Machine Learning-Based Improvement of Smart Contract Security in Fog Computing Using Word2vec and Bert

Tahmina Ehsan[1], Muhammad Usman Sana[1]*, Tayybah Kiren[2]*, Alvena Ehsan[1], Mustabeen Aziz[1], Fateha Minahil[1]
[1]University of Gujrat, Pakistan
[2]University of Engineering and Technology Lahore, Pakistan
***Correspondence**: m.usman@uog.edu.pk; tayybah@uet.edu.pk

Citation | Ehsan. T, Sana. M. U, Kiren. T, Ehsan. A, Aziz. M, Minahil. F, "Machine Learning-Based Improvement of Smart Contract Security in Fog Computing Using Word2vec and Bert", IJIST, Special Issue. pp 79-90, March 2025
Received | Feb 14, 2025 **Revised |** Feb 27, 2025 **Accepted |** March 04, 2025 **Published |** March 07, 2025.

Fog computing extends cloud computing services closer to users, improving efficiency and reducing latency. Smart contracts play a key role in authentication and resource access management within this framework. As the adoption of smart contracts in fog computing grows, ensuring their security has become a major challenge. This study enhances smart contract attack detection in fog computing using machine learning techniques. A dataset of 818 smart contracts was collected from "etherscan.io." Feature extraction was performed using Word2Vec and BERT, while feature selection was done using the information gain method. The Random Forest (RF) and Extra Trees Classifier (ETC) achieved the highest accuracy of 0.91 with Word2Vec, while the LightGBM (LGBM) classifier attained 0.90 accuracy using BERT.
These results demonstrate the effectiveness of machine learning models in improving smart contract security within fog computing environments.

Keywords: Fog Computing; Smart Contract; Machine Learning; Security and Feature Extraction

Introduction:

Cloud computing allows users to access computing resources such as servers, storage, software, databases, and applications over the internet instead of relying on local infrastructure. It operates on a pay-as-you-go model, enabling users to scale resources up or down as needed. IoT devices frequently use cloud resources, and their numbers are increasing daily [1]. While the growth of IoT has created many opportunities for cloud computing, it has also introduced challenges, including cost, data management, security, privacy, bandwidth limitations, network congestion, and latency issues. To address these challenges, Cisco introduced fog computing in 2018 as a bridge between cloud computing and edge computing [2].

Fog computing is a distributed model that extends cloud computing to the network's edge, providing computing, storage, and networking services closer to end users and IoT devices [3]. As shown in Figure 1, it creates an intermediate layer between the cloud and edge computing. This fog layer offers computing and networking resources to edge devices, reducing latency compared to traditional cloud computing. Fog computing is essentially an extension of cloud computing [3] and helps mitigate several cloud-related issues. Additionally, blockchain technology [4] is integrated into the fog layer to enhance security and privacy.

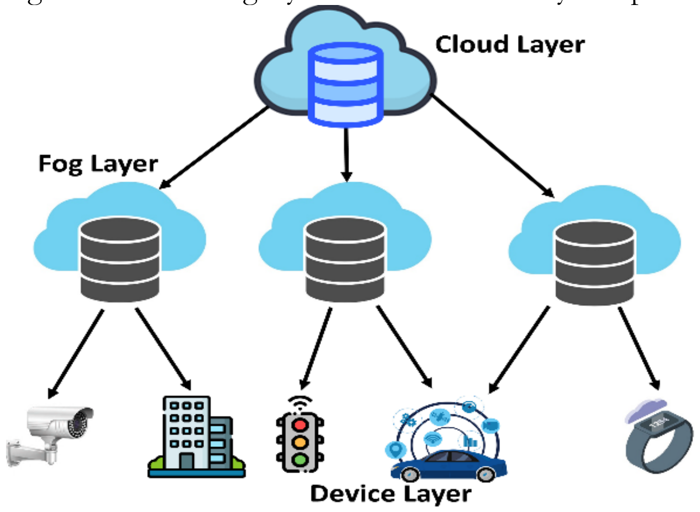

Figure 1. Architecture of fog computing.

Blockchain ensures secure data sharing among fog nodes, IoT devices, and cloud providers while requests, and verification. During user registration, the system assigns a pair of public and private keys, storing the public key within the blockchain. Resource registration lists available fog computing resources and the users who can access them.

To authenticate, a user sends a request using a nonce (a unique identifier) and their public key. The smart contract then follows a challenge-response protocol, sending back the nonce as a challenge. The user signs it with their private key and returns it. The smart contract verifies the signature using the user's public key. If valid, access to fog computing resources is granted [8][9]. Since smart contracts operate independently of external networks, a security breach can affect organizations, miners, and even the entire blockchain network [5][10]. Therefore, researchers must focus on identifying attacks that could compromise smart contract security. In this study, machine learning is applied to detect attacks in smart contracts used for resource access in fog computing.

Framework

Registration Phase

Figure 2 illustrates the user registration process required before accessing resources.

