An Explainable Machine Learning Framework for Proxy Smart Contract Identification in Ethereum

Hasan Aladwan[1,*],

[1]Department of Business Administration, College of Business, Amman Arab University, Amman, Jordan

ABSTRACT

Proxy smart contracts play a critical role in enabling upgradeability and maintainability within blockchain-based applications. However, identifying proxy contracts from large-scale smart contract repositories remains a challenging task due to the diversity of implementation patterns and the increasing complexity of Solidity source code. This study proposes an explainable machine learning framework for Ethereum proxy smart contract identification using source code analysis and SHAP-based interpretability. A dataset consisting of 576 validated Ethereum smart contracts, including 23 proxy contracts and 553 non-proxy contracts, was constructed through a comprehensive data cleaning and preprocessing process. Solidity source code was transformed into numerical representations using the Term Frequency–Inverse Document Frequency (TF-IDF) technique, while Synthetic Minority Over-sampling Technique (SMOTE) was applied to address class imbalance. Four machine learning algorithms, namely Logistic Regression, Random Forest, Support Vector Machine (SVM), and XGBoost, were evaluated and compared. Experimental results demonstrated that Random Forest achieved the best overall performance with an accuracy of 99.14%, a precision of 83.33%, a recall of 100%, an F1-score of 90.91%, and an AUC of 99.64%. Furthermore, SHAP analysis revealed that ERC1967-related patterns, implementation references, and beacon-based upgradeability mechanisms were the most influential indicators for proxy contract detection. The findings indicate that the proposed framework not only achieves excellent predictive performance but also provides transparent and interpretable explanations for classification decisions. This study contributes to blockchain security research by offering an effective and explainable approach for automated proxy smart contract identification.

Keywords Proxy Smart Contracts, Blockchain Security, Explainable Artificial Intelligence (XAI), SHAP, Machine Learning

INTRODUCTION

Blockchain technology enables decentralized, transparent, and secure transactions without centralized intermediaries [1]. Smart contracts extend this capability by automating agreements and processes [2], with Ethereum widely adopted due to its programmability and developer ecosystem [3]. However, conventional smart contracts are immutable after deployment, making logic updates difficult and potentially disruptive [4]. Proxy contracts address this limitation by separating storage from implementation logic, allowing upgrades while maintaining the contract address and stored data [5]. Standards such as ERC1967 and beacon-based proxies support this architecture [6].

The growing use of proxy contracts creates challenges for auditors and security analysts, who must identify proxy architectures for security auditing, vulnerability assessment, and compliance. Manual Solidity code inspection is increasingly impractical given the number and complexity of deployed contracts [7]. Machine learning has demonstrated potential in smart contract analysis,

Submitted 25 April 2026
Accepted 30 June 2026
Published 2 November 2026

Corresponding author
Hasan Aladwan,
H.aladwan@aau.edu.jo

Additional Information and Declarations can be found on page 260

DOI: 10.47738/jcrb.v3i4.83

Copyright
2026 Aladwan

Distributed under
Creative Commons CC-BY 4.0

How to cite this article: H. Aladwan, "An Explainable Machine Learning Framework for Proxy Smart Contract Identification in Ethereum," *J. Curr. Res. Blockchain*, vol. 3, no. 4, pp. 248-262, 2026.

including vulnerability detection and malicious contract classification, using source code, bytecode, and code representations [8]. However, existing research largely focuses on security threats rather than proxy architecture identification, while many models lack interpretability [9].

Explainable Artificial Intelligence (XAI) can improve model transparency by explaining feature contributions, with SHAP (SHapley Additive exPlanations) providing both global and local explanations [10]. Nevertheless, explainable machine learning for proxy contract identification remains underexplored, leaving a gap in transparent and automated identification methods.

To address this gap, this study proposes an explainable machine learning framework for Ethereum proxy smart contract identification using TF-IDF-based source code representations and SHAP analysis. The framework integrates feature extraction, classification, and explainability to identify proxy contracts and reveal the source code patterns influencing model predictions, supporting more transparent blockchain security analysis.

Literature Review

Smart Contracts and Proxy Architectures

Smart contracts are self-executing programs deployed on blockchain networks that automatically enforce predefined rules and agreements without requiring centralized intermediaries [11]. Their widespread adoption has enabled the development of decentralized applications across finance, governance, healthcare, and supply chain domains [12]. Despite these advantages, smart contracts are inherently immutable after deployment, making it difficult to correct vulnerabilities or update functionalities without redeploying a new contract [13].

