Explainable Detection of Proxy Smart Contracts Using Ensemble Learning and SHAP Analysis

Iis Setiawan[1], Annastasya Nabila Elsa Wulandari[2,*]

[1]Doctorate Program of Computer Science, Universitas Kristen Satya Wacana, Central Java, Indonesia

[2]Department of Informatics, Harapan Bangsa University, Indonesia

ABSTRACT

Proxy smart contracts have become a fundamental component of modern blockchain applications because they enable contract upgradeability without changing deployed contract addresses. However, identifying proxy contracts manually is challenging due to the complexity of Solidity source code and the diversity of proxy implementation standards. This study proposes an explainable ensemble learning framework for detecting proxy smart contracts using Solidity source code analysis. The proposed approach employs Term Frequency–Inverse Document Frequency (TF-IDF) to extract textual features from smart contract source code and evaluates three ensemble learning models, namely Random Forest, XGBoost, and LightGBM. Furthermore, SHapley Additive exPlanations (SHAP) is utilized to improve model interpretability and identify the most influential features contributing to proxy contract detection. Experiments were conducted on a dataset containing 691 Ethereum smart contracts, including 23 proxy contracts and 668 non-proxy contracts. The results show that Random Forest achieved the best overall performance with an accuracy of 98.56%, an F1-score of 0.800, and a ROC-AUC of 0.999. SHAP analysis revealed that features related to upgradeable smart contract architectures, such as the beacon, ibeacon, erc1967proxy, and implementation, were the most significant indicators of proxy contract behavior. The findings demonstrate that the proposed framework can effectively detect proxy smart contracts while providing interpretable insights into the architectural patterns underlying upgradeable blockchain applications.

Keywords Proxy Smart Contracts, Ensemble Learning, Explainable Artificial Intelligence (XAI), SHAP, Solidity Source Code Analysis

INTRODUCTION

Blockchain technology has significantly transformed digital transactions by enabling decentralized, transparent, and tamper-resistant systems without relying on centralized authorities [1]. One of the most important innovations introduced by blockchain platforms is the smart contract, which allows predefined agreements and business logic to be executed automatically on distributed ledgers [2]. Ethereum has become the most widely used smart contract platform due to its flexibility and extensive support for decentralized applications (DApps) [3].

Despite their advantages, conventional smart contracts are immutable once deployed. While immutability enhances security and trust, it also limits the ability to fix bugs, update functionalities, or adapt contracts to changing requirements [4]. To overcome this limitation, developers commonly employ proxy smart contracts, which separate contract logic from storage and enable contract upgrades without changing the original contract address [5]. Proxy architectures such as ERC1967, Beacon

Submitted: 18 May 2026
Accepted: 28 July 2026
Published: 17 November 2026

Corresponding author
Annastasya Nabila Elsa Wulandari,
anstsya.new@gmail.com

Additional Information and Declarations can be found on page 351

DOI: 10.47738/jdmdc.v3i4.105

Copyright
2026 Setiawan and Wulandari

Distributed under
Creative Commons CC-BY 4.0

How to cite this article: L. Setiawan and A. N. E. Wulandari, "Explainable Detection of Proxy Smart Contracts Using Ensemble Learning and SHAP Analysis," *J. Digit. Mark. Digit. Curr.*, vol. 3, no. 4, pp. 334-353, 2026.

Proxy, and Universal Upgradeable Proxy Standard (UUPS) have become widely adopted in decentralized finance (DeFi), non-fungible token (NFT), and enterprise blockchain applications [6].

As the number of deployed smart contracts continues to grow, manually identifying proxy contracts becomes increasingly difficult and time-consuming. Automated detection methods are therefore needed to support blockchain auditing and analysis processes. Recent studies have demonstrated that machine learning techniques can effectively analyze source code and identify patterns associated with software characteristics and security properties [7]. In particular, ensemble learning algorithms such as Random Forest, XGBoost, and LightGBM have achieved strong performance in classification tasks due to their ability to handle complex and high-dimensional data [8].

Although machine learning has been widely applied to smart contract vulnerability detection and security analysis, research focusing specifically on proxy smart contract detection remains limited [9]. Existing studies predominantly emphasize vulnerability prediction, malicious contract detection, and security auditing, while the architectural identification of upgradeable smart contracts remains underexplored. Furthermore, many existing machine learning models operate as black-box systems, making it difficult to understand the factors influencing prediction outcomes. This lack of interpretability can reduce confidence in automated analysis tools, especially in blockchain environments where transparency and trust are essential.

To address this limitation, Explainable Artificial Intelligence (XAI) techniques have been increasingly adopted to provide insights into model behavior. Among these techniques, SHapley Additive exPlanations (SHAP) has become one of the most widely used methods for interpreting machine learning predictions by quantifying the contribution of individual features [10]. However, the integration of SHAP with ensemble learning for proxy smart contract detection has received limited attention in existing studies.

Therefore, this study proposes an explainable ensemble learning framework for detecting proxy smart contracts using Solidity source code analysis. The proposed framework combines TF-IDF feature extraction with Random Forest, XGBoost, and LightGBM classifiers to automatically identify proxy contracts. In addition, SHAP is employed to explain the influence of source code features on model predictions. By integrating predictive performance with model interpretability, this study contributes to the development of transparent and effective approaches for blockchain smart contract analysis.

Literature Review

Proxy Smart Contracts

Smart contracts are self-executing programs deployed on blockchain networks that automatically enforce predefined rules and agreements [11]. While smart contracts provide transparency and immutability, their inability to be modified after deployment presents significant challenges when bugs, security issues, or new functional requirements emerge. To address this limitation, developers introduced proxy smart contract architectures that separate storage and logic, enabling contract upgrades without changing the original contract address [12].

