

Privacy-Preserving Fraud Detection via Blockchain and Machine Learning: A Decentralized and Incentive-Driven Framework

Maddala Harika¹, K. Lakshmi Prasuna²

¹Pursuing M.Tech CSE, ²Associate Professor

Department of Computer Science and Engineering

V.K.R, V.N.B & A.G.K. COLLEGE OF ENGINEERING (Autonomous)

Affiliated to JNTUK, Kakinada & Approved by AICTE, New Delhi

An ISO 9001:2015 Certified Institution, Accredited by NAAC with 'A' Grade

Eluru Road, Gudivada, Krishna Dist., Andhra Pradesh - 521301

Abstract:

Detecting fraud in digital environments presents a persistent challenge, particularly where the need for real-time, large-scale data analysis conflicts with stringent privacy requirements. This paper introduces a novel framework that synergizes blockchain technology with advanced machine learning techniques to achieve secure, privacy-preserving, and collaborative fraud detection. The architecture employs blockchain to ensure data integrity, transparency, and decentralized trust, while integrating federated learning and differential privacy to train models without exposing sensitive user information. To incentivize participation, the system incorporates a dynamic, smart contract-based reward mechanism that encourages the contribution of high-quality data. By uniting privacy-aware computation with decentralized infrastructure, the proposed solution offers robust fraud detection capabilities, protects user confidentiality, and fosters cross-organizational cooperation. Experimental results on both synthetic and real-world financial datasets demonstrate the framework's effectiveness, scalability, and adaptability in detecting sophisticated and evolving fraud patterns.

Introduction:

In the digital age, fraud continues to pose a significant threat across sectors such as finance, healthcare, e-commerce, and telecommunications. The increasing volume, velocity, and variety of data, coupled with the growing sophistication of fraudulent activities, have rendered traditional detection systems inadequate. Recent advancements in **machine learning (ML)** have brought new promise to fraud detection by enabling automated and adaptive identification of anomalous behaviors. However, these systems often face challenges related to **data privacy**, **model transparency**, and resistance to adversarial manipulation. Simultaneously, **blockchain technology** has emerged as a decentralized and tamper-proof ledger system offering transparency, traceability, and enhanced security. Its immutability and consensus-driven validation mechanisms make it an appealing infrastructure for fraud-resistant systems. However, blockchain alone is not well-suited for real-time fraud detection or intelligent pattern recognition, areas where ML excels.

This paper proposes a **privacy-preserving and adaptive incentive-based framework** that synergistically combines blockchain and machine learning for fraud detection. The proposed architecture ensures that sensitive data remains secure through on-chain/off-chain hybrid storage models, while ML algorithms operate on encrypted or obfuscated data to maintain user confidentiality. Moreover, we introduce an **incentive mechanism** powered by smart contracts to reward honest participants and data contributors, fostering collaborative fraud intelligence without compromising trust or privacy.

The integration of blockchain and ML not only enhances fraud detection capabilities but also introduces **self-regulating, explainable, and tamper-resistant ecosystems** that are resilient to evolving fraud tactics. By addressing both detection efficacy and data governance, this research aims to set a foundation for scalable and trustworthy fraud detection systems in decentralized environments.

Literature Survey:

Blockchain for Enhanced Security and Transparency

Blockchain technology provides a decentralized, immutable ledger that enhances data integrity and traceability in fraud detection systems. Studies such as Zheng et al. (2018) demonstrate how blockchain ensures tamper-proof transaction records, making fraudulent manipulations more difficult and increasing trust among participants.

Machine Learning for Dynamic Fraud Detection

Machine learning algorithms have been widely adopted for detecting complex fraud patterns by learning from historical transaction data. Research by Ngai et al. (2011) highlights supervised and unsupervised learning techniques for identifying anomalies and suspicious behaviors in real-time.

Privacy-Preserving Mechanisms

Integrating privacy-preserving methods like federated learning and differential privacy with blockchain has been explored to secure sensitive data during fraud detection. For instance, Yang et al. (2019) propose frameworks that allow collaborative model training without exposing raw data, ensuring user privacy while maintaining detection accuracy.