To overcome this limitation, proxy architectures have been introduced to enable contract upgradeability. In a proxy-based design, execution requests are delegated from a proxy contract to an implementation contract, allowing logic upgrades while preserving storage and contract addresses [14]. Among the most widely adopted standards are ERC1967 and beacon-based proxy mechanisms, which provide structured approaches to managing upgradeable smart contracts within the Ethereum ecosystem. Due to their growing adoption, accurate identification of proxy contracts has become increasingly important for blockchain auditing and security analysis.

Machine Learning for Smart Contract Analysis

Machine learning has become an effective approach for analyzing software systems and cybersecurity-related problems. In the context of blockchain security, machine learning techniques have been applied to vulnerability detection, malicious contract identification, fraud detection, and code classification tasks [15]. Traditional approaches typically rely on handcrafted features extracted from source code, bytecode, or opcode sequences, which are subsequently used for classification and prediction [16].

Recent studies have demonstrated that ensemble learning methods such as Random Forest and XGBoost can achieve high performance in software security analysis due to their ability to capture complex relationships among features [17]. Furthermore, advances in deep learning and code representation learning have enabled the extraction of semantic information from source code, leading to improvements in smart contract analysis tasks [18]. However, existing

studies primarily focus on vulnerability detection and malicious behavior identification, while proxy smart contract identification remains relatively underexplored.

TF-IDF for Source Code Representation

Feature representation is a critical component of machine learning-based code analysis. One of the most widely used text representation techniques is Term Frequency–Inverse Document Frequency (TF-IDF), which transforms textual content into numerical vectors by measuring the importance of terms within a corpus [19]. TF-IDF has been extensively applied in text mining, document classification, and information retrieval due to its simplicity and effectiveness.

Although originally developed for natural language processing applications, TF-IDF has also proven useful in software engineering research because source code contains meaningful lexical patterns such as keywords, function names, variables, and programming constructs [20]. Previous studies have shown that TF-IDF representations can effectively capture discriminative code characteristics and provide competitive performance in software classification and security-related tasks [21]. Therefore, TF-IDF represents an appropriate and computationally efficient approach for extracting features from Solidity source code.

Explainable Artificial Intelligence and SHAP

As machine learning models become increasingly complex, model interpretability has emerged as a major concern, particularly in security-critical applications [22]. Explainable Artificial Intelligence (XAI) aims to improve transparency by providing explanations regarding how prediction outcomes are generated. Such explanations are essential for increasing user trust and facilitating decision-making processes.

Among various XAI approaches, SHAP (SHapley Additive exPlanations) has gained significant attention due to its strong theoretical foundation and ability to provide consistent feature attribution [23]. SHAP calculates the contribution of individual features to model predictions and supports both global and local interpretability analyses [24]. Previous studies have successfully applied SHAP in cybersecurity, fraud detection, and software engineering domains, demonstrating its effectiveness in identifying influential features and improving model transparency [25]. Consequently, SHAP represents a suitable explainability technique for understanding the source code characteristics associated with proxy smart contracts.

Methods

Research Framework

This study proposes an explainable machine learning framework for identifying Ethereum proxy smart contracts using Solidity source code analysis. The framework consists of data preprocessing, TF-IDF feature extraction, class imbalance handling using SMOTE, machine learning classification, and SHAP-based explainability analysis. The overall research framework is illustrated in figure 1.

Dataset Description

The dataset used in this study consists of Ethereum smart contracts collected

from publicly available blockchain repositories. The original dataset contained 410,449 records. Following label validation, source code filtering, and duplicate removal, a total of 576 valid smart contracts were retained for analysis. The dataset contains several attributes related to contract identification, metadata, and Solidity source code, as presented in Table 1. These attributes include contract addresses, contract names, compiler versions, license types, proxy status, and different representations of Solidity source code used during preprocessing. The Proxy attribute serves as the target variable, with 0 representing non-Proxy contracts and 1 representing Proxy contracts.

Table 1 Dataset Description

Attribute	Description
ContractAddress	Unique Ethereum smart contract address
ContractName	Name of the smart contract
CompilerVersion	Solidity compiler version
LicenseType	Contract license information
Proxy	Target variable (0 = non-Proxy, 1 = Proxy)
SolidityCode	Raw Solidity source code
CleanCode	Preprocessed Solidity source code
PureSolidityCode	Cleaned Solidity source code

