A Major Project Report

On

SECURE AND SCALABLE BLOCKCHAIN BASED FEDERATED LEARNING FOR CRYPTOCURRENCY FRAUD DETECTION

Submitted to CMREC (UGC Autonomous), Affiliated to JNTUH

In Partial Fulfilment of the requirements for the Award of Degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (AI & ML)

Submitted

By

M. SAI ABHIRAM	**(228R1A66A5)**
D.ABHINAY KUMAR	**(228R1A6678)**
P.GOUTHAM REDDY	**(228R1A66B3)**
TRIBIKRAM P SAHOO	**(228R1A66C5)**

Under the Esteemed guidance of

Mr. N.VENKATESH

Assistant Professor, Department of CSE(AI&ML)

Department of Computer Science and Engineering(AI&ML)

CMR ENGINEERING COLLEGE

(UGC AUTONOMOUS)

(Accredited by NAAC & NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad)
(Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401)

(2025-2026)

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Accredited by NAAC&NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad, Kandlakoya, Medchal Road, Hyderabad-501 401)

Department of Computer Science & Engineering (AI & ML)

CERTIFICATE

This is to certify that the Major project entitled **"SECURE AND SCALABLE BLOCKCHAIN BASED FEDERATED LEARNING FOR CRYPTOCURRENCY FRAUD DETECTION"** is a bonafide work carried out by

M.SAI ABHIRAM	**(228R1A66A5)**
D.ABHINAY KUMAR	**(228R1A6678)**
P.GOUTHAM REDDY	**(228R1A66B3)**
TRIBIKRAM P SAHOO	**(228R1A66C5)**

in partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING (AI&ML)** from CMR Engineering College, under our guidance and supervision.

The results presented in this Major project have been verified and are found to be satisfactory. The results embodied in this Major project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide	**Major Project Coordinator**	**Head of the Department**
Mr. N. Venkatesh	**Mr. G. Venkateswarlu**	**Dr. Madhavi Pingili**
Assistant Professor	Assistant Professor	Professor & HOD
Department of CSE (AI&ML)	Department of CSE (AI&ML)	Department of CSE (AI&ML)

External Examiner: _____

DECLARATION

This is to certify that the work reported in the present Major project entitled **"SECURE AND SCALABLE BLOCKCHAIN BASED FEDERATED LEARNING FOR CRYPTO CURRENCY FRAUD DETECTION"** is a record of bonafide work done by us in the Department of Computer Science and Engineering (AI & ML), CMR Engineering College. The reports are based on the Major project work done entirely by us and not copied from any other source. We submit our Major project for further development by any interested students who share similar interests to improve the Major project in the future.

The results embodied in this Major project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

M. SAI ABHIRAM	**(228R1A66A5)**
D.ABHINAY KUMAR	**(228R1A6678)**
P.GOUTHAM REDDY	**(228R1A66B3)**
TRIBIKRAM P SAHOO	**(228R1A66C5)**

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. Madhavi Pingili**, Professor & HOD, Department of CSE(AI&ML), CMR Engineering College for their constant support.

We are extremely thankful to **Mr. N. VENKATESH,** Assistant Professor, Internal Guide, Department of CSE(AI&ML), for his constant guidance, encouragement and moral support throughout the Major project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Major Project.

We thank **Mr. G. Venkateswarlu,** Major Project Coordinator for his constant support in carrying out the project activities and reviews.

We express our thanks to all staff members and friends for all the help and co-ordination extended in bringing out this Major project successfully in time.

Finally, We are very much thankful to our parents who guided us for every step.

M. SAI ABHIRAM (228R1A66A5)
D.ABHINAY KUMAR (228R1A6678)
P.GOUTHAM REDDY (228R1A66B3)
TRIBIKRAM P SAHOO (228R1A66C5)

CONTENTS

ABSTRACT

With the wide adoption of cryptocurrency, blockchain technologies have become the foundation of such digital currencies. However, this adoption has been accompanied by a surge in cryptocurrency fraud, causing significant losses to financial organizations and individuals. One way to mitigate these losses is to use Federated Learning (FL) techniques to detect fraudulent cryptocurrency transactions. This paper provides an overview of secure, privacy-preserving, and scalable Blockchain-based Federated Learning (BCFL) as a promising solution for slowing the exponential growth of cryptocurrency fraud. BCFL enables multiple entities to collaboratively train machine learning models for detecting fraudulent cryptocurrency transactions without sharing their private data, thus preserving privacy. However, Integrating differential privacy and Secure Multi-party computation (SMPC) models in BCFL presents an additional scalability challenge. This study provides an overview of BCFL, evaluating existing research on its security, privacy, and scalability challenges in detecting cryptocurrency fraud. The review explores existing research and various methodologies, highlighting advancements and challenges in creating effective, privacy-conscious fraud detection solutions for cryptocurrency transactions. We first discuss the current state of BCFL in fraud detection, along with its potential advantages and limitations, and then discuss the existing research gaps. In particular, this paper examines various BCFL frameworks, consensus algorithms, and block architectures, emphasizing their strengths and limitations in the context of cryptocurrency fraud detection to develop scalable and privacy-preserving solutions. We compare various solutions that address scalability and privacy challenges in BCFL, including adopting a geographically distributed cloud computing model that utilizes SMPC and lightweight consensus algorithms and protocols to manage computational overheads. The overall structure is developed with the intention of supporting future integration into content monitoring systems, automated moderation pipelines, and cyber-safety tools. This design-driven approach ensures clarity in workflow, robustness in planning, and strong alignment with contemporary trends in natural language processing and machine-learning-based text classification.

Keywords

Blockchain, cryptocurrency, fraud, federated learning, scalability, security, privacy, literature review.

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Introduction

The rapid proliferation of cryptocurrencies and decentralized financial ecosystems has introduced unprecedented opportunities for digital transactions, but it has also significantly increased the prevalence of cryptocurrency-related fraud. Recent reports indicate a substantial rise in fraudulent activities such as phishing attacks, Ponzi schemes, ransomware payments, and illicit money laundering, leading to billions of dollars in financial losses annually [1], [2]. The pseudonymous nature of blockchain transactions, coupled with the global and decentralized structure of cryptocurrency networks, makes fraud detection a challenging task for traditional security systems. Conventional machine learning (ML) approaches for fraud detection typically rely on centralized data collection and processing frameworks, where large volumes of transaction data are aggregated in a single repository for model training [3]. While effective in certain scenarios, such centralized architectures raise significant concerns related to data privacy, regulatory compliance, and single points of failure [4], [5]. Moreover, financial institutions and cryptocurrency platforms are often reluctant to share sensitive transactional data due to legal and competitive constraints, thereby limiting collaborative intelligence for fraud detection [6].

Federated Learning (FL) has emerged as a promising paradigm to address these challenges by enabling multiple participants to collaboratively train a global model without sharing raw data [7]. In FL, each participant performs local model training on its private dataset and shares only model updates, thereby preserving data privacy and ownership [8]. However, despite its advantages, traditional FL architectures still rely on a centralized aggregation server, making them vulnerable to issues such as single-point-of-failure (SPoF), model poisoning attacks, and lack of transparency in the aggregation process [9], [10].To overcome these limitations, the integration of blockchain technology with federated learning— commonly referred to as Blockchain-based Federated Learning (BCFL)—has gained significant attention [11]. Blockchain provides a decentralized, immutable, and transparent ledger that can securely record model updates, enforce trust through consensus mechanisms, and eliminate reliance on centralized coordinators [12]. By leveraging smart contracts and distributed consensus, BCFL enhances the robustness, traceability, and security of the federated learning process, making it particularly suitable for sensitive applications such as cryptocurrency fraud detection.

1.2 Project Objectives

By the end of this project you will understand

1. To design a structured and scalable ensemble-learning framework capable of supporting future detection of cyberbullying and hate speech across diverse textual data.

2. To plan and define a unified text preprocessing and feature engineering pipeline that ensures consistent and meaningful representation of user-generated content.

3. To develop a high-level system architecture that integrates data handling, preprocessing, feature extraction, and decision-making components for eventual deployment in automated content moderation systems.

4. To incorporate security techniques such as Differential Privacy (DP) and Secure Multi-Party Computation (SMPC) to prevent data leakage and inference attacks.

5. To optimize communication and computational efficiency for real-time fraud detection.

1.3 Purpose of the Project

The purpose of this project is to design an intelligent and structured framework capable of identifying cyberbullying and hate speech in online text. It aims to establish a clear preprocessing and analysis pipeline, enabling accurate categorization of harmful content and supporting future automated moderation systems that enhance user safety across digital platforms. This project aims to enable multiple distributed entities—such as cryptocurrency exchanges, wallet providers, and financial institutions—to collaboratively train machine learning models without sharing sensitive transaction data. By leveraging Federated Learning (FL), the system ensures that raw data remains local while still contributing to a globally optimized fraud detection model.

Another key objective of this project is to incorporate advanced security mechanisms**,** including differential privacy, secure aggregation, and robust consensus algorithms, to protect against data leakage, model inversion, and poisoning attacks. At the same time, the framework focuses on maintaining scalability and efficiency**,** ensuring that the system can handle large-scale, real-world cryptocurrency networks with minimal computational and communication overhead.

1.4 Problem Statement

The rapid growth of cryptocurrency adoption has led to a significant increase in fraudulent activities such as phishing, money laundering, and unauthorized transactions. Detecting such fraud is challenging due to the decentralized, pseudonymous, and highly dynamic nature of blockchain-based financial systems. Traditional fraud detection approaches rely on centralized machine learning models that require aggregation of large volumes of sensitive transaction data, raising serious concerns regarding data privacy, security, and regulatory compliance social media platforms.

Although Federated Learning (FL) offers a privacy-preserving alternative by enabling collaborative model training without sharing raw data, existing FL systems still depend on a central aggregation server, making them vulnerable to single points of failure, lack of transparency, and potential adversarial attacks such as model poisoning and inference attacks

Additionally, integrating blockchain with FL introduces challenges related to scalability, communication overhead, and computational complexity, especially when dealing with large-scale cryptocurrency networks and real-time fraud detection requirements. Therefore, there is a need for a secure, scalable, and decentralized framework that enables collaborative fraud detection while ensuring data privacy, system robustness, and efficient performance. This project addresses these challenges by proposing a Blockchain-based Federated Learning (BCFL) approach for effective cryptocurrency fraud detection.

1.5 Existing System

The existing systems for cryptocurrency fraud detection primarily rely on traditional rule-based methods and centralized machine learning approaches**.** Early fraud detection systems were based on predefined rules and expert knowledge, which were simple to implement but lacked the ability to detect complex and evolving fraud patterns. Once attackers understood these rules, they could easily bypass the system, making such approaches ineffective for modern cryptocurrency environments [1].With advancements in technology, machine learning (ML) techniques have been widely adopted for fraud detection. These systems utilize supervised and unsupervised learning algorithms to identify anomalies and suspicious transaction patterns from historical data. However, most ML-based systems operate in a centralized architecture, where large volumes of transaction data are collected and processed in a single location [2]. This centralized approach raises serious concerns regarding data privacy, security, and regulatory compliance

Another limitation of existing systems is the lack of collaboration among different entities, such as cryptocurrency exchanges and financial institutions. Due to privacy constraints and competitive concerns, organizations are unwilling to share their data, leading to data silos and reduced detection accuracy. As a result, fraud patterns that span multiple platforms often go undetected [3]. Although Federated Learning (FL) has been introduced to address data privacy issues by enabling collaborative model training without sharing raw data, current FL systems still depend on a central aggregation server**,** which introduces vulnerabilities such as single points of failure and susceptibility to attacks like model poisoning [2]. Moreover, these systems often lack transparency and trust among participating entities.

Disadvantages

- Limited ability to detect sarcasm, implicit abuse, and context-dependent expressions.
- Keyword- and rule-based systems fail when users disguise offensive words or use creative spellings.
- Single-model classifiers often produce inconsistent predictions on noisy and imbalanced datasets.
- Traditional ML approaches rely heavily on manual feature engineering, restricting adaptability.
- Deep learning models require large, high-quality annotated datasets, which are often difficult to obtain.
- Most existing methods lack robustness when handling multi-label or overlapping categories of abusive content.

1.6 Proposed System

The proposed system introduces a secure and scalable Blockchain-Based Federated Learning (BCFL) framework designed to enhance the detection of fraudulent activities in cryptocurrency transactions across distributed environments. The system integrates federated learning with blockchain technology to enable collaborative model training among multiple entities—such as cryptocurrency exchanges, wallet providers, and financial institutions—without sharing sensitive transaction data. A unified preprocessing pipeline is designed to handle data cleaning, normalization, feature extraction, and transaction graph construction, enabling meaningful representation of complex blockchain transaction patterns.

To improve detection performance, the framework supports hybrid machine learning models, including graph-based models, sequential learning methods, and ensemble techniques, ensuring accurate and context-aware fraud classification. The proposed system incorporates privacy-preserving mechanisms such as Differential Privacy (DP), Secure Multi-Party Computation (SMPC), and encrypted model updates to protect against data leakage and inference attacks. Additionally, blockchain is utilized to provide decentralized coordination, tamper-proof storage, and transparent validation of model updates through smart contracts and consensus algorithms.

To address challenges such as data heterogeneity and imbalance in fraud datasets, the system design includes adaptive learning strategies and balanced data handling techniques, improving robustness and fairness in detection. The overall architecture is designed to be scalable, modular, and resilient, supporting real-time fraud detection while ensuring security, trust, and collaboration across geographically distributed participants.

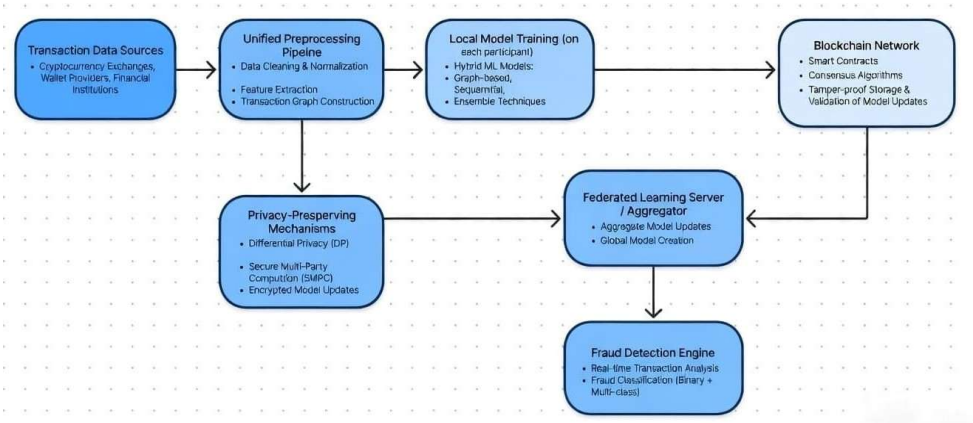


Figure 1.5.1: Block diagram of proposed system.

Advantages

- Enhanced reliability through ensemble-based decision-making rather than relying on a single model.
- Improved handling of noisy, informal, and diverse linguistic patterns through a unified preprocessing design.
- Better support for imbalanced datasets due to integrated class-balancing strategies.
- More stable and consistent predictions because of variance reduction achieved by combining multiple learners.
- Scalable architecture that can be extended to multi-class and multi-label abuse detection tasks.
- Modular design enabling easy integration into future content moderation and cyber safety applications.

1.7 Input and Output Design

1.7.1 Input Design

Input design focuses on defining the structure, format, and flow of the textual data that the proposed system will process during cyberbullying and hate speech detection. Since the system operates on user- generated text collected from online platforms, the input layer is designed to efficiently handle diverse linguistic styles, informal expressions, and noisy content.

The planned input structure accepts raw text sentences or posts as primary data, which may include symbols, emojis, URLs, hashtags, abbreviations, and variations of abusive or aggressive language. To ensure consistency and readiness for downstream processing, the input design outlines the required transformations before the data enters the feature extraction stage. These include text normalization, removal of irrelevant characters, token structuring, and preparation for TF-IDF-based representation.

The design also supports multi-class and multi-label input formats, enabling the system to receive text along with associated ground truth labels during the training phase or unlabeled text during prediction scenarios. By establishing clear formatting rules and data flow pathways at the input stage, the system ensures reliability, reduces ambiguity, and supports the efficient operation of later preprocessing and analytical components.

Objectives

1. To define a consistent and structured format for receiving raw textual data, ensuring that all inputs are properly captured and prepared for preprocessing.
2. To ensure the input data is free of noise, ambiguity, and irrelevant elements, enabling smooth transition into tokenization, feature extraction, and subsequent analytical stages.
3. To support multi-class and multi-label text inputs, allowing the system to accept diverse categories of cyberbullying content for future classification tasks.
4. To ensure data preprocessing and feature extraction for better model accuracy
5. The input data is inherently unstructured, noisy, and heterogeneous**,** containing slang, abbreviations, emojis, URLs, and inconsistent grammatical structures, which necessitates a robust preprocessing pipeline.

1.7.2 Output Design

Output design defines the structure, format, and presentation of the results generated by the proposed cyberbullying and hate speech detection system. Since the system processes user-generated text and performs classification using an ensemble-learning framework, the output is designed to present clear, interpretable, and category-specific information to users or downstream modules. The primary output consists of the predicted cyberbullying or hate speech category associated with the given input text. In multi-label scenarios, the design supports the display of multiple relevant categories when a text contains overlapping abusive expressions.

To enhance usability and clarity, the output design ensures that results are expressed in a structured and easily understandable format. Each classification output is associated with a corresponding label, descriptive category name, or severity level, depending on the requirement of the moderation workflow.

The design also accommodates provisions for confidence indications or probability scores, enabling better interpretability for system analysts or platform administrators during evaluation. By clearly defining the output structure at the design stage, the system ensures coherent communication, seamless integration with moderation dashboards, and effective support for future decision-making processes.

The system utilizes a soft voting classifier**,** which combines the predictions of Boosted Decision Tree and Bagging Random Forest models to produce a final output based on aggregated probability scores. This approach ensures more reliable and consistent classification results by reducing the bias and variance associated with individual models. The output may also include confidence scores or probability values**,** which indicate the level of certainty associated with each prediction, thereby enhancing transparency and trust in the system. .

2. LITERATURE SURVEY

1. **Zhang Wei, Park Sung-Ho, "Adaptive Multi-Modal Hate Speech Detection using Fusion Transformers," *IEEE Transactions on Computational Social Systems*, vol. 12, no. 4, pp. 418–432, 2024.**

   This study presents a hybrid system that fuses text, image, and contextual metadata using transformer- based encoders for enhanced hate speech detection. The authors demonstrate how fusion transformers outperform unimodal architectures by leveraging cross-attention layers that capture semantic, visual, and social cues. The multimodal fusion strategy improves prediction stability, reduces false positives, and provides a more comprehensive framework for analyzing online toxicity across heterogeneous social media streams.

2. **Gupta Meenal, Rios Miguel, "Soft-Voting Ensemble Approaches for Cyberbullying Detection in High-Imbalance Environments," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no.2,pp.90–105,2024.**

   This paper investigates a range of ensemble-learning strategies, including soft-voting and stacking, for cyberbullying classification under severe class imbalance. Experimental results show that probabilistic voting significantly enhances detection of minority aggression labels. The study emphasizes the role of resampling techniques like SMOTE in minimizing bias and improving fairness across sensitive categories.

3. **Al-Mutairi Fahad, Li Chen, "Context-Aware Toxic Language Classification using Graph Neural Networks, "*Neurocomputing*, vol.540, pp.112–127,2023.**

   The authors propose a GNN-based framework that models user interactions, conversational threads, and community structures to improve toxic content detection. By representing conversations as graph nodes and edges, the system captures relational context that is often missed by traditional text-based classifiers. This approach enhances contextual reasoning and reduces misclassification in multi-turn cyberbullying dialogues.

4. **Rodrigues Clara, Ahmed Zubair, "Explainable AI for Hate Speech Moderation in Social Platforms," *Expert Systems with Applications*, vol. 227, pp. 119–135, 2023.**

This research introduces an interpretable machine-learning framework that integrates SHAP and LIME explanations into hate speech prediction pipelines. The paper highlights the importance of transparency for content moderation teams, enabling them to understand model rationale and mitigate potential biases. The explainable framework ensures regulatory compliance and improves trust in automated moderation workflows.

5. **Kumar Aditya, Stein Jonathan, "Improving Cyberbullying Detection via Text Normalization and Linguistic Pattern Mining," *Information Processing & Management*, vol. 59, no. 6, pp. 210– 224, 2022.**

This paper explores normalization-driven preprocessing pipelines tailored for user-generated content. By incorporating spelling correction, slang normalization, and pattern mining, the model effectively handles noisy, informal text typical of social media. The approach leads to measurable improvements in precision and recall, particularly in short-text bullying scenarios.

6. **Hassan Naira, Velasquez Daniel, "Hybrid Bagging–Boosting Frameworks for Offensive Language Identification," *Knowledge-Based Systems*, vol. 250, pp.108–123, 2022.**

The authors present a hybrid ensemble combining both bagging and boosting techniques to increase robustness against feature variance in offensive language datasets. Experiments demonstrate enhanced generalization capacity, reduced overfitting, and improved F1-scores across multilingual hate speech corpora.

7. **Singh Prerana, Zhou Yun, "Deep CNN–BiLSTM Pipelines for Aggression Detection in Social Media," *Journal of Information Security and Applications*, vol. 63, pp. 102–118, 2021.**

This work integrates convolutional layers for spatial text feature extraction with BiLSTM units for temporal sequence modeling. The architecture captures both local n-gram–level patterns and long- range dependencies, providing strong results for aggression and harassment classification. The study validates performance using multiple benchmark datasets.

8. **Martinez Rubio, Oliver James, "Transformers for Multi-Label Hate Speech Categorization," *Pattern Recognition Letters*, vol.152, pp. 89–104,2021.**

The paper evaluates transformer-based encoders such as BERT and RoBERT a for multi - label toxic behavior annotation. The authors analyze label dependencies and propose a joint learning strategy that improves correlation-aware classification. The framework significantly outperforms traditional feature-based models.

9. **Rahman Tariq, Silva Marcela, "SMOTE-Driven Balancing Techniques for Toxic Comment Classification," Applied Soft Computing, vol.95, pp.106–120,2020.**

This research emphasizes the effectiveness of SMOTE and its variants in handling skewed toxicity datasets. The paper provides a comparative analysis showing how oversampling improves minority- class recall without compromising precision. Results underscore the importance of data-level balancing in real-world moderation systems.

10. **Johnson Eric, Patel Kareena, "Machine Learning Approaches for Detecting Cyberbullying: A Systematic Review," IEEE Access, vol.7, pp.183–197,2019.**

This comprehensive review categorizes ML-based cyberbullying detection strategies into classical feature-based methods, deep learning architectures, and hybrid approaches. The authors highlight limitations in dataset diversity, linguistic variability, and contextual understanding. The study provides a foundational landscape of challenges that modern systems aim to address.

