaly detection models. The careful selection of appropriate characteristics significantly impacts the overall performance of the subsequent model that is to be constructed. In addition, the analysis of the contract source code now only examines specific aspects in isolation rather than considering a combination of features. However, it is important to note that the contract source code consists of three distinct components, namely opcode, ABI code, and contact transaction, all of which possess the potential for analysis. Furthermore, this work is centred on examining three source code components as hybrid ensemble characteristics to detect anomalies in smart contracts. The source code exhibits semantic and text-based formatting. Consequently, the procedure of vectorisation is employed to convert it into a numerical dataset, which can then be utilised for training the ML model. The resultant data set obtained through the process of feature vectorisation is merged to construct a hybrid ensemble data set including three distinct combinations of contract source code feature components.

Hence, the primary objective of this study is to ascertain the optimal collection of features that should be recovered from a vast dataset comprising over 10,000 characteristics subsequent to amalgamating the three constituent elements of the source code. The attainment of this objective is accomplished by employing fundamental strategies for reducing features, such as constant-quasi and variance, and generating the most pertinent sets of features using the MRMR-based SULOV approach. The SULOV technique comprises two sequential stages. The initial stage involves identifying and eliminating highly correlated features, resulting in a refined set of uncorrelated features. Moreover, the subsequent stage employs the XGB Feature Importance method to generate a final set of relevant features, prioritised or ranked according to importance. The research methodology effectively decreases the number of features from 17,346 (all features) to the 44 most significant features. The utilisation of pertinent characteristics has a discernible effect on enhancing performance compared to performance achieved by employing the complete collection of features.

The subsequent step involves constructing an ensemble model utilising soft voting, wherein input parameters (pertinent feature sets) are employed to train the model. This study employs a selection technique that involves six combinations of two classifiers functioning as voting estimators. Note that the final model is determined by selecting the combination with the highest-ranking order, which is determined based on the accuracy value. The model under consideration exhibited notable achievement in achieving the best accuracy value (92.99%) compared to a range of separate classifiers. In addition, the proposed model exhibited superior performance compared to the study [38], achieving an accuracy rate of 89.67% while utilising the identical dataset. The investigation of error misclassification rates, namely the FPR and FNR, indicates a comparatively low value compared to individual classifiers' performance. In general, the anomaly detection model generated in this study successfully achieves an ideal detection level, aided by utilising the most relevant features for model performance optimisation. The study's performance was evaluated by substituting the SULOV approach with Boruta inside the framework provided in this study. The experimental findings indicate that the SULOV technique effectively generates 44 pertinent characteristics while exhibiting superior accuracy performance compared to the Boruta technique, which yields 175 features.

Hence, in order to apply the hybrid feature technique for detecting abnormalities in the blockchain network through the analysis of contract source code (including opcode, ABI code, and transaction), it is imperative to have access to a comprehensive labelled data source. This is essential for developing an effective detection model. In essence, source code and ABI code are distinct formats derived from a common origin. The source code encompasses several elements, such as a structured format, semantic meaning, programme logic, operands, conditional expressions, and remark tags, among others. Nevertheless, utilizing this source code for anomaly analysis proves inefficient due to its lack of coherent and comprehensible content, rendering it devoid of any discernible significance for human comprehension. The ABI code, similar to the API, is comprised of programme code that is designed to be easily comprehensible and legible to humans. Consequently, the ABI code is better suited for doing anomaly analysis in comparison to the source code.