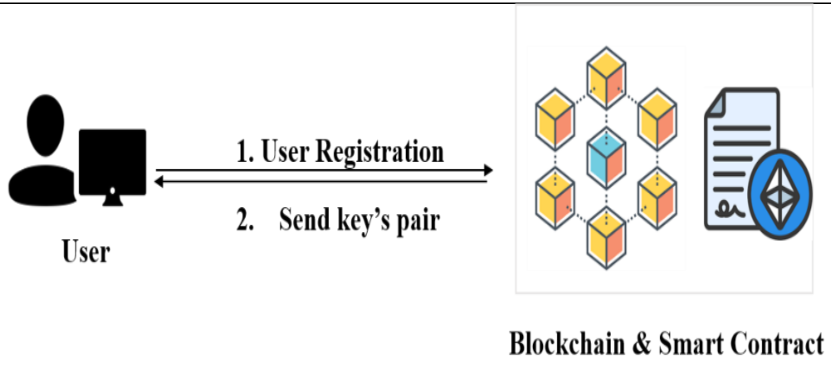

Figure 2. Registration Phase of User.

User Registration:

Users must register through a smart contract to access fog computing resources. Before creating an account, the blockchain-based smart contract verifies and confirms the registration details.

Send Key Pair:

After successful registration, the blockchain generates a cryptographic key pair (public and private keys) for secure authentication. The public key is stored on the blockchain, while the private key is securely sent to the user [11].

Resource Access Granting Phase

Figure 3 illustrates the resource access granting phase, which enables users to access the resources of the fog node.

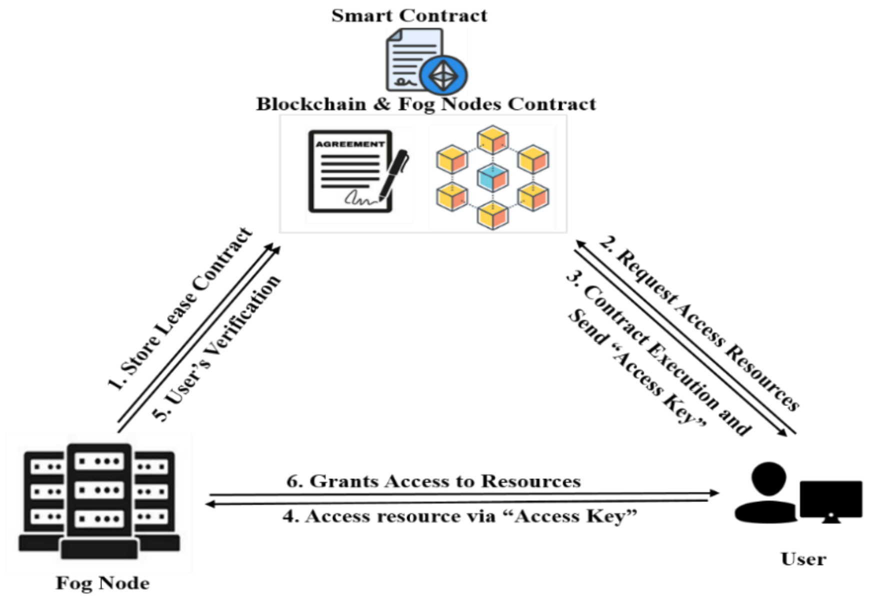
Smart Contract
Blockchain & Fog Nodes Contract
AGREEMENT
1. Store Lease Contract
5. User's Verification
2. Request Access Resources
3. Contract Execution and Send "Access Key"
6. Grants Access to Resources
4. Access resource via "Access Key"
Fog Node
User

Figure 3. Resource Access Granting Phase.

Store Lease Contract:

In this step, the fog node submits its lease contract to the blockchain, defining the terms and conditions for resource access. This ensures secure and transparent resource allocation [11], [12].

Request Access Resources:

A registered user requests access to the fog node's resources. The smart contract verifies the request by matching the user's public key with the registered keys [12] and ensuring compliance with access policies.

Contract Execution and Send “Access key”:

At this stage, after verification, the smart contract retrieves the access contract and securely sends the access key to the user [12].

Access resource via “Access Key”:

After receiving the access key, the user can access the fog node's resources.

User’s Verification:

The fog node verifies the user's identity by matching it with the blockchain ledger [12].

Grants Access to Resources:

After successful verification, access is granted, and a blockchain-based transaction system manages payments for the utilized resources [12].

Unlike previous research, which primarily focused on identifying smart contract vulnerabilities using conventional feature extraction techniques, this study introduces an improved approach by combining Word2Vec and BERT for opcode-based feature extraction. This method enhances the accuracy and efficiency of attack detection in fog computing smart contracts used for resource access and registration. Additionally, the paper evaluates various machine learning classifiers, demonstrating that Random Forest, Extra Trees Classifier, and LightGBM significantly improve security. Compared to previous studies, the proposed framework offers a more reliable, scalable, and precise attack detection technique.