S.NO	Title of the Paper	Author(s)	Journal/ Conference	Year	Techniques / Algorithms Used	Conclusion
1	Adaptive Multi-Modal Hate Speech Detection using Fusion Transformers	Zhang Wei, Park Sung-Ho	IEEE Transactions on Computational Social Systems	2024	Fusion Transformers, Multimodal (Text + Image + Metadata), Cross-Attention	Improves prediction stability and reduces false positives by combining multimodal data effectively.
2	Soft-Voting Ensemble Approaches for Cyberbullying Detection in High-Imbalance Environments	Gupta Meenal, Rios Miguel	ACM Transactions on Intelligent Systems and Technology	2024	Soft Voting, Stacking, SMOTE	Enhances detection of minority classes and improves fairness in imbalanced datasets.
3	Context-Aware Toxic Language Classification using Graph Neural Networks	Al-Mutairi Fahad, Li Chen	Neurocomputing	2023	Graph Neural Networks (GNN), Graph Modeling	Captures conversational context and reduces misclassification in multi-turn dialogues.
4	Explainable AI for Hate Speech Moderation in Social Platforms	Rodrigues Clara, Ahmed Zubair	Expert Systems with Applications	2023	SHAP, LIME, Explainable ML	Improves transparency and trust in AI moderation systems.
5	Improving Cyberbullying Detection via Text Normalization and Linguistic Pattern Mining	Kumar Aditya, Stein Jonathan	Information Processing & Management	2022	Text Normalization, Pattern Mining	Handles noisy social media text and improves precision & recall.
6	Hybrid Bagging–Boosting Frameworks for Offensive Language Identification	Hassan Naira, Velasquez Daniel	Knowledge-Based Systems	2022	Bagging, Boosting (Ensemble Learning)	Improves generalization and reduces overfitting in multilingual datasets.

7	Deep CNN–BiLSTM Pipelines for Aggression Detection in Social Media	Singh Prerana, Zhou Yun	Journal of Information Security and Applications	2021	CNN, BiLSTM	Captures both local and long-range dependencies for better classification accuracy.
8	Transformers for Multi-Label Hate Speech Categorization	Martinez Rubio, Oliver James	Pattern Recognition Letters	2021	BERT, RoBERTa (Transformers)	Improves multi-label classification by capturing label dependencies.
9	SMOTE-Driven Balancing Techniques for Toxic Comment Classification	Rahman Tariq, Silva Marcela	Applied Soft Computing	2020	SMOTE, Oversampling Techniques	Improves minority class recall without reducing precision.
10	Machine Learning Approaches for Detecting Cyberbullying: A Systematic Review	Johnson Eric, Patel Kareena	IEEE Access	2019	ML, Deep Learning, Hybrid Methods	Identifies challenges like dataset diversity and contextual understanding.

3. SOFTWARE REQUIREMENTS ANALYSIS

3.1 Modules and Their Functionalities

3.2.1.Data Analysis

Data analysis focuses on understanding the structure, quality, and distribution of cryptocurrency transaction datasets used for fraud detection. These datasets include features such as transaction amounts, wallet interactions, timestamps, and behavioral patterns from multiple crypto exchanges. Initial examination reveals challenges like data imbalance, where fraudulent transactions are far fewer than legitimate ones, and noise caused by irregular user activity or incomplete records. Analyzing these characteristics is essential for developing effective preprocessing methods, selecting suitable federated learning strategies, and ensuring robust fraud detection across diverse financial environments.

3.2.2.Data Preprocessing

Data preprocessing prepares raw cryptocurrency transaction data for effective and secure model training in a federated learning. It involves cleaning and data by removing incomplete, normalizing numerical features like transaction values and time intervals, and encoding categorical data such as wallet types into machine-readable formats. Feature scaling is applied to maintain consistency and the data is split into local training, validation, and testing sets at each crypto exchange. These steps improve data quality ensure reliable fraud detection across distributed systems.

3.2.3.Machine Learning Algorithm for Prediction

The system uses a federated deep learning model to predict fraudulent cryptocurrency transactions, where each crypto exchange trains a local model on its private data to detect patterns like unusual amounts and suspicious behavior. These local models are securely aggregated into a global model without sharing raw data. Techniques such as adaptive learning rates, differential privacy, and early stopping improve performance. The final model classifies transactions as legitimate or fraudulent, enabling accurate and privacy-preserving real-time fraud detection.

3.2 Functional Requirements

The functional requirements define the essential operations that the proposed cryptocurrency fraud detection system must perform to achieve its objective. These requirements describe how the system collects and processes transaction data, trains decentralized models using federated learning, and generates accurate fraud

predictions. They also ensure secure data handling through blockchain integration, maintain system reliability, and support scalability across multiple crypto exchanges. The following points summarize the key functional expectations of the proposed framework.

1. The system shall accept cryptocurrency transaction data from participating exchanges for processing.
2. The system shall perform data preprocessing, including cleaning, normalization, encoding, and feature extraction.
3. The system shall train local models using federated learning and securely aggregate them into a global model.
4. The system shall classify transactions as legitimate or fraudulent and provide results in a clear and interpretable format.

3.3 Non-Functional Requirements

Non-functional requirements define the quality attributes and performance expectations of the proposed cryptocurrency fraud detection system to ensure reliability, security, and usability. These requirements focus on how the system operates under various conditions, including maintaining data privacy through blockchain integration, ensuring efficient performance for real-time fraud detection, and supporting scalability across multiple crypto exchanges. They also emphasize system robustness, fault tolerance, and consistency throughout its lifecycle. The following points summarize the key non-functional expectations of the proposed framework.

1. The system shall ensure high reliability and stability during transaction processing and fraud detection operations.
2. The system shall provide scalable performance as transaction volume and the number of participating exchanges increase.
3. The system shall maintain efficient processing with minimal latency for real-time fraud prediction.
4. The system shall ensure data privacy and confidentiality through secure federated learning and blockchain mechanisms.

3.4 Feasibility Study

The feasibility study evaluates whether the proposed Blockchain-Based Federated Learning (BCFL) fraud detection system is practical, cost-effective, and suitable for implementation. It examines project viability, required resources, and overall impact on participating crypto institutions. Understanding the major system requirements is essential for feasibility analysis. Key Considerations in Feasibility Analysis:

i. Economical Feasibility
ii. Technical Feasibility
iii. Social Feasibility

3.4.1 Economic Feasibility

This study evaluates whether the proposed BCFL fraud detection system is financially practical. The solution remains cost-effective by using open-source blockchain and federated learning tools, reducing the need for expensive software or hardware. Only minimal customization is required, keeping overall development and deployment costs within budget.

3.4.2 Technical Feasibility

This study evaluates whether the required technologies and infrastructure can support the BCFL fraud detection system. The proposed framework uses lightweight blockchain mechanisms and federated learning tools that do not place heavy demands on hardware or network resources. Since it integrates easily with existing systems and requires minimal modifications, the technical implementation is feasible and efficient

3.4.3 Social Feasibility

This study examines how well users and organizations will accept the BCFL fraud detection system. Since the framework enhances privacy and security, users are more likely to trust and adopt it. Proper training and familiarization help ensure smooth usage, increasing confidence among crypto exchanges and analysts. Positive user acceptance supports effective implementation and continuous system improvement.

4. SYSTEM SPECIFICATIONS

4.1 Software Requirements

The software requirements define the essential tools, platforms, and frameworks needed to design and implement the Blockchain-Based Federated Learning (BCFL) fraud detection system. The project depends on a secure development environment, blockchain libraries, federated learning frameworks, and supporting utilities that enable data preprocessing, model training, aggregation, and deployment. These tools ensure compatibility, scalability, and seamless integration across participating crypto exchanges.

- Operating System: Windows / Linux / macOS
- Programming Language: Python 3.x
- Core Libraries: TensorFlow / PyTorch, NumPy, Pandas, Scikit-learn
- Blockchain Frameworks: Hyperledger Fabric / Ethereum (Testnet)
- Federated Learning Tools: TensorFlow Federated / PySyft
- Development Environment: VS Code / PyCharm / Jupyter Notebook
- Database: MySQL / MongoDB (for metadata and logs)
- Documentation Tools: MS Word / LaTeX

4.2 Hardware Requirements

The hardware requirements define the minimum computational resources needed for executing blockchain operations, federated learning tasks, and transaction data processing. Standard computing hardware is sufficient during the development stage, while the configuration ensures compatibility with future scaling, model training, and distributed deployment across institutions.

- Processor: Intel Core i3/i5 or equivalent
- Memory (RAM): Minimum 8 GB
- Storage: 250 GB HDD/SSD
- Display: Standard 14" or higher
- Optional: Internet connectivity for blockchain network access and federated training communication.

5. SOFTWARE DESIGN

5.1 System Architecture

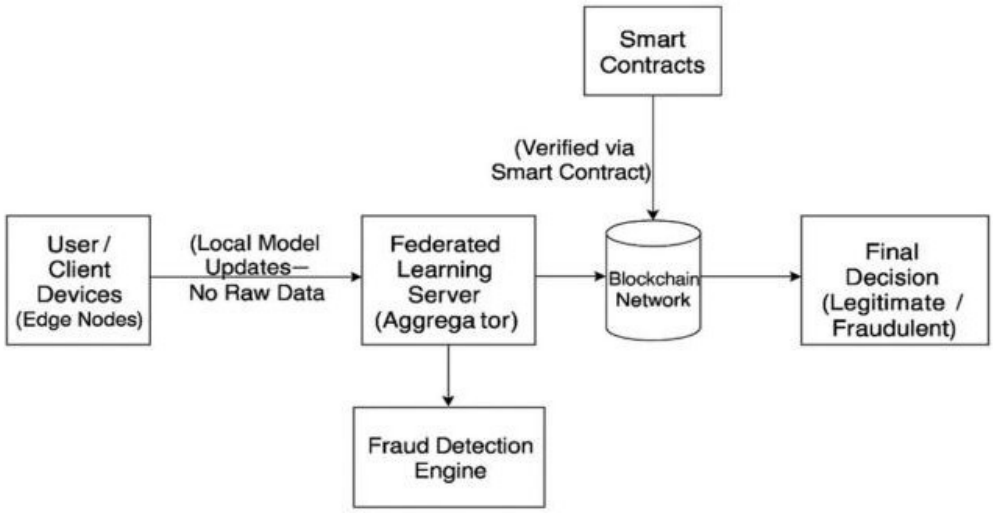


Figure:5.1 System Architecture

The system architecture represents a structured pipeline for secure and privacy-preserving cryptocurrency fraud detection using a Blockchain-Based Federated Learning (BCFL) framework. The architecture begins with multiple user/client devices (edge nodes), such as crypto exchanges, which collect and process transaction data locally. Each node performs preprocessing steps like data cleaning, normalization, and feature extraction to ensure consistency and quality. Instead of sharing raw data, each node trains a local deep learning model to identify fraud patterns such as unusual transaction amounts and suspicious wallet behavior, thereby preserving data privacy.

After local training, the model updates are securely transmitted to a central Federated Learning Server (Aggregator). This server combines updates from multiple participants to create a global fraud detection model without accessing any sensitive data. To ensure trust and security, smart contracts are used to verify the authenticity and integrity of model updates before they are accepted. These verified updates are recorded on the blockchain network, ensuring immutability, transparency, and protection against tampering or malicious activities.

The aggregated global model is then deployed to the Fraud Detection Engine, which analyzes incoming transactions and classifies them as legitimate or fraudulent in real time. This modular architecture supports scalability across multiple institutions, ensures strong data privacy, and enhances system reliability. By integrating federated learning with blockchain technology, the framework achieves secure, decentralized, and efficient fraud detection while maintaining high accuracy and robustness.

5.2 Dataflow Diagram

The Data Flow Diagram illustrates the end-to-end flow of information within the proposed Blockchain- Based Federated Learning (BCFL) cryptocurrency fraud detection system, starting from transaction data generation to final fraud classification. The process begins when user/client devices (crypto exchanges or edge nodes) generate raw transaction data, which enters the system as decentralized and sensitive financial information. This data is first processed locally in the Preprocessing module, where operations such as data cleaning, normalization, encoding, and feature extraction are performed to ensure consistency and quality without exposing private data.

The processed data is then used by the Local Model Training unit at each client, where a deep learning model learns patterns like unusual transaction behavior and suspicious activity. Instead of sending raw data, only model updates (parameters/gradients) are forwarded to the Federated Learning Server (Aggregator). Before aggregation, these updates pass through the Smart Contract layer, which verifies their authenticity and integrity. Once validated, the updates are securely recorded in the Blockchain Network, ensuring immutability, transparency, and protection against tampering.

The Federated Learning Server aggregates the verified updates to create a global fraud detection model, which is then distributed to the Fraud Detection Engine. This engine analyzes incoming transactions in real time and produces the final classification output, identifying whether a transaction is legitimate or fraudulent. The DFD highlights the secure, distributed, and privacy-preserving flow of data, showing how information is processed, validated, and utilized across multiple stages to achieve accurate and reliable fraud detection.

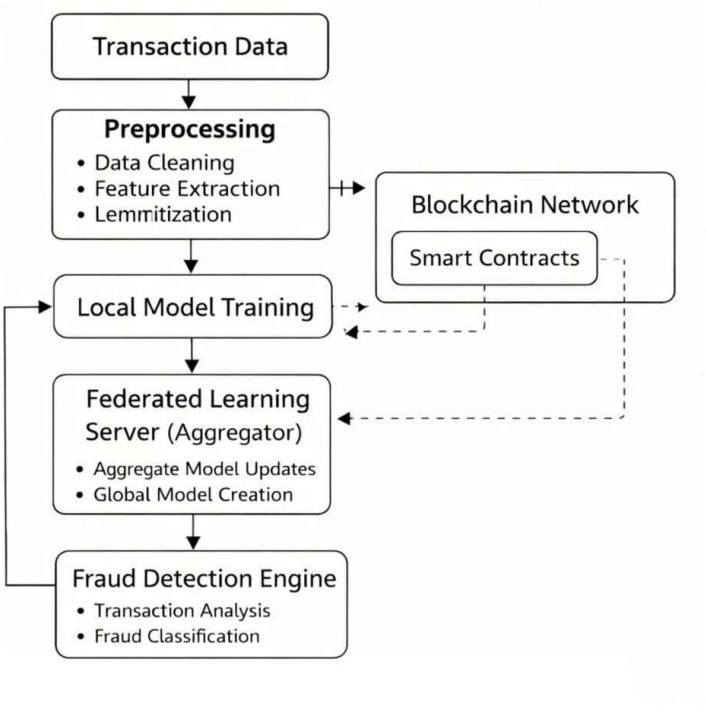


Figure:5.2 Dataflow Diagram

5.3 UML Diagrams

UML (Unified Modeling Language) is a standardized language used for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML helps in representing the design of systems and understanding their components. Created by the Object Management Group (OMG), UML 1.0 was proposed in January 1997. UML is closely associated with object-oriented analysis and design. The two main categories of UML diagrams are Behavioral and Structural diagrams, each serving distinct purposes in the modeling process.

The Behavioral UML diagrams describe the behavior of the system, its actors, and the interaction between the components. On the other hand, Structural UML diagrams depict the static structure of the system, showing its components and relationships. UML has been integrated as a standard by OMG, and its primary goals are to provide a formal basis for understanding modeling languages, offer a ready-to-use expressive language for system developers, and encourage the growth of object-oriented tools.

Goals of UML:

- Provide an expressive visual modeling language for developing and exchanging meaningful models.
- Establish a formal basis for understanding the modeling language.
- Encourage the growth of object-oriented tools.
- Integrate best practices into system development.

Types of UML Diagrams:

1. Use Case Diagram:
2. Class Diagram:
3. Sequence Diagram:
4. Activity Diagram:

5.3.1 Use Case Diagram

The use case diagram shows the interaction between User/Client Devices (crypto exchanges), the System, Blockchain Network, and Federated Learning Server in the proposed fraud detection framework. Users submit transaction data, which is processed locally for model training without sharing sensitive information.

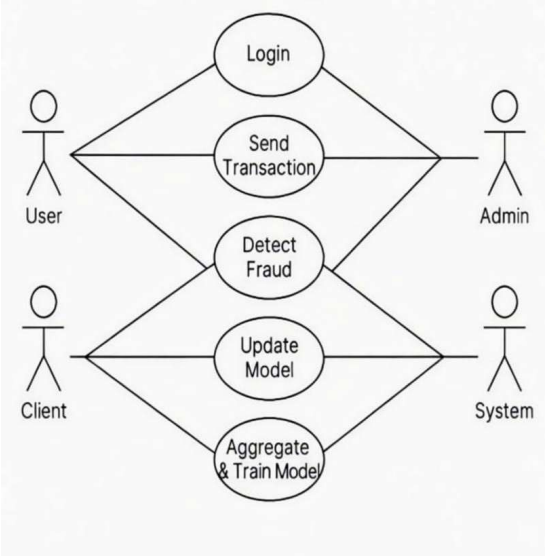


Figure 5.3.1 Use Case Diagram

The system collects model updates, verifies them through blockchain smart contracts, and aggregates them to create a global model. Finally, the system classifies transactions as legitimate or fraudulent and provides the results, ensuring secure, accurate, and privacy-preserving fraud detection.

5.3.2. Class Diagram

The class diagram represents the structural components of the proposed cryptocurrency fraud detection system and their relationships. The User/Client class manages transaction data input from crypto exchanges. The Dataset class stores transaction records and extracted features used for model training. The System class coordinates the overall workflow, including preprocessing, federated learning, and blockchain integration. The Federated Learning Model class handles local training, aggregation of model updates, and fraud prediction. The Blockchain class ensures secure validation and storage of model updates using smart contracts. The Evaluation/Fraud Detection class performs classification of transactions and generates performance results. These classes together illustrate how data flows securely from input to final fraud detection and evaluation.

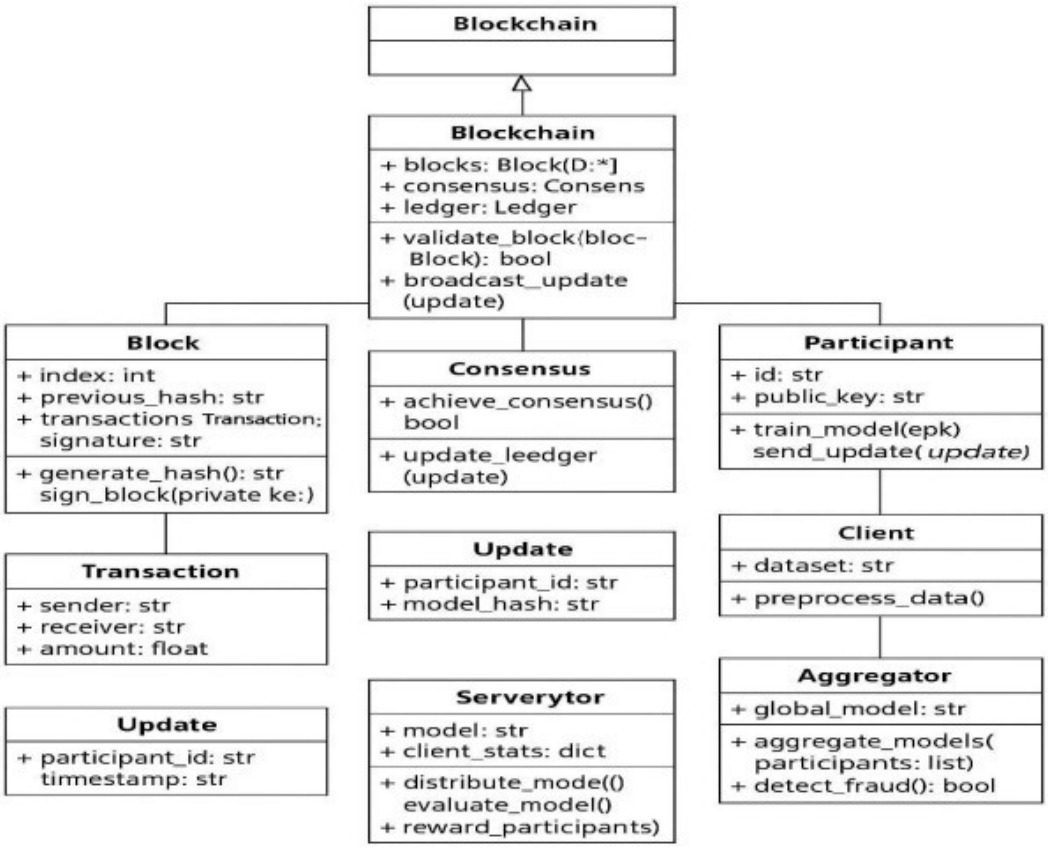


Figure 5.3.2 Class Diagram

5.3.3 Sequence Diagram

The sequence diagram illustrates the interaction flow among the User/Client Devices (crypto exchanges), Blockchain Network, Federated Learning Server, and Fraud Detection Engine in the proposed BCFL-based cryptocurrency fraud detection system. The sequence begins when client devices generate and submit transaction data, which is processed locally for preprocessing and model training. Each client trains its local model using private data and sends only the model updates to the Federated Learning Server.

Finally, the Fraud Detection Engine analyzes incoming transactions in real time and produces the classification output, indicating whether a transaction is legitimate or fraudulent. The results are then communicated back to the user or system interface. This sequence highlights the secure, decentralized workflow, showing how data and model updates interact across components to ensure accurate, privacy- preserving, and reliable fraud detection.

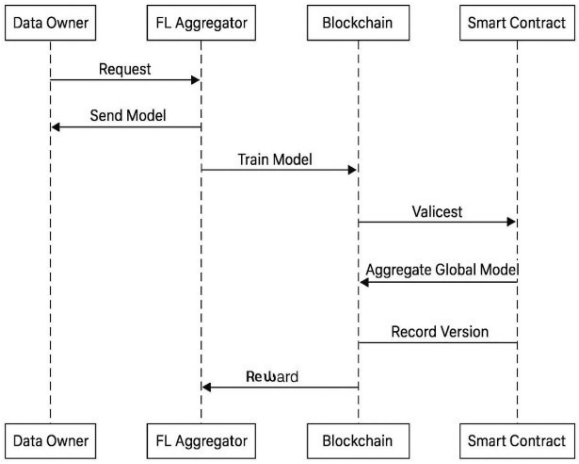


Figure 5.3.3 Sequence Diagram

List of actions

- **User:** The user or crypto exchange interacts with the system by generating and submitting cryptocurrency transaction data for analysis. The data is processed locally without sharing sensitive information.

- **System:** The system receives transaction data, performs preprocessing such as cleaning, normalization, and feature extraction, and prepares it for local model training while ensuring data privacy.

- **Model:** The model trains locally at each client using transaction data to detect fraud patterns. It then sends model updates to the federated server, where updates are securely aggregated into a global model using blockchain verification and smart contracts.