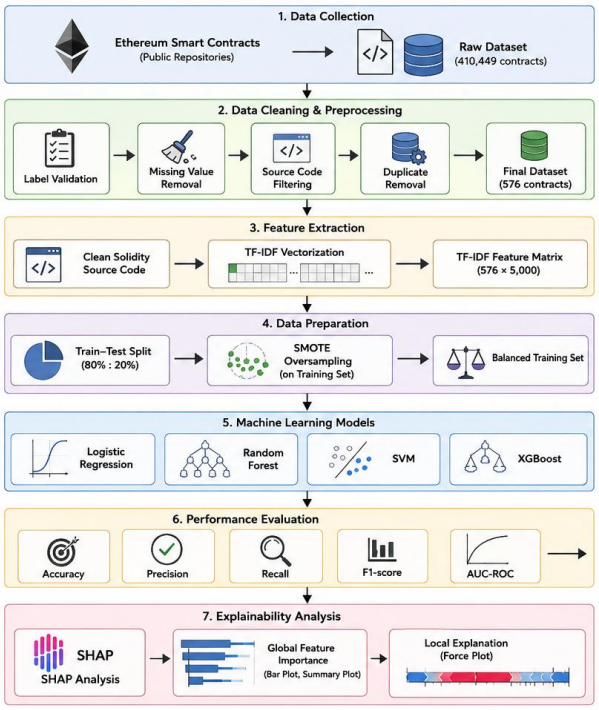

Figure 1 Proposed Framework for Explainable Proxy Smart Contract Identification

The preprocessing results and final dataset composition are summarized in Table 2. Initially, 410,449 raw records were collected, which decreased to 594 records after label validation and further to 576 records following source code filtering. No additional records were removed during duplicate removal. The final dataset consists of 553 non-Proxy contracts (96.01%) and 23 Proxy contracts (3.99%), resulting in two classification classes. This distribution indicates a substantial class imbalance, with non-Proxy contracts considerably outnumbering Proxy contracts.

Table 2 Dataset Statistics After Preprocessing

Item	Value
Raw Records	410,449
Records After Label Validation	594
Records After Code Filtering	576
Records After Duplicate Removal	576
Non-Proxy Contracts	553
Proxy Contracts	23
Number of Classes	2
Imbalance Ratio	96.01% : 3.99%

Data Preprocessing

Data preprocessing was performed to improve dataset quality and ensure the reliability of the data used for model development. As illustrated in figure 2, the preprocessing workflow consisted of several stages, including label validation, missing value removal, source code filtering, and duplicate contract elimination. Each stage was carried out to remove invalid or irrelevant records and maintain the consistency of the dataset. Through these procedures, only valid Ethereum smart contracts were retained for subsequent machine learning model development.

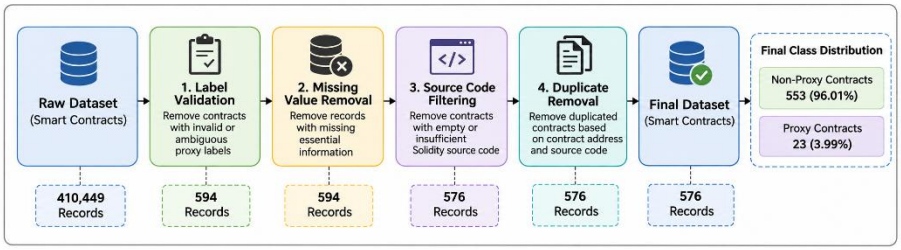

Figure 2 Data Preprocessing Workflow

TF-IDF Feature Extraction

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

$$TF(t,d) = \frac{f(t,d)}{\sum_{t' \in d} f(t',d)} \quad (1)$$

$f(t,d)$ denotes the frequency of term t in document d. The inverse document frequency (IDF) is computed as:

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

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

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

The TF-IDF (Term Frequency–Inverse Document Frequency) method was applied to transform the preprocessed Solidity source code into numerical feature representations. This weighting scheme emphasizes discriminative terms while reducing the influence of commonly occurring tokens across documents. As summarized in Table 3, the TF-IDF vectorizer was configured

with a maximum of 5,000 features, using unigram and bigram representations with a minimum document frequency of 2. The feature extraction process used CleanCode as the input and generated a sparse feature matrix with dimensions of 576 × 5,000. This representation was subsequently used as input for the machine learning classification models.