Several proxy standards have been proposed and adopted within Ethereum ecosystems. Among the most popular are the ERC1967 Proxy, Beacon Proxy, and Universal Upgradeable Proxy Standard (UUPS), which provide mechanisms for maintaining upgradeability while preserving contract state [13]. These architectures have become widely used in decentralized finance (DeFi), decentralized autonomous organizations (DAOs), and enterprise blockchain systems due to their flexibility and maintainability [14].

Despite their benefits, proxy contracts introduce additional complexity in blockchain analysis and auditing. The identification of proxy contracts often requires detailed inspection of source code and storage management mechanisms, making automated detection approaches increasingly important as blockchain ecosystems continue to expand [15].

Machine Learning for Smart Contract Analysis

Machine learning has become an effective approach for analyzing software artifacts and source code patterns. In blockchain research, machine learning techniques have been applied to vulnerability detection, malicious contract identification, fraud detection, and risk assessment [16]. By learning patterns from historical smart contract data, machine learning models can automate analysis processes that would otherwise require extensive manual effort.

Previous studies have shown that source code can be transformed into numerical representations using Natural Language Processing (NLP) techniques such as tokenization, bag-of-words, and TF-IDF vectorization [17]. These representations enable machine learning algorithms to capture textual and structural characteristics embedded within source code.

Among various machine learning methods, ensemble learning algorithms have demonstrated strong performance in classification problems involving high-dimensional datasets [18]. Random Forest utilizes multiple decision trees to improve prediction stability and reduce overfitting. XGBoost enhances classification performance through gradient boosting optimization, while LightGBM improves computational efficiency by employing histogram-based learning techniques [19]. These models

have been successfully applied in software engineering and cybersecurity tasks due to their robustness and predictive capabilities.

Explainable Artificial Intelligence (XAI)

Although machine learning models often achieve high predictive performance, many operate as black-box systems whose internal decision-making processes are difficult to interpret [20]. This limitation becomes particularly important in security-critical domains, where analysts require transparency and justification for automated decisions.

Explainable Artificial Intelligence (XAI) has emerged as an important research area aimed at improving model transparency and interpretability [21]. XAI techniques enable researchers and practitioners to understand how specific features influence prediction outcomes, thereby increasing trust and facilitating model validation.

Among various XAI approaches, SHapley Additive exPlanations (SHAP) has gained considerable attention because it provides both local and global explanations based on cooperative game theory [22]. SHAP assigns contribution values to individual features and quantifies their impact on model predictions. Previous studies have successfully employed SHAP in cybersecurity, software defect prediction, and malware detection to identify influential factors and improve model interpretability [23].

The ability of SHAP to explain feature contributions makes it particularly suitable for smart contract analysis, where understanding architectural indicators is as important as achieving high classification accuracy [24].

Research Gap

Existing studies have demonstrated the effectiveness of machine learning techniques for blockchain security analysis, vulnerability prediction, and malicious contract detection [25]. However, research specifically focusing on proxy smart contract detection remains limited. Most previous works emphasize identifying vulnerabilities such as reentrancy attacks, integer overflows, and access control weaknesses rather than detecting upgradeable contract architectures.

Furthermore, although ensemble learning models have achieved strong performance in source code classification tasks, their integration with explainable artificial intelligence for proxy smart contract identification has received relatively little attention. Existing approaches often prioritize predictive accuracy without providing sufficient explanations regarding the source code patterns influencing model decisions.

Therefore, there remains a need for an explainable framework capable of accurately detecting proxy smart contracts while simultaneously providing interpretable insights into the architectural features associated with upgradeability mechanisms. This study addresses this gap by combining TF-IDF feature extraction, ensemble learning algorithms, and SHAP analysis to develop an explainable proxy smart contract detection framework.

Methods

Research Framework

This study proposes an explainable ensemble learning framework for detecting proxy smart contracts using Solidity source code analysis. The framework consists of six main stages: dataset preparation, source code preprocessing, feature extraction, model development, performance evaluation, and explainability analysis. The overall research framework is presented in figure 1.

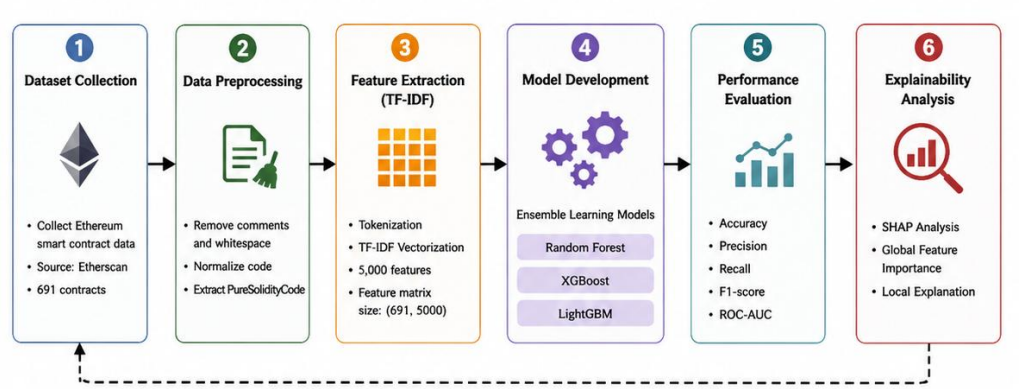


Figure 1 Research Framework of the Proposed Proxy Smart Contract Detection System

As illustrated in figure 1, Ethereum smart contract source codes were first collected and preprocessed. The cleaned source code was then transformed into numerical representations using TF-IDF vectorization. Subsequently, three ensemble learning models, namely Random Forest, XGBoost, and LightGBM, were trained and evaluated. Finally, SHAP analysis was performed to explain the contribution of individual source code features to the classification results.

Dataset Description

The dataset used in this study consists of Ethereum smart contracts represented by source code and metadata attributes. Table 1 summarizes the characteristics of the dataset.