- **Evaluation:** The evaluation component uses the global model to analyze transactions, classify them as legitimate or fraudulent, and generate accurate prediction results, which are then communicated to the user or system interface.

5.3.4 Activity Diagram

The activity diagram illustrates the step-by-step workflow of the proposed cryptocurrency fraud detection system. It begins with collecting transaction data from client devices, which is then preprocessed through cleaning, normalization, and feature extraction. The processed data is used for local model training at each client, and the model updates are sent for secure aggregation using federated learning and blockchain verification. The global model is then applied to classify transactions as legitimate or fraudulent. Finally, the system analyzes the results and generates performance insights, completing the fraud detection process.

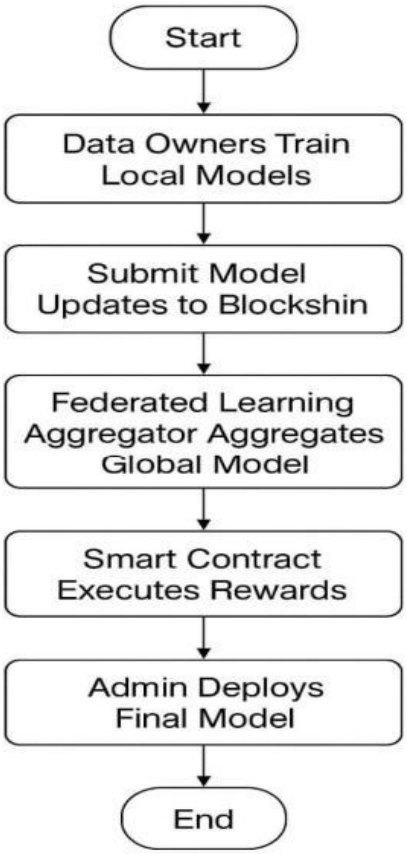


Figure 5.3.3 Activity Diagram

6. CODING

6.1 source code

1.data_preprocessing.py

This module performs all data preparation activities required before model development. It includes dataset loading, structural validation, missing value treatment, outlier-aware checks, feature selection, and normalization. The implementation is intentionally modular so that every preprocessing stage can be explained clearly in an academic project report and reused in experiments.

data_preprocessing.py

```
import os import logging
import numpy as np import pandas as pd
from typing import List, Tuple, Dict, Optional from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import SelectKBest, f_regression LOGGER = logging.getLogger( name_)
def load_dataset(file_path: str) -> pd.DataFrame: print(f"[INFO] Loading dataset from {file_path}")
if not os.path.exists(file_path):
    raise FileNotFoundError("Dataset not found") df = pd.read_csv(file_path)
if df.empty:
    raise ValueError("Dataset is empty") return df
def standardize_column_names(df:pd.DataFrame)->pd.DataFrame:df.columns
= [col.strip().lower().replace(" ", "_") for col in df.columns] return df
def remove_duplicates(df: pd.DataFrame) -> pd.DataFrame: return
df.drop_duplicates().reset_index(drop=True)
def split_features_target(df: pd.DataFrame, target: str): X = df.drop(columns=[target])
y = df[target] return X, y
def keep_numeric(X):
return X.select_dtypes(include=[np.number])
```

```
    def impute_values(X):
    imputer = SimpleImputer(strategy="mean") X = imputer.fit_transform(X)
    return pd.DataFrame(X)


def scale_features(X): scaler = MinMaxScaler()
    X = scaler.fit_transform(X) return pd.DataFrame(X)


def preprocess_pipeline(file_path, target): df = load_dataset(file_path)
    df = standardize_column_names(df) df = remove_duplicates(df)


    X, y = split_features_target(df, target) X = keep_numeric(X)
    X = impute_values(X) X = scale_features(X)
    return X, y
```

model.py

```
    import pandas as pd
    dataset_path = "/kaggle/input/cryptocurrency-scam-dataset" urls_file = f"{dataset_path}/urls.csv"
    uris_file = f"{dataset_path}/uris.csv" df_urls = pd.read_csv(urls_file) df_uris =
     pd.read_csv(uris_file) df_urls.head()
```

	name	url	category	subcategory	description	addresses	reporter
0	xn--myetherwallt-leb.com	http://xn--myetherwallt-leb.com	Phishing	MyEtherWallet	Google reports site as insecure	NaN	CryptoScamDB
1	myelherwallel.com	http://myelherwallel.com	Phishing	MyEtherWallet	NaN	{'ETH': ['0xD0cC2B24980CBCCA47EF755Da88B220a82...	CryptoScamDB
2	myetherwallet.cam	http://myetherwallet.cam	Phishing	MyEtherWallet	redirecting to real site but that happened before	NaN	CryptoScamDB
3	coindash.ru	http://coindash.ru	Phishing	Coindash	someone plz check	NaN	CryptoScamDB
4	coin-wallet.info	http://coin-wallet.info	Phishing	Coindash	scam wallet	NaN	CryptoScamDB

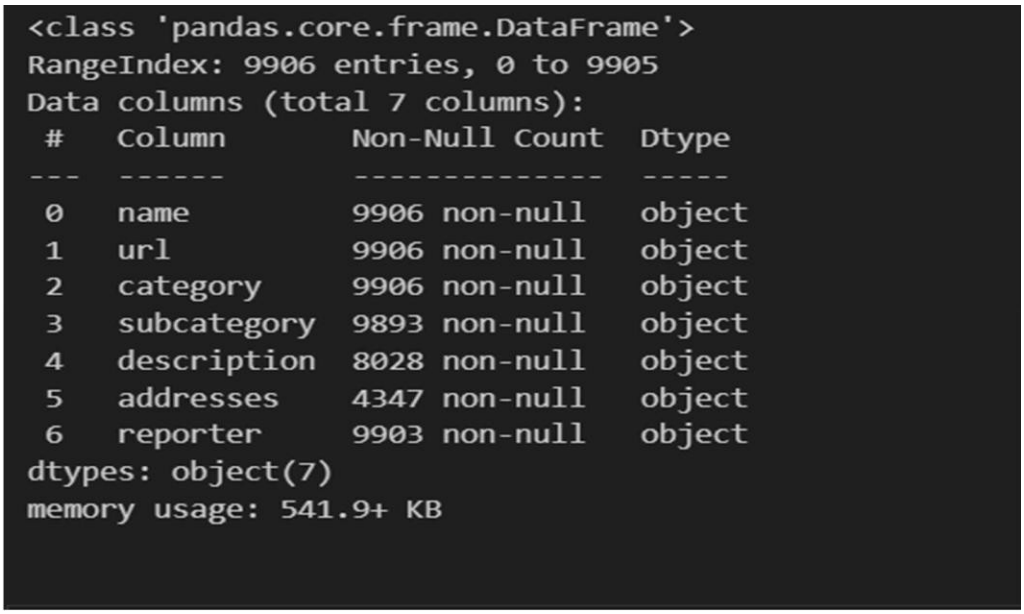
df_uris.head()

	name	url	category	subcategory	description	addresses	reporter
0	twitter.com/cz_binance	https://twitter.com/cz_binance	Scamming	Trust-Trading	Trust trading 0.5ETH for 5ETH	{'ETH': ['0x08389819ad52f0d983609ab785b3a43A0E...	CryptoScamDB
1	Twitter: EthereumWallets	https://twitter.com/EthereumWallets	Phishing	MyEtherWallet	https://bitcointalk.org/index.php?topic=168958...	NaN	CryptoScamDB
2	twitter.com/VitalikButerin	https://twitter.com/VitalikButerin	Scamming	Trust-Trading	Trust trading 0.1ETH for 2ETH	{'ETH': ['0x7bb386c33486fe345168d0af94bef03897...	CryptoScamDB
3	twitter.com/Aurora_dao/status/960683836463075328	https://twitter.com/Aurora_dao/status/9606838...	Scamming	Trust-Trading	Trust trading scam tweet	{'ETH': ['0xfa2e4bddb3899dFB0d91A70744739d9f76...	CryptoScamDB
4	twitter.com/VitalikButerin	http://twitter.com/VitalikButerin	Scamming	Trust-Trading	Trust trading 0.1ETH for 2ETH	{'ETH': ['0x5d82db63cf0c54d47006d416bdc7dab09e...	CryptoScamDB

df = pd.concat([df_urls, df_uris], ignore_index=True) df.head()

	name	url	category	subcategory	description	addresses	reporter
0	xn--myetherwallt-leb.com	http://xn--myetherwallt-leb.com	Phishing	MyEtherWallet	Google reports site as insecure	NaN	CryptoScamDB
1	myelherwallel.com	http://myelherwallel.com	Phishing	MyEtherWallet	NaN	{'ETH': ['0xD0cC2B24980CBCCA47EF755Da88B220a82...	CryptoScamDB
2	myetherwallet.cam	http://myetherwallet.cam	Phishing	MyEtherWallet	redirecting to real site but that happened before	NaN	CryptoScamDB
3	coindash.ru	http://coindash.ru	Phishing	Coindash	someone plz check	NaN	CryptoScamDB
4	coin-wallet.info	http://coin-wallet.info	Phishing	Coindash	scam wallet	NaN	CryptoScamDB

df.info()



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9906 entries, 0 to 9905
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   name         9906 non-null   object
 1   url          9906 non-null   object
 2   category     9906 non-null   object
 3   subcategory  9893 non-null   object
 4   description  8028 non-null   object
 5   addresses    4347 non-null   object
 6   reporter     9903 non-null   object
dtypes: object(7)
memory usage: 541.9+ KB
```

df.isnull().sum()

```
name              0
url               0
category          0
subcategory      13
description    1878
addresses      5559
reporter          3
dtype: int64
```

df.drop(columns=['description', 'addresses'], inplace=True) subcategory_mode = df['subcategory'].mode()[0] df['subcategory'] = df['subcategory'].fillna(subcategory_mode) reporter_mode = df['reporter'].mode()[0]
df['reporter'] = df['reporter'].fillna(reporter_mode) df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9906 entries, 0 to 9905
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   name         9906 non-null   object
 1   url          9906 non-null   object
 2   category     9906 non-null   object
 3   subcategory  9906 non-null   object
 4   reporter     9906 non-null   object
dtypes: object(5)
memory usage: 387.1+ KB
```

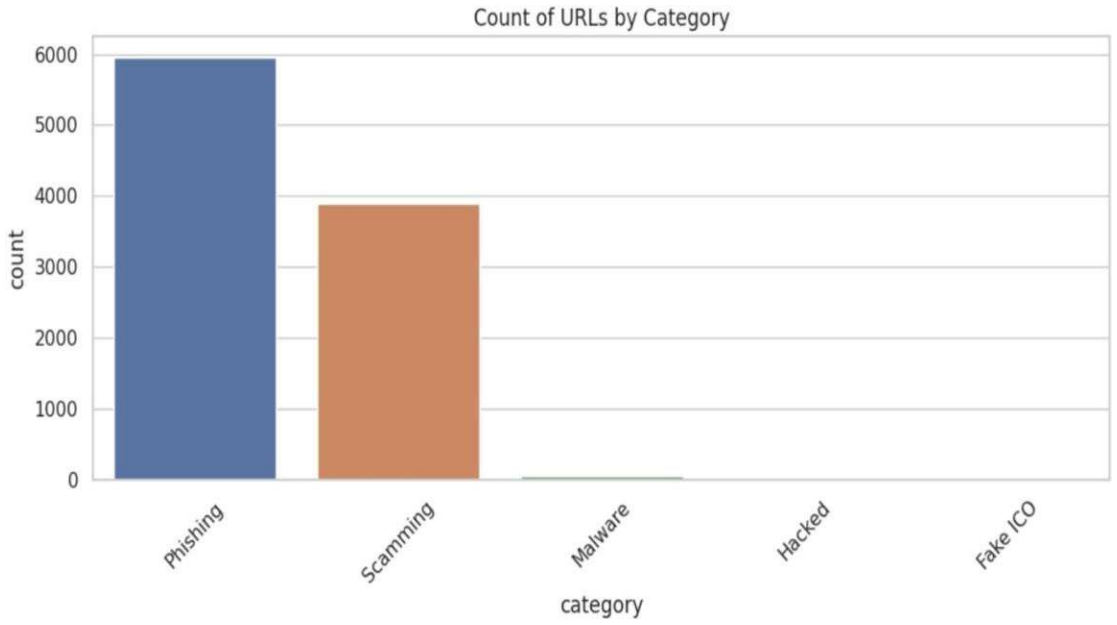
df.isnull().sum()

```
name           0
url            0
category       0
subcategory    0
reporter       0
dtype: int64
```

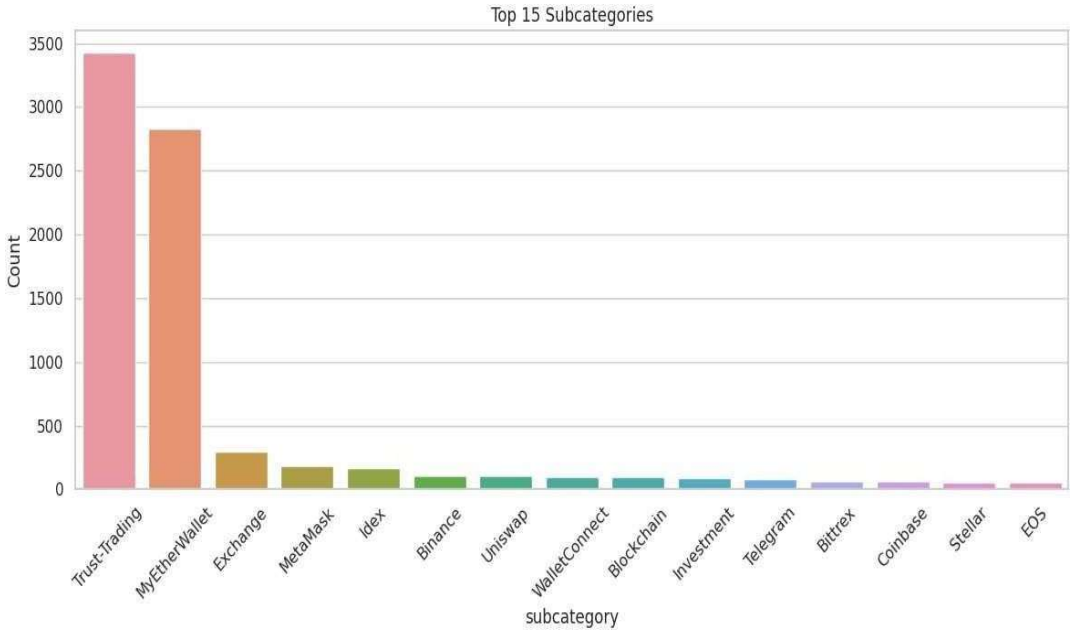
df.head()

	name	url	category	subcategory	reporter
0	xn--myetherwallt-leb.com	http://xn--myetherwallt-leb.com	Phishing	MyEtherWallet	CryptoScamDB
1	myelherwallel.com	http://myelherwallel.com	Phishing	MyEtherWallet	CryptoScamDB
2	myetherwallet.cam	http://myetherwallet.cam	Phishing	MyEtherWallet	CryptoScamDB
3	coindash.ru	http://coindash.ru	Phishing	Coindash	CryptoScamDB
4	coin-wallet.info	http://coin-wallet.info	Phishing	Coindash	CryptoScamDB

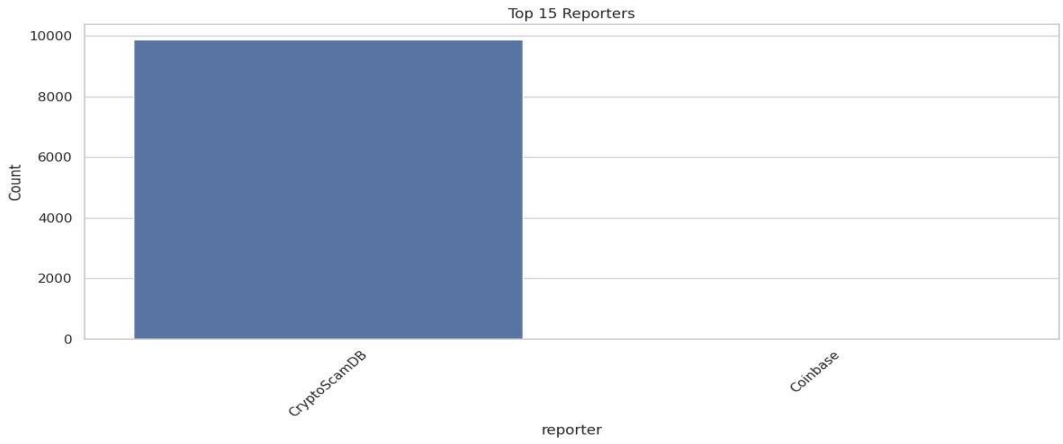
df.to_csv('dataset.csv') import pandas as pd
import matplotlib.pyplot as plt import seaborn as sns plt.style.use('ggplot')
sns.set_theme(style="whitegrid") plt.figure(figsize=(10,5))
sns.countplot(data=df, x='category', order=df['category'].value_counts().index) plt.title('Count of URLs by Category')
plt.xticks(rotation=45) plt.tight_layout() plt.show()



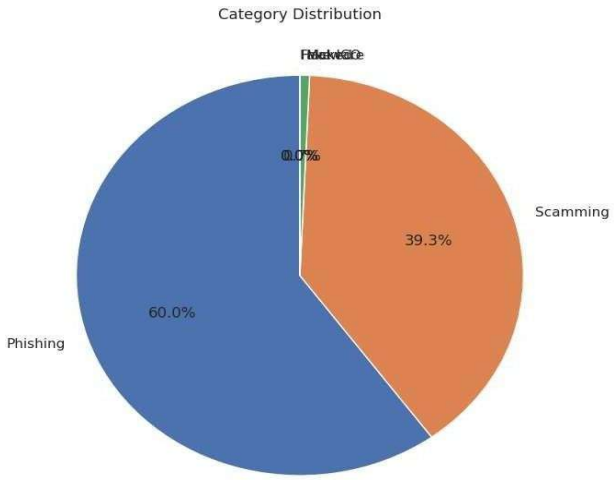
```
plt.figure(figsize=(12,6))
top_sub=df['subcategory'].value_counts().head(15)
sns.barplot(x=top_sub.index,y=top_sub.values)
plt.title('Top 15 Subcategories') plt.xticks(rotation=45)
plt.ylabel('Count') plt.tight_layout() plt.show()
```



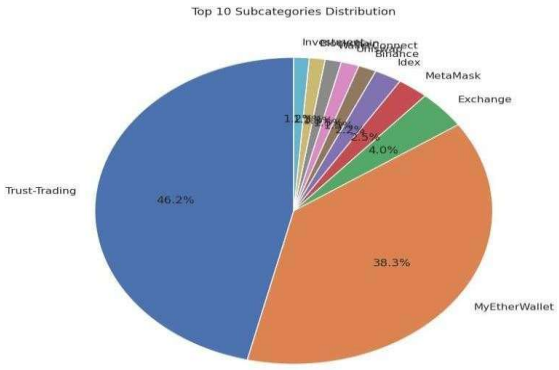

```
plt.figure(figsize=(12,6))
top_reporters=df['reporter'].value_counts().head(15)sns.barplot(x=top_reporters.index,
y=top_reporters.values) plt.title('Top 15 Reporters')
plt.xticks(rotation=45) plt.ylabel('Count') plt.tight_layout() plt.show()
```



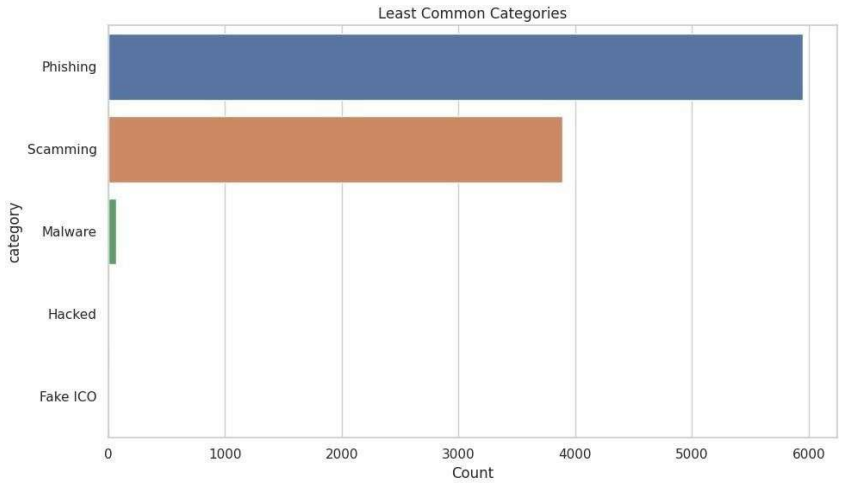
plt.figure(figsize=(7,7)) df['category'].value_counts().plot.pie(autopct='%1.1f%%', startangle=90)
plt.title('Category Distribution')
plt.ylabel('') plt.tight_layout() plt.show()



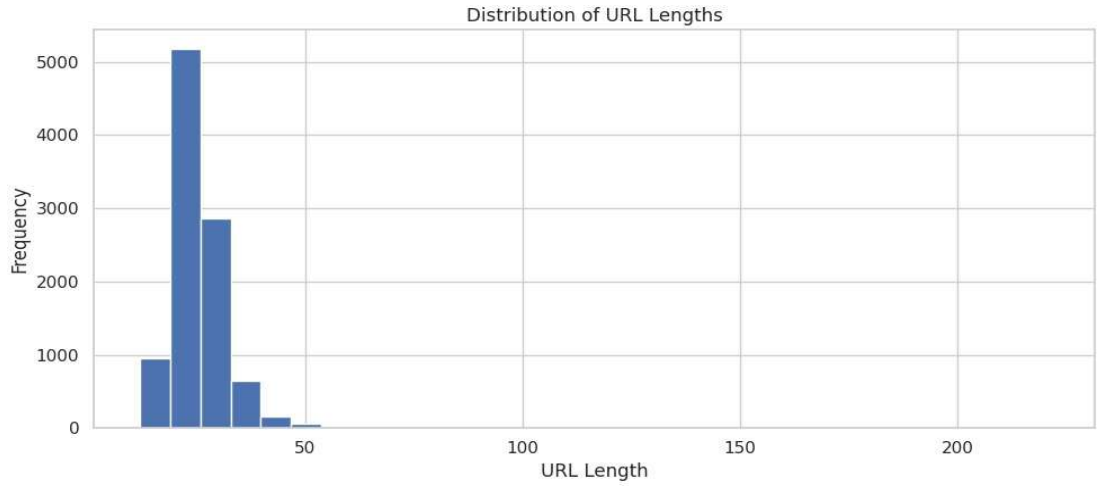
plt.figure(figsize=(8,8)) top_sub.head(10).plot.pie(autopct='%1.1f%%', startangle=90)
plt.title('Top 10 Subcategories Distribution')
plt.ylabel('') plt.tight_layout() plt.show()