The paper is organized as follows:

- **Section II** reviews related literature on fog computing, blockchain integration, and smart contract security.
- **Section III** outlines the research methodology, including data collection, feature extraction, and model selection.
- **Section IV** presents the experimental results and analysis.
- **Section V** concludes the study and discusses future research directions.

Objectives:

The objectives of this research are:

- To analyze attacks on smart contracts used in fog computing for resource access.
- To identify the most effective machine learning algorithms for detecting and preventing these threats.
- To evaluate the performance of machine learning models using F1 score, accuracy, precision, and recall.
- To improve smart contract attack detection by applying opcode-based feature extraction and selection techniques.

Literature Review:

Fog computing extends cloud computing but also inherits some of its challenges. Due to its proximity to IoT devices, it faces several security and privacy issues. Researchers have explored various solutions, including authorization, access control, and authentication, to ensure secure data transmission in fog computing. In [10], the author introduced a deep learning-based detection method to classify smart contracts as either malicious or safe. Techniques such as Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and Artificial Neural Networks (ANN) were used for classification. The author utilized the BigQuery dataset for binary classification, achieving a maximum accuracy of 99.03%. Additionally, the Receiver Operating Characteristic (ROC) curve was provided to compare the performance of these models.

In [13], the author proposed a system for user registration and authentication in fog computing. This system uses smart contracts for registration and securely stores user information in a ledger. Compared to existing systems, it reduces registration and authentication costs. Additionally, it supports multiple user accounts and compares their costs, also known as gas values. In [6], the author analyzes 49,502 real-time smart contracts for various vulnerabilities, including Callstack, integer overflow, timestamp, Time of Day (TOD), and re-entrancy, achieving a high accuracy rate of 99%. The research converts contract code into bytecode and opcodes. Then, n-gram features are extracted from the opcodes, and machine learning algorithms such as XGBoost, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM) are applied. This approach enhances the speed and accuracy of vulnerability detection.

In [14], the author introduces a framework for identifying and classifying vulnerabilities in smart contracts, such as excessive gas consumption, unfixed compiler versions, implicit visibility levels, inappropriate use of pure functions, unchecked low-level calls, and frozen ether. These vulnerabilities are detected using publicly available datasets, including AutoMESC, which reports a 5.2% occurrence of high-severity vulnerabilities and suggests solutions for addressing them. In [12], the author explains how blockchain technology is used to manage resource access. Smart contracts facilitate this process by eliminating third-party dependencies within the network. These contracts are self-executing lines of code created by organizations, institutions, or other entities. In resource access scenarios, both buyers and sellers rely on smart contracts to define terms and conditions. If any condition is violated, the contract becomes invalid or is terminated.

In [15], the author identifies multiple vulnerabilities in smart contracts, including timestamp issues, re-entrancy, Time of Day (TOD) attacks, integer underflow, and overflow. Using the Bi-LSTM model, the author achieves an accuracy of 88.12%. A total of 5,450 smart contracts were collected from the Etherscan website to detect these vulnerabilities. First, the contract code is converted into opcodes, then a feature matrix is generated, and Bi-LSTM is applied for analysis. In [16], the author classifies smart contracts as normal or abnormal using an ensemble model. A dataset of 1,904 smart contracts was gathered from the Etherscan website. Features were extracted from the source code using TF-IDF, while opcode features were derived using the n-gram technique. Applying the ensemble model, the author achieved an accuracy of 89.67%.

In [17], the author analyzed 5,735 smart contracts, generating semantic trees based on their code and utilizing Graph Neural Networks (GNN) and Graph Matching Networks. The model achieved a 92.63% accuracy in detecting block info dependency vulnerabilities. Other vulnerabilities identified include re-entrancy, block info dependency, timestamp dependency, and TX.Origin issues. In [18], the author detects DDoS attacks in smart contracts using an IoT-based dataset [19]. The BotIoT dataset was used for this purpose. Features extracted from IoT sensors were stored in fog nodes before being transferred to the blockchain via smart contracts. Different classifiers, including Random Forest, Decision Tree, and Support Vector Machine, were applied, achieving an accuracy of 99.9%.

In [15], the author enhances smart contract security through machine learning. A total of 835 smart contracts were analyzed, with 455 classified as safe and 380 as malicious. A binary classification approach was used. The contract source code was first converted into opcodes, and a feature matrix was created. Various machine learning models, including KNN, Random Forest (RF), Decision Tree (DT), Logistic Regression (LR), Support Vector Machine (SVM), and Naïve Bayes (NB), were applied. The RF classifier achieved the highest accuracy at 85%.