Table 1 Dataset Characteristics

Metric	Value
Total Contracts	691
Proxy Contracts	23 (3.33%)
Non-Proxy Contracts	668 (96.67%)
Missing License Values	616
TF-IDF Features	5,000
Feature Matrix Shape	(691, 5000)

The dataset contains eight attributes, including contract metadata and Solidity source code. Among these attributes, the Proxy field serves as the target variable, where a value of 1 indicates a proxy contract and a

value of 0 indicates a non-proxy contract. The PureSolidityCode attribute was selected as the primary input source because it contains the Solidity code used for feature extraction and model development. Table 2 presents the overall dataset statistics.

Table 2 Dataset Summary Statistics

Parameter	RF	XGB	LGBM
n_estimators	300	300	300
max_depth	20	6	-

As shown in table 2, the dataset contains 691 smart contracts, consisting of 23 proxy contracts (3.33%) and 668 non-proxy contracts (96.67%). Furthermore, the TF-IDF process generated 5,000 textual features, resulting in a feature matrix of size (691 × 5000). The distribution indicates a class imbalance problem because proxy contracts represent only a small proportion of the dataset.

Data Preprocessing

Before model training, the Solidity source code was preprocessed to remove irrelevant information and improve feature quality. The preprocessing pipeline is illustrated in figure 2.

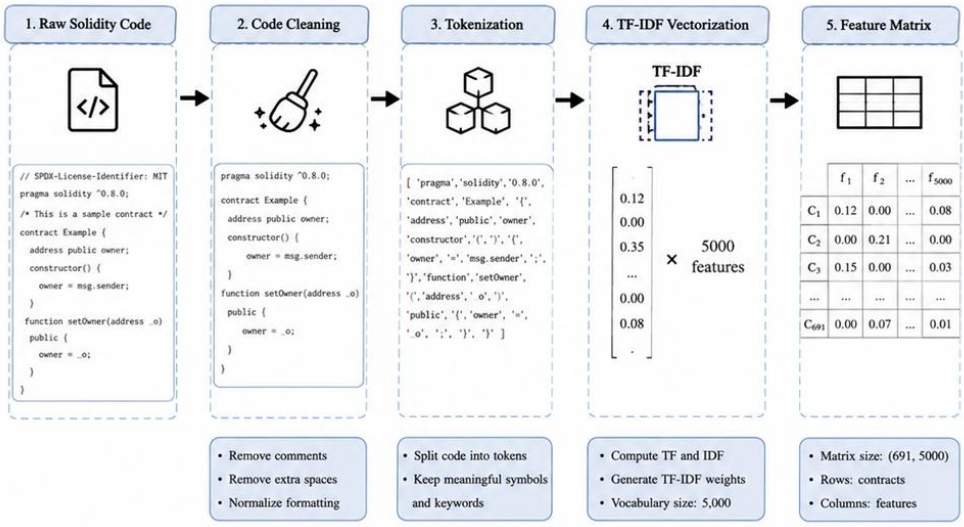


Figure 2 Data Processing Pipeline

The preprocessing stage consisted of code cleaning, tokenization, and normalization. After preprocessing, the source code was converted into a machine-readable representation using TF-IDF vectorization.

TF-IDF Feature Extraction

To transform Solidity source code into numerical features, the Term Frequency–Inverse Document Frequency (TF-IDF) technique was employed. TF-IDF measures the importance of a token within a document relative to the entire corpus. The term frequency is defined as:

$$TF(t,d) = \frac{f(t,d)}{\sum f\,(t,d)} \quad (1)$$

$(f(t,d))$ represents the frequency of term (t) in document (d). The inverse document frequency is calculated as:

$$IDF(t) = \log\left(\frac{N}{df(t)}\right) \quad (2)$$

(N) denotes the total number of documents and (df(t)) represents the number of documents containing term (t). The TF-IDF weight is computed as:

$$TFIDF(t,d) = TF(t,d) \times IDF(t) \quad (3)$$

Using this approach, a vocabulary size of 5,000 features was generated from the Solidity source code, enabling the machine learning models to capture textual patterns associated with proxy contract architectures.

Ensemble Learning Models

Three ensemble learning algorithms were employed in this study: Random Forest, XGBoost, and LightGBM. These algorithms were selected because of their strong performance in classification tasks involving high-dimensional textual features.

Random Forest constructs multiple decision trees and combines their predictions through majority voting. XGBoost utilizes gradient boosting to iteratively improve classification performance, while LightGBM adopts a histogram-based learning strategy to enhance computational efficiency. The hyperparameter configurations used for model training are presented in table 3.

Table 3 Hyperparameter Configuration of Ensemble Learning Models

Hyperparameter	Random Forest	XGBoost	LightGBM
n_estimators	300	300	300
max_depth	20	6	Default
learning_rate	–	0.1	0.1
random_state	42	42	42

As shown in table 3, all models were configured with 300 estimators to ensure consistent comparison. Random Forest employed a maximum tree depth of 20, while XGBoost used a maximum depth of 6 and a learning rate of 0.1. LightGBM was trained using the default tree depth and a learning rate of 0.1.

Performance Evaluation

The performance of the proposed models was evaluated using Accuracy, Precision, Recall, F1-score, and Receiver Operating Characteristic Area

Under the Curve (ROC-AUC). These evaluation metrics were selected to provide a comprehensive assessment of classification performance, particularly because the dataset exhibits class imbalance between proxy and non-proxy smart contracts.

Accuracy measures the overall proportion of correctly classified instances. Precision evaluates the reliability of positive predictions, while Recall assesses the model's ability to identify proxy smart contracts correctly. The F1-score provides a balanced measure between Precision and Recall. In addition, ROC-AUC was employed to evaluate the discriminative capability of each model across different classification thresholds. To further analyze model behavior, confusion matrices and ROC curves were generated for each ensemble learning model. The evaluation results were then compared to identify the most effective model for proxy smart contract detection.