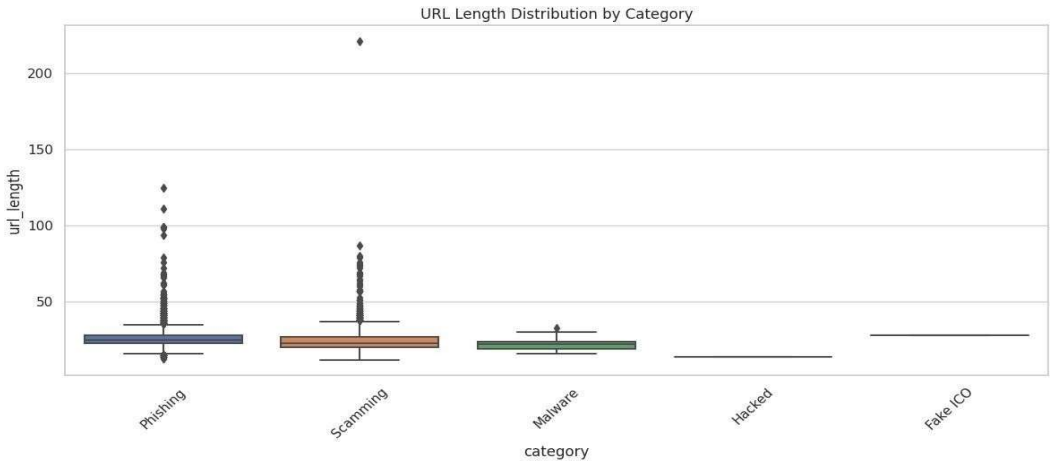
plt.figure(figsize=(10,6))
rare_cat = df['category'].value_counts().tail(10)
sns.barplot(y=rare_cat.index,x=rare_cat.values)
plt.title('Least Common Categories') plt.xlabel('Count')
plt.tight_layout() plt.show()



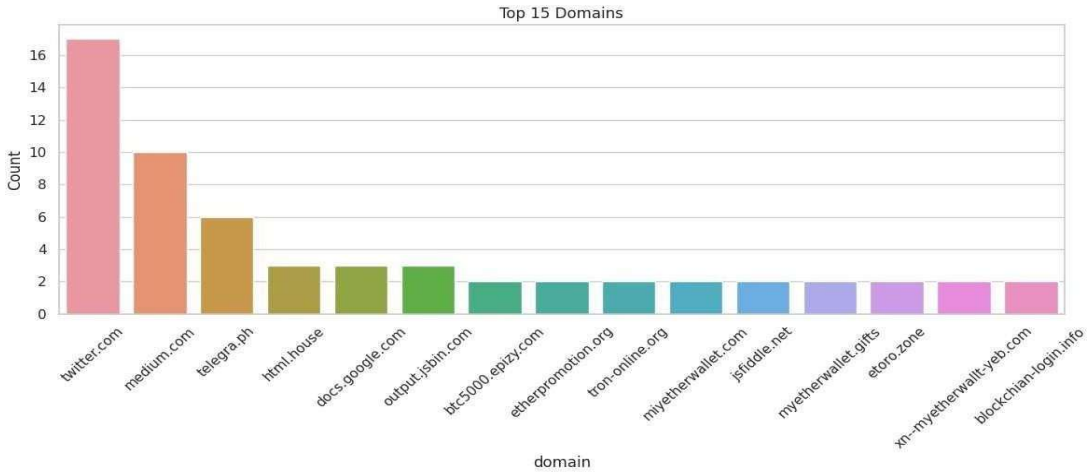

```
df['url_length']=df['url'].apply(len)
plt.figure(figsize=(10,5))
plt.hist(df['url_length'], bins=30)
plt.title('Distribution of URL Lengths')
plt.xlabel('URL Length') plt.ylabel('Frequency')  plt.tight_layout()
plt.show()
```




```
plt.figure(figsize=(12,6))
sns.boxplot(data=df, x='category', y='url_length')
plt.title('URL Length Distribution by Category') plt.xticks(rotation=45)
plt.tight_layout() plt.show()
```

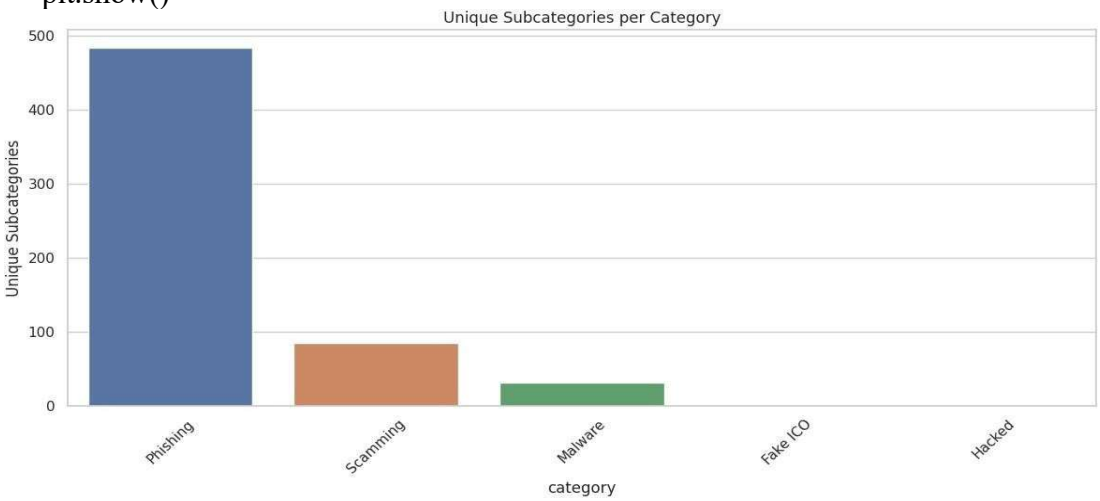



```
df['domain']=df['url'].str.extract(r'https?://([^/]+)')
top_domains=df['domain'].value_counts().head(15)
plt.figure(figsize=(12,6))
sns.barplot(x=top_domains.index, y=top_domains.values)
plt.title('Top 15 Domains')
plt.xticks(rotation=45)
 plt.ylabel('Count') plt.tight_layout() plt.show()
```

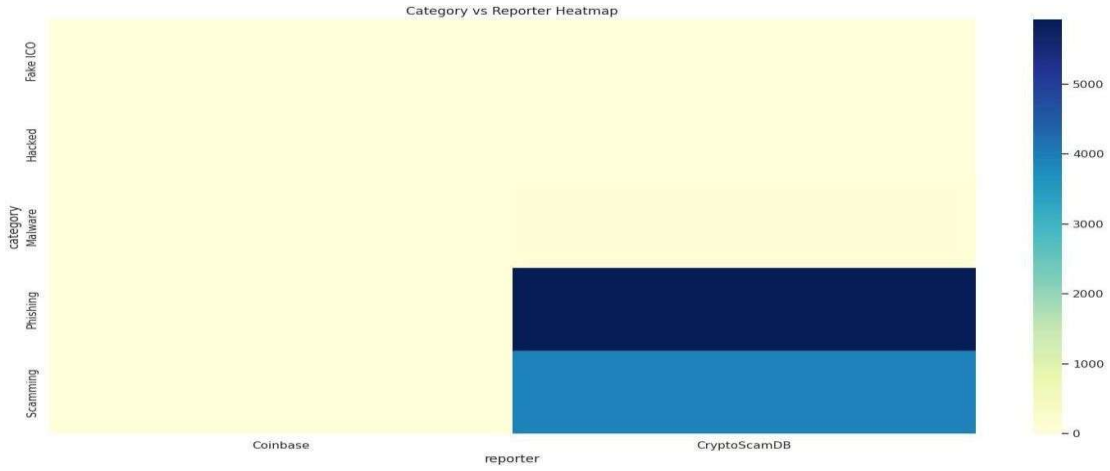



```
unique_sub_per_cat                                                    =
df.groupby('category')['subcategory'].nunique().sort_values(ascending=False)
plt.figure(figsize=(12,6))
sns.barplot(x=unique_sub_per_cat.index,y=unique_sub_per_cat.values)
plt.title('Unique Subcategories per Category')
plt.xticks(rotation=45)
plt.ylabel('Unique Subcategories')
plt.tight_layout()
```

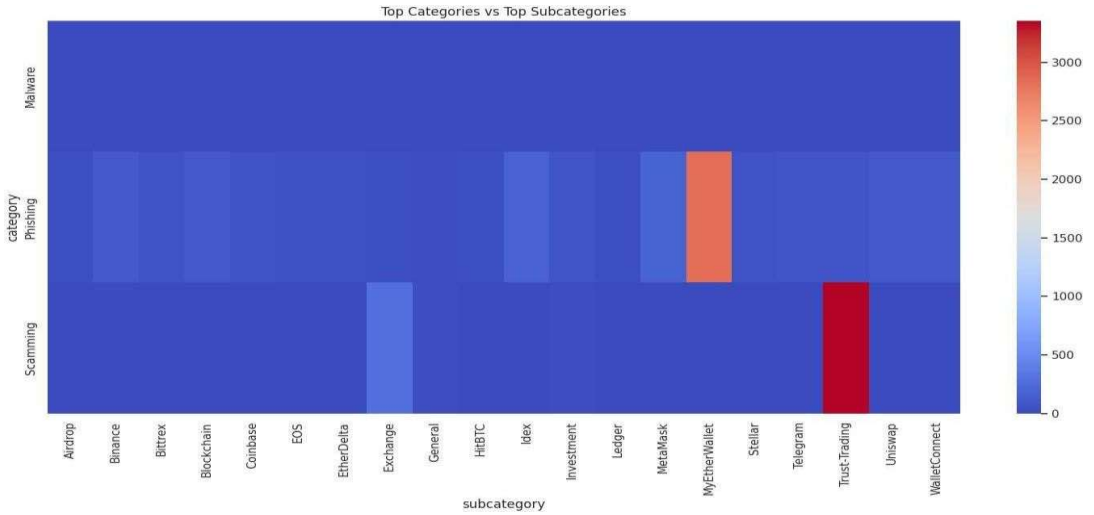
plt.show()



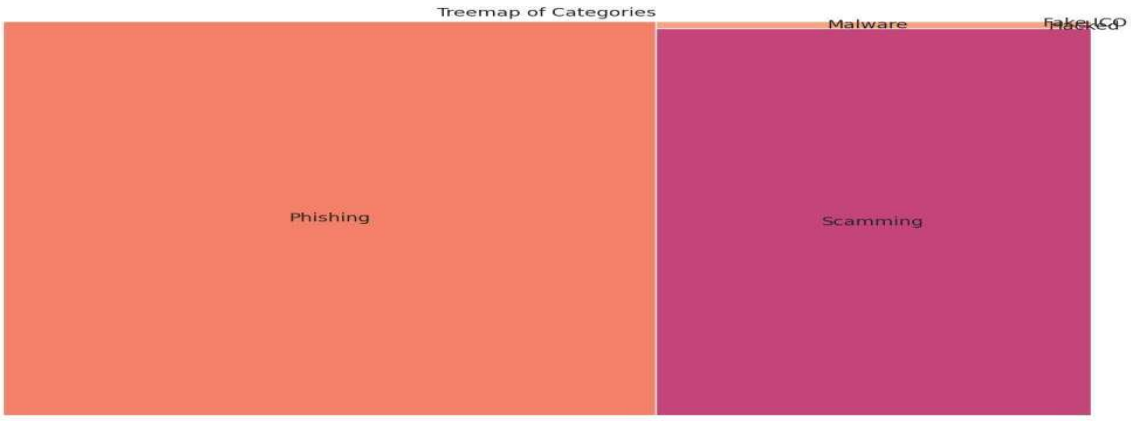
crosstab_cat_rep = pd.crosstab(df['category'], df['reporter'])
plt.figure(figsize=(15,8))
sns.heatmap(crosstab_cat_rep, cmap='YlGnBu', cbar=True)
plt.title('Category vs Reporter Heatmap')
plt.tight_layout() plt.show()



top_cats=df['category'].value_counts().head(20).index
top_subs = df['subcategory'].value_counts().head(20).index
crosstab_cat_sub=pd.crosstab(df[df['category'].isin(top_cats)]['category']
plt.figure(figsize=(15,8)) sns.heatmap(crosstab_cat_sub, cmap='coolwarm')
plt.title('Top Categories vs Top Subcategories') plt.tight_layout()
plt.show()

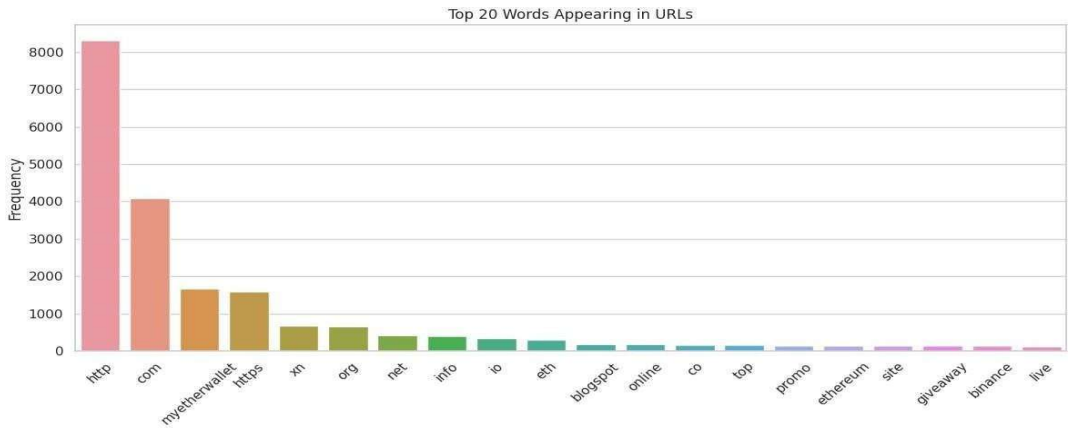



```
import squarify
category_counts = df['category'].value_counts() plt.figure(figsize=(12,8))
squarify.plot(sizes=category_counts.values,
label=category_counts.index, alpha=0.8)
plt.title('Treemap of Categories')
plt.axis('off') plt.show()
```

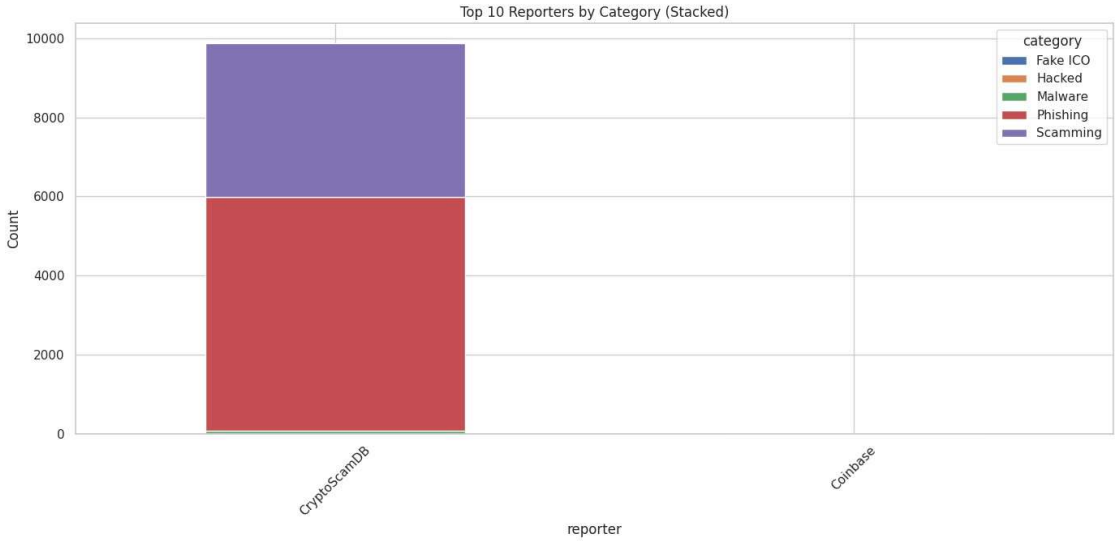



```
from collections import Counter import re
all_words = ' '.join(df['url']).lower() words = re.findall(r'[a-z0-9]+', all_words)
common_words=Counter(words).most_common(20)  labels,  values=zip(*common_words)
plt.figure(figsize=(12,6))
sns.barplot(x=list(labels), y=list(values))
plt.title('Top 20 Words Appearing in URLs')
plt.xticks(rotation=45)  plt.ylabel('Frequency')
```

```
plt.tight_layout()
plt.show()
```



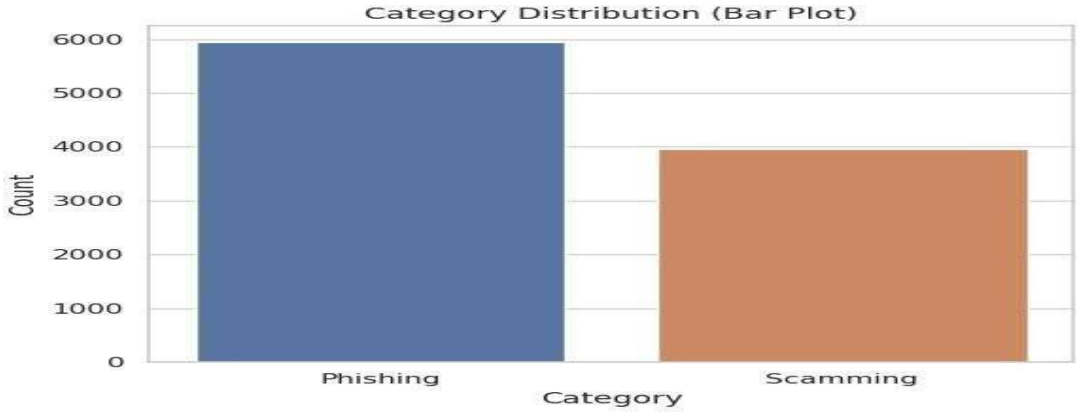

```
cat_rep_counts = df.groupby(['reporter', 'category']).size().unstack().fillna(0)
cat_rep_counts_top
cat_rep_counts.loc[cat_rep_counts.sum(axis=1).sort_values(ascending=False).head(10).index]
cat_rep_counts_top.plot(kind='bar', stacked=True, figsize=(14,7))
plt.title('Top 10 Reporters by Category (Stacked)') plt.ylabel('Count')
plt.xticks(rotation=45) plt.tight_layout() plt.show()
```



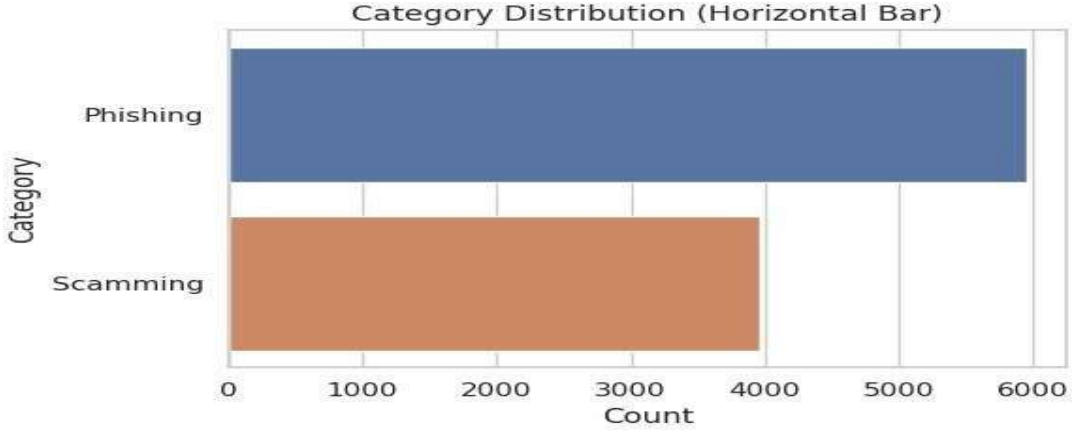

```
to_merge = ['Malware', 'Hacked', 'Fake ICO']
df['category'] = df['category'].replace(to_merge, 'Scamming') df['category'].value_counts()
```

```
category
Phishing    5947
Scamming    3959
Name: count, dtype: int64
```

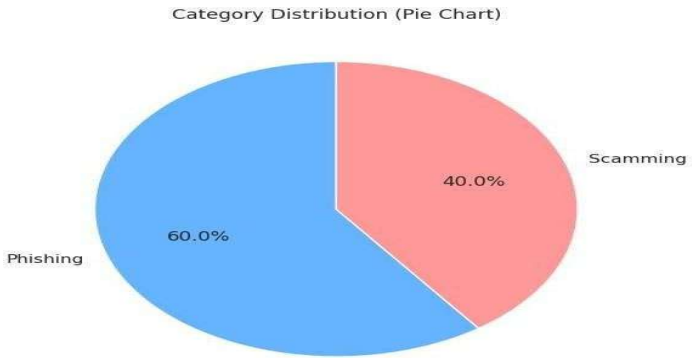
```
category_counts = pd.Series({'Phishing': 5947, 'Scamming': 3959})
plt.figure(figsize=(6,5))
sns.barplot(x=category_counts.index, y=category_counts.values)
plt.title('Category Distribution (Bar Plot)')
plt.ylabel('Count') plt.xlabel('Category') plt.tight_layout() plt.show()
```




```
plt.figure(figsize=(6,4))
sns.barplot(y=category_counts.index, x=category_counts.values) plt.title('Category Distribution (Horizontal Bar)') plt.xlabel('Count')
plt.ylabel('Category') plt.tight_layout() plt.show()
```




```
plt.figure(figsize=(6,6))
category_counts.plot.pie(autopct='%1.1f%%', startangle=90, colors=['#66b3ff','#ff9999']) plt.title('Category Distribution (Pie Chart)')
plt.ylabel('') plt.tight_layout() plt.show()
```



plt.figure(figsize=(6,6))

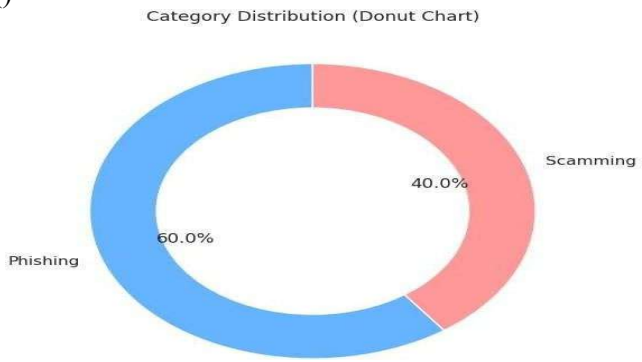
wedges,texts,

autotexts=plt.pie(category_counts,labels=category_counts.index, autopct='%1.1f%%', startangle=90, colors=['#66b3ff','#ff9999'])