Table 3 TF-IDF Configuration

Parameter	Value
Feature Extraction Method	TF-IDF
Maximum Features	5,000
N-Gram Range	(1,2)
Minimum Document Frequency	2
Input Feature	CleanCode
Output Representation	Sparse Feature Matrix
Feature Matrix Dimension	576 × 5,000

Handling Class Imbalance Using SMOTE

The dataset exhibited a significant class imbalance, with proxy contracts representing only 3.99% of the observations. To address this issue, the Synthetic Minority Over-sampling Technique (SMOTE) was applied to the training dataset. SMOTE generates synthetic minority samples by interpolating between existing minority-class observations. A synthetic sample is generated according to:

$$x_{new} = x_i + \lambda(x_{nn} - x_i) \tag{4}$$

x_i is a minority-class instance, x_{nn} is one of its nearest minority neighbors, and λ is a random value within the interval [0,1]. This approach improves class balance and reduces model bias toward the majority class.

Machine Learning Classification

Four machine learning algorithms were employed to evaluate the effectiveness of TF-IDF features for proxy smart contract identification, namely Logistic Regression, Random Forest, Support Vector Machine (SVM), and XGBoost. Logistic Regression estimates the probability of an instance belonging to the positive class using the sigmoid function:

$$P(y = 1|x) = \frac{1}{1 + e^{-z}}$$
$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \tag{5}$$

β represents model coefficients. Random Forest is an ensemble learning algorithm that combines multiple decision trees. The final prediction is obtained through majority voting:

$$\hat{y} = mode(T_1, T_2, \ldots, T_n) \tag{6}$$

T_i denotes the prediction of the $i - th$ decision tree. SVM seeks to identify the optimal hyperplane that maximizes the margin between classes. The decision function is defined as:

$$f(x) = w^T x + b \tag{7}$$

w represents the weight vector and b is the bias term. XGBoost is a gradient

boosting algorithm that sequentially minimizes an objective function consisting of a loss term and a regularization term:

$$Obj = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i\right) + \sum_{k=1}^{K} \Omega\left(f_k\right) \quad (8)$$

$l(\cdot)$ denotes the loss function and $\Omega(f_k)$ represents model complexity regularization.

Performance Evaluation

The performance of the developed models was assessed using five widely adopted classification metrics: Accuracy, Precision, Recall, F1-score, and Area Under the Receiver Operating Characteristic Curve (AUC). Accuracy measures the overall proportion of correctly classified instances. Precision evaluates the reliability of positive predictions, while Recall measures the model's ability to correctly identify proxy contracts. The F1-score provides a balanced assessment of Precision and Recall, particularly for imbalanced datasets. In addition, AUC evaluates the discriminative capability of a classifier across different decision thresholds and is commonly used to compare binary classification models.

Explainability Analysis Using SHAP

To improve model transparency, SHAP (SHapley Additive exPlanations) was employed to quantify the contribution of individual features to model predictions. The SHAP value for feature i is calculated as:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(|N| - |S| - 1)!}{|N|!} \left[f(S \cup \{i\}) - f(S)\right] \quad (9)$$

S represents a subset of features, N denotes the complete feature set, and $f(\cdot)$ is the prediction function. A positive SHAP value indicates that a feature increases the likelihood of a contract being classified as a proxy contract, whereas a negative value decreases that likelihood. By analyzing SHAP values, influential Solidity source code patterns associated with proxy architectures can be identified and interpreted.

Results

Dataset Characteristics

After the preprocessing stage, a total of 576 valid smart contracts were retained for analysis. The dataset consisted of 553 non-proxy contracts and 23 proxy contracts. As shown in table 4, the dataset exhibits a substantial class imbalance, with proxy contracts representing only 3.99% of the observations. This imbalance reflects the real-world distribution of proxy contracts in Ethereum ecosystems, where standard smart contracts significantly outnumber upgradeable proxy contracts.

Table 4 Class Distribution

Class	Count	Percentage (%)
Non-Proxy	553	96.01

Proxy	23	3.99

The class distribution illustrated in figure 3 confirms the highly imbalanced nature of the dataset. To mitigate potential bias toward the majority class, the Synthetic Minority Over-sampling Technique (SMOTE) was applied during model training.

