IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 19 February 2024, accepted 11 March 2024, date of publication 19 March 2024, date of current version 26 March 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3378736*

RESEARCH ARTICLE

# Securing Smart Contracts in Fog Computing: Machine Learning-Based Attack Detection for Registration and Resource Access Granting

**TAHMINA EHSAN[1], MUHAMMAD USMAN SANA[2], MUHAMMAD USMAN ALI[3], ELIZABETH CARO MONTERO[4,5,6], EDUARDO SILVA ALVARADO[4,7,8], SIROJIDDIN DJURAEV[9], AND IMRAN ASHRAF[10]**

[1]Information Technology Development, University of Gujrat, Gujrat 50700, Pakistan
[2]Department of Software Engineering, University of Gujrat, Gujrat 50700, Pakistan
[3]Department of Computer Science, University of Gujrat, Gujrat 50700, Pakistan
[4]Universidad Europea del Atlántico, 39011 Santander, Spain
[5]Universidad Internacional Iberoamericana Arecibo, Arecibo, PR 00613, USA
[6]Universidade Internacional do Cuanza, Cuito, Bié, Angola
[7]Universidad Internacional Iberoamericana, Campeche 24560, Mexico
[8]Universidad de La Romana, La Romana, Dominica
[9]Department of Software Engineering, New Uzbekistan University, Tashkent 100007, Uzbekistan
[10]Department of Information and Communication Engineering, Yeungnam University, Gyeongsan-si 38541, South Korea

Corresponding author: Imran Ashraf (ashrafimran@live.com)

This work was supported by European University of Atlantic.

**ABSTRACT** Smart contracts are becoming increasingly popular for managing transactions or activities in fog computing environments. However, the use of smart contracts for registration and resource access granting is vulnerable to various types of attacks that can compromise their security. Detecting these attacks can be challenging, as attackers can use sophisticated techniques to evade detection. This research uses a machine learning-based approach for detecting different attacks on smart contracts used for registration and resource access granting in fog computing. Data is collected from online Ethereum's official site "etherscan.io". Different feature extraction methods and machine learning models are tested. Using accuracy, precision, recall, F1 score, cross-validation, and computational time, the performance of models is evaluated. Results indicate that extreme gradient boosting (XGB) and random forest (RF) provide the highest accuracy of 80% using the term frequency-inverse document frequency (TF-IDF) approach. The light gradient boost classifier provides the highest accuracy of 81% with the Bag of Word (BoW) approach. Similarly, the extra tree provides the highest accuracy of 83% using the N-gram technique. Furthermore, performance using TF-IDF is slightly poorer than BoW and N-gram, however, it has less computational complexity.

**INDEX TERMS** Smart contracts fog computing, machine learning, cyber security, cyber attacks.

## I. INTRODUCTION

Cloud computing is a computing paradigm where users can access computing resources such as servers, storage, software, databases, and applications over the internet rather than relying on the local computing infrastructure. In cloud computing, users typically pay for the resources that they use on a pay-as-you-go basis, allowing them to scale up or down as needed. A large amount of cloud computing resources are accessed by the Internet of Things (IoT) devices, but the amount of IoT devices is increasing day by day [1]. Undoubtedly, the growth of IoT devices brought many opportunities for cloud computing, but it has also created new challenges and issues, such as cost, data volume

The associate editor coordinating the review of this manuscript and approving it for publication was Nitin Gupta.

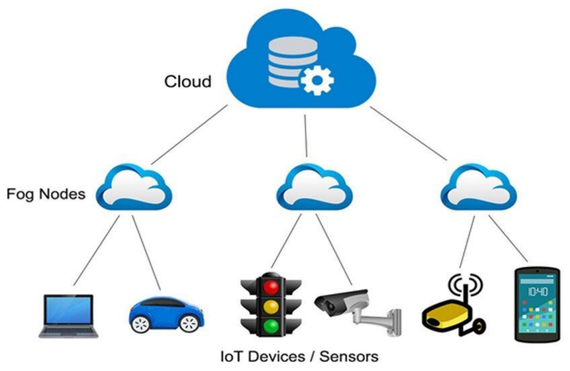

**FIGURE 1. Architecture of fog computing.**

and management, security and privacy, bandwidth, network congestion, and latency issues. To resolve different issues of cloud computing in 2018, Cisco introduced fog computing between cloud computing and edge computing [2].

Fog computing is a distributed computing paradigm that extends the cloud computing paradigm to the edge of the network, as shown in Figure 1. It is designed to provide computing, storage, and networking services closer to end-users and IoT devices [3]. It is creating a layer between the cloud and the edge layer. The fog layer provides a set of computing and networking resources to the edge devices because it is near compared to the cloud computing layer.

Fog computing is the extension of cloud computing [4] designed to resolve different issues of cloud computing [5]. To secure fog computing, security services such as authorization, authentication, control of access, privacy, reliability, accessibility, and non-repudiation are required. Access control is a security service that provides proper access to resources among individuals or users, devices, applications, and services. Access control is critical when it comes to implementing security for an IoT application [6].

Blockchain is used to solve the security and privacy issues of the fog layer [3]. Blockchain is used for data sharing, when fog nodes, IoT devices, and cloud providers are sharing data to maintain data privacy and safety from data tempering. The fog layer also used its consensus mechanism for all fog nodes to agree on the same state of the system. Fog computing involves multiple users and devices, and each has its own identity and access rights. Blockchain provides a decentralized and secure way for managing the identities and access rights without involving a third party. In the authentication process of the user, blockchain is used in the form of a smart contract. Authentication is a process that is used to ensure that users are, who they claim to be. Malicious nodes and unregistered entities are the main targets of the authentication process [7]. The fog layer is connected to multiple devices such as IoT devices, cloud providers, and fog nodes, which creates a complex environment for their authentication.

Blockhain-based security frameworks have been presented for fog computing. For example, a security service architecture is provided in [8] which is implemented on the fog computing layer. This model provides an efficient privacy and authentication process compared to the existing systems. Different authentication approaches are used in fog computing such as identity and access management, biometric authentication, and blockchain-based authentication [9]. Identity and access management systems are used to manage both user and device identities and access rights in fog computing through usernames and passwords, digital certification, and signatures. Biometric authentication can be used in the form of fingerprints, iris scans, and facial recognition to authenticate users. Blockchain-based authentication is used for managing the user and device identities in a decentralized way in fog computing, by using the public and private key pairs and smart contracts to authenticate users and grant access to fog computing resources. The choice of authentication method or technique depends on the security and privacy level of the systems. It is important to carefully evaluate the strengths and weaknesses of each authentication method and select the best one suitable for the specific fog computing system.

In blockchain-based authentication, a smart contract is used for user registration, resource registration, authentication request, challenge-response protocol, verification, and resource access. User registration is used to register the identity of the user [10] in the blockchain-based system, which generates the public and private key pair for the user, and the public key is added to the blockchain. Resource registration is used to register the resources of the fog node and also the list of authorized users who can access the resources. The authentication request is sent to the fog node when the user wants to access the fog computing resources. Resource access grants the user access to the resource of the fog node, after that the user can access and use the resources. During verification, the smart contract verifies the user by using the public key. If the public key is valid, then the user is authenticated. The smart contract checks the user's public key against the list of authorized users, which is stored on the blockchain. If the user is authorized to access the resource, the smart contract grants access to the resource. If any vulnerability or attack affects the security of the smart contract, then multiple problems or challenges occur that affect the security or privacy of the user's data [11].

Smart contracts are rules or policy agreements that are not enforced by the outside network. Any attacks on the smart contract can put the entire network of the blockchain in danger including miners, as well as, the organizations [4]. Different attacks necessitate efficient intrusion detection systems. Phishing attacks, decentralized autonomous organization (DAO) attacks, parity wallet attacks, gambling attacks, spam tokens, and Ponzi attacks, are among the few attacks that exploit vulnerabilities in the security of the smart contract. This study makes the following contributions

- Integration of machine learning with smart contracts to enhance security in fog computing environments during registration and resource access granting process.

- Investigation of various machine learning models with three feature engineering approaches, term frequency-inverse document frequency (TF-IDF), bag of words (BoW), and N-gram for attack detection in fog computing.
- Conducting experiments with self-collected datasets to evaluate the efficacy of different models and feature engineering techniques.
- Comparison of performance metrics across different models and feature engineering methods to identify optimal combinations for attack detection in fog computing.
- Analysis of execution time differences between feature engineering approaches, providing insights into computational efficiency considerations.

## II. BACKGROUND

### A. SMART CONTRACT

A smart contract is a self-executing contract with the terms and conditions of the seller and buyer, client, and host, which is executed transparently and securely. Blockchain is also used in the authentication of the users before the resource allocation or access control [4].