However, a notable limitation of our study is the difficulty in obtaining raw data sets of ABI codes from the Etherscan blockchain explorer using the API. This phenomenon arises due to the Ethereum architecture's utilisation of the EVM for processing the compiled source code of the contract, as well as its acceptance of unpublished source code sourced from the Ethereum network. Consequently, the proportion of smart contracts deployed on the Ethereum network and providing source code is quite small. Nevertheless, the accessibility of the published source code on Ethereum is not immediate, as it necessitates users to undertake a verification process prior to its appearance on the network. The difficulty in acquiring the raw data set ABI code has had repercussions on the thoroughness of preparing other data sets, such as opcodes and transactions. This is due to their reliance on the same contract address, which introduces the potential for an uneven distribution of data sets. The utilisation of a limited dataset sample is prone to yield a model that exhibits elevated performance or overfitting. However, it is important to note that this model may not be ideal in actuality. The deployment of this model inside a real-world implementation ecosystem may have implications for the accuracy and effectiveness of anomaly detection capabilities. Nevertheless, several research studies have employed solidity decompiler tools, such as pyevmasm [53] and evmdis, to undertake the reverse engineering of opcodes into source code. However, it is important to note that these endeavours are not without their limitations, as there exists a potential danger of failure during the conversion process. This failure may occur when the resulting source code format does not adhere to the original standard. Therefore, it is important to conduct comprehensive research or utilise highly reliable techniques in order to effectively convert the opcode format to the source code with greater accuracy than the original source code.

The limitations of the study are also related to the architecture of Ethereum, which is divided into two chains (on-chain and off-chain). The term "on-chain data" pertains to data that is produced within the blockchain network itself. This encompasses many elements, such as accounts, transactions, and contracts, including those according to the ERC-20 and ECR-721 standards. Off-chain data pertains to the transmission of data originating from external sources, which is then routed through the API channel to the smart contract within the blockchain network. This encompasses a wide range of data types, including but not limited to meteorological data, supply chain data, financial data, and any other data accessible through an API. In the practical implementation of DApp, a hybrid approach is often employed, including both off-chain and on-chain operations.Hence, the primary objective of this research is to concentrate on the identification of anomalies exclusively within on-chain data, with a specific emphasis on leveraging contract data analysis techniques. One of the data sets utilised in anomaly analysis of the contract source code comprises account transactions associated with the contract address. Nevertheless, the study has certain drawbacks as it primarily concentrates on the utilisation of external transaction data sets, specifically regular transactions, while neglecting to incorporate internal transaction data sets. The ledger on-chain captures external transaction data, mostly involving EOA associated with hacker groups who exploit smart contract manipulation. Internal transactions are initiated when a corresponding entry is made in the external transaction ledger, as viewed from an operational perspective. Nevertheless, the ledger does not document internal transactions that occur off-chain and impact the Ether balance. Consequently, investigating anomalies using the internal transaction dataset focuses on examining patterns of money transfers, balances, transactions, timestamps, and amounts. Therefore, for a more comprehensive examination of blockchain anomaly detection, it is imperative to consider both types of transactions in the investigation of this proposal. Nevertheless, it is essential to acknowledge that abnormalities or incursions can also manifest beyond the confines of the blockchain network, commonly referred to as off-chain occurrences. To enhance the comprehensiveness of this study, it would be advantageous to incorporate off-chain data sets.

A complete list of abbreviations is summarised in *Appendix I.*

## 6.Conclusion and future work

Smart contracts play a vital role in the development of DApp on blockchain networks. The emergence of blockchain 3.0 has sparked a transformative shift in the realm of DApp, which emerged subsequent to the advent of smart contracts during the era of blockchain 2.0. The utilisation of blockchain technology has led to the transformation of Bitcoin transfer transactions into DApp, which offers enhanced security and reliability as an application platform. Nevertheless, smart contracts have emerged as a breeding ground for hackers involved in deceptive practices, including but not limited to ponzi

scams, honeypots, HYIP, phishing, and similar illicit activities. The impact is experienced by individuals or investors who have incurred substantial financial losses due to deceptive practices in bitcoin investment endeavours. Hence, implementing a robust fraud detection system that facilitates the timely identification of irregularities is of utmost significance. Therefore, integrating science data adaption solutions, AI, ML, and blockchain technology can potentially enhance the efficiency of anomaly detection systems. Applying ML techniques to analyse extensive datasets derived from the Ethereum network poses significant challenges. Performing manual analysis by individually examining transactions on Etherscan.io is unfeasible due to the significant time, financial, and human resource investments it necessitates, as well as the elevated risk of erroneous data analysis.