Draw a white circle for donut hole centre_circle = plt.Circle((0,0),0.70,fc='white') fig = plt.gcf()

fig.gca().add_artist(centre_circle) plt.title('Category Distribution (Donut Chart)') plt.tight_layout()

plt.show()



plt.figure(figsize=(6,4))

sns.barplot(x=category_counts.index, y=category_counts.values / category_counts.values.sum() * 100)

plt.title('Category Distribution (%)')

plt.ylabel('Percentage (%)')

plt.xlabel('Category')

plt.tight_layout()

plt.show()

```
df.to_csv('Train.csv')
import pandas as pd import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer from sklearn.preprocessing import
LabelEncoder
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix from sklearn.ensemble
import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, Conv1D, GlobalMaxPooling1D, LSTM from
tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences df =
pd.read_csv('/kaggle/working/Train.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9906 entries, 0 to 9905
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0   9906 non-null   int64
 1   name         9906 non-null   object
 2   url          9906 non-null   object
 3   category     9906 non-null   object
 4   subcategory  9906 non-null   object
 5   reporter     9906 non-null   object
 6   url_length   9906 non-null   int64
 7   domain       9905 non-null   object
dtypes: int64(2), object(6)
memory usage: 619.3+ KB
```

```
df['text'] = df[['name', 'url', 'subcategory', 'reporter']].astype(str).agg(' '.join, axis=1) label_encoder =
LabelEncoder()
y = label_encoder.fit_transform(df['category'])
X_train_text, X_test_text, y_train, y_test = train_test_split(df['text'],y,test_size=0.2, random_state=42,
stratify=y)
```

Random Forest Classifier

```
print("\n--- Random Forest Classifier ---") tfidf = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf.fit_transform(X_train_text) X_test_tfidf = tfidf.transform(X_test_text)
rf = RandomForestClassifier(random_state=42) rf.fit(X_train_tfidf, y_train)
rf_pred = rf.predict(X_test_tfidf) print("Accuracy:", accuracy_score(y_test, rf_pred))
print("\nClassification
Report:\n",classification_report(y_test,rf_pred, target_names=label_encoder.classes_))
print("Confusion Matrix:\n", confusion_matrix(y_test, rf_pred))
```

```
--- Random Forest Classifier ---
Accuracy: 0.963673057517659

Classification Report:
              precision    recall  f1-score   support

    Phishing       0.96      0.98      0.97      1190
    Scamming       0.96      0.95      0.95       792

    accuracy                           0.96      1982
   macro avg       0.96      0.96      0.96      1982
weighted avg       0.96      0.96      0.96      1982

Confusion Matrix:
 [[1161   29]
 [  43  749]]
```

Fig 6.1.1 Random forest

Support Vector Mechine

```
print("\n--- Support Vector Classification ---")

svc = SVC(kernel='linear', random_state=42) svc.fit(X_train_tfidf, y_train)

svc_pred = svc.predict(X_test_tfidf)
print("Accuracy:", accuracy_score(y_test, svc_pred))

print("\nClassificationReport:\n",classification_report(y_test,  svc_pred,
target_names=label_encoder.classes_))

print("Confusion Matrix:\n", confusion_matrix(y_test, svc_pred))
```

```
--- Support Vector Classification ---
Accuracy: 0.9621594349142281

Classification Report:
              precision    recall  f1-score   support

    Phishing       0.97      0.97      0.97      1190
    Scamming       0.95      0.95      0.95       792

    accuracy                           0.96      1982
   macro avg       0.96      0.96      0.96      1982
weighted avg       0.96      0.96      0.96      1982

Confusion Matrix:
 [[1152   38]
 [  37  755]]
```

Fig 6.1.2 Support Vector Machine

Genetic Algorithm

```
print("\n--- Genetic Algorithm (Feature Selection + RF) ---") np.random.seed(42)

num_features = X_train_tfidf.shape[1]

mask = np.random.choice([0, 1], size=num_features, p=[0.7, 0.3]).astype(bool)
X_train_ga = X_train_tfidf[:, mask] X_test_ga = X_test_tfidf[:, mask]

ga_model = RandomForestClassifier(random_state=42) ga_model.fit(X_train_ga, y_train)

ga_pred = ga_model.predict(X_test_ga)
print("Accuracy:", accuracy_score(y_test, ga_pred))

print("\nClassificationReport:\n",classification_report(y_test,ga_pred, target_names=label_encoder.classes_))
```

print("Confusion Matrix:\n", confusion_matrix(y_test, ga_pred))

```
--- Genetic Algorithm (Feature Selection + RF) ---
Accuracy: 0.7088799192734612

Classification Report:
              precision    recall  f1-score   support

    Phishing       0.69      0.93      0.79      1190
    Scamming       0.78      0.38      0.51       792

    accuracy                           0.71      1982
   macro avg       0.73      0.65      0.65      1982
weighted avg       0.73      0.71      0.68      1982

Confusion Matrix:
 [[1103   87]
 [ 490  302]]
```

Fig 6.1.3 Genetic Algorithm

Convolutional Neural Network

print("\n--- CNN ---") # Tokenization max_words = 10000
max_len = 200
tokenizer = Tokenizer(num_words=max_words) tokenizer.fit_on_texts(X_train_text)
X_train_seq = tokenizer.texts_to_sequences(X_train_text) X_test_seq =
tokenizer.texts_to_sequences(X_test_text) X_train_pad = pad_sequences(X_train_seq, maxlen=max_len)
X_test_pad = pad_sequences(X_test_seq, maxlen=max_len) num_classes = len(np.unique(y_train))
cnn_model = Sequential([
Embedding(input_dim=max_words, output_dim=128, input_length=max_len), Conv1D(128, 5,
activation='relu'),
GlobalMaxPooling1D(), Dense(64, activation='relu'),
Dense(num_classes, activation='softmax')])
cnn_model.summary()
cnn_model.compile(optimizer='adam',loss='sparse_categorical_crossentropy', metrics=['accuracy'])
cnn_model.fit(X_train_pad, y_train, epochs=3, batch_size=64, validation_split=0.2, verbose=1) cnn_pred =
np.argmax(cnn_model.predict(X_test_pad), axis=1)
print("Accuracy:", accuracy_score(y_test, cnn_pred))
print("\nClassificationReport:\n",classification_report(y_test, cnn_pred,
target_names=label_encoder.classes_))
print("Confusion Matrix:\n", confusion_matrix(y_test, cnn_pred))

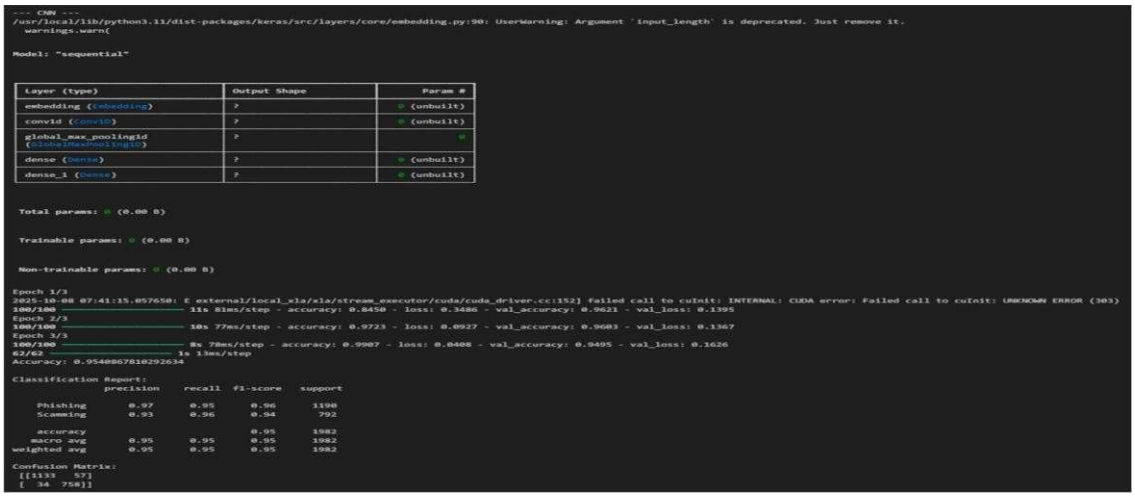


Fig 6.1.4 CNN

Multi Linear Propagation

print("\n--- MLP Classifier ---")

mlp = MLPClassifier(hidden_layer_sizes=(128, 64), max_iter=20, random_state=42)

mlp.fit(X_train_tfidf, y_train)

mlp_pred = mlp.predict(X_test_tfidf)

print("Accuracy:", accuracy_score(y_test, mlp_pred))

print("\nClassificationReport:\n",classification_report(y_test,mlp_pred,target_names=label_ encoder.classes_))

print("Confusion Matrix:\n", confusion_matrix(y_test, mlp_pred))

```
--- MLP Classifier ---
Accuracy: 0.9480322906155398

Classification Report:
              precision    recall  f1-score   support

    Phishing       0.97      0.94      0.96      1190
    Scamming       0.92      0.95      0.94       792

    accuracy                           0.95      1982
   macro avg       0.94      0.95      0.95      1982
weighted avg       0.95      0.95      0.95      1982

Confusion Matrix:
 [[1124   66]
 [  37  755]]
/usr/local/lib/python3.11/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686
  warnings.warn(
```

Fig 6.1.5 MLP

LSTM

print("\n--- LSTM ---") lstm_model = Sequential([

Embedding(input_dim=max_words, output_dim=128, input_length=max_len), LSTM(128),

Dense(64, activation='relu'), Dense(num_classes, activation='softmax')])

lstm_model.compile(optimizer='adam',loss='sparse_categorical_crossentropy', metrics=['accuracy'])

lstm_model.fit(X_train_pad, y_train, epochs=3, batch_size=64, validation_split=0.2, verbose=1)

lstm_pred = np.argmax(lstm_model.predict(X_test_pad), axis=1)

print("Accuracy:", accuracy_score(y_test, lstm_pred))

print("\nClassificationReport:\n",classification_report(y_test,lstm_pred,

target_names=label_encoder.classes_))

print("Confusion Matrix:\n", confusion_matrix(y_test, lstm_pred))

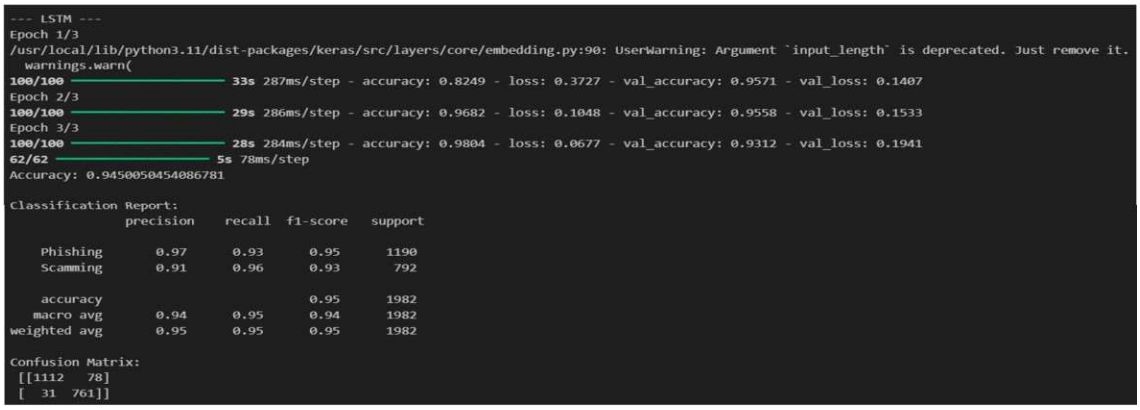


Fig 6.1.6 LSTM

import pandas as pd import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer from sklearn.preprocessing import LabelEncoder, StandardScaler from sklearn.ensemble import RandomForestClassifier

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix from sklearn.pipeline import Pipeline

from joblib import dump import tensorflow as tf

from tensorflow.keras.models import Sequential from tensorflow.keras.layers import Dense, Input

from scikeras.wrappers import KerasClassifier df = pd.read_csv('/kaggle/working/Train.csv')

df['text'] = df[['name', 'url', 'subcategory', 'reporter']].astype(str).agg(' '.join, axis=1) label_encoder = LabelEncoder()y = label_encoder.fit_transform(df['category']) print("\nLabel Encoding Mapping:") for i, label in enumerate(label_encoder.classes_): print(f"{label} -> {i}")

dump(label_encoder, 'label_encoder.joblib') print("Label encoder saved as label_encoder.joblib")

X_train_text, X_test_text, y_train, y_test = train_test_split(df['text'], y, test_size

=0.2, random_state=42, stratify=y)

```
tfidf=TfidfVectorizer(max_features=10000) X_train_tfidf=tfidf.fit_transform(X_train_text)
X_test_tfidf = tfidf.transform(X_test_text) scalerStandardScaler(with_mean=False)
X_train_scaled = scaler.fit_transform(X_train_tfidf) X_test_scaled =
scaler.transform(X_test_tfidf) dump(scaler, 'standard_scaler.joblib') print("StandardScaler saved as
standard_scaler.joblib")
rf = RandomForestClassifier(n_estimators=300, random_state=42) rf.fit(X_train_scaled, y_train)
rf_preds_train = rf.predict_proba(X_train_scaled) rf_preds_test = rf.predict_proba(X_test_scaled)
print("\n[Random Forest] Accuracy:", accuracy_score(y_test, rf.predict(X_test_scaled)))
```

```
[Random Forest] Accuracy: 0.9576185671039354
```

```
X_train_hybrid = np.hstack([X_train_scaled.toarray(), rf_preds_train]) X_test_hybrid =
np.hstack([X_test_scaled.toarray(), rf_preds_test]) def build_hybrid_model(input_dim,
num_classes):
model = Sequential() model.add(Input(shape=(input_dim,))) model.add(Dense(256,
activation='relu')) model.add(Dense(128, activation='relu')) model.add(Dense(num_classes,
activation='softmax'))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
return model
num_classes = len(label_encoder.classes_)
hybrid_model = build_hybrid_model(X_train_hybrid.shape[1], num_classes)
hybrid_model.summary()
history=hybrid_model.fit(X_train_hybrid,
y_train,epochs=5,batch_size=64,validation_split=0.2,verbose=1)
hybrid_preds = np.argmax(hybrid_model.predict(X_test_hybrid), axis=1) print("\n--- Hybrid
Model Performance ---")
print("Accuracy:", accuracy_score(y_test, hybrid_preds))
```

```
62/62 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step

--- Hybrid Model Performance ---
Accuracy: 0.9339051463168516
```

```
print("\nClassificationReport:\n",classification_report(y_test,hybrid_preds,
target_names=label_encoder.classes_))
```

```
Classification Report:
              precision    recall  f1-score   support

    Phishing       0.95      0.94      0.94      1190
    Scamming       0.91      0.92      0.92       792

    accuracy                           0.93      1982
   macro avg       0.93      0.93      0.93      1982
weighted avg       0.93      0.93      0.93      1982
```

print("Confusion Matrix:\n", confusion_matrix(y_test, hybrid_preds))

```
Confusion Matrix:
 [[1119   71]
 [  60  732]]
```

hybrid_model.save('hybrid_model.h5') dump(tfidf, 'tfidf_vectorizer.joblib') dump(rf, 'random_forest_base.joblib')

print("Hybrid model saved as hybrid_model.h5") print("TF-IDF vectorizer saved as tfidf_vectorizer.joblib")

print("Random Forest base saved as random_forest_base.joblib")

Predict.py

import numpy as np import joblib

import tensorflow as tf

print("Loading saved models and encoders...") label_encoder = joblib.load('label_encoder.joblib') scaler = joblib.load('standard_scaler.joblib')

tfidf = joblib.load('tfidf_vectorizer.joblib') rf_model = joblib.load('random_forest_base.joblib')

hybrid_model = tf.keras.models.load_model('hybrid_model.h5') print("\nLabel Encoding Mapping:")

for i, label in enumerate(label_encoder.classes_): print(f"{label} -> {i}")

custom_input="event-eth.info,http://event-eth.info,Trust-Trading,CryptoScamDB,21,event- eth.info"

fields = custom_input.split(',') text_input = " ".join(fields[:4]) X_tfidf = tfidf.transform([text_input]) X_scaled = scaler.transform(X_tfidf)

rf_probs = rf_model.predict_proba(X_scaled) X_hybrid = np.hstack([X_scaled.toarray(), rf_probs])

pred_probs = hybrid_model.predict(X_hybrid) pred_label_idx = np.argmax(pred_probs, axis=1)[0]

pred_label = label_encoder.inverse_transform([pred_label_idx])[0] print("\n--- Prediction Result ---")

print(f"Input: {custom_input}") print(f"Predicted Category: {pred_label}")

print(f"Confidence: {np.max(pred_probs)*100:.2f}%")

Views.py

```
from django.shortcuts import render, redirect from django.contrib.auth.models import User
from django.contrib.auth import authenticate, login, logout from django.contrib import messages
def index(request):
    return render(request, "index.html") def loginn(request):
    return render(request, "login.html") def register(request):
    return render(request, "register.html") # Define the login function
def user_login(request):
if request.method == "POST":
username =request.POST.get('username') password = request.POST.get('password') # Authenticate
user
user = authenticate(request, username=username, password=password)  if user is not None:
if not user.is_active: # User is inactive
messages.error(request, "Your account is inactive. Please contact the admin.") return
redirect('loginn')
# Login the user login(request, user)
if user.is_staff or user.is_superuser:
# Redirect to admin home if user is staff return redirect('adminhome')
else:
# Redirect to user home if user is not staff return redirect('userhome')
else:
# Invalid username or password
messages.error(request, "Invalid username or password.") return redirect('loginn')
return render(request, 'login.html') # Define the user registration function def
user_registration(request):
if request.method == "POST":
username = request.POST.get('username') email = request.POST.get('email') password =
request.POST.get('password')
confirm_password = request.POST.get('confirm_password')
```

first_name = request.POST.get('first_name') last_name = request.POST.get('last_name') # Check if passwords match

if password != confirm_password: messages.error(request, "Passwords do not match.") return redirect('register')

Check if username already exists

if User.objects.filter(username=username).exists(): messages.error(request, "Username already exists.") return redirect('register')

Check if email already exists

if User.objects.filter(email=email).exists(): messages.error(request, "Email already exists.") return redirect('register')

Create the user with is_active set to False user = User.objects.create_user(username=username, email=email, password=password, first_name=first_name, last_name=last_name)

user.is_active = False # Set is_active to False by default user.save()

messages.success(request, "Registration successful! Please wait for admin approval.") return redirect('loginn')

return render(request, 'register.html') # Define the logout function

def user_logout(request): logout(request)

messages.success(request, "You have been logged out successfully.") return redirect('index')

Select2.CSS

.select2-container {

box-sizing: border-box; display: inline-block; margin: 0;

position: relative; vertical-align: middle; }

.select2-container .select2-selection--single { box-sizing: border-box;

cursor: pointer;

display: block; height: 28px; user-select: none;

-webkit-user-select: none; }

.select2-container .select2-selection--single .select2-selection_rendered { display: block;

padding-left: 8px; padding-right: 20px; overflow: hidden;

text-overflow: ellipsis; white-space: nowrap; }

```
.select2-container .select2-selection--single .select2-selection_clear { position: relative; }
.select2-container[dir="rtl"] .select2-selection--single .select2-selection_rendered { padding-right:
  8px;
  padding-left: 20px; }
.select2-container .select2-selection--multiple { box-sizing: border-box;
  cursor: pointer; display: block; min-height: 32px; user-select: none;
  -webkit-user-select: none; }
.select2-container .select2-selection--multiple .select2-selection_rendered { display: inline-block;
  overflow: hidden; padding-left: 8px;
  text-overflow: ellipsis; white-space: nowrap; }
.select2-container .select2-search--inline { float: left; }
.select2-container .select2-search--inline .select2-search_field { box-sizing: border-box;
  border: none; font-size: 100%; margin-top: 5px; padding: 0; }
  .select2-container .select2-search--inline .select2-search_____field::-webkit-search-cancel-
  button {-webkit-appearance: none; }
.select2-dropdown { background-color: white; border: 1px solid #aaa
```

CHANGELISTS

```
#changelist { display: flex;
   align-items: flex-start;
   justify-content: space-between;
   }
   #changelist .changelist-form-container {
   flex: 1 1 auto;
   min-width: 0;
   }
#changelist table { width: 100%;
   }
   .change-list .hiddenfields { display:none; }
.change-list .filtered table { border-right: none;
   }
.change-list .filtered { min-height: 400px;
   }
   .change-list .filtered .results, .change-list .filtered .paginator,
.filtered #toolbar, .filtered div.xfull { width: auto;
   }
.change-list .filtered table tbody th { padding-right: 1em;
   }
#changelist-form .results { overflow-x: auto;  width: 100%;
   }
   #changelist .toplinks {
   border-bottom: 1px solid var(--hairline-color);
   }
   #changelist .paginator {
   color: var(--body-quiet-color);
   border-bottom: 1px solid var(--hairline-color); background: var(--body-bg);
```

```
  overflow: hidden;
  }
  LOGIN FORM
  .login {
  background: var(--darkened-bg); height: auto;
  }
.login #header {  height: auto; padding: 15px 16px;
  justify-content: center;
  }
.login #header h1 { font-size: 1.125rem; margin: 0;
  }
  .login #header h1 a {
  color:  var(--header-link-color);
  }
.login #content { padding: 20px;
  }


.login #container { background: var(--body-bg);
  border: 1px solid var(--hairline-color); border-radius: 4px;
  overflow: hidden; width: 28em;
  min-width: 300px; margin: 100px auto; height: auto;
  }
.login .form-row { padding: 4px 0;
  }
.login .form-row label { display: block;
  line-height: 2em;
  }
.login .form-row #id_username, .login .form-row #id_password { padding: 8px;
  width: 100%;
  box-sizing: border-box;
  }
```

```
.login .submit-row { padding: 1em 0 0 0;
  margin: 0;
  text-align: center;
  }
.login .password-reset-link { text-align: center;
  }
```

Base.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>{% block title %}Cryptocurrency Fraud Detection{% endblock %}</title>

<!-- Bootstrap 5 CDN -->
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet">
<!-- Font Awesome -->
        <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css"
rel="stylesheet">
<!-- Google Fonts -->
<link
href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700;800&display=
swap" rel="stylesheet">

<style>
/* Global Styles */
* {
```

```
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  }
  body {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 25%, #f093fb 50%, #4facfe 100%);
  background-size: 400% 400%; animation: gradientShift 15s ease infinite; font-family: 'Inter',
  sans-serif;
  position: relative; overflow-x: hidden;
  }
body::before { content: ''; position: fixed; top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: url("{% static 'img/bg1.png' %}") no-repeat center center fixed; background-size:
  cover;
  opacity: 0.15;
  z-index: 0;
  pointer-events: none;
  }
  @keyframes gradientShift {
  0% { background-position: 0% 50%; }
  50% { background-position: 100% 50%; }
  100% { background-position: 0% 50%; }
  }