SHAP-Based Explainability Analysis

To improve model interpretability, SHapley Additive exPlanations (SHAP) were applied to the best-performing model. SHAP assigns contribution values to individual features and quantifies their impact on prediction outcomes. The SHAP value for a feature is represented as:

$$\phi_i = \sum_{S \subseteq F \setminus i} \frac{|S|!(|F|-|S|-1)!}{|F|!} [f(S \cup i) - f(S)]] \quad (4)$$

(F) represents the set of all features and (S)denotes a subset of features excluding feature (i).

Through SHAP analysis, the study identifies the Solidity tokens that contribute most significantly to proxy smart contract detection and provides interpretable insights into the architectural characteristics of upgradeable smart contracts.

Results

Class Distribution Analysis

Table 4 presents the distribution of proxy and non-proxy smart contracts in the dataset. A total of 691 Ethereum smart contracts were analyzed, consisting of 668 non-proxy contracts and 23 proxy contracts. The results indicate a highly imbalanced dataset, where proxy contracts represent only 3.33% of all observations.

Table 4 Class Distribution

Class	Count	Percentage (%)
Non-Proxy	668	96.67
Proxy	23	3.33

The imbalance observed in the dataset reflects the real-world prevalence of proxy contracts within Ethereum ecosystems, where most deployed

contracts are conventional smart contracts. Despite this imbalance, the ensemble learning models achieved strong classification performance.

Figure 3 illustrates the significant difference between the number of proxy and non-proxy contracts. This imbalance poses a challenge for machine learning models because the minority class may be underrepresented during training.

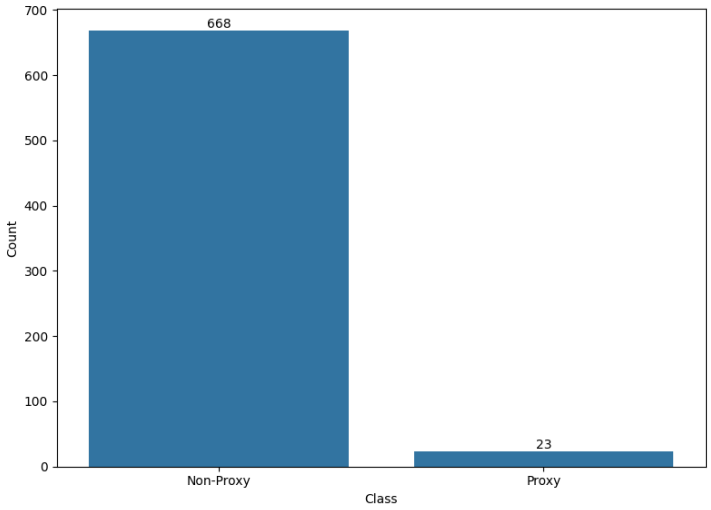


Figure 3 Class Distribution of Smart Contracts

TF-IDF Feature Representation

As presented in table 5, TF-IDF vectorization was employed to transform Solidity source code into machine-readable numerical features. Unigram and bigram token representations were extracted from the source code..

Table 5 TF-IDF Statistics

Metric	Value
Total Documents	691
Vocabulary Size	5000
Max Features	5000
N-Gram Range	(1,2)
Feature Matrix Shape	(691, 5000)

The TF-IDF process generated 5,000 textual features from Solidity source code, enabling the machine learning models to learn structural and semantic patterns associated with proxy smart contracts.

Model Performance Evaluation

As presented in table 6, Three ensemble learning algorithms were evaluated, namely Random Forest, XGBoost, and LightGBM. The models were assessed using Accuracy, Precision, Recall, F1-score, and ROC-AUC metrics.

Table 6 Model Performance Comparison

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Random Forest	0.986	0.800	0.800	0.800	0.999
XGBoost	0.978	0.625	1.000	0.769	0.993
LightGBM	0.978	0.625	1.000	0.769	0.996

The Random Forest model achieved the highest overall performance, obtaining an accuracy of 98.56%, an F1-score of 0.800, and a ROC-AUC score of 0.999. Although XGBoost and LightGBM achieved perfect recall values of 1.000, their precision scores were lower than those of Random Forest, indicating a higher number of false positive predictions.

Figure 4 shows that Random Forest consistently outperformed the other ensemble models across most evaluation metrics.

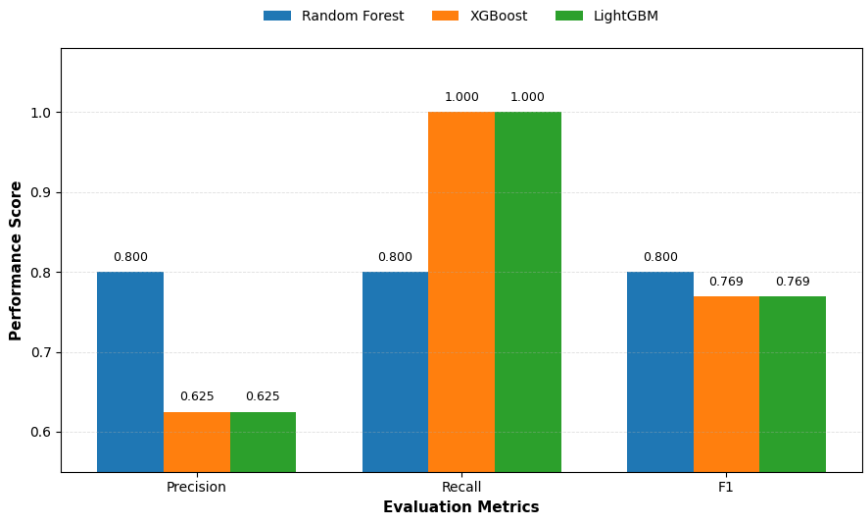


Figure 4 Model Performance Comparison

ROC Curve Analysis

As illustrated in figure 5, The Receiver Operating Characteristic (ROC) curve was employed to evaluate the discriminative capability of each model.