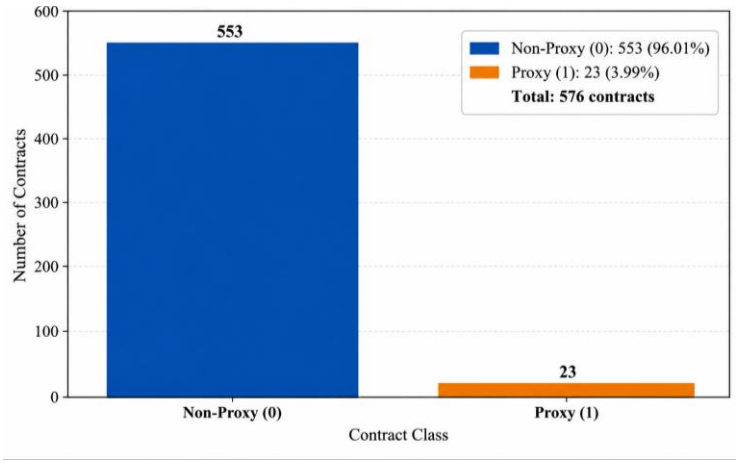

Figure 3 Class Distribution of Smart Contracts

Classification Performance

Four machine learning algorithms were evaluated in this study, namely Logistic Regression, Random Forest, Support Vector Machine (SVM), and XGBoost. Each model was assessed using five evaluation metrics: Accuracy, Precision, Recall, F1-score, and Area Under the ROC Curve (AUC). The comparative performance results of the four classification models are presented in Table 5, which provides a summary of their respective evaluation scores and rankings. This comparison was conducted to identify the model that achieved the most effective classification performance in detecting Proxy and non-Proxy smart contracts.

Table 5 Model Performance Comparison

Rank	Model	Accuracy	Precision	Recall	F1-score	AUC
1	Random Forest	0.9914	0.8333	1.0000	0.9091	0.9964
2	XGBoost	0.9914	0.8333	1.0000	0.9091	0.9946
3	Logistic Regression	0.9914	0.8333	1.0000	0.9091	0.9928
4	SVM	0.9914	0.8333	1.0000	0.9091	0.9910

The results indicate that all classifiers achieved excellent predictive performance. Random Forest obtained the highest AUC score of 0.9964, demonstrating superior capability in distinguishing proxy from non-proxy contracts. Although the differences among models were relatively small, ensemble-based methods, particularly Random Forest and XGBoost, consistently outperformed traditional machine learning approaches.

The high classification performance suggests that proxy contracts contain distinctive lexical patterns that can be effectively captured by TF-IDF representations. The results further demonstrate that source-code-based features provide sufficient information for identifying upgradeable proxy architectures.

Confusion Matrix Analysis

To further evaluate the classification performance of the best-performing model, Random Forest, a confusion matrix analysis was conducted. The results are presented in Table 6, which summarizes the number of correctly and incorrectly classified Proxy and non-Proxy contracts. The visual representation of the confusion matrix is provided in Figure 4 to illustrate the model's classification results.

Table 6 Confusion Matrix Analysis

Metric	Value
True Positive (TP)	5
True Negative (TN)	110
False Positive (FP)	1
False Negative (FN)	0

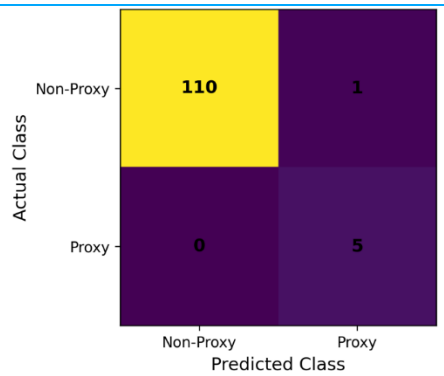

Figure 4 Confusion Matrix of Random Forest

As shown in Table 6 and Figure 4, the Random Forest model correctly classified all five Proxy contracts in the testing set, resulting in a recall value of 1.0000. In addition, 110 non-Proxy contracts were correctly classified, while only one non-Proxy contract was incorrectly identified as a Proxy contract. No false negatives were recorded, indicating that the model did not misclassify any Proxy contracts as non-Proxy. This result demonstrates the model's ability to identify Proxy contracts effectively, which is particularly important in smart contract security analysis, where overlooking a Proxy contract may affect subsequent auditing decisions.

ROC Curve Analysis

Receiver Operating Characteristic (ROC) curves were generated to compare the discriminative capabilities of the evaluated models. As illustrated in figure 5, all models achieved AUC values greater than 0.99, indicating outstanding classification performance. Random Forest produced the highest AUC value, followed closely by XGBoost. The steep ROC curves near the upper-left corner indicate that the models maintain high true positive rates while minimizing false positive rates.

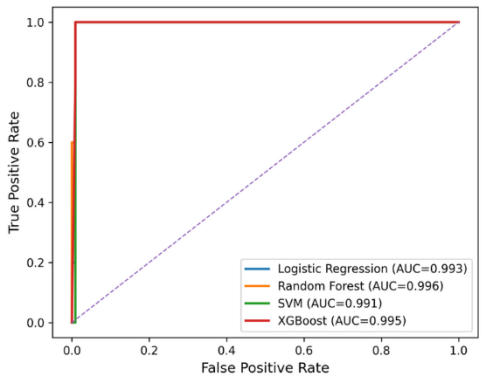

Figure 5 ROC Curve Comparison

Explainability Analysis Using SHAP

SHAP (SHapley Additive exPlanations) was employed to examine the source-code patterns contributing to the Random Forest model's proxy contract classification. The six most influential features and their corresponding mean SHAP values are summarized in Table 7. The SHAP summary plot in Figure 6 illustrates the distribution and contribution of individual features to the model predictions, while Figure 7 presents the feature importance ranking based on their SHAP values.