Therefore, this study has explored the method of analysing the three components of the contract source code to detect anomalies in the contract. The contract source code component combines opcode, ABI code, and contract transactions to produce a hybrid feature set. Combining these three source code feature components has produced a feature size exceeding 10 thousand features after going through the feature vectorisation transformation process based on TF-IDF and N-Gram methods to produce vector number values to facilitate ML model processing. Since the dimensions of the features are too large, this study has proposed two main processes, namely the basic feature reduction method (constant-quasi and variance) and the SULOV method, to finalise the most relevant set of features. As a result, the 44 most relevant feature sets are generated from the full feature set (17,346) without lowering the performance level of the model. This set of relevant features becomes the input to train an ensemble model based on ensemble soft voting. The strategy in this study is the selection of combinations of classifiers that act as voting estimators to be selected as the final model based on the highest ranking (highest accuracy value). Correspondingly, the model produced successfully obtained the highest accuracy value (92.99%) compared to various other classifiers, and its performance was better than the results of other studies using the same data set. The analysis of blockchain data presents a significant problem due to the dynamic nature of the blockchain environment, which frequently undergoes modifications to enhance security measures, upgrade software versions, and other similar objectives. Consequently, modifying features inside the blockchain network leads to revising existing ML models, as the metadata set or features have undergone changes by adding or removing features. In addition, the hybrid technique, which involves integrating multiple source code feature components, creates an extensive feature set. Hence, this phenomenon underscores the necessity for advanced processing devices, including high-capacity RAM, storage, and CPUs.

Hence, one of the prospective areas of research in the future involves exploring alternative NLP techniques (FastText, Word2Vec, uni-gram, bi-gram, doc2vec, to name a few), apart from TF-IDF and N-Gram, for text-based processing in order to convert source code based on semantic code (text-based). A necessity arises to conduct a comparative examination of various strategies due to the fact that varying approaches yield disparate outcomes. Therefore, the optimal method choice will enhance the efficacy of the anomaly detection model by scrutinising the source code of smart contracts on Ethereum.

## Acknowledgment

The Research Management and Innovation Centre (RMIC), University of Sultan Zainal Abidin, funded this study.

## Conflicts of interest

The authors have no conflicts of interest to declare.

## Author's contribution statement

**Sabri Hisham:** Models and method selection, building a framework, conducting experiments, analysis of experimental results, draft writing, checking for plagiarism, and proofreading**. Mokhairi Makhtar:** Supervision, give an opinion, input on draught revision, and final revision. **Azwa Abdul Aziz:** Supervision, exchange of sample manuscripts, and draught evaluation comments.

## References

[1] Hu T, Liu X, Chen T, Zhang X, Huang X, Niu W, et al. Transaction-based classification and detection approach for Ethereum smart contract. Information Processing & Management. 2021; 58(2):102462.

[2] Chen W, Guo X, Chen Z, Zheng Z, Lu Y, Li Y. Honeypot contract risk warning on Ethereum smart contracts. In international conference on joint cloud computing 2020 (pp. 1-8). IEEE.

[3] Bitcoin NS. Bitcoin: a peer-to-peer electronic cash system. 2008.

[4] Wu J, Yuan Q, Lin D, You W, Chen W, Chen C, et al. Who are the phishers? phishing scam detection on Ethereum via network embedding. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2020; 52(2):1156-66.

[5] Zheng Z, Chen W, Zhong Z, Chen Z, Lu Y. Securing the Ethereum from smart ponzi schemes: identification