Adaptive Incentive Schemes

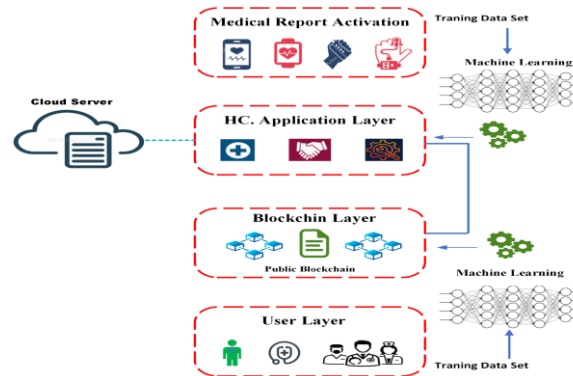
Incentive mechanisms are critical for encouraging honest participation and data sharing in decentralized fraud detection networks. Research such as Kshetri (2017) discusses adaptive reward systems built on smart contracts that dynamically adjust incentives based on user behavior and contribution to fraud mitigation.

Hybrid Approaches Combining Blockchain and ML

Recent work focuses on hybrid architectures that leverage blockchain's decentralized consensus and machine learning's predictive capabilities. For example, Fan et al. (2020) introduce systems where

blockchain validates data integrity, and ML models analyze transaction patterns for fraud, achieving robustness and scalability.

System Architecture



System Implementations:

1. Data Collection and Preprocessing Module

Purpose: Collect transactional and user behavior data from various sources while ensuring data anonymization to protect user privacy.

Functions:

Interface with transactional systems (e.g., banking, e-commerce).

Clean and normalize data to a standard format.

Encrypt sensitive data fields before storing or processing.

Generate feature vectors for machine learning models.

2. Blockchain Network Module

Purpose: Provide a decentralized, tamper-proof ledger to record transactions and model updates securely.

Functions:

Implement a permissioned blockchain for trusted nodes participation.

Store hashed transaction records and alerts.

Manage smart contracts that automate fraud reporting and incentive distribution.

Ensure data immutability and auditability.

3. Machine Learning-based Fraud Detection Module

Purpose: Analyze incoming data streams and detect fraudulent transactions using adaptive learning models.

Functions:

Deploy supervised and unsupervised models (e.g., Random Forest, Autoencoders).

Continuously update models with new labeled data (online learning).

Perform anomaly detection and score transactions in real-time.

Generate alerts for suspicious activities.

Privacy-Preserving Mechanisms Module

Purpose: Protect user data privacy throughout the fraud detection process.

Functions:

Implement homomorphic encryption or secure multi-party computation (SMPC) for encrypted data processing.

Use differential privacy techniques to prevent data leakage.

Anonymize user identities before analysis or sharing.

Control access permissions via blockchain smart contracts.

Incentive Management Module

Purpose: Motivate participants (e.g., nodes, users, data providers) to contribute data and verify fraud reports.

Functions: Design adaptive incentive mechanisms rewarding accurate fraud detection and reporting.

Utilize smart contracts to automate reward distribution.

Track participant contributions and reputation scores on-chain.

Adjust incentives based on fraud detection accuracy and participation levels.

6. Alert and Reporting Module

Purpose: Notify stakeholders and authorities about detected fraud in a timely and secure manner.

Functions:

- Generate real-time alerts for suspicious transactions.
- Provide dashboards with fraud analytics and system status.
- Allow secure sharing of fraud reports with authorized parties.
- Log alert histories on blockchain for transparency.

User Interface Module

Purpose: Facilitate interaction for users, administrators, and auditors.

Functions:

- User-friendly portals for monitoring transaction status.
- Interfaces for submitting feedback or dispute claims.
- Administrative tools to configure models and incentives.

Visualization of fraud patterns and system metrics.

Increasing digital transaction making new ways for the fraudsters to perform fraud transaction and to detect such fraud transaction, financial organization heavily dependent on machine learning algorithms which required accurate dataset for correct prediction. In the existing system all organizations were using single centralized server to train ML algorithms whose database can be easily tamper by database administrator without getting detected. ML algorithms trained on tamper data will start predicting incorrect transaction and enable fraudsters to make successful transaction.

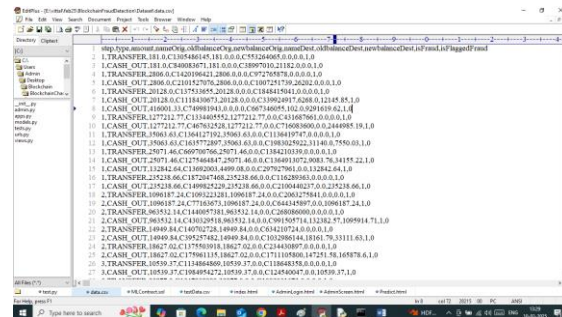
To avoid above issue author of this paper employing Blockchain technology to maintain ML models whose database cannot be tamper in any manner and Blockchain has inbuilt support for data privacy and security. Blockchain store each record as block/transaction and associate each block with unique hashcode and this hashcode get verified for subsequent block storage, if data tamper in any block then result to hashcode mismatch and get detected. This process of verification make Blockchain tamper proof.