On the blockchain, a smart contract is self-executing code. In basic terms, it is a digital agreement or contract which is represented by a computer code, that runs on the distributed ledger. They are to conduct predetermined processes when particular requirements are fulfilled, making them perfect for creating completely automated systems.

### B. SIGNIFICANCE OF SECURING SMART CONTRACTS IN FOG COMPUTING

Smart contracts offer significant benefits and security precautions and are essential to fog computing, especially in the phases of resource access granting and user registration. Smart contracts automate and optimize processes in fog computing environments, reducing the need for human involvement and improving efficiency in operation. They run autonomously to carry out predetermined rules and conditions, eliminating the need for middlemen and providing truthful transactions.

In fog computing, security is important, and essential for improving security protocols. Smart contracts ensure the authenticity and privacy of transactions and reduce the possibility of manipulation and unwanted access. Decentralization is also improving security because it removes the single points of failure and decreases the dependence on centralized authorities [12]. Furthermore, smart contracts can mitigate insider risks and unlawful access to resources by executing access control techniques based on preset constraints and privileges. Smart contracts ensure the continued security of fog computing environments by detecting and responding to security threats through mechanisms for enforcement and continual monitoring.

In fog computing, failures in security can result in financial losses, harm to reputation, compromised integrity, hacking of data, and unavailability of service. To effectively manage these risks, machine learning-based solutions provide automated processes, the identification of anomalies, flexibility, the ability to adapt, and improved accuracy. Machine learning improves the overall safety condition of fog computing environments by utilizing these skills, which allow for adaptive mitigation of risks and reduce the possible effect of security breaches [13].

### C. MACHINE LEARNING AND SMART CONTRACT

Blockchain technology has been popular over the past several years. This technology enables individuals to interact directly with one another through an extremely secure and distributed system, without the need for a third party [14], [15]. Machine learning, in addition to its strengths, can assist in dealing with many of the restrictions that blockchain-based systems face. The combined use of both of these technologies, machine learning, and blockchain technology, has the potential to produce very effective and helpful solutions [16].

Numerous real-world fields are already starting to employ and do considerable research on machine learning [17]. Thousands of records per day can be used to train machine learning models, which can then be used to tackle a variety of economic and social problems. The field of machine learning has also begun to influence blockchain technology. No doubt blockchain technology has many advantages, but it also has some unavoidable drawbacks. Machine learning has enhanced the way it is perceived while also aiming to address the shortcomings of blockchain technology [18] and provide reliable and effective solutions, especially in the field of security and privacy. The machine learning approach can be used with the smart contract in the blockchain platform to improve its security. Recently, machine learning has been a prominent or effective approach for attack detection in smart contracts [19].

Machine learning is used for attack detection during registration and resource access processes in fog computing. Machine learning algorithms enable real-time detection and response to threats without requiring human intervention. These algorithms are flexible and improve over time. They are capable of detecting abnormal patterns in huge amounts of data, stopping possible attacks before they do a great deal of damage, and ensuring the reliability of smart contracts.

Ethereum is the first platform that supports smart contracts. The smart contracts are managed or working on the blockchain platform. Smart contracts are present in the manner of contract accounts in Ethereum [20], as shown in Figure 2. For example, a smart contract is used to maintain the network activities through agreement, and machine learning can be used with smart contracts for user validation, and identity verification, also identify cyberattacks and stop them in real-time. It makes it difficult for attackers to steal or hack confidential data from the network. It can analyze the patterns or behavior of smart contracts and highlight the security drawbacks or flaws that exist in the smart contract because security or privacy must always be maintained.

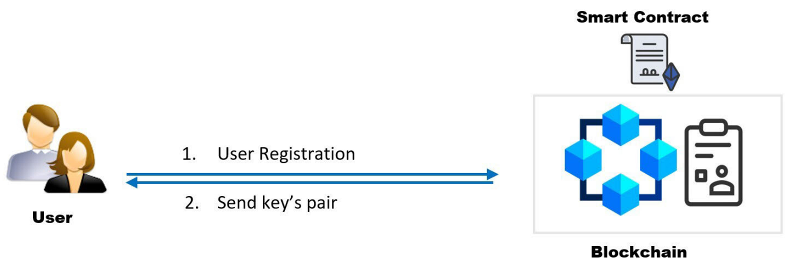

**FIGURE 2.** Registration phase of user.

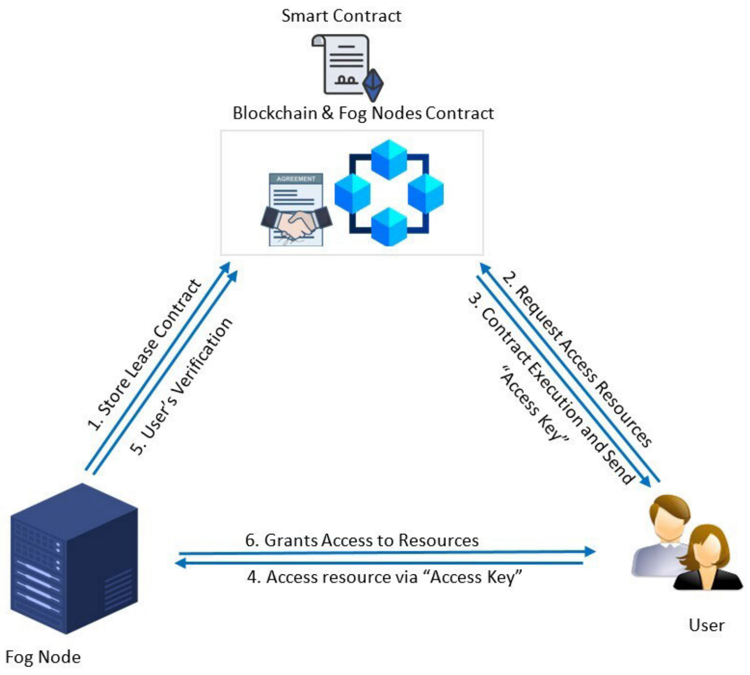
Smart Contract
Blockchain & Fog Nodes Contract
AGREEMENT
1. Store Lease Contract
5. User's Verification
2. Request Access Resources
3. Contract Execution and Send "Access Key"
6. Grants Access to Resources
4. Access resource via "Access Key"
Fog Node
User

**FIGURE 3.** Resource access granting phase.

### 1) REGISTRATION PHASE

i) **User Registration**: When a user wants to access the resources of the fog node, they first need to register using the smart contract [21]. Users fill in the information and this information is stored in the blockchain. In the blockchain, a smart contract is used to register and verify the user's information. If the user's information fulfills the requirements for registration, the user account is created [21].

ii) **Send Key's pair**: After storing the information, Blockchain generates the pair of the public and private keys and sends it to the user.

### 2) RESOURCE ACCESS GRANTING PHASE

Figure 3 shows the resource access granting process which comprises the following phases

i) **Store Lease Contract**: In this step, the fog node submits its self-executing lease contract on the Ethereum blockchain if it is ready to share its resources. The leasing contract specifies how the agreement is to be carried out and is written as a section of executable code [22]. According to this contract, corresponding events are triggered when the customer delivers a predetermined amount of digital coins to the provider, indicating that the requirements of the agreement are completed.

ii) **Request Access Resources**: A user can check the contract from the blockchain if he attempts to use a fog node's resources. The user sends the request to the blockchain network for the fog node resource. Smart contract checks their public key with the registered public keys, if exist, and are matched with the user public key [22]. The smart contract checks its requirements and required resources.

iii) **Contract Execution and Send "Access key"**: In this step, the smart contract executes the contract and sends the access key of the fog node to the user [22] if the user is already registered in the network, then the secret key of fog node is provided to the user for accessing the resources. The user can transfer the payment for the resources as coins or ethers, or a smart contract can reserve the payment amount from the user's account.

iv) **Access resource via "Access Key"**: After getting the access key of the fog node, the user accesses its resources.

v) **User's Verification**: The fog node verifies the user's identity from the blockchain that accesses the resources.
vi) **Grants Access to Resources**: After the verification step, the fog node is granted access to the user to its resources. After completion of the process, payment is transferred to the fog node.

### D. OBJECTIVES OF RESEARCH

The objective of this research is to use a machine learning-based approach for detecting attacks in smart contracts which are used in resource management in fog computing. The objectives are

i) Identify and categorize attacks on smart contracts used in fog computing during registration and resource access granting because if smart contracts are tempered by the attacker, he can make the unauthorized access of resources and loss of registered data during registration.
ii) Determine the most effective machine learning techniques for detecting and preventing these attacks.
iii) Evaluate the performance of the chosen machine learning approach in terms of accuracy, precision, recall, and F1 score.
iv) Use different opcode-based feature extraction methods to enhance attack detection in smart contracts.