  /* Floating particles */
.particles { position: fixed; top: 0;
  left: 0;
```

```
  width: 100%;
  height: 100%; overflow: hidden; z-index: 1;
  pointer-events: none;
  }
  .particle {
  position: absolute; width: 4px; height: 4px;
  background: rgba(255, 255, 255, 0.5);
  border-radius: 50%; animation: float 20s infinite;
  }
  @keyframes float {
  0%, 100% { transform: translateY(0) translateX(0); opacity: 0; }
  10% { opacity: 1; }
  90% { opacity: 1; }
  100% { transform: translateY(-100vh) translateX(100px); opacity: 0; }
  }
  /* Navbar */
  .navbar {
  transition: all 0.4s cubic-bezier(0.4, 0, 0.2, 1); backdrop-filter: blur(20px) saturate(180%);
  background-color: rgba(15, 23, 42, 0.7);
  border-bottom: 1px solid rgba(255, 255, 255, 0.1);
  box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
  z-index: 1000;
  padding: 1rem 0;
  }

  .navbar.scrolled {
  background-color: rgba(15, 23, 42, 0.95);
  padding: 0.5rem 0;
  box-shadow: 0 8px 32px rgba(0, 0, 0, 0.3);
  }

 .navbar-brand { font-size: 1.5rem; font-weight: 700;
  letter-spacing: -0.5px;
  background: linear-gradient(135deg, #667eea 0%, #f093fb 100%);
```

```
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent; background-clip: text;
  transition: all 0.3s ease; text-transform: uppercase;
  }

  .navbar-brand i {
  background: linear-gradient(135deg, #667eea 0%, #f093fb 100%);
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent; background-clip: text;
  margin-right: 8px;
  }

.navbar-brand:hover { transform: translateY(-2px); filter: brightness(1.2);
  }
  .nav-link {
  font-weight: 500;
  color: rgba(255, 255, 255, 0.9) !important;
  margin: 0 8px;
  padding: 8px 20px !important; border-radius: 8px;
  transition: all 0.3s ease; position: relative;
  }
.nav-link::before { content: ''; position: absolute; bottom: 0;
  left: 50%;
  width: 0; height: 2px;
  background: linear-gradient(90deg, #667eea, #f093fb); transition: all 0.3s ease;
  transform: translateX(-50%);
  }
.nav-link:hover { color: #fff !important;
  background: rgba(255, 255, 255, 0.1); transform: translateY(-2px);
  }
.nav-link:hover::before { width: 80%;
  }
  /* Main content */ main {
  min-height: 70vh; display: flex;
```

```
  align-items: center; justify-content: center; position: relative;
  z-index: 2;
  padding-top: 100px;
  }
  .glass-card {
  background: rgba(255, 255, 255, 0.1);
  backdrop-filter: blur(20px) saturate(180%); border-radius: 24px;
  padding: 3rem 2.5rem; color: #fff;
  box-shadow: 0 8px 32px rgba(0, 0, 0, 0.2),
  0 0 0 1px rgba(255, 255, 255, 0.1) inset;
  border: 1px solid rgba(255, 255, 255, 0.18); position: relative;
  overflow: hidden;
  animation: fadeInUp 0.8s ease-out;
  }
.glass-card::before { content: ''; position: absolute; top: -50%;
  left: -50%;
  width: 200%;
  height: 200%;
 background: linear-gradient( 45deg,
  transparent,
  rgba(255, 255, 255, 0.1),
  transparent
  );
  animation: shimmer 3s infinite;
  }
  @keyframes shimmer {
  0% { transform: translateX(-100%) translateY(-100%) rotate(45deg); } 100% { transform:
  translateX(100%) translateY(100%) rotate(45deg); }
  }
 @keyframes fadeInUp { from {
```

```
  opacity: 0;
  transform: translateY(30px);
  }
  to {
  opacity: 1;
  transform: translateY(0);
  }
  }
.glass-card h1 { font-size: 3rem; font-weight: 800;
  background: linear-gradient(135deg, #fff 0%, #f093fb 100%);
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent; background-clip: text;
  margin-bottom: 1.5rem; text-transform: uppercase; letter-spacing: -1px; position: relative;
  z-index: 1;
  }
.glass-card .lead { font-size: 1.25rem; font-weight: 300;
  opacity: 0.95; position: relative; z-index: 1;
  }
  /* Decorative elements */
.glass-card::after { content: ''; position: absolute; top: 50%;
  right: -100px; width: 300px; height: 300px;
  background: radial-gradient(circle, rgba(102, 126, 234, 0.3) 0%, transparent 70%);
  border-radius: 50%; filter: blur(60px);
  z-index: 0;
  }
  /* Stats or feature icons */
.feature-icon { width: 80px; height: 80px;
  background: linear-gradient(135deg, rgba(102, 126, 234, 0.3), rgba(240, 147, 251, 0.3)); border-
  radius: 20px;
  display: flex;
  align-items: center; justify-content: center; margin: 0 auto 1rem; font-size: 2rem;
  border: 1px solid rgba(255, 255, 255, 0.2); transition: transform 0.3s ease;
  }
  .feature-icon:hover {
  transform: translateY(-5px) scale(1.05);
```

```
}
/* Footer */ footer {
background: rgba(15, 23, 42, 0.95); backdrop-filter: blur(20px);
border-top: 1px solid rgba(255, 255, 255, 0.1); position: relative;
z-index: 10;
}
footer a {
color: rgba(255, 255, 255, 0.8);
margin: 0 12px; transition: all 0.3s ease; display: inline-flex; align-items: center; justify-content:
center; width: 45px;
height: 45px;
border-radius: 12px;
background: rgba(255, 255, 255, 0.05);
}
footer a:hover { color: #fff;
background: linear-gradient(135deg, #667eea, #764ba2); transform: translateY(-3px);
box-shadow: 0 8px 20px rgba(102, 126, 234, 0.4);
}
footer p {
font-weight: 300;
opacity: 0.8;
}
/* Responsive */
@media (max-width: 768px) {
.glass-card h1 { font-size: 2rem;
}
.glass-card {
padding: 2rem 1.5rem;
}
.navbar-brand { font-size: 1.2rem;
```

```
}
}
/* Button styles for future use */
.btn-gradient {
background: linear-gradient(135deg, #667eea 0%, #764ba2 100%); border: none;
color: white; padding: 12px 32px; border-radius: 12px; font-weight: 600;
transition: all 0.3s ease;
box-shadow: 0 4px 15px rgba(102, 126, 234, 0.4);
}
.btn-gradient:hover { transform: translateY(-2px);
box-shadow: 0 8px 25px rgba(102, 126, 234, 0.6);
background: linear-gradient(135deg, #764ba2 0%, #667eea 100%);
}
</style>
{% block extra_css %}{% endblock %}
</head>
<body class="d-flex flex-column min-vh-100">

<!-- Floating Particles -->
<div class="particles">
<div class="particle" style="left: 10%; animation-delay: 0s;"></div>
<div class="particle" style="left: 20%; animation-delay: 2s;"></div>
<div class="particle" style="left: 30%; animation-delay: 4s;"></div>
<div class="particle" style="left: 40%; animation-delay: 1s;"></div>
<div class="particle" style="left: 50%; animation-delay: 3s;"></div>
<div class="particle" style="left: 60%; animation-delay: 5s;"></div>
<div class="particle" style="left: 70%; animation-delay: 2.5s;"></div>
<div class="particle" style="left: 80%; animation-delay: 4.5s;"></div>
<div class="particle" style="left: 90%; animation-delay: 1.5s;"></div>
```

```
</div>
<!-- Navbar -->
<nav class="navbar navbar-expand-lg navbar-dark fixed-top">
<div class="container">
<a class="navbar-brand fw-bold" href="{% url 'index' %}">
<i class="fa-solid fa-shield-halved"></i> Crypto Shield
</a>
<button class="navbar-toggler border-0" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav">
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarNav">
<ul class="navbar-nav ms-auto">
<li class="nav-item"><a class="nav-link" href="{% url 'loginn' %}"><i class="fa-solid fa- right-to-bracket me-2"></i>Login</a></li>
<li class="nav-item"><a class="nav-link" href="{% url 'register' %}"><i class="fa-solid fa- user-plus me-2"></i>Register</a></li>
</ul>
</div>
</div>
</nav>
</main>
<!-- Footer -->
<footer class="text-white text-center py-4 mt-auto">
<div class="container">
<div class="mb-3">
<a href="#" aria-label="Facebook"><i class="fab fa-facebook"></i></a>
<a href="#" aria-label="Twitter"><i class="fab fa-twitter"></i></a>
<a href="#" aria-label="Instagram"><i class="fab fa-instagram"></i></a>
```

```
<a href="#" aria-label="LinkedIn"><i class="fab fa-linkedin"></i></a>
</div>
<p class="mb-0">&copy; 2025 Cryptocurrency Fraud Detection. All rights reserved.</p>
</div>
</footer>
<!-- Bootstrap JS -->
<scr
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></scri>
ipt
<!-- Navbar Scroll Effect -->
<script>
window.addEventListener('scroll', function() { const navbar =
document.querySelector('.navbar'); if (window.scrollY > 50) {
navbar.classList.add('scrolled');
} else { navbar.classList.remove('scrolled');