In propose work author experimenting with various ML algorithms such as Passive Aggressive Classifier (PAC), Stochastic Gradient (SGD), Perceptron and Naïve Bayes and then training all this algorithm with incremental support to update model weights with new and old training data. Each algorithm performance is evaluated in terms of accuracy, precision, confusion matrix, recall and FSCORE. Model with best accuracy will be updated to Blockchain and all organization can train local model and update best model weights to Blockchain. Anytime we can obtained weights from Blockchain and then can perform prediction. Due to high amount of data so Blockchain will take more time for mining so author giving incentives based on Difficulty level. Difficulty level can be calculated by splitting data into multiple parts and each part will get trained incrementally. Parts with high number of records may consume more Blockchain mining time.

To train and test above algorithms performance author has used 'Financial Transaction dataset' which can be downloaded from below KAGGLE website

<https://www.kaggle.com/datasets/ealaxi/paysim1/data>

In below screen sowing dataset details

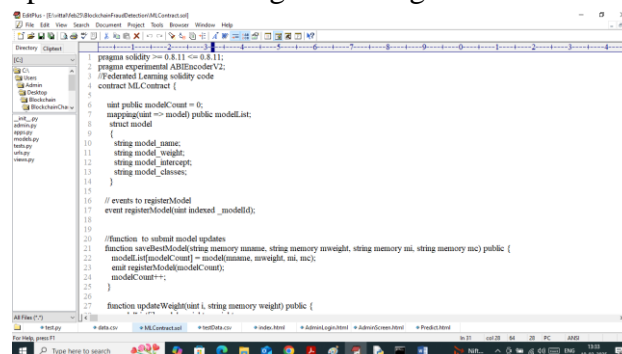


In above dataset screen first row contains dataset column names and remaining rows contains dataset values and in each record contains class label as 0 (normal) and 1 (fraud). So by using above dataset will train and test each algorithm performance.

Note: to calculate difficulty level we have divided dataset into two parts where first part contains 10000 records as Difficulty Level1 and second part contains 30000 records as difficulty level2. Both levels of data get trained incrementally.

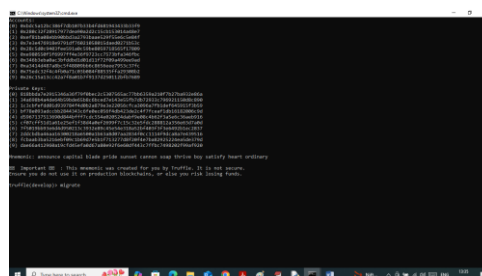
Blockchain Smart Contract Deployment

Blockchain can store and retrieve data using Smart Contract which can be designed using SOLIDITY programming. Smart contract contains functions which can be called using any programming language to store and retrieve data. In propose work to manage ML weights we have designed following contract



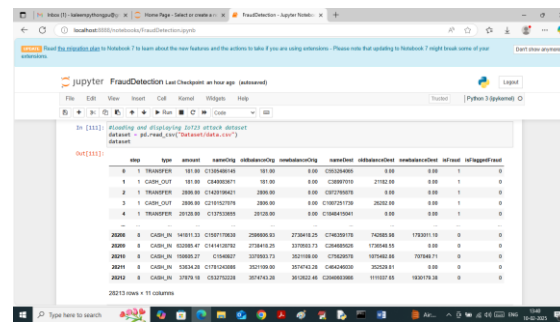
In above contract we have defined function to manage model weights and now we need to deploy above contract to Blockchain Ethereum using below steps

First go inside 'hello-eth/node-modules/bin' folder and then look and double click on 'runBlockchain.bat' file to get below page

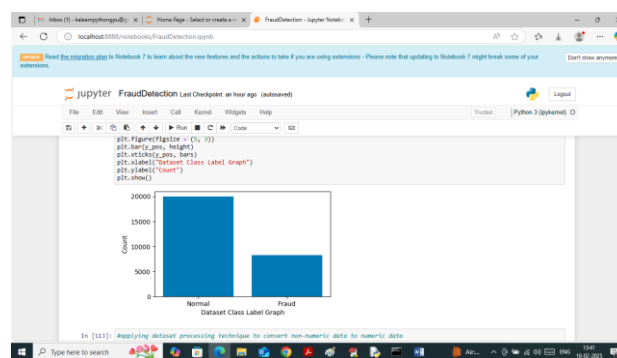


In above screen Ethereum started with default accounts and private keys and then type command as 'migrate' and then press enter key to get below page