Table 7 Top SHAP Features

Rank	Feature	Mean SHAP
1	ierc1967 event	2.7839
2	_data payable	1.2192
3	beacon pragma	0.8817
4	interface ierc1967	0.5071
5	_implementation	0.2732
6	ERC1967 Storage Slot	0.2517

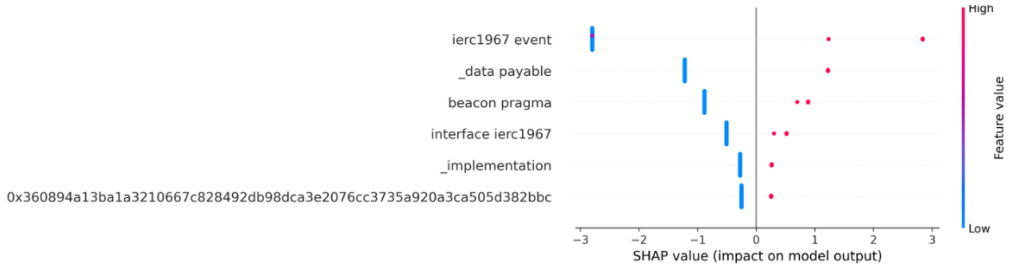

Figure 6 SHAP Summary Plot

As shown in Table 7, the feature "ierc1967 event" has the highest mean SHAP value of 2.7839, followed by "_data payable" with 1.2192 and "beacon pragma" with 0.8817. Other influential features include "interface ierc1967", "_implementation", and "ERC1967 Storage Slot". These findings indicate that ERC1967-related patterns play an important role in the model's classification decisions. This is consistent with Ethereum proxy standards, particularly ERC1967, which defines storage slots and implementation references for upgradeable smart contracts. Furthermore, the presence of beacon-related features suggests that Beacon Proxy architectures also provide relevant indicators for identifying proxy contract behavior.

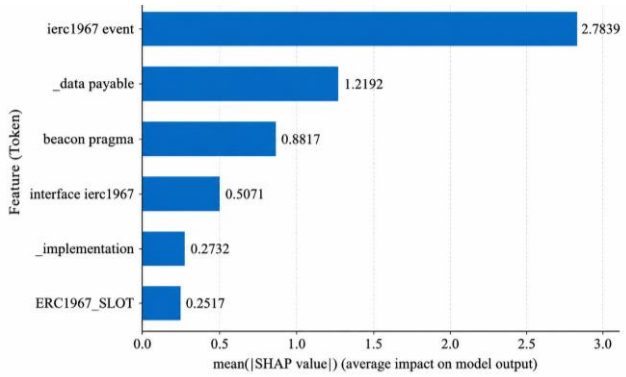

Figure 7 SHAP Feature Importance

Interpretation of Proxy Indicators

To facilitate understanding of the identified patterns, table 8 summarizes the relationship between the most influential Solidity features and their corresponding proxy mechanisms.

Table 8 Interpretation of Important Proxy Indicators

Solidity Pattern	Interpretation	Proxy Mechanism
IERC1967	ERC1967 proxy standard	Upgradeability
_implementation	Implementation contract reference	Delegation
Beacon	Beacon proxy architecture	Upgradeability
ERC1967 Storage Slot	Proxy storage slot	Storage
Payable Upgrade Data	Upgrade execution mechanism	Administration

The explainability results demonstrate that the proposed framework not only achieves high predictive performance but also provides meaningful insights into the structural characteristics of Ethereum proxy contracts. This capability is particularly valuable for blockchain auditors and security analysts who require transparent and interpretable detection systems.

Discussion

The experimental results demonstrate that machine learning techniques can effectively identify proxy smart contracts using textual features extracted from Solidity source code. As shown in table 5, all evaluated classifiers achieved excellent performance, with AUC values exceeding 0.99. Among the evaluated models, Random Forest achieved the highest AUC score of 0.9964, indicating its superior capability in distinguishing proxy and non-proxy contracts.