### E. RESEARCH QUESTION

To meet the above-defined objectives, we have defined the following research questions

i) How can machine learning be applied to identify attacks on smart contracts that are utilized in the environment of fog computing to provide resource access and registration of the users?
ii) In a fog computing environment, how can be secured a smart contract against an intruder or attacker?
iii) How can opcode analysis be applied as a feature extraction method that is fast and reliable for identifying attacks on smart contracts?
iv) Which techniques and models are most effective in predicting attacks on smart contracts used in fog computing environments to give resource access and registration to the users?

This paper is structured as follows. Section III provides literature relevant to research problems such as fog computing and attack or vulnerability detection from the smart contract. In Section IV, the methodology of our research has been explained in detail. This section also discusses how we collect data, make a dataset, explore data analysis on the dataset, and feature extraction techniques. In Section V, the results of this research have been presented, as well as, the discussion of which feature extraction technique and model provides the best results. In Section VI, the conclusion of this research and future directions are discussed.

## III. LITERATURE REVIEW

Fog computing is an extension of cloud computing that inherits different issues from the cloud [23]. Due to the closeness to IoT devices, many security and privacy-related problems are faced and reported. In the literature, researchers provided different approaches for service authorization, access control, and authentication in fog computing for secure transmission. For example, in [24], the author provided a queuing theory-based cuckoo search (QTCS) model to allocate the resources in fog computing by using the priority-based method. This model enhances the quality of the services, power consumption, resource allocation, and management of resources efficiently.

The study [25] provided a deep learning-based detection scheme for a malicious and safe class of smart contracts. Different deep learning techniques are used for the detection of both classes such as long short-term memory (LSTM), gated recurrent unit (GRU), and artificial neural networks (ANN). The authors used the BigQuery dataset with binary classification and the highest result of these classifiers is 99.03%. In [21], the authors proposed a system for the registration and authentication of the user in fog computing. The authors used the smart contract for registration and stored the information and data of the user in the secure ledger. The proposed system consumes less cost for the registration and authentication process as compared to the existing systems. The study also conducted multiple user accounts for this system and compared its cost with existing systems.

In [26], the author used 49502 real-time smart contracts with different vulnerabilities such as call stack, integer overflow, timestamp, TOD, reentrance, and integer overflow, and reported a high accuracy of 99%. The author converted the contract code into byte code and opcode, followed by the extraction of the n-gram feature from the opcode. Different machine learning models are applied such as extreme gradient boosting (XGBoost), K nearest neighbor (KNN), and support vector machine (SVM), etc. The authors improve the speed and accuracy of vulnerability detection using machine learning. Similarly, the study [27] provided a mutual authentication scheme for fog computing. The scheme is used for authentication and secure key exchange processes in fog computing. The proposed scheme used elliptic curve cryptography and the hash function for the authentication and key exchange process. It provides security from cyber-attacks such as man-in-attack and replay attacks etc.

The authors provide a framework in [28] that is used for mining and classifying the smart contract concerning various vulnerabilities such as extra gas consumption, compiler version not fixed, implicit visibility level, unchecked low-level call, frozen ether, etc. These vulnerabilities are studied using the AutoMESC dataset and various fixes are suggested. In [22], the authors provided the process of granting resource access using the blockchain. The smart contract is used to access the resources, the big advantage of the smart contract is removing the third dependency in the network.

The study [29] proposed a fraud detection model for investors who are investing using smart contracts in Ethereum. The author used the 3203 smart contracts Ponzi and non-Ponzi attacks that were collected using web scrapping from the Etherscan website. The machine learning classifiers are used for detection such as J48, random forest, and 0-day model. Models show promising results. In [30], the author provides the algorithm for resource allocation in fog computing. They used a modified whale-optimized resource allocation algorithm. The author used a fog node with limited resources and allocated its resources with effective results using two phases. In the first phase, the task is classified while in the second phase involves task offloading. The algorithm provides better performance and successful completion of the task as compared to other algorithms such as the shortest job first.

In [31], the authors detect multi-label vulnerabilities such as timestamp, reentrancy, TOD, integer underflow, and overflow using the Bi-LSTM with an accuracy of 88.12%. The author collects 5450 smart contracts from the Etherscan website and detects multiple vulnerabilities in the smart contract such as integer underflow, integer overflow, reentrance, timestamp, and transaction order dependency. Similarly, in [32], the author detects the normal and abnormal smart contracts using an ensemble model. The author collects 1904 smart contracts from the Etherscan website. The author extracts the features from the source code of the smart contract using TF-IDF and n-gram techniques. The proposed ensemble model obtains an accuracy of 89.67%.

The study [33] collects the byte code of a smart contract from Etherscan. The author converts the byte code into images and removes the noise after removing the noise and extracting features. CNN classifier is applied to the features with an accuracy of 95.85%.

In [34], the author collects 5735 smart contracts and semantic trees from the code of the smart contract. Later, graph neural networks, and graph matching networks are applied with a reported accuracy of 92.63%. Various vulnerabilities are detected in the study including reentrance, block info dependency, timestamp dependency, etc. Similarly, [35] proposes a distributed denial of service (DDoS) attack detection approach. The authors use the BoT-IoT dataset for experiments. Different classifiers are utilized including random forest (RF), decision tree (DT), and SVM for attack detection.

The study [36], used a framework known as Contract-Fuzzer for the detection of vulnerabilities in smart contracts. Using this framework, different types of vulnerabilities are detected such as gasless, freezing ether, reentrance, block number dependency, exception disorder, and dangerous delegate calls. The authors used 9960 smart contracts from the Etherscan website for vulnerability detection of the smart contract. The proposed framework provided less false positive rate compared to existing approaches. Similarly, [37] collected 2194 verified smart contracts from the Ethersan website. The author used a machine learning framework for vulnerability detection in smart contracts. Experiments involve the detection of reentrancy, arbitrary_memory_access, block_dependency, TOD amount, assertion_failure, ether lock, and integer overflow/underflow. The author used SVM and LSTM which provide 87.5% and 80.9% accuracy, respectively.

In the same vein, [26] collected 78 smart contracts from SmartBugs for vulnerability detection using machine learning. The collected dataset contains several classes such as short address, access control, bad randomness, unchecked low level, denial of services, and reentrance. The author used two classifiers for the prediction of the results such as KNN and stochastic gradient descent with accuracy of 84% and 82%, respectively. In [38], the authors downloaded the dataset from the BigQuery. The author used binary and multi-classification for the vulnerability detection of smart contracts. The authors used multiple classifiers on multi and binary classification data but KNN provided the highest accuracy 99.5% on multiclass data, and RF provided the highest 97.0% accuracy on the binary class dataset.

In [39], the authors detect the Ponzi schemes by using the opcode context characteristics of smart contract and contract account characteristics. The authors collect the labeled 3590 non-Ponzi and 200 Ponzi smart contracts from the XBlock website. After acquiring the dataset, the yellow paper of Ethereum is used for converting the bytecode and hexadecimal values into opcode values. The author uses the N-gram technique for feature extraction from the opcode of smart contracts. In addition, the adaptive synthetic sampling (ADASYN) technique is used to deal with the unbalancing issue of the dataset. After resolving this issue, the Adaboost classifier is trained on the Ponzi contracts features to detect the attacks.

In study [40], the author detects the Ponzi schemes from the smart contract. The author uses the three datasets with binary classification such as Ponzi and non-Ponzi. The first dataset consists of 3588 non-Ponzi and 200 Ponzi scheme smart contracts. In the second dataset, 167 Ponzi scheme addresses of smart contracts extracted from other sources and 180 Ponzi scheme smart contracts, which are extracted from XBlock. Now, the dataset consists of 547 Ponzi scheme addresses of smart contracts and 3588 non-Ponzi scheme smart contracts, these are a total of 4135 addresses of smart contracts for experiments. Then, a Control Flow Graph is used to extract the n-gram Term Frequency and n-gram Term Frequency-Inverse Document Frequency features. SVM_SMOTE algorithm was used to resolve the issue of oversampling in the dataset. It balanced positive and negative samples of the dataset and utilized the SVM_SMOTE-based Random Forest algorithm for the detection of the Ponzi scheme smart contracts and it provide 95% accuracy.