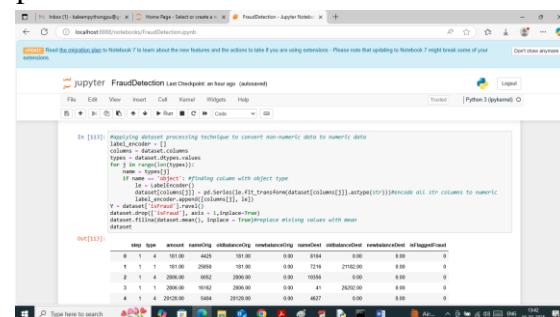




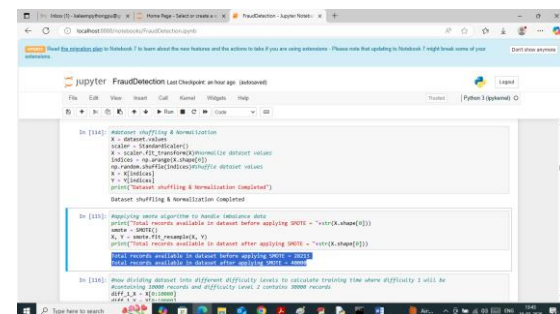
In above screen loading and displaying dataset values and above dataset contains some non-numeric values but ML take only numeric values so by applying Label Encoder class can convert non-numeric to numeric data



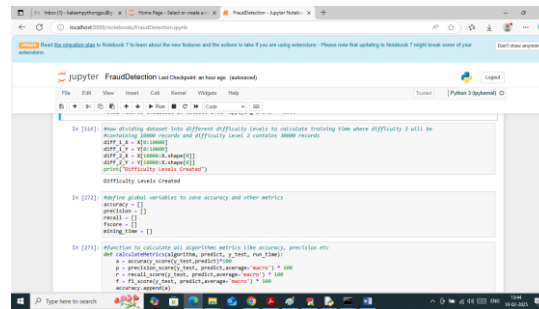
In above screen visualizing graph of normal and fraud transaction exists in dataset



In above screen applying label encoder class to convert non-numeric data to numeric data and then replacing missing values with MEAN



In above screen applying processing technique in first block to shuffle and normalized dataset values. In second block applying SMOTE technique to handle imbalance issues. In above screen before applying smote dataset were having 28000 records and after applying smote dataset size increased to 40000



```

In [18]: # Now dividing dataset into two different difficulty levels to calculate training time where difficulty 1 will be
# generating 10000 records and difficulty level 2 contains 30000 records
diff_1_x = X[:10000]
diff_1_y = y[:10000]
diff_2_x = X[10000:]
diff_2_y = y[10000:]
print('Difficulty Levels Created')

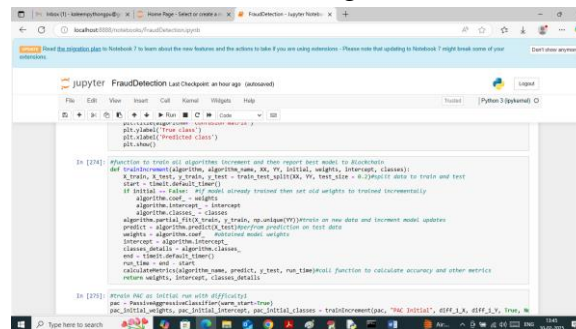
Difficulty Levels Created

In [27]: # Define global variables to save accuracy and other metrics
accuracy = 0
precision = 0
recall = 0
f1_score = 0
mining_time = 0

In [28]: # Function to calculate all algorithm metrics like accuracy, precision etc
def calculate_metrics(algorithm, predict, y_test, run_time):
    a = accuracy_score(y_test, predict)
    p = precision_score(y_test, predict, average='macro')
    r = recall_score(y_test, predict, average='macro')
    f = f1_score(y_test, predict, average='macro')
    return accuracy, precision, recall, f1_score, run_time

In [29]: # Function to train all algorithm incrementally and then report best model to Blockchain
def train_incremental(algorithm, X_train, y_train, X_test, y_test, start_time, end_time, initial_weights, initial_intercept, initial_classes):
    # Train the model incrementally
    algorithm.partial_fit(X_train, y_train, incremental=True)
    # Calculate metrics
    accuracy, precision, recall, f1_score, run_time = calculate_metrics(algorithm, algorithm.predict(X_test), y_test, run_time)
    # Update metrics
    accuracy = (accuracy * start_time + accuracy * (end_time - start_time)) / end_time
    precision = (precision * start_time + precision * (end_time - start_time)) / end_time
    recall = (recall * start_time + recall * (end_time - start_time)) / end_time
    f1_score = (f1_score * start_time + f1_score * (end_time - start_time)) / end_time
    run_time = (run_time * start_time + run_time * (end_time - start_time)) / end_time
    # Return metrics
    return accuracy, precision, recall, f1_score, run_time
    
```