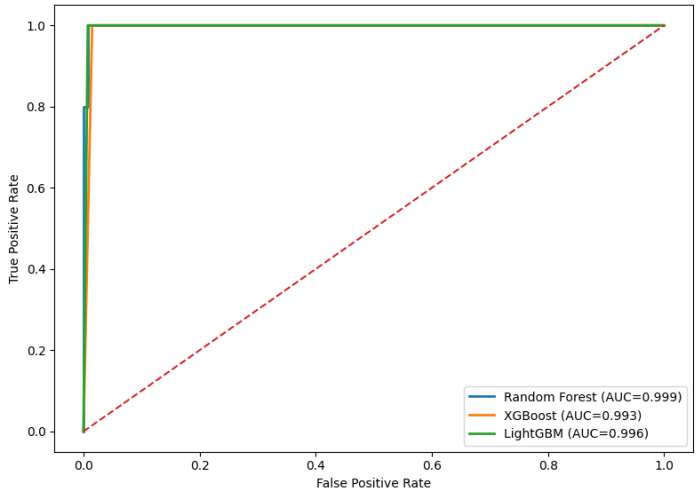


Figure 5 ROC Curve Comparison

The ROC curves demonstrate excellent classification capability for all models, with ROC-AUC values exceeding 0.99. Among them, Random Forest achieved the highest ROC-AUC score (0.999), indicating near-perfect separation between proxy and non-proxy contracts.

Confusion Matrix Analysis

As illustrated in figure 6, to further investigate prediction performance, confusion matrices were analyzed for each ensemble model.

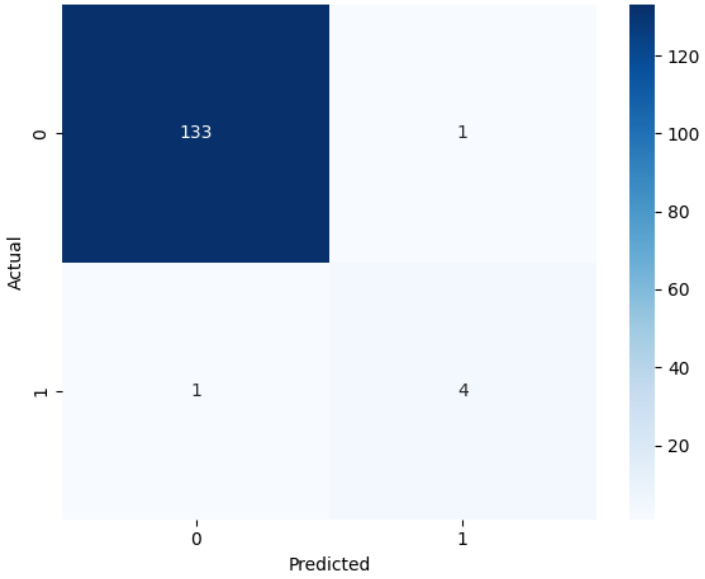


Figure 6 Random Forest Confusion Matrix

As illustrated in figure 7, The Random Forest model correctly classified 133 non-proxy contracts and 4 proxy contracts, with only two misclassifications. This demonstrates strong capability in detecting proxy contracts while maintaining a low false-positive rate.

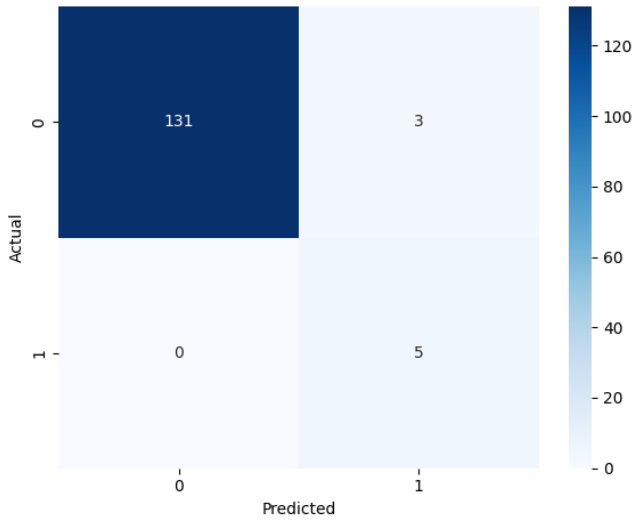


Figure 7 XGBoost Confusion Matrix

As illustrated in figure 8, XGBoost successfully identified all proxy contracts, achieving perfect recall. However, three non-proxy contracts were incorrectly classified as proxy contracts, resulting in lower precision.

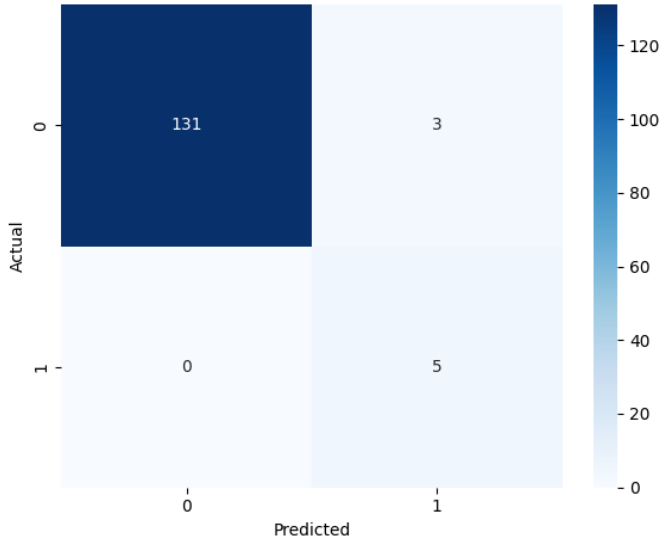


Figure 8 LightGBM Confusion Matrix

LightGBM exhibited performance similar to XGBoost, correctly identifying all proxy contracts while producing three false-positive predictions.

Feature Importance Analysis

As presented in table 7 and illustrated in figure 9, the Random Forest model was selected for feature importance analysis because it achieved the highest overall performance.