using static features. ACM Transactions on Software Engineering and Methodology. 2023; 32(5):1-28.
[6] Deepa N, Pham QV, Nguyen DC, Bhattacharya S, Prabadevi B, Gadekallu TR, et al. A survey on blockchain for big data: approaches, opportunities, and future directions. Future Generation Computer Systems. 2022; 131:209-26.
[7] Huang J, He D, Obaidat MS, Vijayakumar P, Luo M, Choo KK. The application of the blockchain technology in voting systems: a review. ACM Computing Surveys (CSUR). 2021; 54(3):1-28.
[8] Christidis K, Devetsikiotis M. Blockchains and smart contracts for the internet of things. IEEE Access. 2016; 4:2292-303.
[9] Berdik D, Otoum S, Schmidt N, Porter D, Jararweh Y. A survey on blockchain for information systems management and security. Information Processing & Management. 2021; 58(1):102397.
[10] Belchior R, Vasconcelos A, Guerreiro S, Correia M. A survey on blockchain interoperability: past, present, and future trends. ACM Computing Surveys (CSUR). 2021; 54(8):1-41.
[11] Qin K, Zhou L, Gervais A. Quantifying blockchain extractable value: how dark is the forest? In symposium on security and privacy (SP) 2022 (pp. 198-214). IEEE.
[12] Rahouti M, Xiong K, Ghani N. Bitcoin concepts, threats, and machine-learning security solutions. IEEE Access. 2018; 6:67189-205.
[13] Liu L, Tsai WT, Bhuiyan MZ, Peng H, Liu M. Blockchain-enabled fraud discovery through abnormal smart contract detection on Ethereum. Future Generation Computer Systems. 2022; 128:158-66.
[14] Wood G. Ethereum: a secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper. 2014; 151(2014):1-32.
[15] Szabo N. Formalizing and securing relationships on public networks. First Monday. 1997; 2(9).
[16] Buterin V. Ethereum white paper: a next generation smart contract & decentralized application platform. First Version. 2014; 53.
[17] Buterin V. A next-generation smart contract and decentralized application platform. White Paper. 2014; 3(37):1-27.
[18] Cheng Z, Hou X, Li R, Zhou Y, Luo X, Li J, et al. Towards a first step to understand the cryptocurrency stealing attack on Ethereum. In international symposium on research in attacks, intrusions and defenses (RAID 2019) 2019 (pp. 47-60). USENIX Association.
[19] Sallam A, Rassem T, Abdu H, Abdulkareem H, Saif N, Abdullah S. Fraudulent account detection in the Ethereum's network using various machine learning techniques. International Journal of Software Engineering and Computer Systems. 2022; 8(2):43-50.
[20] Camino R, Torres CF, Baden M, State R. A data science approach for detecting honeypots in Ethereum. In international conference on blockchain and cryptocurrency (ICBC) 2020 (pp. 1-9). IEEE.
[21] Hu B, Zhou C, Tian YC, Qin Y, Junping X. A collaborative intrusion detection approach using blockchain for multimicrogrid systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2019; 49(8):1720-30.
[22] Preuveneers D, Rimmer V, Tsingenopoulos I, Spooren J, Joosen W, Ilie-zudor E. Chained anomaly detection models for federated learning: an intrusion detection case study. Applied Sciences. 2018; 8(12):1-21.
[23] Nguyen TD, Pham LH, Sun J, Lin Y, Minh QT. Sfuzz: an efficient adaptive fuzzer for solidity smart contracts. In proceedings of the ACM/IEEE 42nd international conference on software engineering 2020 (pp. 778-88).
[24] Fan S, Fu S, Xu H, Zhu C. Expose your mask: smart ponzi schemes detection on blockchain. In international joint conference on neural networks (IJCNN) 2020 (pp. 1-7). IEEE.
[25] Vasek M, Moore T. Analyzing the bitcoin ponzi scheme ecosystem. In financial cryptography and data security: FC 2018 international workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao 2019 (pp. 101-12). Springer Berlin Heidelberg.
[26] Bartoletti M, Carta S, Cimoli T, Saia R. Dissecting ponzi schemes on Ethereum: identification, analysis, and impact. Future Generation Computer Systems. 2020; 102:259-77.
[27] Zhou Y, Kumar D, Bakshi S, Mason J, Miller A, Bailey M. Erays: reverse engineering Ethereum's opaque smart contracts. In 27th USENIX security symposium (USENIX Security 18) 2018 (pp. 1371-85).
[28] Tug S, Meng W, Wang Y. CBSigIDS: towards collaborative blockchained signature-based intrusion detection. In international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData) 2018 (pp. 1228-35). IEEE.
[29] Wang W, Song J, Xu G, Li Y, Wang H, Su C. Contractward: automated vulnerability detection models for Ethereum smart contracts. IEEE Transactions on Network Science and Engineering. 2020; 8(2):1133-44.
[30] Zhang L, Wang J, Wang W, Jin Z, Zhao C, Cai Z, et al. A novel smart contract vulnerability detection method based on information graph and ensemble learning. Sensors. 2022; 22(9):1-25.
[31] Chen W, Zheng Z, Cui J, Ngai E, Zheng P, Zhou Y. Detecting ponzi schemes on Ethereum: towards healthier blockchain technology. In proceedings of the 2018 world wide web conference 2018 (pp. 1409-18).
[32] Chen W, Zheng Z, Ngai EC, Zheng P, Zhou Y. Exploiting blockchain data to detect smart ponzi schemes on Ethereum. IEEE Access. 2019; 7:37575-86.
[33] Jung E, Le TM, Gehani A, Ge Y. Data mining-based Ethereum fraud detection. In international conference on blockchain (Blockchain) 2019 (pp. 266-73). IEEE.