In above screen dividing dataset into two difficulty levels were difficulty 1 having 10000 records and difficulty 2 having 30000 records. In next blocks defining function to calculate accuracy and other metrics

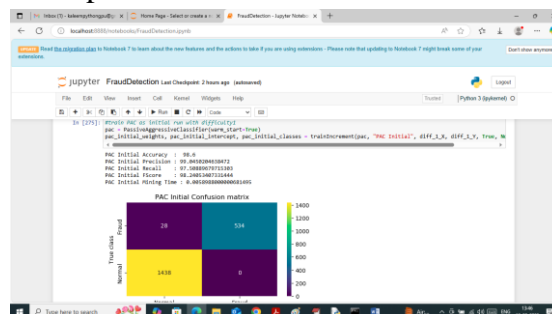


```

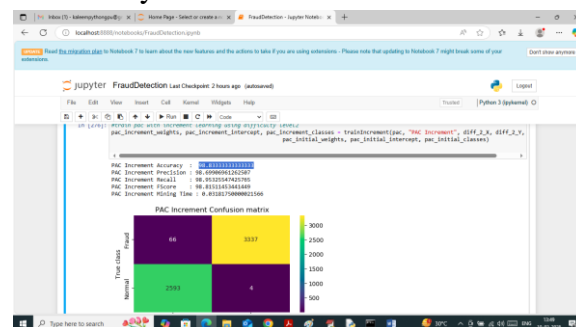
In [24]: # Function to train all algorithm incrementally and then report best model to Blockchain
def train_incremental(algorithm, X_train, y_train, X_test, y_test, start_time, end_time, initial_weights, initial_intercept, initial_classes):
    # Train the model incrementally
    algorithm.partial_fit(X_train, y_train, incremental=True)
    # Calculate metrics
    accuracy, precision, recall, f1_score, run_time = calculate_metrics(algorithm, algorithm.predict(X_test), y_test, run_time)
    # Update metrics
    accuracy = (accuracy * start_time + accuracy * (end_time - start_time)) / end_time
    precision = (precision * start_time + precision * (end_time - start_time)) / end_time
    recall = (recall * start_time + recall * (end_time - start_time)) / end_time
    f1_score = (f1_score * start_time + f1_score * (end_time - start_time)) / end_time
    run_time = (run_time * start_time + run_time * (end_time - start_time)) / end_time
    # Return metrics
    return accuracy, precision, recall, f1_score, run_time

In [25]: # Define global variables to save accuracy and other metrics
accuracy = 0
precision = 0
recall = 0
f1_score = 0
mining_time = 0
    
```

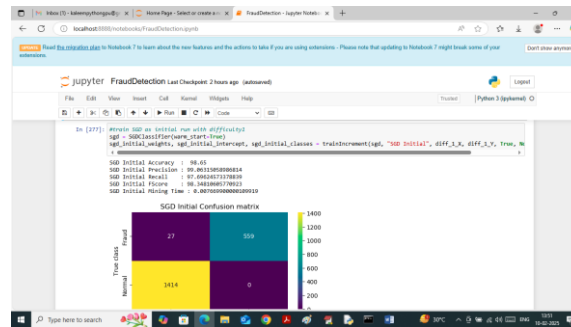
In above screen defining function to train algorithm incrementally and then calculate accuracy and other metrics and then getting weights to update to Blockchain



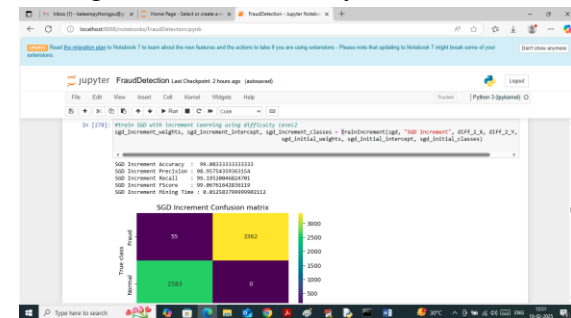
In above screen training PAC algorithm initially with difficulty 1 (10000 records) and then PAC got 98.6% accuracy and can see other metrics like precision, recall and FSCORE. In above confusion matrix graph x-axis represents Predicted Labels and y-axis represents True Labels (normal or fraud) and then yellow and light green boxes in diagonal represents correct prediction count and remaining blue boxes represents incorrect prediction count which are very few. In above screen can see training time also



