Blockchain Integrated Machine Learning System for Cryptocurrency Fraud Detection

Rakesh Joshi
IIT Bombay
Email: rakeshjoshi@iitb.ac.in

***Abstract*—This report explains the development of a system that combines machine learning and blockchain to detect fraudulent transactions and store the results in a permanent and verifiable way. A Random Forest model is trained using real Ethereum transaction data to identify suspicious behavior. The prediction results are then stored on the Ethereum Sepolia test network using a smart contract. This design makes fraud detection both intelligent and transparent.**

***Index Terms*—Fraud Detection, Machine Learning, Blockchain, Ethereum, Smart Contracts, Random Forest.**

I. Introduction

Modern financial systems process millions of transactions every day, and this makes them a common target for fraud. Most existing systems depend on centralized databases, which means data can be altered, deleted, or attacked. To reduce this risk, this project combines machine learning with blockchain. Machine learning is used to detect fraud patterns, while blockchain is used to store results in a way that cannot be changed later.

Each transaction is represented as a vector $\mathbf{x} = [b, t, v]$, where b is the block height, t is the timestamp, and v is the transaction value. The aim is to learn a function $f(\mathbf{x})$ that outputs $y \in \{0, 1\}$, where 1 means fraud and 0 means a normal transaction. The main idea is not only to predict fraud, but also to make every prediction verifiable by anyone.

II. Dataset and Preprocessing

The dataset contains 254,973 Ethereum transactions with information such as transaction hash, block height, timestamp, sender, receiver, value, and an error flag. The column `isError` is used as the label. Columns like transaction hash and wallet addresses were removed because they are identifiers and do not help in learning patterns. After cleaning, the final features used were BlockHeight, TimeStamp, and Value.

Only about 6.13% of the transactions were marked as fraud or error, which means the data is highly imbalanced. To handle this, class balancing methods were used during training. Missing values were filled with zero, and features with no variation were removed so that the model does not learn useless information.

III. Machine Learning Model

Fraud detection is treated as a binary classification problem. Random Forest was chosen because it is reliable, handles noisy data well, and reduces overfitting. It works by creating many decision trees and letting them vote on the final result. If most trees say a transaction is fraudulent, then the system marks it as fraud.

Three models were tested: Logistic Regression, Random Forest, and XGBoost. They were compared using accuracy, precision, recall, F1-score, and ROC-AUC. Random Forest performed the best with an F1-score of 0.8163, meaning it achieved a good balance between catching fraud and avoiding false alarms. This model was trained again using the full training data and saved for use in the application.

IV. Blockchain Design

Blockchain was used to make sure that once a fraud decision is made, it can never be changed. For every transaction, a digital fingerprint is created using SHA-256 hashing. If even a small part of the input changes, the hash becomes completely different, which makes tampering easy to detect.

The smart contract stores the hash of the transaction data, the fraud result, the confidence score, and the name of the model used. Once this data is written to the blockchain, no one can edit or delete it. This creates a permanent audit trail for every AI decision.

V. System Architecture

The system is built in layers. First, the machine learning layer predicts whether a transaction is fraud. Then, the hashing layer creates a fingerprint of the input. After that, the blockchain layer stores the result in a smart contract. Finally, the application layer connects everything and allows users to interact with the system.

The overall flow is simple: user data goes to the AI model, the model gives a result, the data is hashed, and everything is written to the blockchain.

VI. Implementation

The model was trained by minimizing classification errors, while handling class imbalance using balanced weights. The smart contract was written in Solidity and deployed on the Sepolia test network. It prevents duplicate records and emits an event whenever a new record is added.

The backend was built using FastAPI and Web3.py. It loads the trained model, receives transaction data, makes predictions, generates hashes, signs transactions using a private key, and sends them to the blockchain. The frontend was created using

Streamlit and provides a simple form where users can enter transaction details and see the result.

To make the system more stable, a self-healing feature was added. If the model file is missing or broken, a small replacement model is trained automatically so the system can still run.

VII. Results

The system works as expected. It can detect fraud in real time and store each result on the Sepolia blockchain. Every stored record can be checked using Etherscan, which proves that the data is really on the blockchain and has not been changed.

VIII. Conclusion

This project shows that AI decisions do not have to remain hidden inside closed systems. By storing fraud predictions on a public blockchain, the system becomes transparent and trustworthy. Machine learning provides intelligence, and blockchain provides security. Together, they create a system that can be used in future financial and automated platforms where trust is critical.