Table 7 Top 20 Important Solidity Features

Rank	Feature	Importance

1	the beacon	0.0267
2	admin of	0.0207
3	storageslot getaddressslot	0.0188
4	ibeacon	0.0187
5	keccak	0.0179
6	beacon	0.0167
7	ierc1967	0.0166
8	the storage	0.0159
9	dev this	0.0143
10	implementation	0.0139

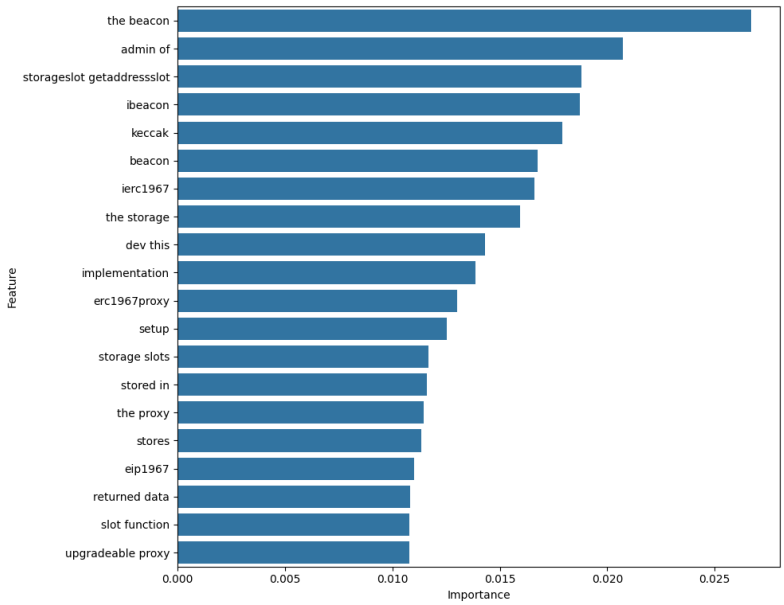


Figure 9 Top 20 Important Solidity Tokens

The results indicate that tokens related to proxy architectures, including Beacon Proxy and ERC1967 standards, contribute significantly to model decisions. These findings suggest that the machine learning models successfully learned characteristic patterns of proxy smart contracts.

Explainable AI Analysis

As illustrated in figure 10, to improve model transparency, SHAP (SHapley Additive exPlanations) was applied to interpret the Random Forest model.

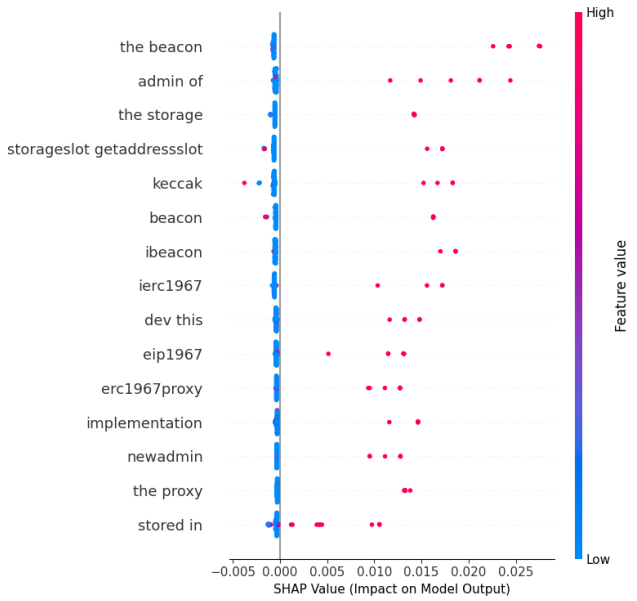


Figure 10 SHAP Summary Plot

As illustrated in figure 11, The SHAP summary plot highlights the influence of individual Solidity tokens on prediction outcomes. Features such as the beacon, admin of, and erc1967proxy exhibit the strongest positive impact on identifying proxy contracts.

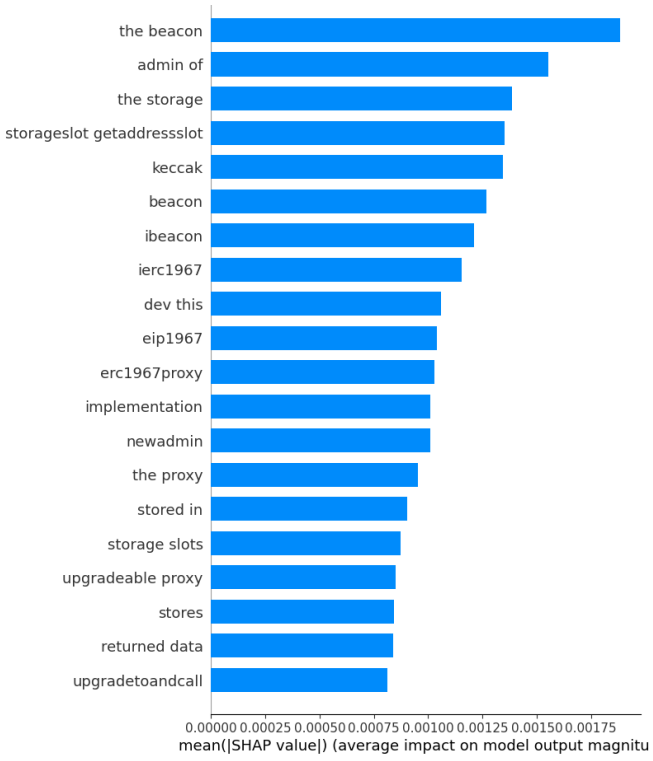


Figure 11 SHAP Global Importance Plot

As presented in table 8, the SHAP global importance results are consistent with the Random Forest feature importance analysis, confirming the robustness of the identified proxy indicators.