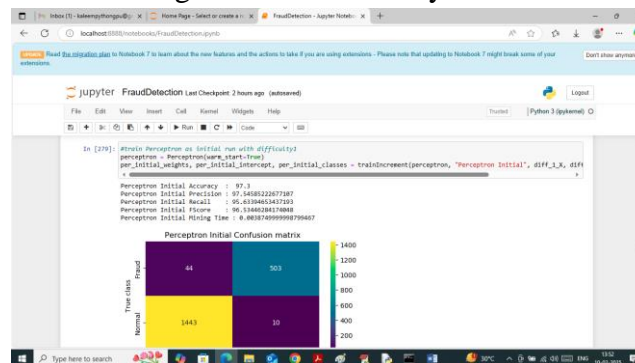
In above screen PAC trained incrementally on difficulty level 2 (30000 records) and then got 98.83% accuracy and can see other metrics output also



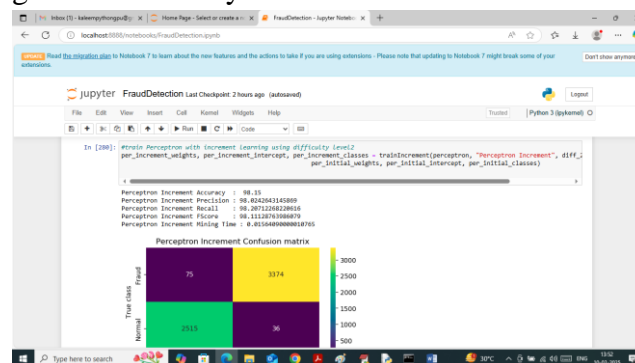
In above screen SGD on initial data got 98.65% accuracy



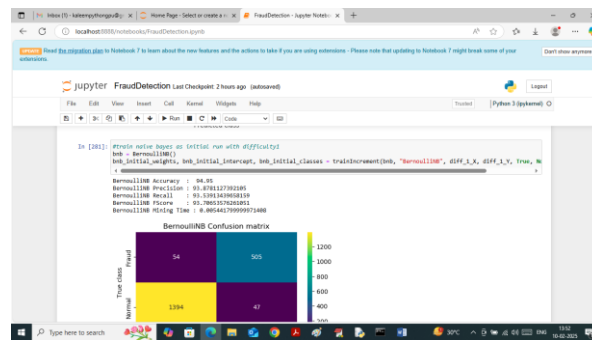
In above screen SGD on increment data got 99.83% accuracy



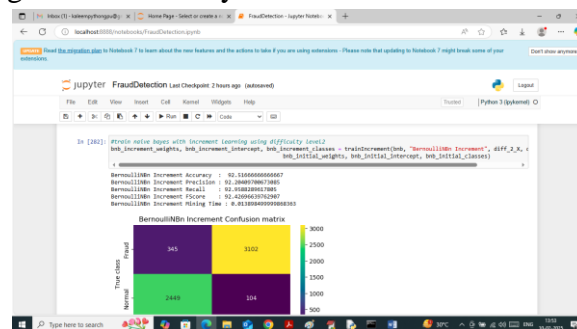
In above screen Perceptron got 97% accuracy on initial data



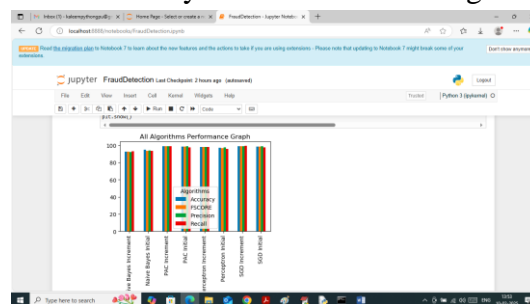
In above screen perceptron got 98% accuracy on increment training



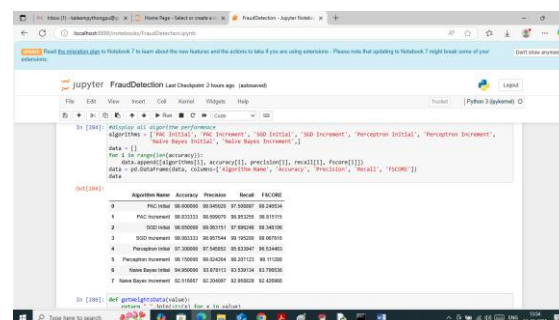
In above screen Naïve Bayes got 94% accuracy on initial data