[34] Yan Z, Susilo W, Bertino E, Zhang J, Yang LT. AI-driven data security and privacy. Journal of Network and Computer Applications. 2020; 172:102842.
[35] Peng H, Li J, Wang S, Wang L, Gong Q, Yang R, et al. Hierarchical taxonomy-aware and attentional graph capsule RCNNs for large-scale multi-label text classification. IEEE Transactions on Knowledge and Data Engineering. 2019; 33(6):2505-19.
[36] Pham T, Lee S. Anomaly detection in bitcoin network using unsupervised learning methods. arXiv preprint arXiv:1611.03941. 2016.
[37] Bogner A. Seeing is understanding: anomaly detection in blockchains with visualized features. In proceedings of the international joint conference on pervasive and ubiquitous computing and proceedings of the international symposium on wearable computers 2017 (pp. 5-8). ACM.
[38] Aljofey A, Rasool A, Jiang Q, Qu Q. A feature-based robust method for abnormal contracts detection in Ethereum blockchain. Electronics. 2022; 11(18):1-24.
[39] Chen W, Li X, Sui Y, He N, Wang H, Wu L, et al. Sadponzi: detecting and characterizing ponzi schemes in Ethereum smart contracts. Proceedings of the ACM on Measurement and Analysis of Computing Systems. 2021; 5(2):1-30.
[40] Kamišalić A, Kramberger R, Fister JI. Synergy of blockchain technology and data mining techniques for anomaly detection. Applied Sciences. 2021; 11(17):1-37.
[41] Kumar N, Singh A, Handa A, Shukla SK. Detecting malicious accounts on the Ethereum blockchain with supervised learning. In cyber security cryptography and machine learning: fourth international symposium, Be'er Sheva, Israel, proceedings 2020 (pp. 94-109). Springer International Publishing.
[42] Awang MK, Makhtar M, Udin N, Mansor NF. Improving customer churn classification with ensemble stacking method. International Journal of Advanced Computer Science and Applications. 2021; 12(11):277-85.
[43] Awang MK, Makhtar M, Mamat AR. Ensemble selection and combination based on cost function for UCI datasets. Journal of Theoretical and Applied Information Technology. 2021; 99(16):4015-25.
[44] Hisham S, Makhtar M, Aziz AA. Combining multiple classifiers using ensemble method for anomaly detection in blockchain networks: a comprehensive review. International Journal of Advanced Computer Science and Applications. 2022; 13(8):404-22.
[45] Baba NM, Makhtar M, Fadzli SA, Awang MK. Current issues in ensemble methods and its applications. Journal of Theoretical & Applied Information Technology. 2015; 81(2):266-76.
[46] Wang L, Cheng H, Zheng Z, Yang A, Zhu X. Ponzi scheme detection via oversampling-based long short-term memory for smart contracts. Knowledge-Based Systems. 2021; 228:107312.
[47] Lu P, Cai L, Yin K. SourceP: smart ponzi schemes detection on Ethereum using pre-training model with data flow. arXiv preprint arXiv:2306.01665. 2023.
[48] Zhang L, Chen W, Wang W, Jin Z, Zhao C, Cai Z, et al. Cbgru: a detection method of smart contract vulnerability based on a hybrid model. Sensors. 2022; 22(9):1-24.
[49] Durieux T, Ferreira JF, Abreu R, Cruz P. Empirical review of automated analysis tools on 47,587 Ethereum smart contracts. In proceedings of the 42nd international conference on software engineering 2020 (pp. 530-41). ACM/IEEE.
[50] Grieco G, Song W, Cygan A, Feist J, Groce A. Echidna: effective, usable, and fast fuzzing for smart contracts. In proceedings of the 29th SIGSOFT international symposium on software testing and analysis 2020 (pp. 557-60). ACM.
[51] Huang J, Zhou K, Xiong A, Li D. Smart contract vulnerability detection model based on multi-task learning. Sensors. 2022; 22(5):1-24.
[52] Ferreira JF, Cruz P, Durieux T, Abreu R. Smartbugs: a framework to analyze solidity smart contracts. In proceedings of the 35th international conference on automated software engineering 2020 (pp. 1349-52). IEEE/ACM.
[53] Fan S, Fu S, Xu H, Cheng X. Al-SPSD: anti-leakage smart ponzi schemes detection in blockchain. Information Processing & Management. 2021; 58(4):102587.
[54] Chen J, Xia X, Lo D, Grundy J, Luo X, Chen T. Defectchecker: automated smart contract defect detection by analyzing EVM bytecode. IEEE Transactions on Software Engineering. 2021; 48(7):2189-207.
[55] Vivar AL, Castedo AT, Orozco AL, Villalba LJ. An analysis of smart contracts security threats alongside existing solutions. Entropy. 2020; 22(2):1-29.
[56] Torres CF, Steichen M. The art of the scam: demystifying honeypots in Ethereum smart contracts. In 28th USENIX security symposium (USENIX Security 19) 2019 (pp. 1591-607).
[57] Sun X, Lin X, Liao Z. An ABI-based classification approach for Ethereum smart contracts. In international conference on dependable, autonomic and secure computing, international conference on pervasive intelligence and computing, international conference on cloud and big data computing, 2021 (pp. 99-104). IEEE.
[58] Asha J, Meenakowshalya A. Fake news detection using n-gram analysis and machine learning algorithms. Journal of Mobile Computing, Communications & Mobile Networks. 2021; 8(1):33-43.
[59] Aljofey A, Jiang Q, Rasool A, Chen H, Liu W, Qu Q, et al. An effective detection approach for phishing websites using URL and HTML features. Scientific Reports. 2022; 12(1):1-19.
[60] Zhao Z, Anand R, Wang M. Maximum relevance and minimum redundancy feature selection methods for a marketing machine learning platform. In international conference on data science and advanced analytics (DSAA) 2019 (pp. 442-52). IEEE.