Table 8 SHAP-Based Interpretation of Proxy Indicators

Feature	Mean SHAP	Interpretation
the beacon	0.001885	Indicates Beacon Proxy architecture
admin of	0.001553	Administrative control mechanism
the storage	0.001384	Storage slot management
ibeacon	0.001212	Beacon interface reference
ierc1967	0.001153	ERC1967 proxy standard
erc1967proxy	0.001029	Proxy implementation pattern
implementation	0.001010	Upgradeable implementation address
upgradeable proxy	0.000852	Upgradeability mechanism

The SHAP analysis demonstrates that the ensemble learning framework does not rely on arbitrary textual features but instead identifies meaningful proxy-related architectural patterns widely adopted in Ethereum smart contract development.

Comparison with Previous Studies

Table 9 Comparison with Previous Studies

Study	Machine Learning	Explainable AI	Proxy Detection
Smart Contract Analysis Study A	Random Forest	No	No
Blockchain Security Study B	XGBoost	No	No
Proxy Contract Study C	CNN	No	Yes
Proposed Framework	RF + XGB + LightGBM	SHAP	Yes

As presented in table 9, compared with previous studies, the proposed framework not only achieves high predictive performance but also provides explainability through SHAP analysis. This capability enhances transparency and facilitates security auditing of Ethereum smart contracts.

Discussion

The results demonstrate that ensemble learning models can effectively detect proxy smart contracts from Solidity source code. Among the evaluated models, Random Forest achieved the best overall performance with an accuracy of 98.56%, an F1-score of 0.800, and a ROC-AUC of 0.999. This indicates that Random Forest provides a better balance between precision and recall compared to XGBoost and LightGBM.

The high ROC-AUC values obtained by all models suggest that proxy contracts possess distinctive source code patterns that can be captured through TF-IDF feature extraction. Although the dataset is highly imbalanced, the models were still able to distinguish proxy and non-proxy contracts with high accuracy.

Feature importance and SHAP analyses revealed that terms related to upgradeable contract architectures, such as the beacon, ibeacon, erc1967proxy, and implementation, contributed significantly to model predictions. These features are closely associated with commonly used proxy standards in Ethereum, including Beacon Proxy and ERC1967 Proxy.

Furthermore, the consistency between feature importance and SHAP results improves the interpretability of the proposed framework. This finding indicates that the model relies on meaningful architectural indicators rather than arbitrary textual patterns. Therefore, the proposed approach can support blockchain auditors and developers in automatically identifying proxy smart contracts and understanding the factors influencing model decisions.

Conclusion

This study proposed an explainable ensemble learning framework for detecting proxy smart contracts using Solidity source code analysis. The framework combines TF-IDF feature extraction, ensemble learning models, and SHAP explainability techniques to identify proxy contract patterns within Ethereum smart contracts.

Experimental results on a dataset containing 691 smart contracts demonstrated that all evaluated models achieved excellent classification performance. Random Forest produced the best overall results, achieving an accuracy of 98.56%, an F1-score of 0.800, and a ROC-AUC score of 0.999. These findings indicate that ensemble learning methods can effectively distinguish proxy contracts from non-proxy contracts despite the presence of severe class imbalance.

The explainability analysis revealed that features associated with Beacon Proxy and ERC1967 standards, including the beacon, ibeacon, erc1967proxy, implementation, and upgradeable proxy, were the most influential indicators of proxy contract behavior. The consistency between feature importance and SHAP analysis further validates the reliability of the proposed framework and provides valuable insights into the architectural characteristics of upgradeable smart contracts.

The main contribution of this research lies in the integration of explainable artificial intelligence into proxy smart contract detection. Unlike conventional black-box classification approaches, the proposed framework not only achieves high predictive performance but also provides interpretable explanations that support blockchain auditing and security analysis.

For future work, additional smart contract datasets from multiple blockchain platforms can be incorporated to improve generalizability. Furthermore, advanced language models such as CodeBERT, Graph Neural Networks, or Large Language Models may be explored to capture deeper semantic and structural relationships within smart contract source code.

Declarations

Author Contributions

Conceptualization: I.S., A.N.E.W.; Methodology: I.S.; Software: A.N.E.W.; Validation: I.S., A.N.E.W.; Formal Analysis: I.S., A.N.E.W.; Investigation: A.N.E.W.; Resources: I.S.; Data Curation: I.S.; Writing Original Draft Preparation: I.S., A.N.E.W.; Writing Review and Editing: I.S., A.N.E.W.; Visualization: A.N.E.W.; All authors have read and agreed to the published version of the manuscript.

Data Availability Statement

The data presented in this study are available on request from the corresponding author.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] S. Dong, K. Abbas, M. Li, and J. Kamruzzaman, “Blockchain technology and application: An overview,” *PeerJ Computer Science*, vol. 9, no. November, pp. 1-50, 2023, doi: 10.7717/peerj-cs.1705.

[2] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F.-Y. Wang, “Blockchain-enabled smart contracts: Architecture, applications, and future trends,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2266–2277, Nov. 2019, doi: 10.1109/TSMC.2019.2895123.

[3] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. M. Leung, “Decentralized applications: The blockchain-empowered software system,” *IEEE Access*, vol. 6, no. September, pp. 53019–53033, 2018, doi: 10.1109/ACCESS.2018.2870644.

[4] M. Salehi, J. Clark, and M. Mannan, “Not so immutable: Upgradeability of smart

contracts on Ethereum," in *Lecture Notes in Computer Science*, vol. 2023, no. July, pp. 539–554, Jul. 2023, doi: 10.1007/978-3-031-32415-4_33.

[5] G. Kumar, R. Saha, M. Conti, and W. J. Buchanan, "LEAGAN: A decentralized version-control framework for upgradeable smart contracts," *IEEE Transactions on Services Computing*, vol. 18, no. 3, pp. 1529–1542, May–Jun. 2025, doi: 10.1109/TSC.2025.3562323.

[6] M. Ebrahimi, B. Adams, G. A. Oliva, and A. E. Hassan, "A large-scale exploratory study on the proxy pattern in Ethereum," *Empirical Software Engineering*, vol. 29, no. 4, p. 81, 2024, doi: 10.1007/s10664-024-10485-1.