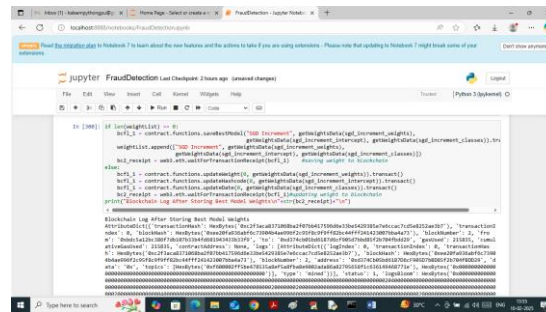
In above screen naïve bayes got 92% accuracy on increment training



In above screen visualizing all algorithm performance where x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars.

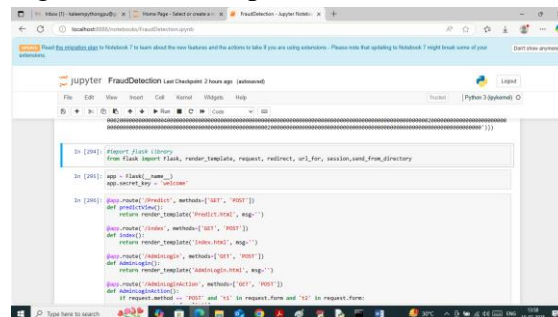


In above screen showing all algorithms performance in tabular format and in all algorithms SGD with increment training got high accuracy

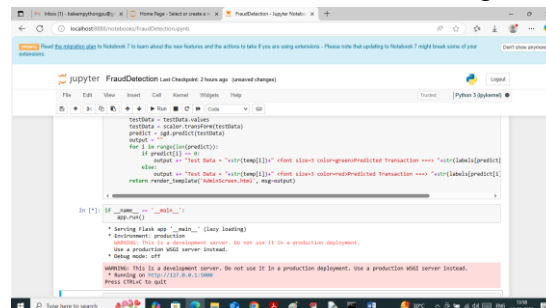


In above screen saving SGD weights to Blockchain as it got high accuracy and consider as best model. In above screen after saving weights to Blockchain we got all log details which contains details like Block No, transaction no, hash code and many other details.

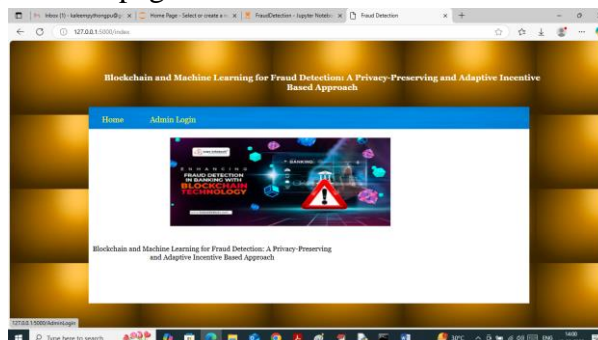
In above screens we trained ml models incrementally and then save best weights in blockchain and now will detect fraud transaction using below web output



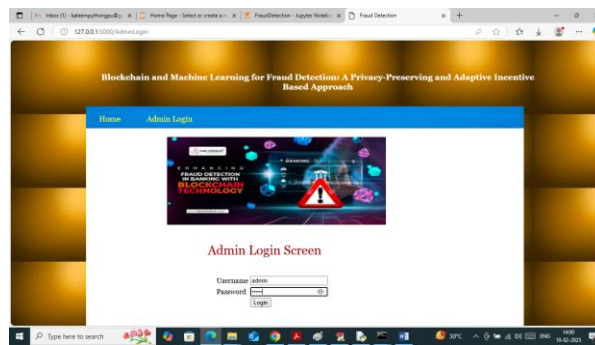
In above screen run all flask blocks to start FLASK web server and get below page



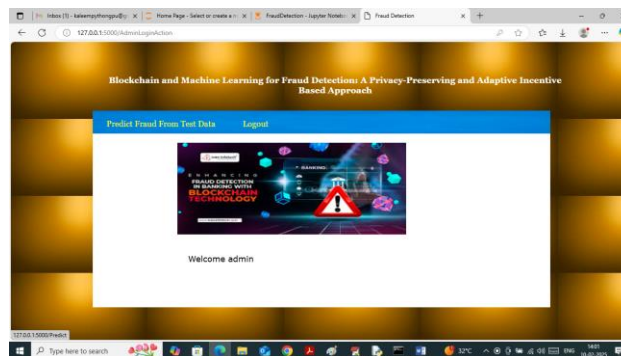
In above screen flask server started and now open browser and enter URL as <http://127.0.0.1:5000/index> and then press enter key to get below page