In study [41], the author improves the fast detection of vulnerable smart contracts because the detection of large-scale smart contracts is very difficult and critical. The

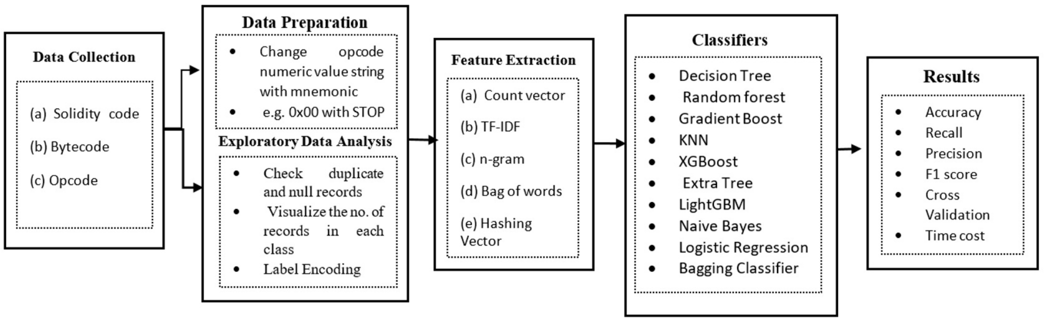

**FIGURE 4.** Workflow of the adopted methodology.

author discusses the two methods of analysis which are code analysis and Learning method. The author reduces the dimension of features to improve the efficiency or speed of the detection. In this research, the author uses the two datasets. The first dataset has 871 vulnerable and 2129 non-vulnerable smart contracts. Second, the author selects the 3000 smart contracts Contractward. The open-source tool Oyente labels these addresses with six labels such as integer Underflow, Integer Overflow, Callstack Depth, TOD, Timestamp, and Re-entrancy for analysis of performance. The author uses the Block-gram technique to improve the efficiency of the detection process and then compares the results with n-gram features to evaluate its performance. Different classifiers are used for the comparison of their performance. The Block-gram and N-gram technique take 0.0002s, and 0.09s for the first dataset using NB. The Block-gram and N-gram techniques take 0.0003s, and 0.09s for attack detection for the second dataset using NB.

A probabilistic buckshot approach is presented in [42] for robustness and smoothing the handover process for ad-hoc networks. In addition, a heuristic model is also implemented for identifying the best cluster head. For securing the data, a lightweight encryption algorithm is also adopted. Results indicate a 20% to 23% improvement in energy-sensitive sensor networks.

## IV. MATERIALS AND METHODS

The goal of this research is to create a machine learning-based approach for precisely and automatically identifying numerous smart contract attacks. An overview of the proposed approach is given in Figure 4.

### A. DATA COLLECTION

Due to the restricted availability of open-source solidity code, researchers and developers who require access to a large number of smart contracts for analysis and experimentation confront difficulties [37]. The solution proposed in this study uses an API key to extract contracts with "Solidity, bytecode, and opcode" from the Ethereum official website https://etherscan.io/ [43]. In this research, 1353 smart contracts are collected for experiments.

#### 1) DATASET LABELING

The collected smart contracts are labeled using the "Label Word Cloud" feature in the Etherscan.io website https://etherscan.io/labelcloud. Using this feature, smart contracts are labeled with four labels including DAO, parity bug, gambling, and spam token.

But still, unlabeled addresses remain in the file which are not found on this website. The other addresses heist, phish-hack, and the exploit are labeled using forta [44] which provides the different scanning nodes for the blockchain component and provides the labeled addresses [45] for data analysis and machine learning. Ponzi label using https://github.com/BuptHxz/DetectionOfPonziContract which addresses are used in [46]. After labeling the addresses, the solidity code, bytecode, and opcode of the smart contract are collected from the Etherscan using the API key.

#### 2) CLASSES

In this research, different malicious classes of smart contracts are collected. These classes are briefly discussed below

i) Ponzi: A Ponzi scheme is a type of investment scam in which cash from new investors is used to pay out claim profits to current investors. Ponzi scheme managers frequently entice new investors by offering to put money in possibilities that claim to provide great returns with no risk [46].
ii) Phish-hack: Phishing attacks are used on the smart contract to deceive people into exposing private keys, usernames, passwords, or other authentication information to access their accounts or money [5].
iii) Gambling: Gambling attacks are used to gain unauthorized access to the resources of the blockchain using an unauthorized way or method.
iv) Parity Bug: The adversaries sent two transactions to carry out the assault, intending to take control of multisign so that all the money could be taken out. The parity multisign wallet library contract was launched after the assault was over. It did, however, have a flaw that allowed anybody to run initWallet. Because the attack was carried out twice, it is known as parity wallet hacks 1 and 2. By starting a call to initWallet, the attacker in the initial attack was able to change

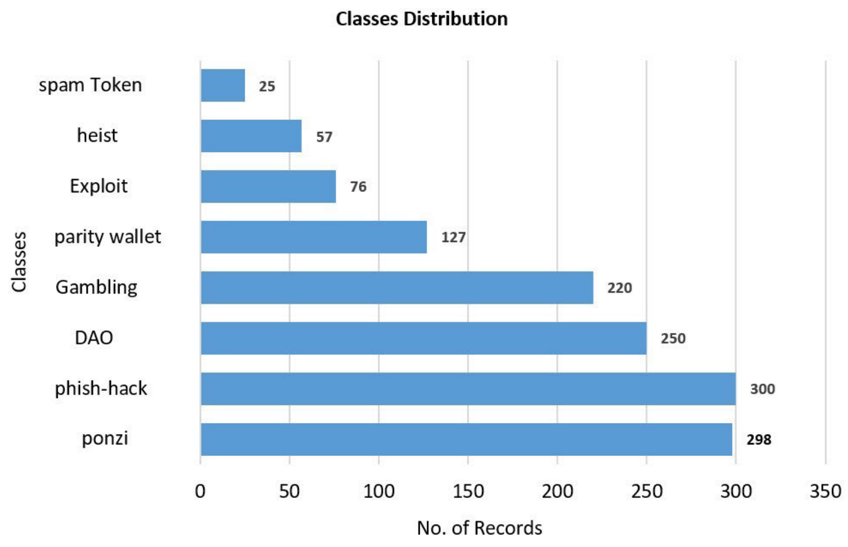

**FIGURE 5.** **Number of samples in each class.**

the wallet's state. As a result, the attacker gained the owner's trust and was able to steal the money without being stopped [47].

v) Exploit: Exploit tokens are tokens that have been produced and spread as a result of taking advantage of flaws in the blockchain and smart contract infrastructure. These tokens are frequently linked to malicious behavior or attacks that abuse flaws in the system [48].

vi) Heist: Unauthorized persons or attackers steal money from a user and move it to their accounts. As a result, its participants suffer huge financial losses [49].

vii) DAO: The DAO contract has serious weaknesses that let attackers take money. A flaw allowed an attacker to repeatedly request money from the smart contract before the balance was updated. The flaw was caused by errors in the code, where the developer of the smart contract neglected to account for the possibility of a recursive call. As a result, it made it possible for attackers to steal millions of dollars worth of ether in the first few hours [50].

viii) Spam Token: Spam tokens are tokens that are involved in scams or spam. The name or symbols of the token have the same attributes such as script or code, and URL [51].

## B. DATA PREPARATION

The collected opcode of each smart contract has some hexadecimal values that start with ''0x''. For example ''0 $\times$ 00'' value and its mnemonics value is STOP. The values that have the value of the mnemonic in Ethereum yellow paper are replaced [31], [52].

## C. DUPLICATION REMOVAL

In this step, duplicate values and null records are checked and removed from the dataset. Figure 5 shows the distribution of the number of records for each class of the collected dataset.

## D. LABEL ENCODING

Label encoding is the process of converting the categorical variables to numeric values that can be used with machine learning models. The collected dataset has eight classes including Ponzi, phish-hack, DAO, gambling, parity wallet, exploit, heist, and spam token. Using label encoding, these are replaced with numeric numbers 0 to 7.

## E. FEATURE EXTRACTION

In this step, features are extracted using the feature extraction techniques from the opcode of the smart contract. The opcode of a smart contract is similar to a natural language that is readable or understandable by humans. In the literature, different feature extraction techniques are used for feature extraction from the opcode. This study adopts N-gram, TF-IDF, and BoW for their wide use and better results. These techniques are briefly explained below.

### 1) N-GRAM TECHNIQUE

N-grams are contiguous collections of objects from a repository of text. The number $n$ in n-grams indicates the size of items or words that are to be taken into consideration; for example, $n$ is equal to 1 is used for unigram, $n$ equal to 2 is used for bigram, $n$ equal to 3 is used for trigram, and so on [53]. To identify unusual schemes, we extract the n-gram characteristics from the contract opcode sequences. In natural language processing (NLP), n-grams are frequently used. In addition, malware detection tasks also use it frequently [32].