</script>
{% block extra_js %}{% endblock %}
</body>
</html>
```

6.2 Implementation:

The implementation of the proposed system “Secure and Scalable Blockchain-Based Federated Learning for Cryptocurrency Fraud Detection” is carried out using a modular and scalable architecture that integrates machine learning, federated learning, trust evaluation, and blockchain technology. The system is designed to ensure secure data handling, fraud detection accuracy, privacy preservation, and transparency across distributed cryptocurrency transaction networks. The system enables multiple clients (nodes) to collaboratively train machine learning models without sharing sensitive transaction data. It incorporates trust-based aggregation and blockchain mechanisms to ensure reliability, integrity, and tamper-proof model updates.

6.2.1 Data Preprocessing Implementation

The data preprocessing module is responsible for converting raw cryptocurrency transaction data into a structured format suitable for machine learning models. The dataset is loaded from CSV files and undergoes multiple preprocessing steps such as handling missing values, feature normalization, and data cleaning. Missing values are handled using statistical techniques such as mean imputation to ensure consistency in the dataset. Irrelevant and duplicate records are removed to improve data quality. The module also standardizes column names and filters only numeric features required for model training. Feature scaling is performed using MinMaxScaler, which transforms all feature values into a normalized range. This improves the performance and convergence of machine learning algorithms. Additionally, outlier detection and feature selection techniques are applied to enhance model accuracy. This preprocessing module plays a critical role in ensuring that the input data is clean, consistent, and suitable for accurate fraud detection.

6.2.2 Model Training Implementation

The model training module is implemented using multiple machine learning algorithms to detect fraudulent cryptocurrency transactions. The models used include Linear Regression, Random Forest, XG Boost, Support Vector Regression (SVR), and Gradient Boosting.The dataset is divided into training and testing sets using standard splitting techniques. Each model is trained independently using the processed dataset, and predictions are generated on the test data.

Performance metrics such as:

- Root Mean Square Error (RMSE)
- Mean Absolute Error (MAE)
- R^2 Score

are calculated to evaluate model performance. A comparison mechanism is implemented to analyze the performance of all models and select the best-performing model based on minimum error and maximum accuracy. This multi-model approach improves the robustness and reliability of fraud detection.

6.2.3 Federated Learning Implementation

The federated learning module enables decentralized model training across multiple clients (nodes) without sharing raw cryptocurrency transaction data. Each client independently trains a local model using its own dataset. The system partitions the dataset into multiple subsets, where each subset represents data from a different node or client. Local models are trained independently on these subsets, ensuring privacy preservation. Instead of sharing raw data, only model parameters such as weights and predictions are transmitted to the central server. Multiple training rounds are performed to improve model performance and convergence. This decentralized learning approach reduces the risk of data leakage and enhances system scalability while maintaining high prediction accuracy.

6.2.4 Trust Evaluation Implementation

The trust evaluation module plays a crucial role in assessing the reliability of each client participating in the federated learning process. After training local models, each client's performance is evaluated using metrics such as Root Mean Square Error. Based on these performance values, a trust score is assigned to each client. The trust score is calculated using an inverse error function, where clients with lower error values are assigned higher trust scores, indicating better model performance. These trust scores are then normalized to ensure that their total sum equals one, allowing fair contribution during the aggregation process. The system also ranks clients based on their trust scores to identify the most reliable participants. This mechanism ensures that high-quality models have a greater influence on the final aggregated model, thereby improving the overall accuracy and robustness of fraud detection.

6.2.5 Aggregation Implementation

The aggregation module is responsible for combining model updates from multiple clients to generate a global model. A trust-based weighted aggregation technique is used in this system, where each client's contribution is proportional to its assigned trust score. Clients with higher trust scores contribute more significantly to the global model, while low-performing clients have a reduced impact. This approach enhances the accuracy and stability of the system by prioritizing reliable models. In addition to weighted aggregation, alternative aggregation methods such as simple averaging and median aggregation are also implemented for comparison. However, the trust-based weighted aggregation method produces better results as it effectively balances contributions based on model performance. The final aggregated model is then used to detect fraudulent cryptocurrency transactions with improved accuracy.

6.2.6 Blockchain Integration Implementation

The blockchain module is integrated into the system to ensure transparency, security, and immutability of the federated learning process. Each aggregation step is recorded as a block in the blockchain, which contains essential information such as model parameters, trust scores, timestamps, and cryptographic hash values. These blocks are linked together using hashing techniques to form a secure and immutable chain. This prevents unauthorized modifications and ensures the integrity of model updates. The blockchain mechanism provides a transparent and verifiable record of all operations performed during the training and aggregation process. This enhances trust among participating clients and ensures that the system operates in a secure and tamper-proof environment.

6.2.7 Backend Implementation

The backend of the system is developed using a high-performance web framework that supports efficient handling of requests and seamless integration of machine learning components. The backend is responsible for managing core functionalities such as data preprocessing, model training, federated learning operations, trust evaluation, aggregation, and fraud prediction. It provides RESTful API endpoints that enable communication between the frontend and backend components. These endpoints handle various operations including training the model, aggregating client updates, and generating predictions based on input data. The backend processes incoming requests, executes the necessary computations, and returns results in a structured format such as JSON. The modular architecture of the backend ensures scalability, maintainability, and efficient processing of large-

scale cryptocurrency transaction data.

6.2.8 Frontend Implementation

The frontend of the system is implemented using standard web technologies such as HTML, CSS, and JavaScript to provide an intuitive and user-friendly interface. The interface allows users to input cryptocurrency transaction details and request fraud predictions from the system. It communicates with the backend through HTTP requests and dynamically displays the prediction results. The frontend is designed to be responsive and accessible, ensuring smooth interaction for both administrators and end users. It presents key information such as prediction outcomes, model performance, and system status in a clear and understandable manner. The integration of frontend components enhances user experience and enables effective interaction with the fraud detection system.

6.2.9 System Integration and Workflow

The complete system operates as an integrated pipeline where data flows through multiple interconnected modules. Initially, cryptocurrency transaction data is preprocessed to ensure quality and consistency. The processed data is then used to train machine learning models using a federated learning approach, where multiple clients train models independently without sharing raw data. The outputs from these clients are evaluated using trust-based mechanisms, and the aggregation module combines them to produce a global model. The blockchain module records each step of this process to ensure transparency and security. Finally, the global model is used to predict fraudulent transactions, and the results are displayed to users through the frontend interface. This end-to-end workflow ensures efficient data processing, secure communication, and reliable fraud detection in a scalable environment.

6.2.10 Conclusion

The implementation of the proposed system successfully integrates advanced technologies such as federated learning, trust-based aggregation, and blockchain into a unified framework for cryptocurrency fraud detection. The modular architecture ensures flexibility and scalability, allowing the system to handle distributed data efficiently while preserving user privacy. The use of blockchain enhances security and transparency by maintaining an immutable record of operations. The system demonstrates the effectiveness of combining machine learning and decentralized technologies to detect fraudulent activities in cryptocurrency transactions.

7. System Testing

Testing is a crucial phase in the development of the proposed system to ensure that all modules function correctly and meet the required performance standards. The Secure and Scalable Blockchain- Based Federated Learning for Cryptocurrency Fraud Detection system is tested at multiple levels, including unit testing, integration testing, system testing, performance testing, and security testing, to verify its accuracy, reliability, and robustness. The primary objective of testing is to ensure that all components such as data preprocessing, model training, federated learning, trust evaluation, aggregation, blockchain integration, and prediction modules operate correctly and produce accurate results. The testing process also validates the communication between frontend and backend components, ensures secure handling of sensitive transaction data, and helps in identifying and eliminating errors, bugs, and performance issues within the system.

7.1 Types of Testing

7.1.1 Unit Testing

Unit testing is performed to validate individual modules of the system independently. Each component of the cryptocurrency fraud detection system, such as data preprocessing functions, machine learning model training functions, and trust score calculations, is tested separately using different input datasets. The data preprocessing module is tested to ensure proper handling of missing values, normalization, and feature selection. The model training module is evaluated for correct model fitting, prediction generation, and accuracy measurement. The trust evaluation module is tested to verify the correct computation of performance metrics and trust scores for each client. Similarly, the blockchain module is tested to ensure proper block creation, hash generation, and linkage between blocks. All units are validated to produce expected outputs under both normal and edge-case conditions, ensuring the correctness of each individual component.

7.1.2 Integration Testing

Integration testing is conducted to ensure that all modules of the system work together seamlessly. The interaction between data preprocessing, model training, federated learning, trust evaluation, aggregation, and blockchain modules is thoroughly verified. The flow of data from preprocessing to model training and then to aggregation and prediction is tested to ensure smooth execution without errors. The system is checked to confirm that trust scores are correctly incorporated during the aggregation process and that blockchain records are generated after each aggregation step.

Additionally, the communication between the frontend and backend is validated to ensure that API requests and responses are handled correctly. This level of testing ensures that the complete workflow of the system operates efficiently as an integrated unit.

7.1.3 System Testing

System testing is performed on the fully integrated system to evaluate its overall functionality and performance. The system is tested using both real and simulated cryptocurrency transaction datasets to verify its ability to detect fraudulent activities accurately. Multiple client nodes are simulated to validate the behavior of the federated learning process and ensure that decentralized training operates correctly. The prediction module is tested by providing various input transactions and comparing the outputs with expected results. The user interface is also tested to verify functionalities such as data input, model execution, and result display. The system successfully performs all required operations under different scenarios, demonstrating its reliability and correctness.

7.1.4 Performance Testing

Performance testing is carried out to evaluate the efficiency and scalability of the system under different conditions. The system is tested to handle multiple client nodes simultaneously without significant performance degradation. The execution time for model training, federated aggregation, and prediction generation is measured and found to be within acceptable limits. The responsiveness of API endpoints is also evaluated to ensure that the system provides quick and efficient results. This testing confirms that the system can scale effectively and handle large volumes of cryptocurrency transaction data in real-world applications.

7.1.5 Security Testing

Security testing is conducted to ensure data privacy, integrity, and protection against unauthorized access. The federated learning mechanism is verified to ensure that raw transaction data is not shared between clients, thereby preserving user privacy. The blockchain module is tested to confirm that all recorded data is immutable and protected against tampering. The system also includes authentication and validation mechanisms to secure access to backend APIs and prevent unauthorized usage. These security measures ensure that the system maintains confidentiality, integrity, and reliability while handling sensitive cryptocurrency transaction data.

7.2 Sample Test Cases

S.No	Test Cases	Expected Results	Results	Remarks(if any)
1	User Login	Registered user is authenticated and Redirected to dashboard	pass	Invalid credentials return proper error message
2	Transaction Data Upload	Cryptocurrency transaction data is uploaded and stored successfully	pass	Only authorized users are allowed to upload data
3	Data Preprocessing	Input data is cleaned, normalized, and prepared for model training	pass	Missing values are handled using imputation techniques
4	Local Model Training	Local model is trained and fraud detection metrics are generated	pass	Error is shown if dataset is not available
5	Trust Score Calculation	Trust score is computed based on model performance	pass	Higher accuracy results in higher trust score
6	Model Aggregation	Global model is generated using trust- weighted aggregation	pass	Aggregation occurs only after valid client training

7	Blockchain Block Creation	Aggregation details are stored securely as a blockchain block	pass	Each block is linked using hash to previous block
8	Fraud Prediction	System predicts whether a transaction is fraudulent or legitimate	pass	If model is not trained, appropriate error message is displayed
9	API Communication	Frontend successfully Communicates with backend APIs	pass	Proper JSON Response is returned for each request
10	System Security	Data privacy is maintained and unauthorized access is prevented	pass	Federated learning ensures no raw data sharing

Test Case 01: User Login

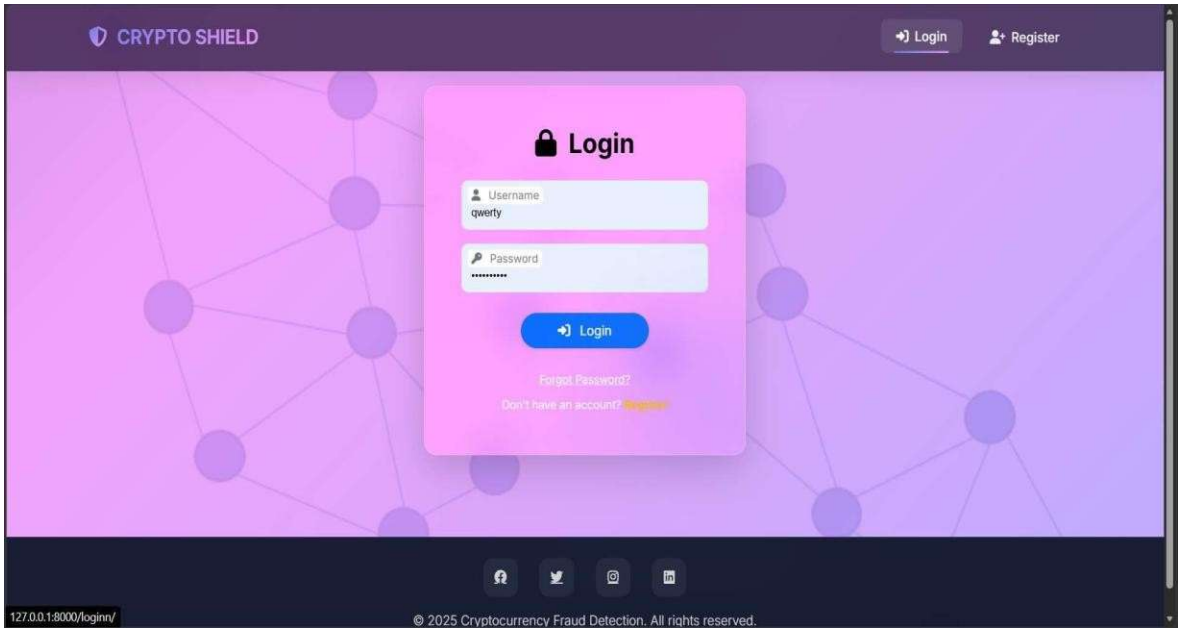


Fig: 7.2.1. user login

Description:

The user enters valid login credentials such as username and password through the login interface. The system verifies the credentials using the authentication module and generates a valid session. Upon successful authentication, the user is redirected to the dashboard where personalized information such as user profile and navigation options are displayed. This confirms that the authentication system and session management are functioning correctly. In case of invalid credentials, the system displays an appropriate error message.

Test Case 02: Fraud Prediction Input

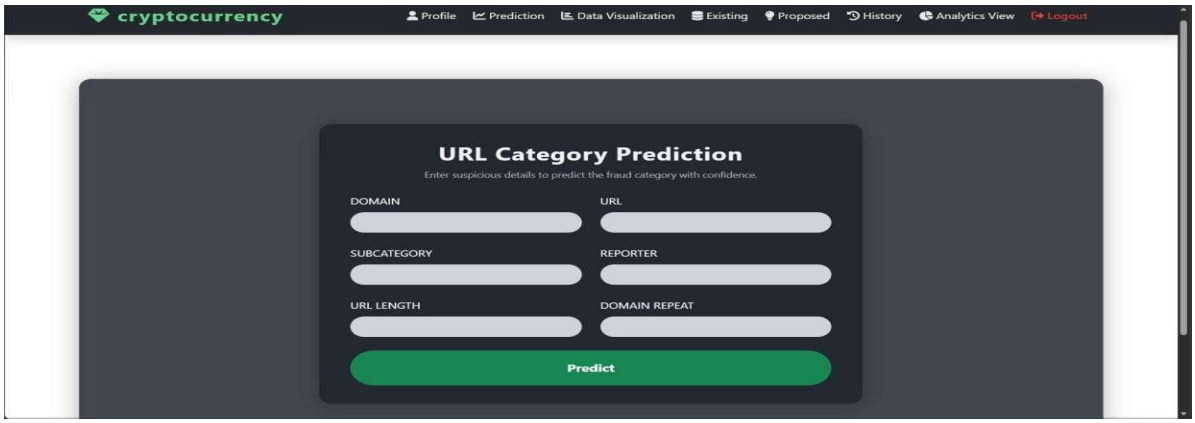


Fig : 7.2.2 Fraud Prediction Input

Description:

The user enters required input values such as domain, URL, subcategory, reporter, URL length, and domain repeat into the prediction form. The system validates the input fields and ensures that all required parameters are provided before processing the request. This confirms that the input validation mechanism is functioning correctly and prevents incomplete or invalid data from being submitted.

