SmartChain

Python Blockchain with AI/ML Integration

Designed by Muhammed Sayees

1. Overview

SmartChain is a Python-based blockchain platform enhanced with AI and ML capabilities for fraud detection, dynamic mining difficulty adjustment, anomaly detection, and transaction pattern analysis. It combines the fundamentals of blockchain technology with modern intelligence to create a robust, secure, and adaptive network.

2. Modern & Advanced Features

- Federated Learning for collaborative anomaly detection across multiple nodes.
- Explainable AI to provide human-readable explanations for flagged transactions.
- Graph Neural Network-based transaction network analysis for advanced insights.
- Privacy-Preserving Analytics with differential privacy techniques.
- Smart Contract Integration to trigger AI-based rules on-chain.
- Proof-of-Stake (PoS) simulation alongside PoW for hybrid consensus.
- Real-time WebSocket notifications for critical events and alerts.
- Tokenomics Simulation for economic modeling and supply dynamics.
- Reputation Scoring and Trust Mechanisms for nodes and users.
- Integration with ChatGPT API as an NLP-driven transaction assistant.

3. AI/ML Components

Anomaly Detection

Isolation Forest & One-Class SVM for fraud and unusual activity detection.

Difficulty Prediction

Regression models to predict and adjust mining difficulty in real-time.

User Classification

K-Means & DBSCAN clustering for categorizing user roles (regular, miner, whale).

Graph Analytics

GNN techniques for analyzing transaction relationships.

4. Tech Stack

- Python 3.8+
- Flask / FastAPI for RESTful API
- ReportLab for PDF generation
- scikit-learn, pandas, NumPy for ML
- Plotly for data visualization
- Web3.py for smart contract interactions
- SQLite / LevelDB for on-disk storage
- Docker for containerization

5. Project Structure

• blockchain/block.py – Block data structure and hashing functions

• blockchain/chain.py – Chain management and consensus logic

• blockchain/transaction.py – Transaction model and validation

• ml/fraud_detection.py – Training and inference scripts for anomaly detection

• ml/difficulty_model.py – Model for dynamic difficulty adjustment

• ml/preprocess.py – Data cleaning and feature engineering

• app.py – Flask/FastAPI server exposing endpoints

• templates/ – HTML frontend for dashboard (optional)

• requirements.txt – Project dependencies

• README.md – Detailed documentation

6. Future Enhancements & Roadmap

• Implement Proof-of-Stake (PoS) and Delegated PoS consensus algorithms.

• Build a React/Next.js front-end with real-time charts and maps.

• Integrate zero-knowledge proofs for enhanced privacy.

• Develop mobile client SDK for light nodes.

• Add multi-language support for broader adoption.