[61] Gollapalli M, Alansari A, Alkhorasani H, Alsubaii M, Sakloua R, Alzahrani R, et al. A novel stacking ensemble for detecting three types of diabetes mellitus using a Saudi Arabian dataset: Pre-diabetes, T1DM, and T2DM. Computers in Biology and Medicine. 2022; 147:1-12.
[62] Farhana N, Firdaus A, Darmawan MF, Ab RMF. Evaluation of Boruta algorithm in DDoS detection. Egyptian Informatics Journal. 2023; 24(1):27-42.

**Sabri Hisham** earned his Bachelor's degree in computer science (Industrial Computing) from Universiti Teknologi Malaysia (UTM) in 2001 and his Master's degree in software engineering from Universiti Malaysia Pahang (UMP) in 2014. He is currently a PhD student at Universiti Sultan Zainal Abidin's Department of Computer Science in the Faculty of Computing and Informatics at Terengganu, Malaysia. He is also the Head of Infostructure at Universiti Malaysia Pahang's Information and Technology Department. He is also a Blockchain Solidity Smart Contract and Ethereum Expert and certified professional. Blockchain, Bitcoin, ML, IoT, Mobile Apps, Web Applications, SCADA, and Telemetry Systems are among his current research interests.
Email: sabrihisham@ump.edu.my

**Prof Mokhairi Makhtar** earned his PhD in 2012 from the University of Bradford in the United Kingdom. He is currently a Professor at Universiti Sultan Zainal Abidin (UniSZA) in Terengganu, Malaysia, in the Department of Computer Science. Machine Learning, Ensemble Method, Data Mining, Soft Computing, Timetabling and Optimisation, Natural Language Processing, E-Learning, and Deep Learning are some of his current research interests.
Email: mokhairi@unisza.edu.my