In [20], the author presents a comprehensive approach to detecting vulnerabilities in smart contracts using machine learning, automated auditing tools, and reduced manual effort and execution time. The proposed model outperformed traditional methods, achieving an effectiveness rate of 80%. In [15], the author applies a machine-learning approach to detect abnormal smart contracts. A total of 835 smart contracts were collected from the Etherscan website, with 455 classified as normal and 380 as abnormal. Among the abnormal contracts, 327 were identified as scams, while 53 were found to be vulnerable. The dataset was preprocessed before applying various machine learning models for evaluation. The Random Forest (RF) model achieved an accuracy of 0.85, Logistic Regression (LR) reached 0.81, while K-Nearest

Neighbors (KNN) and Decision Trees (DT) scored 0.77. The Support Vector Machine (SVM) and Naïve Bayes (NB) models obtained accuracy results of 0.75 and 0.71, respectively. In [21], the author focuses on identifying Ponzi schemes, a type of fraud that lures new investors with false promises of high returns. A dataset of 3,786 smart contracts was sourced from the Kaggle website, containing four key features: address, opcode, label, and creator. After preprocessing the dataset and extracting relevant features, the author evaluated hybrid classifiers. By combining the strengths of XGBoost and GRU models, the study achieved an impressive accuracy of 96.8%.

The following are the research questions:

- How can machine learning be applied to detect attacks on smart contracts used for registration and resource access in fog computing environments?
- What methods can be used to protect smart contracts from attackers in a fog computing environment?
- How can opcode analysis serve as an efficient and effective feature extraction technique for detecting attacks in smart contracts?
- Which algorithms and models are best suited for detecting attacks on smart contracts used for registration and resource access in fog computing environments?

Material and Methods

This research focuses on attack detection using machine learning with a three-labeled dataset. The methodology is illustrated in Figure 4.

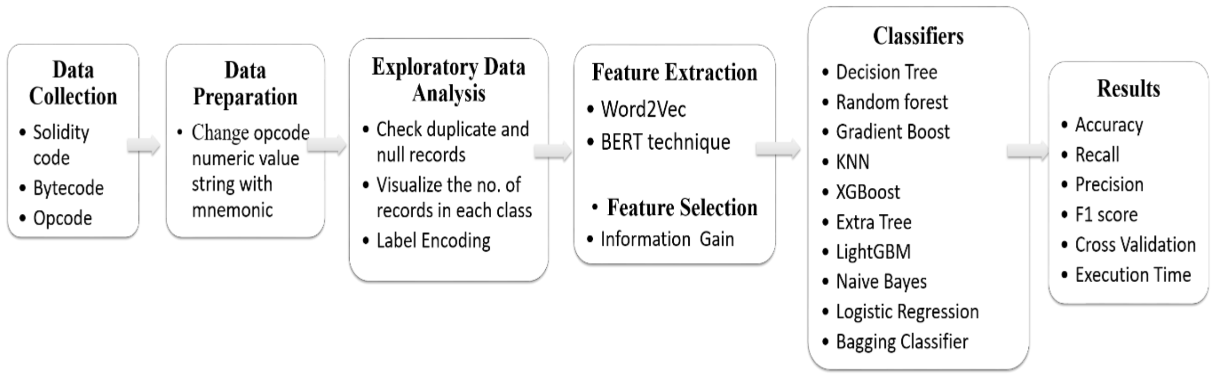

Figure 4. Proposed Methodology

Dataset:

In this research, 818 smart contracts were collected from Ethereum's official website, “etherscan.io,” along with their Solidity code, bytecode, and opcode. The dataset is categorized into three labels: Ponzi (using Forta [22]), Phish-hack [9], and Gambling [23]. It includes 300 smart contracts under the Phish-hack label, 298 under Ponzi, and 220 under Gambling.

Table 1. Composition of Dataset

Label	Number of Smart Contract	Source
Ponzi	298	Forta [22]
Phish-hack	300	[9]
Gambling	220	[23]
Total	**818**	Etherscan.io

Data Preparation:

The opcode of a smart contract contains various hexadecimal values, starting with '0x'. These values are replaced with their corresponding mnemonic representations using [15]. Next, null records are verified, and the labels are encoded as follows: Ponzi ('0'), Phish-hack ('1'), and Gambling ('2').

Feature Extraction:

This research utilizes Word2Vec and BERT techniques to extract features from the opcode of smart contracts.

Feature Selection:

In this research, the information gain technique is used for feature selection, extracting features with a threshold above 0.03.

Classifiers:

Various machine learning classifiers are used, including Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), K-Nearest Neighbors (KNN), Gradient Boost (GB), Bagging Classifier (BC), Naïve Bayes (NB), Extra Trees Classifier (ETC), Light Gradient Boost Machine (LGBM), and Extreme Gradient Boost (XGB).

Result and Discussions:

Various techniques can be used for feature extraction; however, this research employs two methods: Word2Vec and BERT. The results of both are discussed below.

Word2Vec Technique:

In this research, the Word2Vec technique is applied for feature extraction, with the results presented in Table 1.