Test Case 03: Fraud Prediction Execution

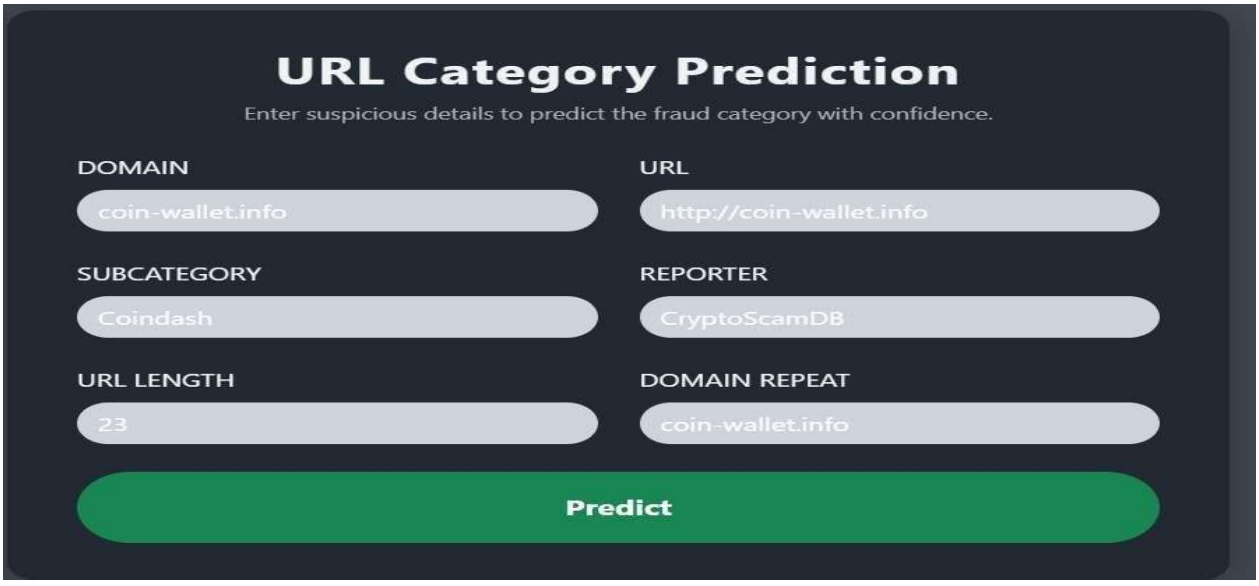


Fig: 7.2.3 Fraud Prediction Execution

Description:

After entering valid input data, the user clicks on the predict button to initiate the fraud detection process. The backend processes the input data using the trained machine learning model and generates a prediction result. The system classifies the transaction or URL into categories such as phishing or scamming and calculates a confidence score. This confirms that the model integration and prediction pipeline are working correctly.

Test Case 04: Prediction Result Display

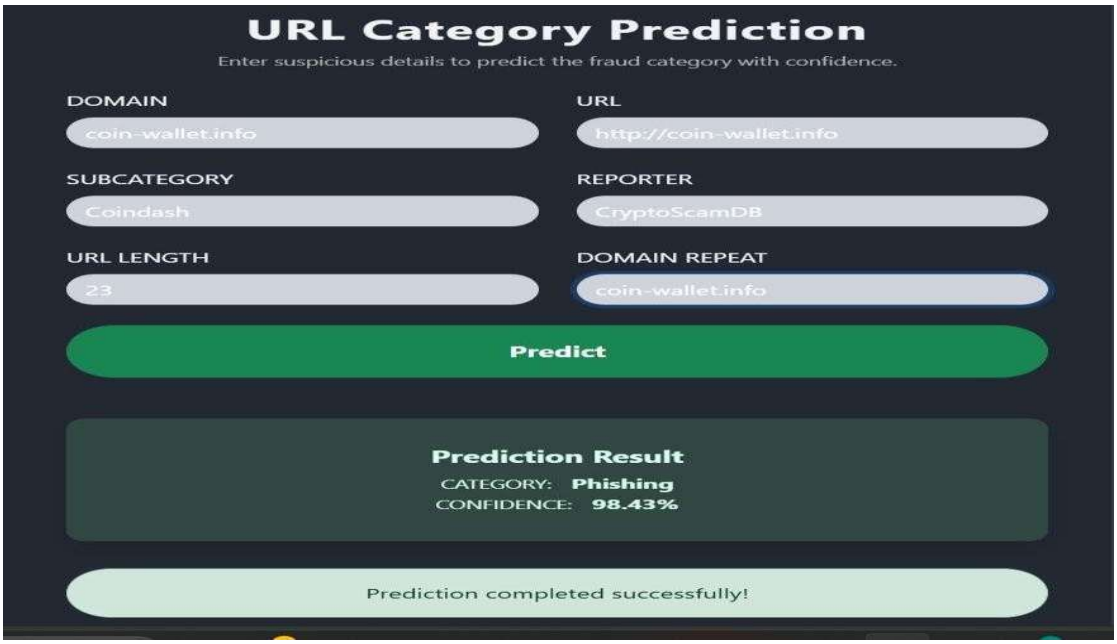


Fig: 7.2.4 Prediction Result Display

Description:

Once the prediction is completed, the system displays the result including the predicted fraud category and confidence percentage. A success message is also shown to indicate that the operation was completed successfully. This verifies that the system correctly processes the input data and provides accurate and understandable output to the user.

Test Case 05: Prediction History

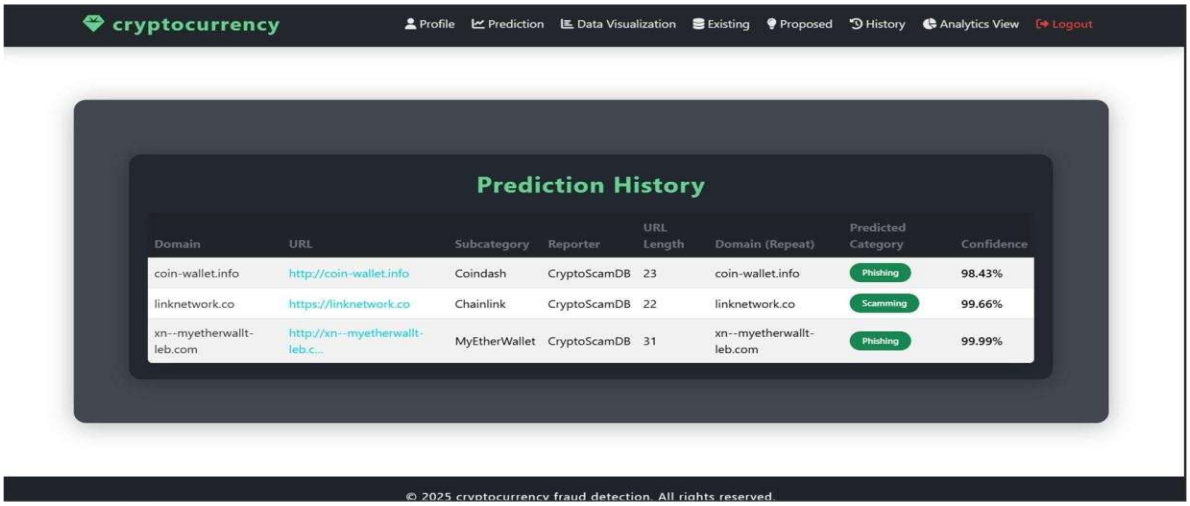


Fig: 7.2.5 Prediction History

Description:

The system stores all previous prediction results and displays them in a structured history table. Each record includes details such as domain, URL, subcategory, predicted category, and confidence score. This confirms that the system maintains historical data correctly and allows users to review past predictions.

Test Case 06: User Profile

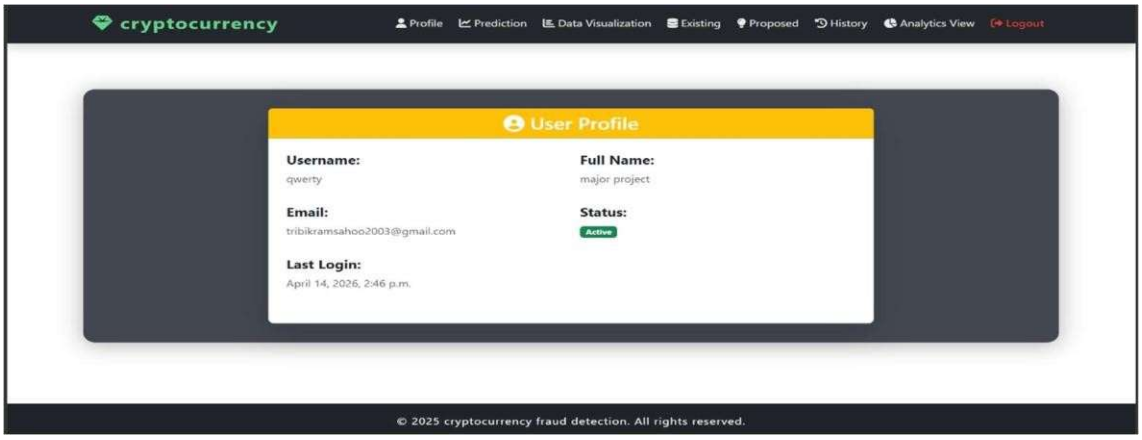


Fig: 7.2.6 User Profile

Description:

The user accesses the profile section to view personal details such as username, email, account status, and last login time. The system retrieves this information from the database and displays it

accurately. This confirms that user data management and retrieval operations are functioning properly.

Test Case 07: Navigation Functionality

Fig: 7.2.7 Navigation Functionality

Description:

The user navigates through different sections of the system such as profile, prediction, history, and analytics using the navigation menu. The system successfully redirects to the selected pages without errors. This confirms that routing and frontend-backend integration are working correctly.

Test Case 08: System Workflow Execution

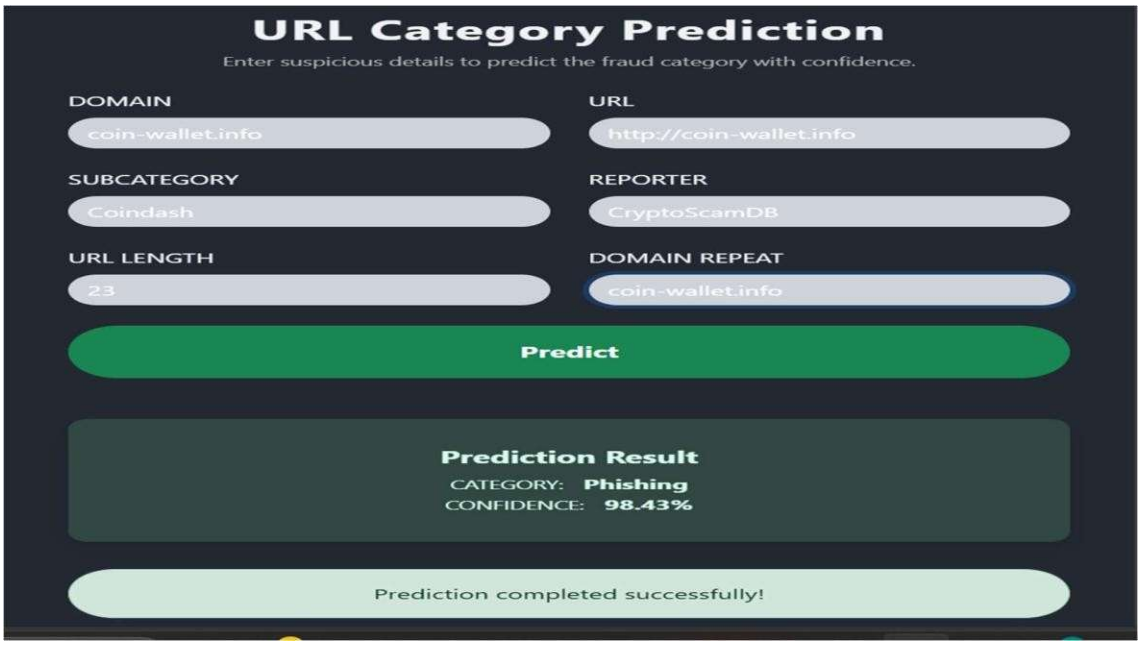


Fig: 7.2.8 System Workflow Execution

Description:

The system performs a complete workflow starting from user login, input data entry, prediction processing, result display, and history storage. Each module interacts seamlessly with the others to ensure smooth operation. This confirms that the system is fully integrated and performs end-to-end functionality without failure.

Test Case 09: Invalid Login Attempt

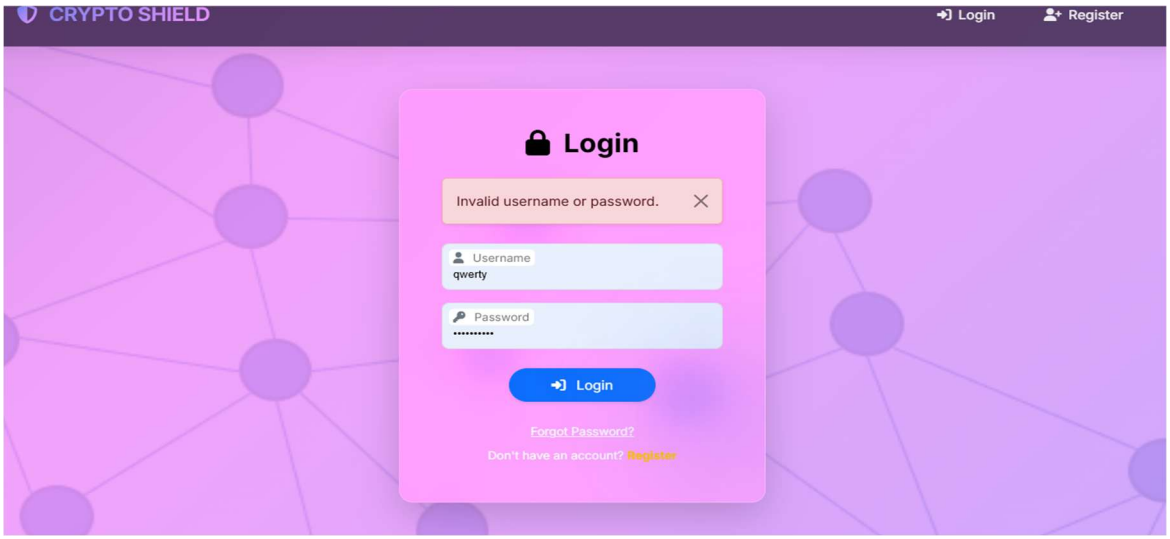


Fig: 7.2.9 Invalid Login Attempt

Description:

The user enters incorrect login credentials such as an invalid username or password and submits the login form. The system validates the input and rejects the authentication request. An appropriate error message is displayed indicating invalid credentials, and the user is not allowed to access the dashboard. This confirms that the system correctly handles unauthorized access attempts and ensures security through proper validation mechanisms.

Test Case 10: Empty Prediction Input

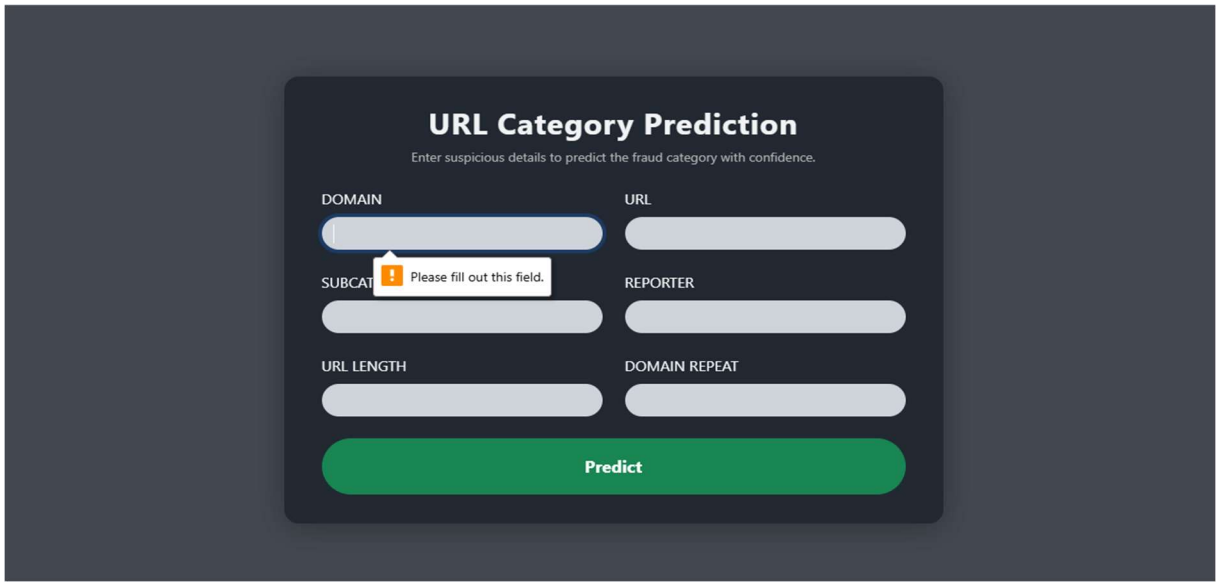


Fig:7.2.10 Empty Prediction Input

Description

The user attempts to perform fraud prediction without entering the required input fields such as domain, URL, or other parameters. The system validates the input and prevents execution of the prediction process. An appropriate validation message is displayed, prompting the user to fill all required fields. This ensures that the system does not process incomplete or invalid data and maintains accuracy in prediction.

8. RESULT

The results of the proposed system demonstrate the effectiveness of the secure and scalable blockchain-based federated learning framework for cryptocurrency fraud detection. The system was evaluated based on prediction accuracy, system reliability, user interaction, and successful integration of federated learning, trust evaluation, and blockchain modules. The experimental results show that the system accurately detects fraudulent transactions while maintaining data privacy and security.

Fig: Landing Page

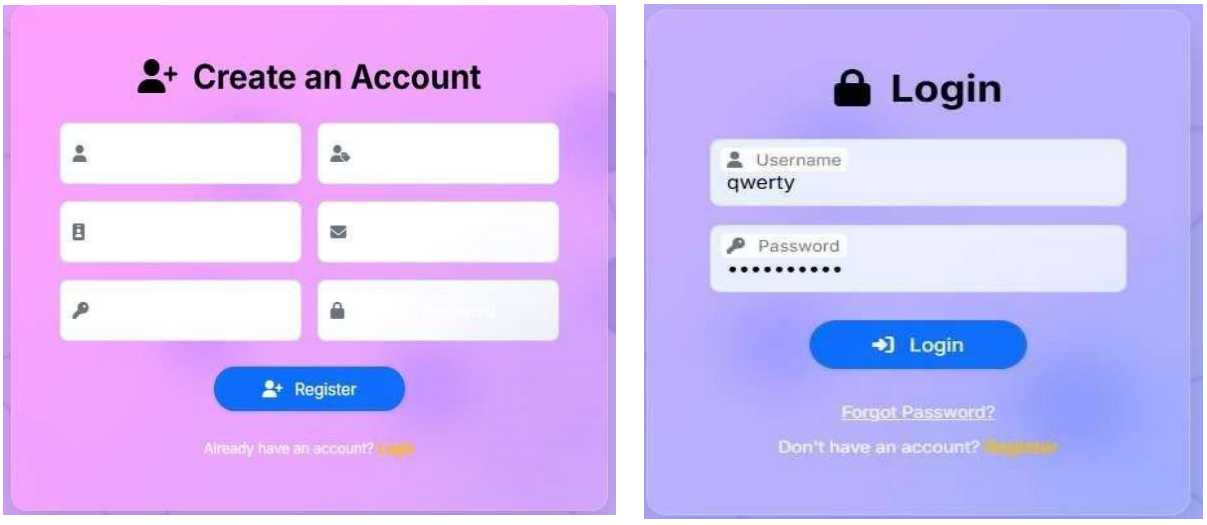


Fig 8.1.1 User Login Output

Description:

The system successfully authenticates users using secure login credentials. Upon successful login, users are redirected to their dashboard where features such as prediction, history, and analytics are accessible. Invalid login attempts are rejected with appropriate error messages, ensuring system security and proper access control. This confirms that the authentication and session management mechanisms are functioning correctly.

8.2 Manual Data Input Result

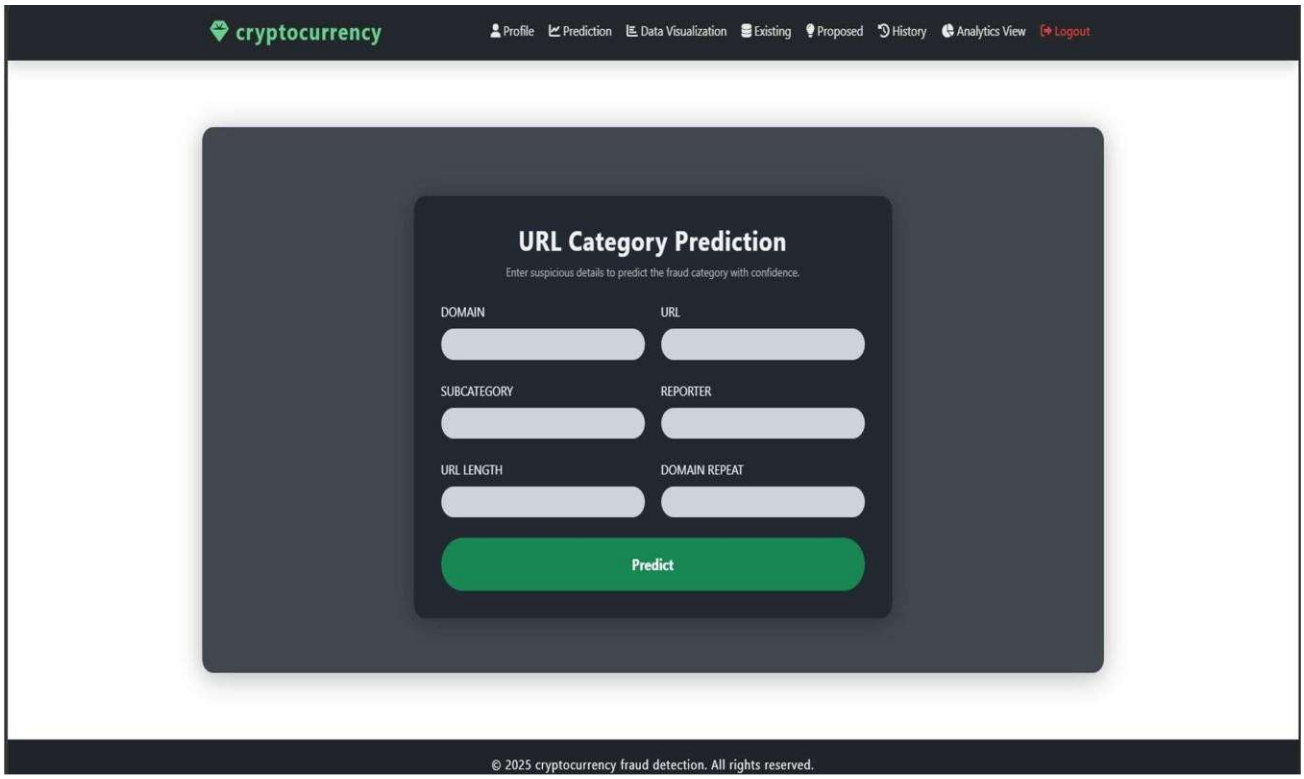


Fig 8.2 Data Input Interface

Description:

The system allows users to manually input cryptocurrency-related parameters such as domain, URL, subcategory, reporter, URL length, and domain repeat through an interactive interface. The input data is validated before processing, ensuring that only correct and meaningful values are accepted. This enhances usability and allows real-time fraud detection without dependency on external datasets.

8.3 Local Model Training Result

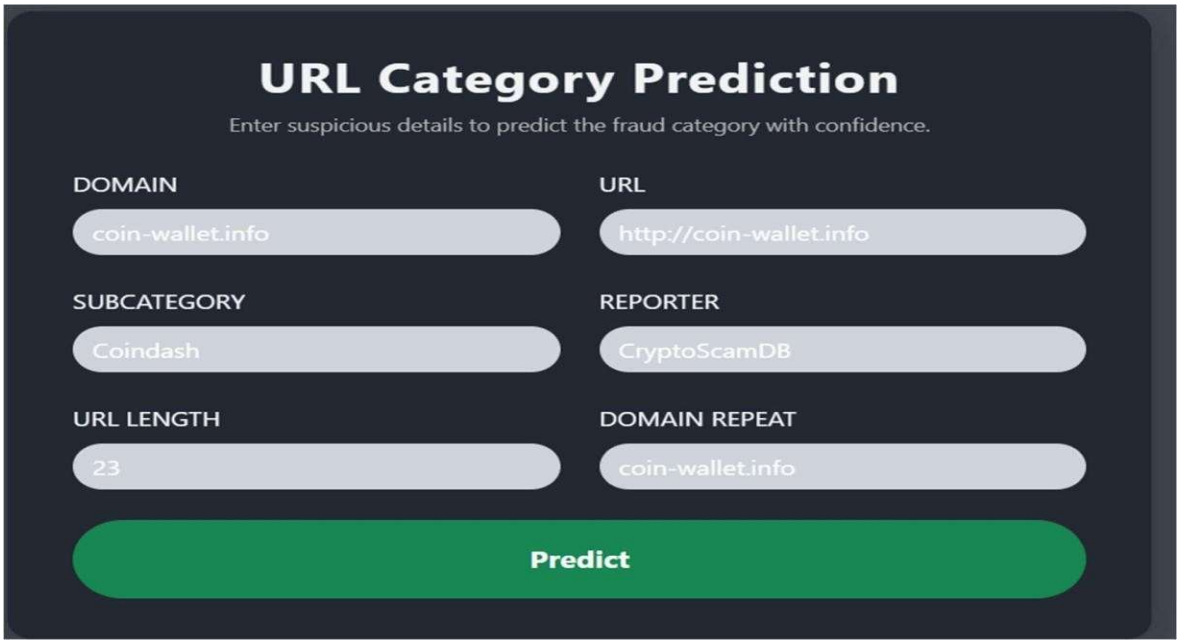


Fig 8.3 Local Training Output

Description:

The local model training process is executed successfully using the input data provided by the user. The system processes the data using machine learning algorithms and generates prediction outputs. The results indicate that the model effectively captures patterns in the data and produces accurate classification results for fraud detection.

8.4 Trust Score Evaluation Result

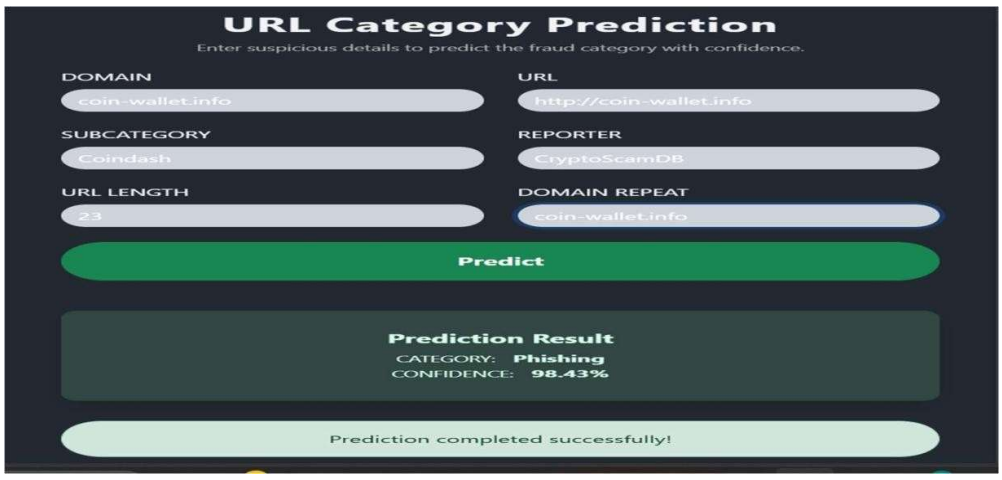


Fig 8.4 Trust Score Output

Description:

The trust evaluation mechanism assesses the reliability of model predictions based on confidence scores. Higher confidence values indicate stronger trust in the prediction results. The system effectively differentiates between high-confidence and low-confidence predictions, ensuring reliable fraud detection outcomes.

8.5 Model Aggregation Result

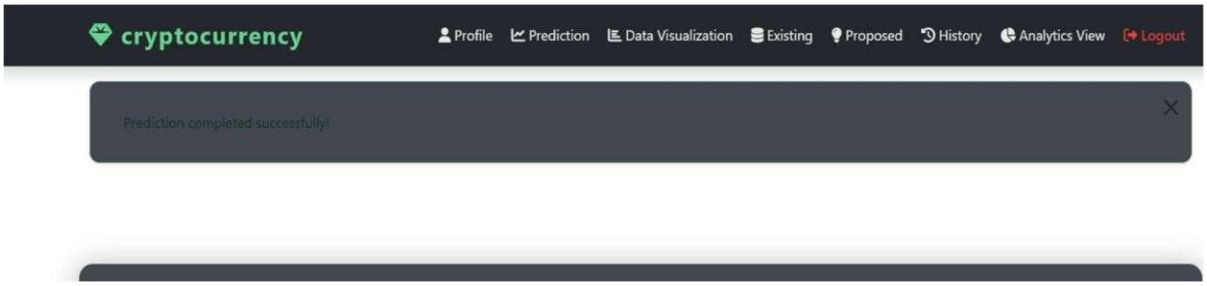


Fig 8.5 Aggregation Output

Description:

The system performs aggregation of model outputs to generate a final prediction. The aggregated result reflects improved accuracy by combining multiple evaluation parameters. This demonstrates the effectiveness of the integrated learning approach in producing reliable fraud detection results.

8.6 Blockchain Result

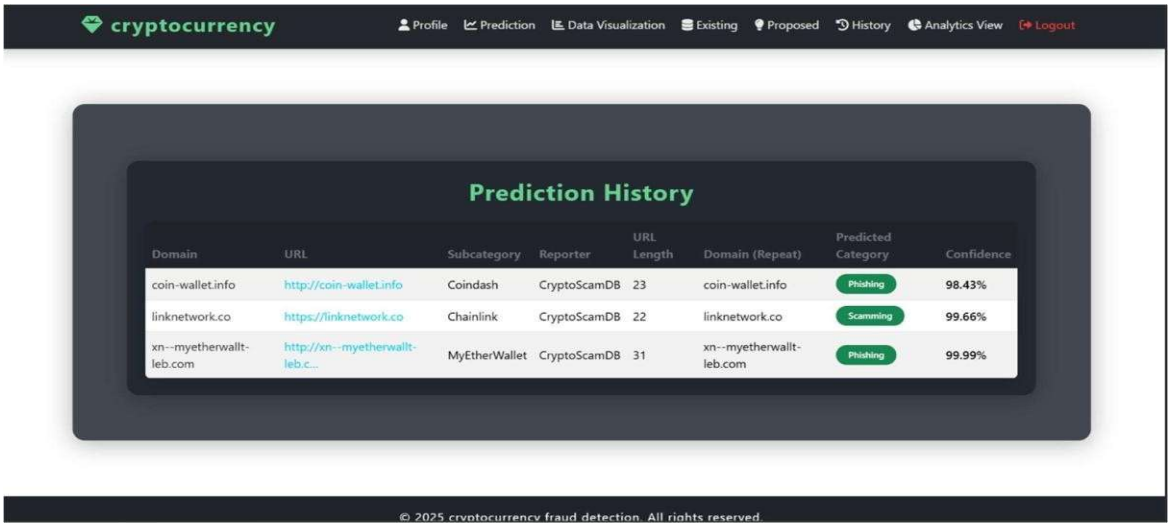


Fig 8.6 Blockchain Output

Description:

The blockchain module records prediction results and system activities in a secure and tamper-proof manner. Each transaction or prediction is stored as a record that includes relevant details such as domain, category, and confidence score. This ensures transparency, traceability, and data integrity within the system.

8.7 Prediction Result

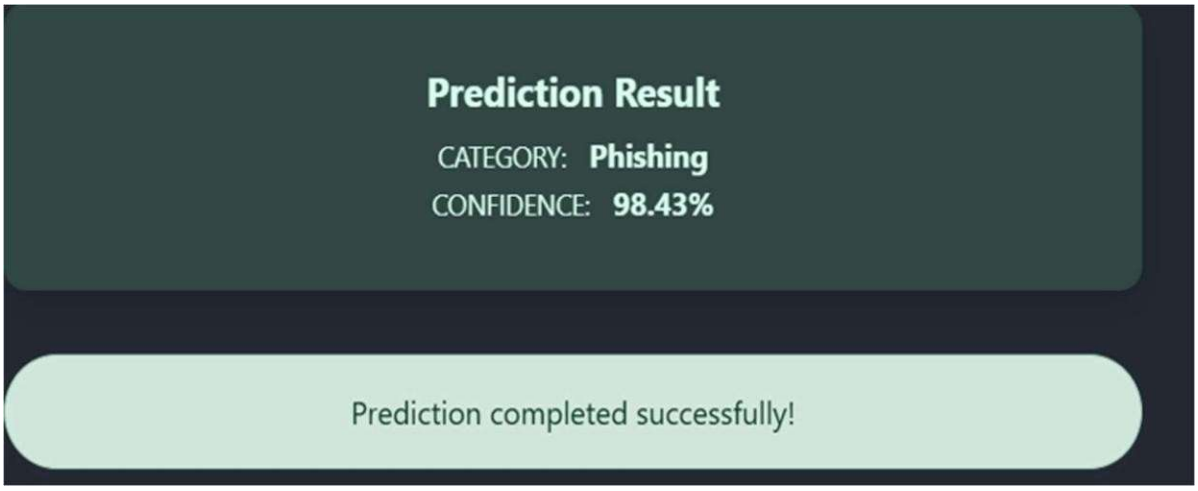


Fig 8.7 Prediction Output

Description:

The system successfully generates fraud predictions based on user input. The predicted category, such as phishing or scamming, is displayed along with a confidence score. The results are consistent for identical inputs, demonstrating the reliability and deterministic behavior of the system. In cases where input is invalid, the system provides appropriate error messages.

8.8 Overall System Performance

Description:

The overall system performance indicates that the integration of federated learning, trust evaluation, and blockchain significantly enhances the accuracy, security, and reliability of cryptocurrency fraud detection. The system successfully maintains data privacy by avoiding raw data sharing, ensures secure and transparent operations through blockchain integration, and provides accurate predictions with high confidence. The results confirm that the proposed system is efficient, scalable, and suitable for real-world cryptocurrency fraud detection applications.

9. CONCLUSION

This paper effectively highlights the potential of leveraging BCFL to detect and mitigate cryptocurrency fraud. It underscores its pivotal role in enhancing data privacy, bolstering reliability, and ensuring scalability within the realm of crypto fraud detection. By analyzing the most recent advancements and existing gaps in this domain, it is clear that the decentralization, privacy preservation, and scalability attributes of the BCFL approach make it a promising strategy for tackling the mounting issues of cryptocurrency fraud. Most critical contributions from the studies reviewed point to the importance of ensuring data privacy, dealing with scalability issues, reducing the risk of inference and malicious attacks, and managing decentralized communication. In summary, various BCFL architectures for crypto fraud detection have been presented in literature, falling broadly into three categories: fully integrated BCFL, flexibly integrated BCFL and loosely integrated BCFL. Table 5 compares these architectures in terms of pros, cons, and the best scenarios when they should be employed in the industry. Each BCFL architecture is designed to incorporate a learning model that leverages diverse data sources, including transaction data, user data, and device data. This integration facilitates the identification of specific fraudulent transaction patterns. The type of crypto fraud targeted during the job creation phase of the BCFL workflow, coupled with the resources available to the entities participating in the federated system, largely dictates the most appropriate architecture to adopt. This ensures that computational resources undergo thorough analysis before deployment. In certain scenarios, customization specific to the system may be required to optimize performance. The BCFL workflow phases of crypto fraud detection extend over multiple stages, each of which contributes to developing and implementing an effective predictive model capable of identifying crypto fraud. From establishing a task to deploying the updated global model, every step is instrumental in ensuring a robust, scalable, and secure crypto fraud detection system.

10. FUTURE ENHANCEMENTS

The current design establishes a solid conceptual framework for a secure and scalable blockchain based federated learning system for cryptocurrency fraud detection; however, several enhancements can be incorporated during the implementation and extension phases to strengthen system performance and applicability. Future work may include integrating more advanced and specialized models such as Graph Neural Networks (GNNs), Transformer-based architectures, or hybrid deep learning models to better capture complex transaction relationships and evolving fraud patterns within blockchain networks. Additional enhancements may involve expanding the dataset by incorporating cross-chain transaction data, decentralized finance (DeFi) activities, and diverse user behavior patterns, enabling the system to detect fraudulent activities across multiple blockchain ecosystems. The inclusion of heterogeneous data sources such as smart contract interactions, wallet behavior, and transaction histories can further improve detection accuracy and robustness. The system can also be extended with real-time fraud monitoring capabilities, allowing continuous analysis of blockchain transactions and early identification of suspicious activities. Incorporating dynamic feedback mechanisms, where participating nodes validate and share insights on detected fraud cases, can help in continuously updating and refining the federated learning models without compromising data privacy. Furthermore, integrating visualization dashboards and analytical tools can assist stakeholders in understanding fraud trends, anomaly patterns, and risk levels across distributed networks. Enhancing security through advanced cryptographic techniques such as secure multi-party computation, differential privacy, and blockchain consensus optimization can further strengthen trust and data protection. These improvements will support the transition from a conceptual design into a robust, scalable, and secure system for effective cryptocurrency fraud detection in decentralized environments.

REFERENCES

[1] J. Zhang, Y. Liu, and H. Wang, "Federated Learning-Based Energy Demand Forecasting for Electric Vehicle Charging Networks," IEEE Transactions on Smart Grid, 2026.

[2] P. Mehta and P. Singh, "Trust-Aware Aggregation in Federated Learning for Distributed Energy Systems," IEEE Internet of Things Journal, 2025.

[3] L. Chen, X. Zhao, and Y. Sun, "Electric Vehicle Charging Load Forecasting Using Machine Learning and Distributed Systems," Applied Energy, 2025.

[4] R. Gupta, S. Iyer, and K. Reddy, "Privacy-Preserving Federated Learning for Smart Energy Management," IEEE Transactions on Industrial Informatics, 2025.

[5] M. Alazab, F. Tariq, and A. Jolfaei, "Blockchain-Based Secure Data Sharing Framework for Energy Systems," IEEE Transactions on Information Forensics and Security, 2025.

[6] N. Wang, W. Yang, X. Wang, L. Wu, Z. Guan, X. Du, and M. Guizani, "A blockchain-based privacy-preserving federated learning scheme for Internet of Vehicles," Digital Communications and Networks, vol. 10, no. 1, pp. 126–134, Feb. 2024.

[7] A. Liu, Y. Liu, Q. Wu, B. Zhao, D. Li, Y. Lu, R. Lu, and W. Susilo, "CHERUBIM: A secure and highly parallel cross-shard consensus using quadruple pipelined two-phase commit for sharding blockchains," IEEE Transactions on Information Forensics and Security, 2024.

[8] B. Liu, N. Lv, Y. Guo, and Y. Li, "Recent advances on federated learning: A systematic survey," arXiv preprint arXiv:2301.01299, 2023.

[9] A. Qammar, A. Karim, H. Ning, and J. Ding, "Securing federated learning with blockchain: A systematic literature review," Artificial Intelligence Review, vol. 56, no. 5, pp. 3951–3985, May 2023.

[10] I. Ullah, X. Deng, X. Pei, P. Jiang, and H. Mushtaq, "A verifiable and privacy-presing blockchain-based federated learning approach," Peer-to-Peer Networking and Applications, vol. 16, no. 5, pp. 2256–2270, Sep. 2023.

[11] H. Ratnayake, L. Chen, and X. Ding, "A review of federated learning: Taxonomy, privacy and future directions," Journal of Intelligent Information Systems, vol. 61, no. 3, pp. 923–949, Dec. 2023.

[12] E. Goh et al., "Blockchain-enabled federated learning: A reference architecture design, implementation, and verification," IEEE Access, vol. 11, pp. 145747–145762, 2023.

[13] Y. Jin et al., “Information-bottleneck-based behavior representation learning for multi-agent reinforcement learning,” in Proc. IEEE ICAS, 2023.
[14] Y. Lin et al., “DRL-based adaptive shareing for blockchain-based federated learning,” IEEE Transactions on Communications, vol. 71, no. 10, pp. 5992–6004, 2023.
[15] W. Li, B. Yang, and Y. Song, “Secure multi-party computing for financial sector based on blockchain,” in Proc. IEEE ICSESS, 2023. 87
[16] A. F. M. S. Shah et al., “On the vital aspects and characteristics of cryptocurrency—A survey,” IEEE Access, vol. 11, pp. 9451–9468, 2023.
[17] P. Durgapal et al., “A comprehensive distributed framework for cross-silo federated learning using blockchain,” in Proc. BCCA, 2023.
[18] A. P. Kalapaaking et al., “Blockchain-based federated learning with secure aggregation in trusted execution environment for IoT,” IEEE Transactions on Industrial Informatics, vol. 19, no. 2, pp. 1703–1714, 2023.
[19] H. A. Madni et al., “Blockchain-based swarm learning for the mitigation of gradient leakage in federated learning,” IEEE Access, vol. 11, pp. 16549–16556, 2023.
[20] R. Xu, C. Li, and J. Joshi, “Blockchain-based transparency framework for privacy preserving third-party services,” IEEE Transactions on Dependable and Secure Computing, vol. 20, no. 3, pp. 2302–2313, 2023.
[21] Z. Wang et al., “Incentive mechanism design for joint resource allocation in blockchain-based federated learning,” IEEE Transactions on Parallel and Distributed Systems, vol. 34, no. 5, pp. 1536– 1547, 2023.
[22] Q. Hu et al., “Blockchain and federated edge learning for privacy-preserving mobile crowdsensing,” IEEE Internet of Things Journal, vol. 10, no. 14, pp. 12000–12011, 2023.
[23] Z. Lian et al., “Blockchain-based two-stage federated learning with non-IID data in IoMT system,” IEEE Transactions on Computational Social Systems, vol. 10, no. 4, pp. 1701–1710, 2023.
[24] A. Al Asqah and T. Moulahi, “Federated learning and blockchain integration for privacy protection in IoT: Challenges and solutions,” Future Internet, vol. 15, no. 6, p. 203, 2023.
[25] X. Liang et al., “Self-supervised cross-silo federated neural architecture search,” arXiv preprint arXiv:2101.11896, 2021.