### 2) TF-IDF TECHNIQUE

TF-IDF is another widely used approach for NLP tasks. It comprises TF and IDF which are calculated separately. TF counts the number of occurrences of unique words while IDF counts the documents across which unique words are

**TABLE 1.** Results using TF-IDF technique.

| Algorithm | Accuracy | Precision | Recall | F1-Score | CV Accuracy | Execution Time(s) |
|---|---|---|---|---|---|---|
| LR | 0.64 | 0.68 | 0.64 | 0.60 | 0.66 | 3.45 |
| DT | 0.75 | 0.74 | 0.75 | 0.74 | 0.74 | 2.76 |
| RF | 0.80 | 0.81 | 0.80 | 0.79 | 0.81 | 9.16 |
| KNN | 0.74 | 0.74 | 0.74 | 0.72 | 0.75 | 1.73 |
| GB | 0.77 | 0.77 | 0.77 | 0.76 | 0.80 | 330.4 |
| BC | 0.78 | 0.77 | 0.78 | 0.77 | 0.79 | 6.82 |
| NB | 0.65 | 0.63 | 0.65 | 0.63 | 0.63 | 0.31 |
| ETC | 0.79 | 0.79 | 0.79 | 0.78 | 0.81 | 5.33 |
| LGBM | 0.78 | 0.79 | 0.78 | 0.78 | 0.80 | 71.54 |
| XGB | 0.80 | 0.81 | 0.80 | 0.79 | 0.80 | 52.16 |

found. TF-IDF assigns higher weights for those terms which appear less frequently thereby giving higher importance to rare words. So, TF-IDF is used to calculate the importance of the word in the document or text [54].

### 3) BAG OF WORDS TECHNIQUE

By measuring the number of times every word appears, the BoW model converts any text into fixed-length vectors. Vectorization is a common term used to describe this procedure. Its simplicity makes it cost-effective to compute, and where placement or contextual information is irrelevant, simpler is sometimes
better.

## F. CLASSIFIERS

Different classifiers are used in this research such as logistic regression (LR), DT, RF, XGB, extra tree classifier (ETC), gradient boost (GB), KNN, NB, bagging classifier (BC), and light boost classifier (LGBM).

Cross-validation is to divide the data set into groups and treat each group as a validation dataset to evaluate the model [55]. In this approach, 10-fold cross-validation is used for every model.

## G. MODEL EVALUATION METRICS

In this research, accuracy, precision, recall, F1 score, cross-validation, and time cost are used for the evaluation of the models.

Accuracy, precision, recall, and F1 score are calculated using the following equations, respectively

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

In addition to these metrics, computational time is also considered for performance comparison. The time cost means the time that is used by the model for training and testing of the results.

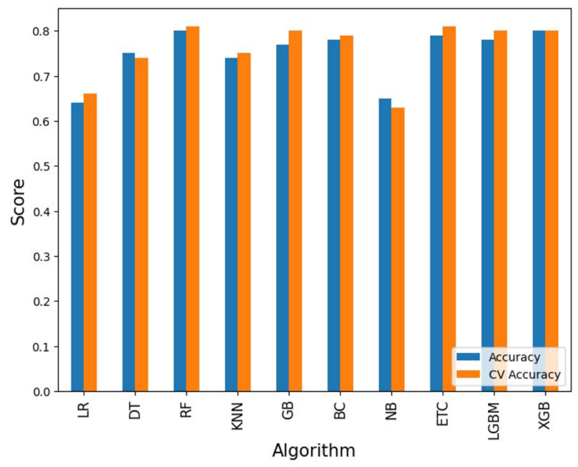

**FIGURE 6.** Accuracy vs cross-validation accuracy using TF-IDF features.

## V. RESULTS AND DISCUSSION

Several machine learning models are used for experiments in this study along with various feature extraction approaches like TF-IDF, BoW, and n-gram. For experiments, the dataset is split into 80% to 20% subsets. 80% of the data is used for training and 20% is used for testing.

## A. EXPERIMENTAL RESULTS USING TF-IDF TECHNIQUE

Different classifiers are applied to the extracted features using the TF-IDF technique. Experimental results of all models are given in Table 1. Results also contain computational time and cross-validation accuracy. Results demonstrate that the NB classifier is the best when execution time is considered, as it takes only 0.31 seconds. The best accuracy score of 0.80 is provided by the RF and XGB which take approximately 9 seconds and 52 seconds, respectively.

Validation results are illustrated in Figure 6 which provides a comparative evaluation of accuracy and cross-validation accuracy. Results show that all the models have pretty much similar trends where cross-validation accuracy is marginally increased, except for DT and NB where the cross-validation is slightly reduced.

Figure 7 shows a visual presentation of the accuracy, precision, recall, and F1 score of all models. RF and XGB show superior results compared to other models, 0.80 accuracy score. Moreover, RF and XGB show similar

**TABLE 2.** Results using bag-of-words technique.

| Algorithm | Accuracy | Precision | Recall | F1-Score | CV Accuracy | Execution Time(s) |
|---|---|---|---|---|---|---|
| LR | 0.79 | 0.79 | 0.79 | 0.79 | 0.77 | 351.58 |
| DT | 0.76 | 0.75 | 0.76 | 0.75 | 0.74 | 1.88 |
| RF | 0.79 | 0.80 | 0.79 | 0.78 | 0.80 | 9.16 |
| KNN | 0.71 | 0.72 | 0.71 | 0.70 | 0.69 | 0.70 |
| BC | 0.79 | 0.81 | 0.79 | 0.78 | 0.78 | 9.92 |
| ETC | 0.80 | 0.82 | 0.80 | 0.79 | 0.81 | 3.89 |
| XGB | 0.80 | 0.82 | 0.80 | 0.80 | 0.81 | 18.10 |
| GB | 0.77 | 0.78 | 0.77 | 0.77 | 0.79 | 181.91 |
| NB | 0.65 | 0.64 | 0.65 | 0.64 | 0.62 | 0.46 |
| LGBM | 0.81 | 0.83 | 0.81 | 0.80 | 0.81 | 30.19 |

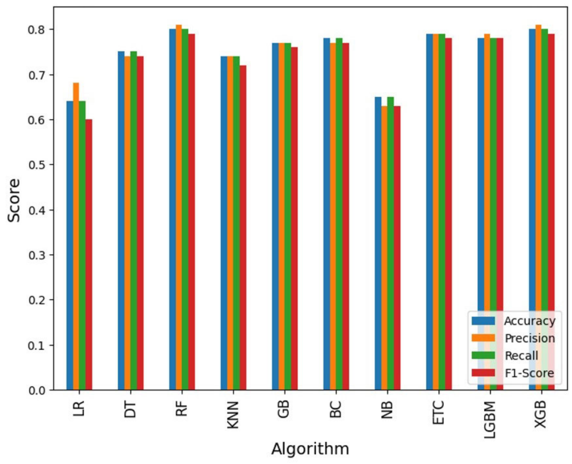

**FIGURE 7.** Experimental results of models using TF-IDF features.

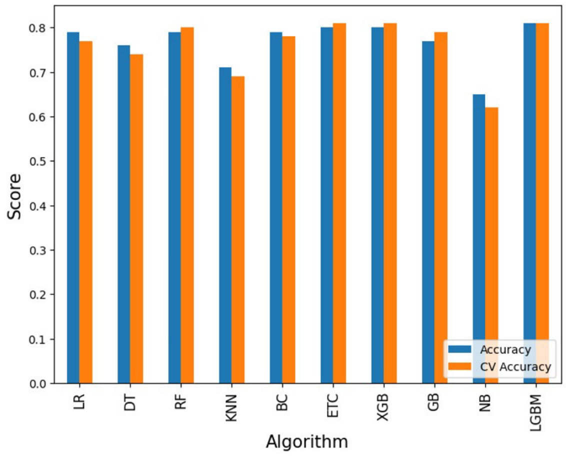

**FIGURE 8.** Accuracy vs cross-validation accuracy using BoW features.

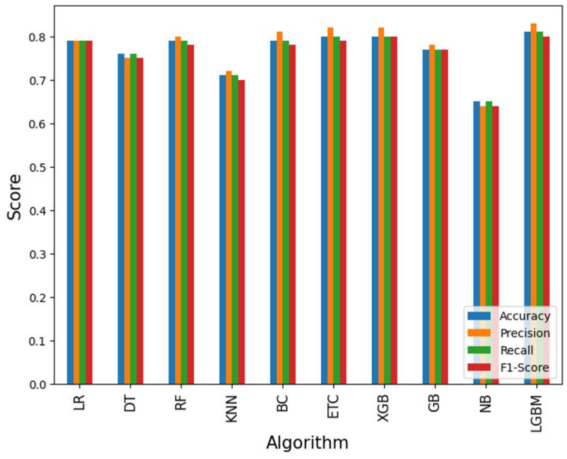

**FIGURE 9.** Experimental results using BoW technique.

results concerning precision, recall, and F1 score indicating robust performance of these models.

ETC shows superior results compared to other models, followed by XGB with a 0.82 accuracy score. Moreover, ETC and XGB show similar results concerning precision, recall, and F1 score indicating robust performance of these models.

### B. EXPERIMENTAL RESULTS USING BAG OF WORDS TECHNIQUE

In addition to TF-IDF, the BoW approach is also employed in this study for performance comparison. Table 2 presents the results of all models using the BoW technique. Results demonstrate that the performance of models is improved when used with BoW features, compared to the TF-IDF approach. The XGB model provides the best results with 105 sec of execution time.