Table 2. Results Using the Word2Vec Technique

Algorithm	Accuracy	Precision	Recall	F1-Score	CV Accuracy	Execution Time(s)
LR	0.84	0.85	0.84	0.84	0.79	1.67
DT	0.80	0.80	0.80	0.80	0.82	0.09
RF	0.91	0.92	0.91	0.91	0.88	8.78
KNN	0.88	0.91	0.88	0.89	0.85	0.02
GB	0.87	0.88	0.87	0.87	0.87	114.90
BC	0.86	0.87	0.86	0.86	0.86	0.29
NB	0.65	0.67	0.65	0.65	0.69	0.01
ETC	0.91	0.92	0.91	0.91	0.88	2.17
LGBM	0.87	0.87	0.87	0.87	0.87	3.21
XGB	0.85	0.85	0.85	0.85	0.87	7.00

The RF and ETC classifiers delivered the best performance with execution times of 8s and 2s, respectively. The NB classifier achieved an accuracy score of 0.88, while the GB and LGBM classifiers both attained 0.87 accuracy. The BC classifier followed with an accuracy of 0.86, and the XGB classifier reached 0.85 accuracy. The DT and NB classifiers produced lower results, with accuracy scores of 0.80 and 0.65, respectively. Figure 5 illustrates the comparison of all models based on Accuracy, Precision, Recall, and F1-score. According to this figure, the RF and ETC classifiers demonstrated the best performance.

BERT Technique:

In this research, the BERT technique is also used for feature extraction, and its results are presented in Table 2. The LGBM classifier achieved the highest accuracy score of 0.90 with an execution time of 24s. The GB, BC, and ETC classifiers followed closely, each attaining an accuracy of 0.88. The XGB classifier achieved an accuracy of 0.87, while the DT and RF classifiers both reached 0.85. The KNN and LR classifiers obtained accuracy scores of 0.84 and 0.79, respectively. The NB classifier recorded the lowest accuracy, scoring 0.66.

Figure 6 presents a comparison of all models based on Accuracy, Precision, Recall, and F1-score. According to this figure, the LGBM classifier delivers the best performance.

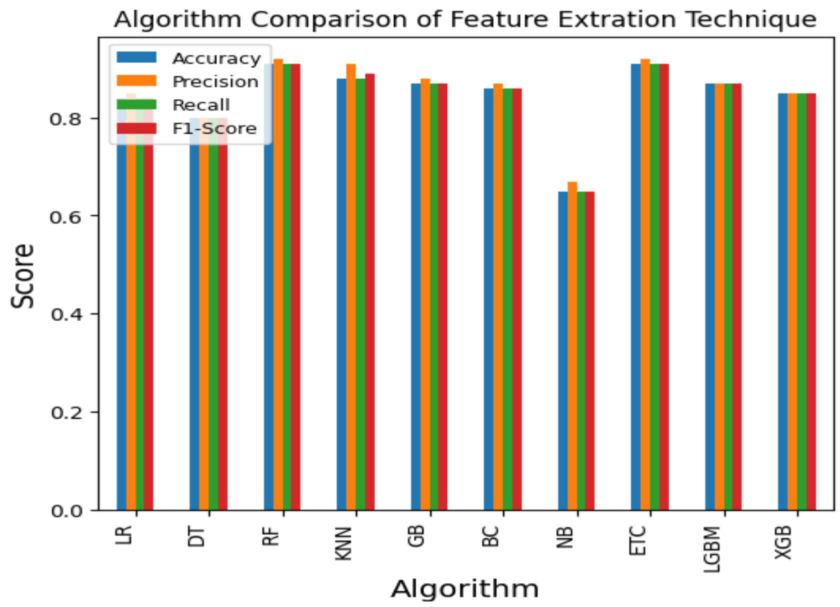

Figure 5. ML algorithms result using the Word2Vec technique.

Table 3. Results Using the Bert Technique

Algorithm	Accuracy	Precision	Recall	F1-Score	CV Accuracy	Execution Time(s)
LR	0.79	0.80	0.79	0.78	0.80	3.41
DT	0.85	0.86	0.85	0.85	0.83	0.52
RF	0.85	0.85	0.85	0.85	0.89	17.54
KNN	0.84	0.88	0.84	0.84	0.86	0.03
GB	0.88	0.88	0.88	0.88	0.87	744.09
BC	0.88	0.89	0.88	0.88	0.86	1.33
NB	0.66	0.68	0.66	0.63	0.70	0.03
ETC	0.88	0.89	0.88	0.88	0.90	10.47
LGBM	0.90	0.91	0.90	0.90	0.89	24.08
XGB	0.87	0.87	0.87	0.87	0.89	46.78