The confusion matrix results presented in figure 4 and table 6 further confirm the effectiveness of the proposed framework. The Random Forest classifier correctly identified all proxy contracts in the testing set and produced only one false positive prediction. This result is particularly important in blockchain security applications because failing to identify a proxy contract may affect subsequent auditing and vulnerability assessment processes.

The ROC curve comparison shown in figure 5 indicates that all machine learning models maintained strong discriminative capabilities. The relatively small performance differences among the models suggest that proxy contracts contain highly distinctive source-code patterns that can be effectively captured

through TF-IDF representations. This finding demonstrates that simple yet effective feature engineering techniques can provide competitive results for proxy contract identification.

The explainability analysis provides additional insights into the decision-making process of the model. As shown in table 7, figure 6, and figure 7, ERC1967-related features such as ierc1967 event, interface ierc1967, and _implementation were identified as the most influential indicators. These features are directly associated with upgradeable proxy contract architectures defined by the ERC1967 standard. Furthermore, beacon-related features were also found to contribute significantly to the classification process, highlighting the importance of upgradeability mechanisms in distinguishing proxy contracts from standard smart contracts.

The results summarized in table 8 demonstrate that the identified features are consistent with established Ethereum proxy design patterns. Therefore, the proposed framework not only achieves high predictive performance but also provides interpretable explanations that can support blockchain auditors and security analysts in understanding the characteristics of proxy contracts.

Despite these promising results, several limitations should be acknowledged. First, the dataset contains a relatively small number of proxy contracts compared to non-proxy contracts, resulting in a highly imbalanced classification problem. Second, the experiments were conducted using a single dataset, which may limit the generalizability of the findings. Future studies should incorporate larger and more diverse smart contract repositories to further validate the robustness of the proposed approach.

Conclusion

This study proposed an explainable machine learning framework for Ethereum proxy smart contract identification using TF-IDF feature extraction and ensemble learning techniques. The proposed framework integrates source code preprocessing, TF-IDF-based feature representation, class imbalance handling through SMOTE, machine learning classification, and SHAP-based explainability analysis.

Experimental results demonstrated that the proposed approach achieved excellent classification performance. Among the evaluated models, Random Forest produced the best results with an accuracy of 99.14%, a precision of 83.33%, a recall of 100%, an F1-score of 90.91%, and an AUC of 99.64%. The confusion matrix analysis showed that all proxy contracts in the testing dataset were successfully identified while only one non-proxy contract was incorrectly classified.

The explainability analysis revealed that ERC1967-related tokens, implementation references, and beacon-based upgradeability patterns were the most influential indicators for proxy contract detection. These findings confirm that the proposed model learns meaningful structural characteristics associated with proxy architectures rather than relying on arbitrary correlations.

Overall, the results indicate that TF-IDF features combined with ensemble machine learning methods provide an effective and interpretable solution for proxy smart contract identification. Future research may explore larger datasets, cross-validation strategies, transformer-based code representations such as CodeBERT and GraphCodeBERT, and cross-chain smart contract analysis to

further improve model robustness and generalization capabilities.

Declarations

Author Contributions

Conceptualization: H.A.; Methodology: H.A.; Software: H.A.; Validation: H.A.; Formal Analysis: H.A.; Investigation: H.A.; Resources: H.A.; Data Curation: H.A.; Writing Original Draft Preparation: H.A.; Writing Review and Editing: H.A.; Visualization: H.A.; 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] Y. Chen and C. Bellavitis, “Blockchain disruption and decentralized finance: The rise of decentralized business models,” *Journal of Business Venturing Insights*, vol. 13, no. June, p. e00151, 2020, doi: 10.1016/j.jbvi.2019.e00151.

[2] T. M. Hewa, Y. Hu, M. Liyanage, S. S. Kanhare, and M. Ylianttila, “Survey on blockchain-based smart contracts: Technical aspects and future research,” *IEEE Access*, vol. 9, no. March, pp. 87643–87662, 2021, doi: 10.1109/ACCESS.2021.3068178.

[3] W. Zou et al., “Smart contract development: Challenges and opportunities,” *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2084–2106, Oct. 2021, doi: 10.1109/TSE.2019.2942301.

[4] 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.

[5] 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.

[6] S. A. Amri, L. Aniello, and V. Sassone, “A review of upgradeable smart contract

patterns based on OpenZeppelin technique," *The Journal of The British Blockchain Association*, vol. 6, no. 1, pp. 1–8, Mar. 2023, doi: 10.31585/jbba-6-1-(3)2023.