Figure 8 provides the cross-validation results of all models using the BoW technique. XGB shows the best performance followed by ETC and LGBM models. The performance of LR is robust with similar accuracy and cross-validation accuracy while NB is the most affected model when cross-validation is used.

Results regarding precision, recall, and F1 score are presented in Figure 9. LGBM shows the best results regarding accuracy, precision, recall, and F1 score with a 0.81 score for each. It is followed by the ETC, XGB with a marginally low accuracy score of 0.80. Other models also perform better with BoW features.

### C. N-GRAM TECHNIQUE

Besides TF-IDF and BoW, the performance of machine learning models is also evaluated using the n-gram approach. Results given in Table 3 indicate that using n-gram features, the best results are obtained by the ETC with a 0.83 accuracy

**TABLE 3.** Results using n-gram technique.

| Algorithm | Accuracy | Precision | Recall | F1-Score | CV Accuracy | Execution Time(s) |
|---|---|---|---|---|---|---|
| LR | 0.77 | 0.78 | 0.77 | 0.76 | 0.78 | 116.64 |
| DT | 0.76 | 0.77 | 0.76 | 0.75 | 0.76 | 11.66 |
| RF | 0.80 | 0.82 | 0.80 | 0.79 | 0.82 | 34.21 |
| KNN | 0.73 | 0.73 | 0.73 | 0.71 | 0.71 | 7.34 |
| BC | 0.79 | 0.79 | 0.79 | 0.78 | 0.80 | 59.34 |
| ETC | 0.83 | 0.85 | 0.83 | 0.82 | 0.82 | 34.02 |
| XGB | 0.79 | 0.80 | 0.79 | 0.78 | 0.81 | 317.55 |
| GB | 0.80 | 0.79 | 0.80 | 0.78 | 0.80 | 2049.06 |
| NB | 0.70 | 0.74 | 0.70 | 0.68 | 0.69 | 8.02 |
| LGBM v0.80 | 0.82 | 0.80 | 0.80 | 0.82 | 126.04 | |

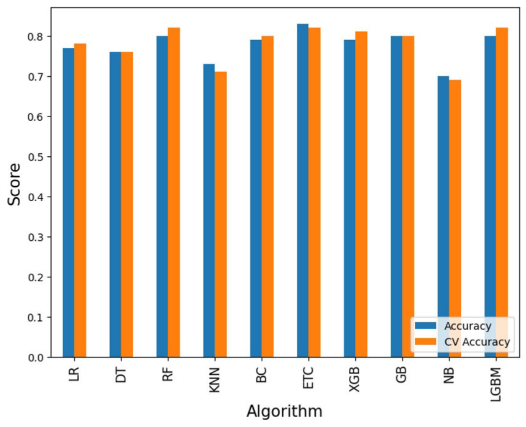

**FIGURE 10.** Accuracy vs cross-validation accuracy using N-gram features.

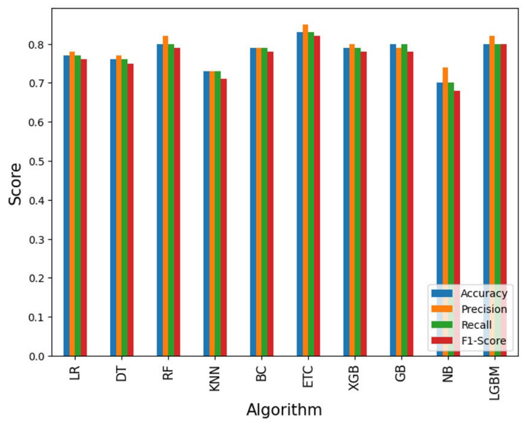

**FIGURE 11.** Experimental results using N-gram technique.

score. It is followed by LGBM, RF, and GB which obtain an equal 0.80 accuracy score with n-gram features. Results are comparatively better than those using the TF-IDF approach, however, inferior to the BoW approach.

Accuracy and cross-validation results are displayed in Figure 10, indicative of the robust performance of ETC, XGB, and RF models. In addition, the LR model shows similar scores for accuracy and cross-validation accuracy.

Results regarding accuracy, precision, recall, and F1 score are illustrated in Figure 11. The performance of XGB is good with the same scores for accuracy, precision, recall, and F1 score indicating the similar good performance of the model for all classes. The figure also shows good results for ETC regarding the true positive rate. Other than NB and KNN, all models show better performance.

### *D. COMPARATIVE PERFORMANCE OF ALL FEATURE EXTRACTION TECHNIQUES*

Experimental results involving TF-IDF, BoW, and n-gram are shown in Table 4. Results indicate that TF-IDF shows slightly poor performance with a 0.80 accuracy score using the RF and XGB model. In comparison, BoW provides better learning capability for the models and shows better performance. The performance of the models is significantly improved when using BoW features, as shown in the previous section. Similarly, other models tend to perform better when used with n-gram features, compared to TF-IDF features. However, execution time is less when TF-IDF features are used.

The confusion matrices for the top four best-performing machine learning models using the TF-IDF techniques are presented in Figure 12. Results show that the extra tree classifier shows the best results with 224 correct predictions. It is followed by the light GBM with 220 correct predictions while RF has the lowest number of correct predictions with 217 correct predictions while 38 predictions are wrong.

### *E. DISCUSSION*

Fog computing presents several difficulties during the registration and resource access phases, including scaling management of resources, reliability problems, security threats, and privacy and data concerns. It is difficult to identify and classify attacks against smart contracts that are utilized in fog computing during the processes of providing resource access and registering users. The study highlights that smart contracts are vulnerable to different attacks and how necessary it is to use a secure mechanism to stop the unauthorized access and manipulation of data.

It is critical to identify the best machine learning methods for recognizing and avoiding these threats. The proposed machine learning-based attack detection system aims to address these by enhancing security, providing real-time threat detection, adapting to evolving threats, and improving accuracy. The results of the study demonstrate the possibilities for improving security measures through the combination

**TABLE 4. Results using all feature extraction techniques.**

| Algorithm | Accuracy | Precision | Recall | F1-Score | CV Accuracy | Execution Time(s) |
|---|---|---|---|---|---|---|
| ETC (TF-IDF technique) | 0.80 | 0.81 | 0.80 | 0.79 | 0.81 | 9.16 |
| XGB (BoW technique) | 0.80 | 0.81 | 0.80 | 0.79 | 0.80 | 52.16 |
| LGBM (BoW technique) | 0.81 | 0.83 | 0.81 | 0.80 | 0.81 | 30.19 |
| ETC (N-gram technique) | 0.83 | 0.85 | 0.83 | 0.82 | 0.82 | 34.02 |

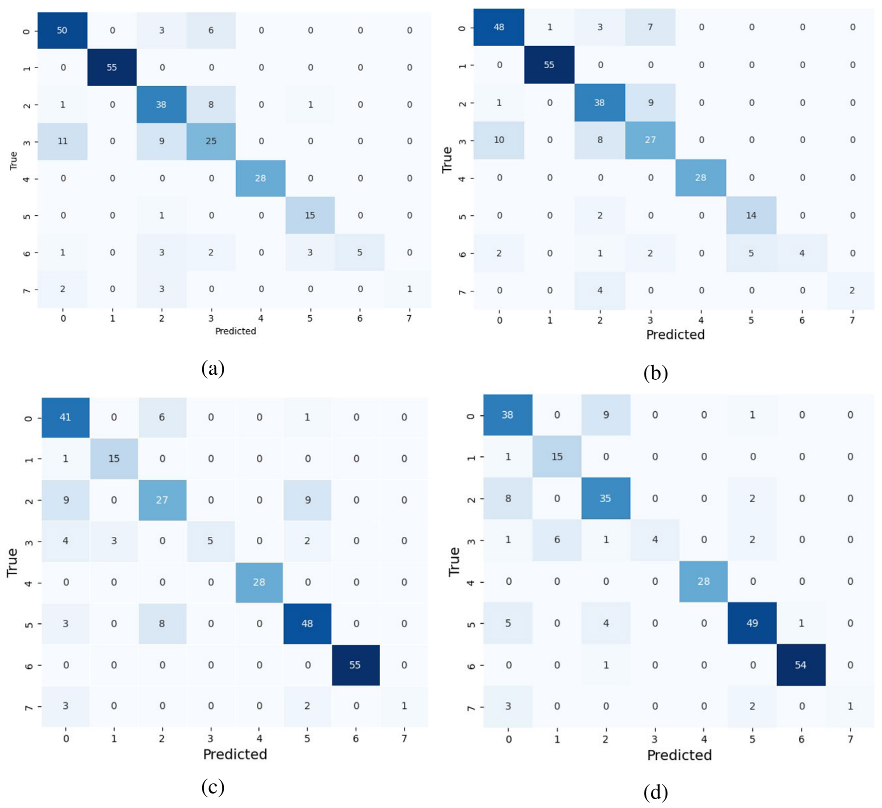

**FIGURE 12. Confusion matrices of machine learning models using TF-IDF features, (a) Random Forest classifier, ((b) XGBoost classifier, (c) LightGBM classifier, and (d) Extra Tree classifier.**

of machine learning with smart contracts. The extra tree classifier with the N-gram feature extraction technique performs better, highlighting how useful they are for detecting and preventing attacks. Different performance metrics are used to evaluate the results of each classifier. There is potential for reducing security threats and maintaining the integrity of fog computing systems through the combination of machine learning and creative feature engineering techniques. These consequences provide possibilities for investigation in the direction of improving fog computing security and building trust in decentralized computing systems.