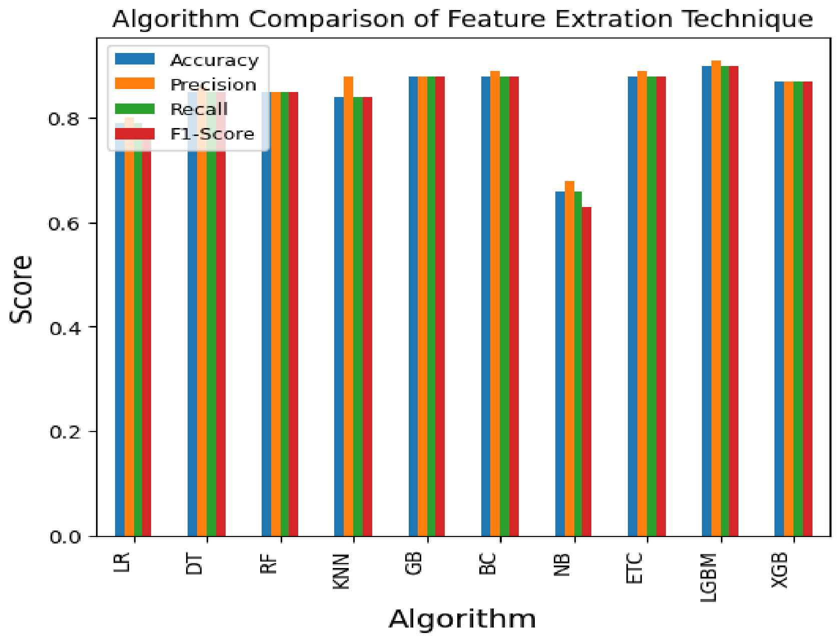

Figure 6. ML algorithms results using the BERT technique.

Fog computing faces several challenges in user registration and resource access, including resource management, security, and privacy concerns within the fog layer. Although smart contracts are used to manage these processes, they remain vulnerable to various attacks. This research enhances security by integrating machine learning for attack detection in smart contracts. Using the Word2Vec feature extraction technique, the Random Forest (RF) and Extra Trees Classifier (ETC) achieved an accuracy of 0.91. Meanwhile, the LightGBM (LGBM) classifier delivered the best performance with an accuracy of 0.90 using the BERT technique. These results highlight the effectiveness of combining machine learning with smart contracts to improve security, scalability, and real-time attack detection. The proposed methodology is highly scalable, enabling it to handle larger systems and diverse datasets, making it ideal for expanding fog computing networks.

Discussion:

The findings of this study demonstrate that machine learning-based security analysis using Word2Vec and BERT for opcode feature extraction significantly enhances smart contract vulnerability detection in fog computing environments. The results show that Random Forest (RF) and Extra Trees Classifier (ETC) achieved the highest accuracy (91%) with Word2Vec, while LightGBM (LGBM) performed best with BERT (90%). This highlights the importance of feature representation in improving attack detection. Compared to traditional methods using n-grams or TF-IDF for feature extraction, the proposed model achieves higher classification accuracy and fewer false positives, making it more effective for large-scale smart contract security monitoring. One key observation is the varying performance of different classifiers in detecting Ponzi schemes, phishing attacks, and gambling-related vulnerabilities. While ensemble models like RF and ETC demonstrated high accuracy and stability, models such as XGBoost and SVM had lower detection rates for specific attack types, indicating that classifier selection plays a crucial role in optimizing smart contract security. BERT-based feature extraction improved the contextual understanding of opcode sequences, making it easier to identify malicious patterns. However, the study did not analyze per-class accuracy, precision, or recall, which could provide deeper insights into each classifier's strengths and weaknesses in detecting different attack types.

Despite promising results, the study focuses solely on opcode-based detection, which, while effective, does not analyze code-level vulnerabilities. Issues such as reentrancy, unchecked external calls, and integer overflow in Solidity-based smart contracts cannot always be identified through opcode analysis alone. Existing research suggests that integrating opcode-based classification with static analysis tools (e.g., Slither, Mythril) could enhance security by detecting both pattern-based and logic-based vulnerabilities. Future research should explore hybrid detection models that combine opcode and code-level analysis to create a more comprehensive security framework.

Another limitation is the model's robustness against adversarial attacks. Machine learning-based security systems are susceptible to adversarial opcode perturbations, where small changes in opcode sequences can mislead classifiers into wrongly identifying malicious contracts as benign. To address this, future work should consider adversarial training techniques or anomaly detection methods to improve resilience against evasion attacks. Additionally, model explainability remains a challenge—techniques such as SHAP or LIME could be used to analyze feature importance and identify opcode sequences that contribute most to classification decisions, increasing trust in the system.

Lastly, the study does not evaluate computational efficiency in real-world fog computing environments. While the proposed model achieves high accuracy, practical deployment requires assessing memory usage, processing latency, and scalability for real-time attack detection. Given the resource constraints of fog computing nodes, lightweight models or edge-optimized ML architectures should be considered to ensure efficient, low-latency security monitoring in decentralized networks.