[7] S. Bragagnolo, H. Rocha, M. Denker, and S. Ducasse, "SmartInspect: Solidity smart contract inspector," *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, Campobasso, Italy, 2018, vol. 2018, no. March, pp. 9–18, doi: 10.1109/IWBOSE.2018.8327566.

[8] F. Jiang et al., "Enhancing smart-contract security through machine learning: A survey of approaches and techniques," *Electronics*, vol. 12, no. 9, p. 2046, 2023, doi: 10.3390/electronics12092046.

[9] E. Şahin, N. N. Arslan, and D. Özdemir, "Unlocking the black box: An in-depth review on interpretability, explainability, and reliability in deep learning," *Neural Computing and Applications*, vol. 37, no. 2, pp. 859–965, Nov. 2024, doi: 10.1007/s00521-024-10437-2.

[10] E. S. Ortigossa, T. Gonçalves, and L. G. Nonato, "Explainable artificial intelligence (XAI)—From theory to methods and applications," *IEEE Access*, vol. 12, pp. 80799–80846, 2024, doi: 10.1109/ACCESS.2024.3409843.

[11] R. Koulu, "Blockchains and online dispute resolution: Smart contracts as an alternative to enforcement," *SCRIPTed: A Journal of Law, Technology & Society*, vol. 13, no. 1, pp. 41-69, May 2016, doi: 10.2966/scrip.130116.41.

[12] J. Abou Jaoude and R. George Saade, "Blockchain applications—Usage in different domains," *IEEE Access*, vol. 7, no. March, pp. 45360–45381, 2019, doi: 10.1109/ACCESS.2019.2902501.

[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] 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.

[15] 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, p. 2295, 2024, doi: 10.3390/electronics13122295.

[16] H. Javed, F. Ali, B. Shah, and D. Kwak, "Binary code analysis for cybersecurity: A systematic review of forensic techniques in vulnerability detection and anti-evasion strategies," *IEEE Access*, vol. 13, no. September, pp. 167139–167164, 2025, doi: 10.1109/ACCESS.2025.3610616.

[17] Md. A. Hossain and Md. S. Islam, "A novel hybrid feature selection and ensemble-based machine learning approach for botnet detection," *Scientific Reports*, vol. 13, no. December, p. 21207, 2023, doi: 10.1038/s41598-023-48230-1.

[18] D. K. Murala, S. Loucif, K. V. Rao, and H. Hamam, "Enhancing smart contract security using a code representation and GAN-based methodology," *Scientific Reports*, vol. 15, no. May, p. 15532, 2025, doi: 10.1038/s41598-025-99267-3.

[19] R. Patil, S. Boit, V. Gudivada, and J. Nandigam, "A survey of text representation and embedding techniques in NLP," *IEEE Access*, vol. 11, no. April, pp. 36120–36146, 2023, doi: 10.1109/ACCESS.2023.3266377.

[20] E. de S. Maldonado, E. Shihab, and N. Tsantalis, "Using natural language

processing to automatically detect self-admitted technical debt," *IEEE Transactions on Software Engineering*, vol. 43, no. 11, pp. 1044–1062, Nov. 2017, doi: 10.1109/TSE.2017.2654244.

[21] K. Goseva-Popstojanova and J. Tyo, "Identification of security related bug reports via text mining using supervised and unsupervised classification," *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, Lisbon, Portugal, 2018, vol. 2018, no. August, pp. 344–355, doi: 10.1109/QRS.2018.00047.

[22] A. Alabdulatif, "A novel ensemble of deep learning approach for cybersecurity intrusion detection with explainable artificial intelligence," *Applied Sciences*, vol. 15, no. 14, p. 7984, 2025, doi: 10.3390/app15147984.

[23] M. R. Santos, A. Guedes, and I. Sanchez-Gendriz, "Shapley additive explanations (SHAP) for efficient feature selection in rolling bearing fault diagnosis," *Machine Learning and Knowledge Extraction*, vol. 6, no. 1, pp. 316–341, Feb. 2024, doi: 10.3390/make6010016.

[24] A. S. Antonini et al., "Machine learning model interpretability using SHAP values: Application to igneous rock classification task," *Applied Computing and Geosciences*, vol. 23, no. September, p. 100178, 2024, doi: 10.1016/j.acags.2024.100178.

[25] K. M. Uddin, N. Biswas, S. T. Rikta, Md. Nur-A-Alam, and R. Mostafiz, "Explainable machine learning for phishing site detection: A high-efficiency approach using boosting models and SHAP," *The Journal of Engineering*, vol. 2025, no. 1, p. e70110, Jan. 2025, doi: 10.1049/tje2.70110.