The use of machine learning-based threat detection in fog computing settings has many benefits, including improved security, flexibility in handling different datasets, real-time processing, and scalability to handle network complexity. However, there are challenges to overcome, such as making sure the data is of high quality, managing limited resources, interpreting complex models, and adjusting to changing conditions. To successfully integrate and effectively enhance security in fog computing environments, these aspects must be balanced.

## VI. CONCLUSION AND FUTURE WORK

Smart contracts are used in the fog computing environment for the registration of the user and resource access granting process. However, smart contracts are prone to several attacks which undermine the security of fog computing. The integration of machine learning with smart contracts can help detect and prevent attacks on fog computing. This study investigates the efficacy of various machine learning models with three feature engineering approaches TF-IDF, BoW, and n-gram in this regard. Experiments are carried out with self-collected datasets using a variety of models. Experimental results reveal that an accuracy score of 0.80 can be achieved with TF-IDF using an XGB and random forest classifiers. BoW features tend to train the models better resulting in superior performance as proven by a 0.81 accuracy score by the light gradient boosting model. N-gram feature also proved to be better than TF-IDF with the extra tree classifier showing a 0.83 accuracy score but on average all models perform better with BoW features. Using TF-IDF features, however, tends to decrease the execution time substantially. For future work, several dimensions can be adopted including the increase in the number of records for better training. In addition, the use of deep learning to fully utilize the potential of large datasets is another possible direction. Furthermore, more attacks on smart contracts can be added in future work.

## REFERENCES

[1] M. Anwer, S. M. Khan, and M. U. Farooq, "Attack detection in IoT using machine learning," *Eng., Technol. Appl. Sci. Res.*, vol. 11, no. 3, pp. 7273–7278, 2021.

[2] Z. Ashi, M. Al-Fawa'reh, and M. Al-Fayoumi, "Fog computing: Security challenges and countermeasures," *Int. J. Comput. Appl.*, vol. 175, no. 15, pp. 30–36, Aug. 2020.

[3] Y. I. Alzoubi, A. Al-Ahmad, and A. Jaradat, "Fog computing security and privacy issues, open challenges, and blockchain solution: An overview," *Int. J. Electr. Comput. Eng.*, vol. 11, no. 6, p. 5081, Dec. 2021.

[4] Y. I. Alzoubi, V. H. Osmanaj, A. Jaradat, and A. Al-Ahmad, "Fog computing security and privacy for the Internet of Thing applications: State-of-the-art," *Secur. Privacy*, vol. 4, no. 2, p. e145, Mar. 2021.

[5] W. Chen, X. Guo, Z. Chen, Z. Zheng, and Y. Lu, "Phishing scam detection on Ethereum: Towards financial security for blockchain ecosystem," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, vol. 7, 2020, pp. 4456–4462.

[6] K. Ragothaman, Y. Wang, B. Rimal, and M. Lawrence, "Access control for IoT: A survey of existing research, dynamic policies and future directions," *Sensors*, vol. 23, no. 4, p. 1805, Feb. 2023.

[7] Y. Du, Z. Wang, and V. C. M. Leung, "Blockchain-enabled edge intelligence for IoT: Background, emerging trends and open issues," *Future Internet*, vol. 13, no. 2, p. 48, Feb. 2021.

[8] T. Hewa, A. Braeken, M. Liyanage, and M. Ylianttila, "Fog computing and blockchain-based security service architecture for 5G industrial IoT-enabled cloud manufacturing," *IEEE Trans. Ind. Informat.*, vol. 18, no. 10, pp. 7174–7185, Oct. 2022.

[9] A. Ali, M. Ahmed, M. Imran, and H. A. Khattak, "Security and privacy issues in fog computing," in *Fog Computing: Theory and Practice*. Hoboken, NJ, USA: Wiley, 2020, pp. 105–137.

[10] S. K. Dwivedi, R. Amin, and S. Vollala, "Smart contract and IPFS-based trustworthy secure data storage and device authentication scheme in fog computing environment," *Peer-to-Peer Netw. Appl.*, vol. 16, no. 1, pp. 1–21, Jan. 2023.

[11] N. A. Gupta, M. Bansal, S. Sharma, D. Mehrotra, and M. Kakkar, "Detection of vulnerabilities in blockchain smart contracts: A review," in *Proc. Int. Conf. Comput. Intell., Commun. Technol. Netw. (CICTN)*, Apr. 2023, pp. 558–562.

[12] H. Farooq, A. Altaf, F. Iqbal, J. C. Galán, D. G. Aray, and I. Ashraf, "DrunkChain: Blockchain-based IoT system for preventing drunk driving-related traffic accidents," *Sensors*, vol. 23, no. 12, p. 5388, Jun. 2023.

[13] I. Ashraf, Y. Park, S. Hur, S. W. Kim, R. Alroobaea, Y. B. Zikria, and S. Nosheen, "A survey on cyber security threats in IoT-enabled maritime industry," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 2, pp. 2677–2690, Feb. 2023.

[14] F. Iqbal, A. Altaf, Z. Waris, D. G. Aray, M. A. L. Flores, I. D. L. T. Díez, and I. Ashraf, "Blockchain-modeled edge-computing-based smart home monitoring system with energy usage prediction," *Sensors*, vol. 23, no. 11, p. 5263, Jun. 2023.

[15] R. Chaganti, R. V. Boppana, V. Ravi, K. Munir, M. Almutairi, F. Rustam, E. Lee, and I. Ashraf, "A comprehensive review of denial of service attacks in blockchain ecosystem and open challenges," *IEEE Access*, vol. 10, pp. 96538–96555, 2022.

[16] Y. Verma. (2021). *How Machine Learning CA be Used With Blockchain Technology*. [Online]. Available: https://analyticsindiamag.com/how-machine-learning-can-be-used-with-blockchain-technology/

[17] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *Social Netw. Comput. Sci.*, vol. 2, no. 3, p. 160, May 2021.

[18] H. Taherdoost, "Blockchain and machine learning: A critical review on security," *Information*, vol. 14, no. 5, p. 295, May 2023.

[19] F. Jiang, K. Chao, J. Xiao, Q. Liu, K. Gu, J. Wu, and Y. Cao, "Enhancing smart-contract security through machine learning: A survey of approaches and techniques," *Electronics*, vol. 12, no. 9, p. 2046, Apr. 2023.

[20] H. Wu, H. Dong, Y. He, and Q. Duan, "Smart contract vulnerability detection based on hybrid attention mechanism model," *Appl. Sci.*, vol. 13, no. 2, p. 770, Jan. 2023.

[21] O. Umoren, R. Singh, Z. Pervez, and K. Dahal, "Securing fog computing with a decentralised user authentication approach based on blockchain," *Sensors*, vol. 22, no. 10, p. 3956, May 2022.

[22] G. Liu, J. Wu, and T. Wang, "Blockchain-enabled fog resource access and granting," *Intell. Converged Netw.*, vol. 2, no. 2, pp. 108–114, Jun. 2021.

[23] T. S. Mir, H. B. Liaqat, T. Kiren, M. U. Sana, R. M. Alvarez, Y. Miró, A. E. P. Barrera, and I. Ashraf, "Antifragile and resilient geographical information system service delivery in fog computing," *Sensors*, vol. 22, no. 22, p. 8778, Nov. 2022.

[24] M. Iyapparaja, N. K. Alshammari, M. S. Kumar, S. S. R. Krishnan, and C. L. Chowdhary, "Efficient resource allocation in fog computing using QTCS model," *Comput., Mater. Continua*, vol. 70, no. 2, pp. 2225–2239, 2022.

[25] R. Gupta, M. M. Patel, A. Shukla, and S. Tanwar, "Deep learning-based malicious smart contract detection scheme for Internet of Things environment," *Comput. Electr. Eng.*, vol. 97, Jan. 2022, Art. no. 107583.

[26] W. Wang, J. Song, G. Xu, Y. Li, H. Wang, and C. Su, "ContractWard: Automated vulnerability detection models for Ethereum smart contracts," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1133–1144, Apr. 2021.