Table 4. Comparison with Existing studies

Reference	Vulnerabilities Addressed	Techniques Used	Key Contributions
[24]	Arbitrary memory access, integer underflow/overflow, transaction dependency	BERT-ATT-BiLSTM (pre-trained language model)	Enhances accuracy across multiple datasets, outperforming earlier methods that struggle with diverse contract designs.
[25]	Transaction dependency, arbitrary memory access, block dependency, assertion failure, integer underflow, ether block, integer overflow	LSTM, Support Vector Machine (SVM)	Effectively detects vulnerabilities in smart contracts.
[20]	Complex vulnerabilities	Conventional techniques	Improves blockchain application reliability and enables rapid vulnerability identification.
[26]	Ponzi and non-Ponzi attacks	Random Forest	Uses binary-labeled data for detecting smart contract attacks.
[27]	Ponzi and non-Ponzi attacks	AdaBoost Classifier	Detects fraudulent contracts to enhance smart contract security.
[28]	Integer overflow, timestamp, integer underflow, reentrancy, call stack depth, transaction order dependency (TOD)	Naïve Bayes	Speeds up weak contract identification, addressing challenges in analyzing large-scale smart contracts.
Our	Ponzi, Phish Hack, Gambling,	LR, DT, RF, XGB, ETC, GB, KNN, NB, BC, and LGBM.	Detect the different attacks of smart contracts using the machine learning classifiers

Conclusion:

Integrating machine learning with smart contracts can significantly enhance attack detection and prevention in fog computing systems. This integration strengthens security, reducing the risk of data breaches and other threats. Machine learning algorithms, trained on large datasets, can identify behavioral patterns and detect anomalies. This research utilized a binary-labeled dataset to detect attacks in smart contracts. The Word2Vec and BERT techniques were applied for opcode feature extraction, followed by the implementation of various machine learning classifiers. The results show that Random Forest (RF) and Extra Trees Classifier (ETC) achieved the highest accuracy (0.91) using Word2Vec, while LightGBM (LGBM) reached 0.90 accuracy with BERT. Other classifiers also performed well in attack detection. This approach helps mitigate attacks, minimizing data loss and system downtime. Future research could explore hybrid techniques to further improve detection accuracy and address emerging security challenges in smart contracts within fog computing environments.

Acknowledgement. Sincere gratitude is expressed to Dr. Muhammad Usman Sana for invaluable guidance and support throughout this research, which greatly contributed to the completion of this paper.

Author's Contribution. Tahmina Ehsan led the research and experiments. Muhammad Usman Sana supervised and reviewed the work. Tayybah Kiren, Alvena Ehsan, Mustabeen Aziz, and Fateha Minahil contributed to data analysis, literature review, and manuscript preparation.

Conflict of interest. The authors declare no conflict of interest regarding this publication.

References:

[1] W. M. Anwer, S. M. Khan, M. U. Farooq, "Attack Detection in IoT using Machine Learning," *Eng. Technol. Appl. Sci. Res.*, vol. 11, no. 3, pp. 7273–7278, 2021, doi: https://doi.org/10.48084/etasr.4202.

[2] M. A. A.-F. Zain Ashi, Mohammad Al-Fawa'reh, "Fog computing: security challenges and countermeasures," *Int. J. Comput. Appl. Technol.*, vol. 175, no. 15, pp. 30–36, 2020, doi: 10.5120/ijca2020920648.

[3] 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. Priv.*, vol. 4, no. 2, p. e145, Mar. 2021, doi: 10.1002/SPY2.145.

[4] A. Ehsan et al., "Enhanced Anomaly Detection in Ethereum: Unveiling and Classifying Threats With Machine Learning," *IEEE Access*, vol. 12, pp. 176440–176456, 2024, doi: 10.1109/ACCESS.2024.3504300.

[5] A. G. & A. M. Yehia Ibrahim Alzoubi, "A systematic review of the purposes of Blockchain and fog computing integration: classification and open issues," *J. Cloud Comput.*, vol. 11, no. 80, 2022, doi: https://doi.org/10.1186/s13677-022-00353-y.

[6] V. C. M. L. Yao Du, Zehua Wang, "Blockchain-Enabled Edge Intelligence for IoT: Background, Emerging Trends and Open Issues," *Futur. Internet*, vol. 13, no. 2, p. 48, 2021, doi: https://doi.org/10.3390/fi13020048.

[7] 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. Informatics*, vol. 18, no. 10, pp. 7174–7185, Oct. 2022, doi: 10.1109/TII.2022.3140792.

[8] 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, doi: 10.1007/S12083-022-01376-7/METRICS.

[9] H. A. S. F. Alaba, "Smart Contracts Security Application and Challenges: A Review," *Cloud Comput. Data Sci.*, 2023, doi: 10.37256/ccds.5120233271.

[10] R. Gupta, M. M. Patel, and S. T. Shukla, Arpit, "Deep learning-based malicious smart contract detection scheme for internet of things environment," *Comput. Electr. Eng.*, vol. 97, p. 107583, 2022, doi: https://doi.org/10.1016/j.compeleceng.2021.107583.