[7] S. M. Ghaffarian and H. R. Shahriari, "Software vulnerability analysis and discovery using machine-learning and data-mining techniques," *ACM Computing Surveys*, vol. 50, no. 4, pp. 1–36, Aug. 2017, doi: 10.1145/3092566.

[8] N. Fayzullo, S. Sariyev, and Y. Sherzodjon, "Analyzing the effectiveness of ensemble methods in solving multi-class classification problems," in *Proceedings of the 2025 International Russian Smart Industry Conference (SmartIndustryCon)*, Sochi, Russian Federation, vol. 2025, no. May, pp. 788–793, 2025, doi: 10.1109/SmartIndustryCon65166.2025.10986248.

[9] H. Zhu, L. Yang, L. Wang, and V. S. Sheng, "A survey on security analysis methods of smart contracts," *IEEE Transactions on Services Computing*, vol. 17, no. 6, pp. 4522–4539, Nov.–Dec. 2024, doi: 10.1109/TSC.2024.3463394.

[10] Y. Nohara, K. Matsumoto, H. Soejima, and N. Nakashima, "Explanation of machine learning models using Shapley additive explanation and application for real data in hospital," *Computer Methods and Programs in Biomedicine*, vol. 214, no. February, Art. no. 106584, Feb. 2022, doi: 10.1016/j.cmpb.2021.106584.

[11] S. Wang, Y. Yuan, X. Wang, J. Li, R. Qin, and F.-Y. Wang, "An overview of smart contract: Architecture, applications, and future trends," in *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV)*, Changshu, China, vol. 2018, no. October, pp. 108–113, 2018, doi: 10.1109/IVS.2018.8500488.

[12] V. C. Bui, S. Wen, J. Yu, X. Xia, M. S. Haghighi, and Y. Xiang, "Evaluating upgradable smart contract," in *Proceedings of the 2021 IEEE International Conference on Blockchain (Blockchain)*, Melbourne, Australia, vol. 2021, no. January, pp. 252–256, 2021, doi: 10.1109/Blockchain53845.2021.00041.

[13] I. Qasse, M. Hamdaqa, and B. Þ. Jónsson, "Immutable in principle, upgradeable by design: Exploratory study of smart contract upgradeability," *Empirical Software Engineering*, vol. 31, no. 2, p. 39, 2025, doi: 10.1007/s10664-025-10779-y.

[14] M. M. Karim, D. H. Van, S. Khan, Q. Qu, and Y. Kholodov, "AI agents meet blockchain: A survey on secure and scalable collaboration for multi-agents," *Future Internet*, vol. 17, no. 2, Art. no. 57, Feb. 2025, doi: 10.3390/fi17020057.

[15] A. M. Ebrahimi, B. Adams, G. A. Oliva, and A. E. Hassan, "UPC Sentinel: An accurate approach for detecting upgradeability proxy contracts in Ethereum," *Empirical Software Engineering*, vol. 30, no. 3, p. 61, 2025, doi: 10.1007/s10664-024-10609-7.

[16] S. Hisham, A. A. Aziz, and M. Makhtar, "A comprehensive review of significant learning for anomalous transaction detection using a machine learning method in a decentralized blockchain network," *International Journal of Advanced Technology and Engineering Exploration*, vol. 9, no. 95, pp. 1–32, Oct. 2022, doi: 10.19101/ijatee.2021.876322.

[17] K. Piasta and R. Kotas, “Comparative analysis of natural language processing techniques in the classification of press articles,” *Applied Sciences*, vol. 15, no. 17, Art. no. 9559, Aug. 2025, doi: 10.3390/app15179559.

[18] H. Ahn *et al*., “Classification by ensembles from random partitions of high-dimensional data,” *Computational Statistics & Data Analysis*, vol. 51, no. 12, pp. 6166–6179, Aug. 2007, doi: 10.1016/j.csda.2006.12.043.

[19] Z. Dai and W. Huang, “Improving energy management practices through accurate building energy consumption prediction: Analyzing the performance of LightGBM, RF, and XGBoost models with advanced optimization strategies,” *Electrical Engineering*, vol. 107, no. 9, pp. 12583–12605, Jun. 2025, doi: 10.1007/s00202-025-03167-8.

[20] V. Buhrmester, D. Münch, and M. Arens, “Analysis of explainers of black box deep neural networks for computer vision: A survey,” *Machine Learning and Knowledge Extraction*, vol. 3, no. 4, pp. 966–989, Dec. 2021, doi: 10.3390/make3040048.

[21] V. Chamola, V. Hassija, A. R. Sulthana, D. Ghosh, D. Dhingra, and B. Sikdar, “A review of trustworthy and explainable artificial intelligence (XAI),” *IEEE Access*, vol. 11, no. July, pp. 78994–79015, 2023, doi: 10.1109/ACCESS.2023.3294569.

[22] L. Wu, “A review of the transition from Shapley values and SHAP values to RGE,” *Statistics*, vol. 59, no. 5, pp. 1161–1183, Apr. 2025, doi: 10.1080/02331888.2025.2487853.

[23] R. Rodríguez-Pérez and J. Bajorath, “Interpretation of machine learning models using Shapley values: Application to compound potency and multi-target activity predictions,” *Journal of Computer-Aided Molecular Design*, vol. 34, no. 10, pp. 1013–1026, 2020, doi: 10.1007/s10822-020-00314-0.

[24] X. Xie *et al*., “Block-gram: Mining knowledgeable features for efficiently smart contract vulnerability detection,” *Digital Communications and Networks*, vol. 11, no. 1, pp. 1–12, Feb. 2025, doi: 10.1016/j.dcan.2023.07.009.

[25] R. Kiani and V. S. Sheng, “Ethereum smart contract vulnerability detection and machine learning-driven solutions: A systematic literature review,” *Electronics*, vol. 13, no. 12, Art. no. 2295, Jun. 2024, doi: 10.3390/electronics13122295.