[27] R. Kalaria, A. S. M. Kayes, W. Rahayu, and E. Pardede, "A secure mutual authentication approach to fog computing environment," *Comput. Secur.*, vol. 111, Dec. 2021, Art. no. 102483.

[28] M. Soud, I. Qasse, G. Liebel, and M. Hamdaqa, "AutoMESC: Automatic framework for mining and classifying Ethereum smart contract vulnerabilities and their fixes," 2022, *arXiv:2212.10660*.

[29] E. Jung, M. Le Tilly, A. Gehani, and Y. Ge, "Data mining-based Ethereum fraud detection," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Jul. 2019, pp. 266–273.

[30] R. Sing, S. K. Bhoi, N. Panigrahi, K. S. Sahoo, N. Jhanjhi, and M. A. AlZain, "A whale optimization algorithm based resource allocation scheme for cloud-fog based IoT applications," *Electronics*, vol. 11, no. 19, p. 3207, Oct. 2022.

[31] S. Qian, H. Ning, Y. He, and M. Chen, "Multi-label vulnerability detection of smart contracts based on bi-LSTM and attention mechanism," *Electronics*, vol. 11, no. 19, p. 3260, Oct. 2022.

[32] A. Aljofey, A. Rasool, Q. Jiang, and Q. Qu, "A feature-based robust method for abnormal contracts detection in Ethereum blockchain," *Electronics*, vol. 11, no. 18, p. 2937, Sep. 2022.

[33] T. H.-D. Huang, "Hunting the Ethereum smart contract: Color-inspired inspection of potential attacks," 2018, *arXiv:1807.01868*.

[34] Y. Zhang and D. Liu, "Toward vulnerability detection for Ethereum smart contracts using graph-matching network," *Future Internet*, vol. 14, no. 11, p. 326, Nov. 2022.

[35] P. Kumar, R. Kumar, G. P. Gupta, and R. Tripathi, "A distributed framework for detecting DDoS attacks in smart contract-based Blockchain-IoT systems by leveraging fog computing," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 6, p. e4112, Jun. 2021.

[36] B. Jiang, Y. Liu, and W. K. Chan, "ContractFuzzer: Fuzzing smart contracts for vulnerability detection," in *Proc. 33rd IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Sep. 2018, pp. 259–269.

[37] Q. O. M. Hasan, "Machine learning based framework for smart contract vulnerability detection in Ethereum blockchain," Ph.D. dissertation, Rochester Inst. Technol., Rochester, NY, USA, 2023.

[38] A. Mezina and A. Ometov, "Detecting smart contract vulnerabilities with combined binary and multiclass classification," *Cryptography*, vol. 7, no. 3, p. 34, Jul. 2023.

[39] M. Wang and J. Huang, "Detecting Ethereum Ponzi schemes through opcode context analysis and oversampling-based AdaBoost algorithm," *Comput. Syst. Sci. Eng.*, vol. 47, no. 1, pp. 1023–1042, 2023.

[40] S. Ji, C. Huang, P. Zhang, H. Dong, and Y. Xiao, "Ponzi scheme detection based on control flow graph feature extraction," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2023, pp. 585–594.

[41] X. Xie, H. Wang, Z. Jian, Y. Fang, Z. Wang, and T. Li, "Block-Gram: Mining knowledgeable features for efficiently smart contract vulnerability detection," *Digit. Commun. Netw.*, Aug. 2023.

[42] P. N. Srinivasu, R. Panigrahi, A. Singh, and A. K. Bhoi, "Probabilistic buckshot-driven cluster head identification and accumulative data encryption in WSN," *J. Circuits, Syst. Comput.*, vol. 31, no. 17, Nov. 2022, Art. no. 2250303.

[43] J. Sui, L. Chu, and H. Bao, "An opcode-based vulnerability detection of smart contracts," *Appl. Sci.*, vol. 13, no. 13, p. 7721, Jun. 2023.
[44] Forta Docs. (2022). *Detection Bots and Templates Utilizing ML*. [Online]. Available: https://docs.forta.network/en/latest/ml-data-resources/
[45] Forta Network. (2022). *Forta Labelled Datasets*. [Online]. Available: https://github.com/forta-network/labelled-datasets
[46] X. He, T. Yang, and L. Chen, "CTRF: Ethereum-based Ponzi contract identification," *Secur. Commun. Netw.*, vol. 2022, pp. 1–10, Mar. 2022.
[47] S. Palladino. (Jul. 2017). *The Parity Wallet Hack Explained*. [Online]. Available: https://blog.zeppelin.solutions/on-the-parity-wallet-multisighack-405a8c12e8f7
[48] Z. Zhang, B. Zhang, W. Xu, and Z. Lin, "Demystifying exploitable bugs in smart contracts," in *Proc. IEEE/ACM 45th Int. Conf. Softw. Eng. (ICSE)*, May 2023, pp. 615–627.
[49] Wallstreetmojo Team. (2023). *What is DAO Heist*. [Online]. Available: https://www.wallstreetmojo.com/dao-heist/
[50] S. Sayeed, H. Marco-Gisbert, and T. Caira, "Smart contract: Attacks and protections," *IEEE Access*, vol. 8, pp. 24416–24427, 2020.
[51] L. Rhue, "Is sunlight an effective disinfectant? Transparency, reputation, and perceived trust of Ethereum tokens," *Transparency, Reputation, Perceived Trust Ethereum Tokens*, Jul. 2018, doi: 10.2139/ssrn.3218394.
[52] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger (2014)," Tech. Rep., 2014. Accessed: Dec. 23, 2023. [Online]. Available: https://www.scaler.com/topics/nlp/ngrams-in-nlp/
[53] S. Madala. (2023). *Introduction to N-Grams in NLP*. [Online]. Available: https://www.scaler.com/topics/nlp/ngrams-in-nlp/
[54] R. Ahuja, A. Chug, S. Kohli, S. Gupta, and P. Ahuja, "The impact of features extraction on the sentiment analysis," *Proc. Comput. Sci.*, vol. 152, pp. 341–348, Jan. 2019.
[55] X. Zhang and C.-A. Liu, "Model averaging prediction by K-fold cross-validation," *J. Econometrics*, vol. 235, no. 1, pp. 280–301, Jul. 2023.

**TAHMINA EHSAN** is currently with the Department of Information Technology Development, University of Gujrat, Gujrat, Pakistan. Her research interests include cloud computing, fog computing, cyber security, machine learning, and deep learning.

**MUHAMMAD USMAN SANA** received the B.E. degree in information technology from the University of Engineering and Technology Taxila, Pakistan, in 2006, the M.S. degree in communication engineering from the Chalmers University of Technology, Sweden, in 2010, and the Ph.D. degree from Xi'an University of Science and Technology, China, in 2023, with major in security information system and engineering. His research interests include the field of technology and innovation, with a special focus on cloud computing, networks security, wireless sensor networks, social network analysis, next generation networks, socially-aware communication, routing, and switching. He is also a keen and responsible reviewer of top international journals indexed by the Web of Science Core Collection. He was a Technical Program Committee Member of the IEEE International Conference on Energy, Power, and Environment (ICEPE-2023).

**MUHAMMAD USMAN ALI** received the M.S. degree in computer network engineering from the University of Engineering and Technology Taxila, Pakistan, in 2008, and the Ph.D. degree from Yeungnam University, South Korea, in 2018. He is currently with the Department of Computer Science, University of Gujrat, Pakistan. His current research interests include multisensor fusion-based indoor positioning systems, Wi-Fi fingerprinting, indoor navigation and mapping, and computer vision technologies.

**ELIZABETH CARO MONTERO** is currently with Universidad Europea del Atlántico, Santander, Spain. She is also affiliated with Universidade Internacional Iberoamericana, Puerto Rico, and Universidade Internacional do Cuanza, Kuito, Angola.

**EDUARDO SILVA ALVARADO** is currently with Universidad Europea del Atlántico, Santander, Spain. He is also affiliated with Universidade Internacional Iberoamericana, Campeche, Mexico, and Universidad de La Romana, La Romana, Dominica.

**SIROJIDDIN DJURAEV** received the Ph.D. degree in information and communication engineering from Yeungnam University, South Korea, in 2022. He is currently with the Department of Software Engineering, New Uzbekistan University, Tashkent, Uzbekistan. His research interests include ad hoc networks, Sybil attack detection, vehicular applications, and vehicular networks.

**IMRAN ASHRAF** received the M.S. degree in computer science from Blekinge Institute of Technology, Karlskrona, Sweden, in 2010, and the Ph.D. degree in information and communication engineering from Yeungnam University, Gyeongsan-si, South Korea, in 2018. He was a Postdoctoral Fellow with Yeungnam University, where he is currently an Assistant Professor with the Information and Communication Engineering Department. His research interests include indoor positioning and localization in 5G and beyond, indoor location-based services in wireless communication, smart sensors for smart cars, and data analytics.