[11] T. Ehsan et al, "Securing Smart Contracts in Fog Computing: Machine Learning-Based Attack Detection for Registration and Resource Access Granting," *IEEE Access*, vol. 12, pp. 42802–42815, 2024, doi: 10.1109/ACCESS.2024.3378736.

[12] et al Liu C. H., Sun J., Ni W., "Blockchain-enabled secure fog platform: Issues, challenges and solutions," *IEEE Commun. Surv. Tutorials*, vol. 22, no. 4, pp. 2488–2521, 2020.

[13] K. D. Otuekong Umoren, Raman Singh, Zeeshan Pervez, "Securing Fog Computing with a Decentralised User Authentication Approach Based on Blockchain," *Sensors (Basel)*, vol. 22, no. 10, p. 3956, 2022, doi: 10.3390/s22103956.

[14] M. Soud, I. Qasse, G. Liebel, and M. Hamdaqa, "AutoMESC: Automatic Framework for Mining and Classifying Ethereum Smart Contract Vulnerabilities and Their Fixes," *Proc. - 2023 49th Euromicro Conf. Softw. Eng. Adv. Appl. SEAA 2023*, pp. 410–417, 2023, doi: 10.1109/SEAA60479.2023.00068.

[15] V. F. Derek Liu, Francesco Piccoli, “Machine Learning Approach to Identify Malicious Smart Contract Opcodes: A Preliminary Study,” *JPS Conf. Proc.*, 2024, [Online]. Available: https://journals.jps.jp/doi/10.7566/JPSCP.43.011002

[16] Q. Q. Ali Aljofey, Abdur Rasool, Qingshan Jiang, “A Feature-Based Robust Method for Abnormal Contracts Detection in Ethereum Blockchain,” *Electronics*, vol. 11, no. 18, p. 2937, 2022, doi: https://doi.org/10.3390/electronics11182937.

[17] et al Wang T., Zhang S., Liu Y., “AI-driven detection of malicious smart contracts in blockchain networks,” *Neural Comput. Appl.*, vol. 34, no. 10, pp. 7945–7962, 2022.

[18] 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, doi: 10.1002/ETT.4112.

[19] et al Rehman R., Khan N., Uddin F., “Distributed denial-of-service attack detection using machine learning in IoT-enabled smart environments,” *IEEE Internet Things J.*, vol. 9, no. 14, pp. 12687–12698, 2022.

[20] S. El Haddouti, M. Khaldoune, M. Ayache, and M. D. Ech-Cherif El Kettani, “Smart contracts auditing and multi-classification using machine learning algorithms: an efficient vulnerability detection in ethereum blockchain,” *Computing*, vol. 106, no. 9, pp. 2971–3003, Sep. 2024, doi: 10.1007/S00607-024-01314-W/METRICS.

[21] J. U. Fahad Hossain, Mehedi Hasan Shuvo, “A hybrid machine learning approach for improved ponzi scheme detection using advanced feature engineering,” *Int. J. Informatics Commun. Technol.*, vol. 14, no. 1, pp. 50–58, 2025, [Online]. Available: https://ijict.iaescore.com/index.php/IJICT/article/view/21270

[22] Forta Network, “Forta-network/labelled-datasets,” GitHub. Accessed: Mar. 27, 2025. [Online]. Available: https://github.com/forta-network/labelled-datasets

[23] R. Buyya and S. N. Srirama, “Fog and edge computing : principles and paradigms,” p. 471, 2019.

[24] P. L. Fei He, Fei Li, “Enhancing smart contract security: Leveraging pre-trained language models for advanced vulnerability detection,” *IET Blockchain*, 2024, doi: https://doi.org/10.1049/blc2.12072.

[25] Qusai Omar Mustafa Hasan, “Machine Learning Based Framework for Smart Contract Vulnerability Detection,” Rochester Institute of Technology. Accessed: Mar. 27, 2025. [Online]. Available: https://books.google.com.pk/books/about/Machine_Learning_Based_Framework_for_Sma.html?id=EvML0AEACAAJ&redir_esc=y

[26] S. Ji, C. Huang, P. Zhang, H. Dong, and Y. Xiao, “Ponzi Scheme Detection Based on Control Flow Graph Feature Extraction,” *Proc. - 2023 IEEE Int. Conf. Web Serv. ICWS 2023*, pp. 585–594, 2023, doi: 10.1109/ICWS60048.2023.00077.

[27] 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, May 2023, doi: 10.32604/CSSE.2023.039569.

[28] T. L. Xueshuo Xie, Haolong Wang, Zhaolong Jian, Yaozheng Fang, Zichun Wang, “Block-gram: Mining knowledgeable features for efficiently smart contract vulnerability detection,” *Digit. Commun. Networks*, vol. 11, no. 1, pp. 1–12, 2025, doi: https://doi.org/10.1016/j.dcan.2023.07.009.