**Mr Azwa Abdul Aziz** earned a degree in computer science in 2002 from Malaysia's Universiti Teknologi Mara. He completed his studies at the Bachelor's level and graduated from Universiti Teknologi Mara (UiTM), Malaysia, in 2004. Then, in 2010, he earned a master's degree in computer science from Malaysia's University of Malaysia Terengganu (UMT). He is recently a lecturer at the Department of Computer Science at Sultan Zainal Abidin University in Terengganu, Malaysia. Included in his research interests are Big Data Analytics, Text Mining, Business Intelligence, and Machine Learning.
Email: azwaaziz@ unisza.edu.my

## Appendix I

| S. No. | Abbreviations | Descriptions |
|---|---|---|
| 1 | 3D | Three Dimensions |
| 2 | ABI code | Application Binary Interface Code |
| 3 | AFL | American Fuzzy Lop |
| 4 | AI | Artificial Intelligence |
| 5 | Al-SPSD | Anti-Leakage Smart Ponzi Scheme Detection |
| 6 | API | Application Programming Interface |
| 7 | AUC | Area under the ROC Curve |
| 8 | BiGRU | Bidirectional Gated Recurrent Unit |
| 9 | BiLSTM | Bidirectional LSTM |
| 10 | CBGRU | Convolutional-Based Bidirectional Gated Recurrent Unit |
| 11 | CNN | Convolutional Neural Network |
| 12 | CPU | Central Processing Unit |
| 13 | DAO | Decentralised Autonomous Organisation |
| 14 | DApp | Decentralised Application |
| 15 | DASP | Decentralized Application Security Project |
| 16 | DeFi | Decentralised Finance |
| 17 | DL | Deep Learning |
| 18 | DT | Decision Tree |
| 19 | EOA | Externally Owned Accounts |
| 20 | ERC | Ethereum Request for Comment |
| 21 | ETC | Extra-Tree Classifier |
| 22 | ETH | Ether |
| 23 | EVM | Ethereum Virtual Machine |
| 24 | FN | False Negative |
| 25 | FNR | False Negative Rate |
| 26 | FP | False Positive |
| 27 | FPR | False Positive Rate |
| 28 | GB | Gradient Boosting Classifier |
| 29 | GBDT | Gradient Boosting Method |
| 30 | GRU | Gated Recurrent Unit |
| 31 | HYIP | High-Yield Investment Programmes |
| 32 | IoT | Internet of Thing |
| 33 | JSON | JavaScript Object Notation |
| 34 | KNN | K-Nearest Neighbours Algorithm |
| 35 | LDA | Linear Discriminant Analysis |
| 36 | LightGBM | Light Gradient-Boosting Machine |
| 37 | LSTM | Long Short-Term Memory |
| 38 | MI | Mutual Information |
| 39 | ML | Machine Learning |
| 40 | MRMR | Minimum Redundancy Maximum Relevance |
| 41 | MS | Milliseconds |
| 42 | MulCas | Multi-view Cascade Ensemble model |
| 43 | MySQL | My's Structured Query Language (MySQL) |
| 44 | NFT | Non-Fungible Token |
| 45 | NLP | Natural Language Processing |
| 46 | Opcode | Operation Code |
| 47 | PSD-OL | Ponzi Schemes Detection Approach Based On Oversampling-Based |
| 48 | RAM | Random Access Memory |
| 49 | RF | Random Forest |
| 50 | SadPonzi | Semantic-Aware Detection Approach for Ponzi |
| 51 | SGD | Stochastic Gradient Descent |
| 52 | SMOTE | Synthetic Minority Oversampling Technique |
| 53 | SULOV | Searching for Uncorrelated List of Variables |
| 54 | TF-IDF | Term Frequency - Inverse Document Frequency |
| 55 | TN | True Negative |
| 56 | TNR | True Negative Rate |
| 57 | TP | True Positive |
| 58 | TPR | True Positive Rate |
| 59 | XGB | eXtreme Gradient Boosting |