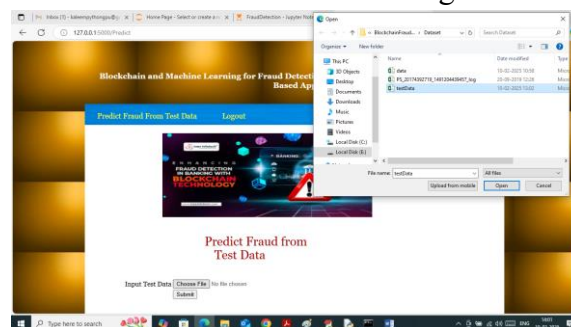
In above screen click on 'Admin Login' link to get below page



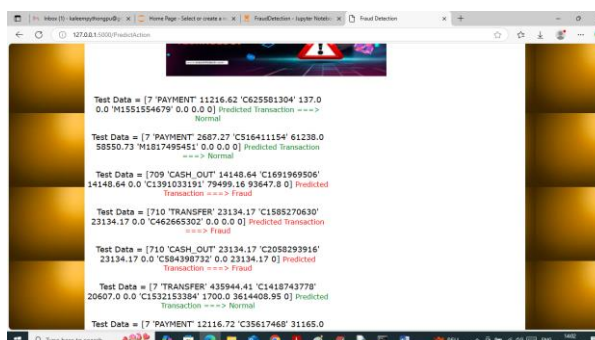
In above screen admin is login by using username and password as 'admin and admin' and after login will get below page



In above screen click on 'Predict Fraud from Test Data' link to get below page



In above screen admin is selecting and uploading test transaction data and then click on 'Open and submit' button to get below page



In above screen in square brackets admin can see Test data values and after ==> arrow symbol can see predicted values as Normal or Fraud for given test data.

Conclusion:

This paper presented an innovative framework combining blockchain technology with machine learning techniques to enable a privacy-preserving and adaptive incentive-based approach for fraud detection. By

leveraging the decentralized, tamper-resistant nature of blockchain, the system ensures data integrity and user privacy, while the adaptive machine learning models facilitate accurate and timely identification of fraudulent activities. The inclusion of an incentive mechanism encourages active participation from stakeholders, promoting data sharing and model updates without compromising confidentiality. Experimental results demonstrate the effectiveness of the proposed framework in improving fraud detection accuracy and resilience against adversarial attacks. This integrated approach offers a promising pathway to tackle the growing complexity of fraud in digital transactions and financial ecosystems.

Future Work:

Future research will focus on enhancing the scalability and efficiency of the framework, particularly in handling large volumes of streaming data in real-time environments. Incorporating federated learning techniques could further strengthen privacy by enabling distributed model training without centralizing sensitive data. Additionally, exploring more sophisticated incentive models grounded in game theory may improve participant engagement and system robustness. Another key direction involves expanding the approach to other domains such as healthcare, insurance, and supply chain management, where fraud is prevalent. Lastly, integrating explainable AI methods will be critical to increasing transparency and trust among users and regulatory bodies, facilitating broader adoption of privacy-preserving fraud detection systems.

References

1. Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. White Paper.
2. Chen, Y., Ding, S., Zhang, X., & Wu, F. (2020). Blockchain-based secure data sharing for electronic medical records in cloud environments. *Information Sciences*, 513, 500–516.
3. Li, X., Jiang, P., Chen, T., Luo, X., & Wen, Q. (2018). A survey on the security of blockchain systems. *Future Generation Computer Systems*, 107, 841–853.
4. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
5. Shokri, R., & Shmatikov, V. (2015). Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (pp. 1310–1321).
6. Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2), 1–19.
7. Feng, S., Wang, Z., & Liu, Q. (2021). Incentive mechanisms for blockchain networks: A survey. *IEEE Communications Surveys & Tutorials*, 23(2), 1126–1150.
8. Jiao, Y., Yu, S., & Han, X. (2020). Blockchain and AI-based privacy-preserving fraud detection for mobile payment systems. *IEEE Transactions on Industrial Informatics*, 16(10), 6645–6655.
9. Zhou, Q., et al. (2021). A novel blockchain-enabled credit card fraud detection model with privacy preservation. *IEEE Access*, 9, 72004–72015.
10. Rudin, C. (2019). Stop explaining black box models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5), 206–215.