AI/ML

Hybrid AI Model for Online Payment Fraud Detection Using Machine Learning

Mahesh, Guru Basava, R. Sudharshan, J A. Mahesha, Bhagyashri Wakde

Department of Computer Science and Engineering, Rajiv Gandhi Institute of Technology, Bengaluru, Karnataka, India

Published November 2026

Abstract

The rapid growth of digital payments has been accompanied by a sharp rise in online payment fraud, creating serious financial and trust-related risks for users, merchants, and financial institutions. Traditional rule-based fraud detection systems depend on predefined thresholds and manually maintained rules, and therefore struggle to keep pace with evolving fraud patterns, producing high false-positive rates and delayed detection. This paper presents a Hybrid AI Model for Online Payment Fraud Detection that combines multiple machine learning classifiers within a single decision framework. The proposed system integrates Support Vector Machine (SVM), Random Forest, and XGBoost classifiers, whose outputs are fused by a soft-voting ensemble to produce a more balanced and reliable final prediction. The severe class imbalance inherent in transaction data is handled using the Synthetic Minority Over-sampling Technique (SMOTE), and numerical features are standardized before training. The framework is further designed to accommodate unsupervised anomaly detection (Isolation Forest and Autoencoders) and expert-defined business rules for detecting novel and known fraud types respectively. The system is implemented as a full-stack web application using a Flask backend, a MySQL database, and an HTML, CSS, JavaScript, and Bootstrap frontend, providing user registration, login, transaction submission, real-time fraud prediction with confidence score, and prediction history. Models were trained on a publicly available transaction dataset of more than six million records and evaluated using accuracy, precision, recall, F1-score, and confusion matrices. The individual models achieved accuracies between 99.87% and 99.93%, and the voting classifier achieved 99.906% on the evaluation subset. The system offers a lightweight, practical, and scalable alternative to computationally heavy deep-learning approaches for secure digital payment environments.

Keywords: online payment fraud; hybrid AI model; machine learning; ensemble learning; SMOTE; voting classifier; class imbalance; XGBoost

1 Introduction

The adoption of internet banking, mobile wallets, card-not-present payments, and instant transfer systems has transformed commerce, but it has also expanded the attack surface available to fraudsters. Online payment fraud now poses a major threat to individuals, businesses, and financial institutions, leading to direct monetary loss, regulatory exposure, and erosion of customer confidence.

Conventional fraud detection systems are largely rule-based. They apply predefined thresholds and manually defined conditions to historical transaction data and flag transactions that violate them. Although such systems are transparent and easy to deploy, they are rigid: they cannot recognize new or previously unseen fraud schemes, they require frequent manual updating, and they generate large numbers of false alerts that disrupt genuine customers and overload analysts.

Machine learning (ML) offers a data-driven alternative. By learning statistical patterns from labelled transaction histories, ML classifiers can capture complex, non-linear relationships between transaction attributes and fraudulent behaviour. However, no single algorithm performs best under every condition. Fraud datasets are extremely imbalanced, with fraudulent records typically forming a tiny fraction of all transactions, and individual models tend to trade precision against recall differently.

The **Hybrid AI Model for Online Payment Fraud Detection** is proposed to address these issues. It combines several ML techniques with a rule-aware, extensible architecture so that the strengths of one model compensate for the weaknesses of another. The central objective is not merely to train another classifier, but to build a complete, deployable system that spans data preprocessing, model training, ensemble prediction, secure storage, and user-facing fraud alerts.

For example:

- Preprocessing converts raw transaction records into clean, scaled, and balanced training data.
- SVM, Random Forest, and XGBoost each learn a different view of fraud behaviour.
- The soft-voting classifier fuses their probabilities into one final decision.
- The Flask backend exposes the trained model to users in real time.
- The MySQL database records every transaction and prediction for traceability.
- The alert and history modules inform users about suspicious activity.

Thus, the system establishes a continuous workflow between **data preparation, model learning, ensemble decision-making, and user-facing fraud reporting**.

2 Problem statement

Online payment systems are an integral part of modern commerce but are increasingly targeted by fraudulent activity. Existing detection approaches suffer from several limitations:

1. **High False Positives** — Rule-based systems raise numerous false alerts, causing unnecessary investigations and disrupting the user experience.
2. **Limited Scalability** — As transaction volumes grow, static rule sets cannot scale effectively, leading to delayed detection and response.
3. **Inability to Adapt** — Rigid rules cannot adjust to new fraud tactics without manual intervention.
4. **Dependence on Predefined Rules** — Novel or previously unseen fraud types are not detected.
5. **Poor Real-Time Detection** — Reliance on batch processing and static rules delays the identification of fraudulent activity.
6. **Lack of Transparency** — Analysts often cannot understand why a particular transaction was flagged.
7. **Class Imbalance** — Fraudulent transactions form a very small proportion of the data, biasing standard classifiers towards the majority class.
8. **Single-Model Weakness** — Any single classifier shows a trade-off between precision and recall and may miss certain fraud patterns.

There is therefore a critical need for a robust, adaptive, and intelligent solution that can accurately detect and prevent fraudulent transactions in real time. The proposed Hybrid AI Model addresses these limitations by integrating multiple machine learning techniques into one system.

3 Objectives

The primary objectives of the proposed system are:

1. To preprocess and analyze online transaction datasets in order to extract significant features that distinguish legitimate from fraudulent transactions.
2. To handle severe class imbalance using SMOTE and to standardize features for stable model training.
3. To design and implement a hybrid AI model by integrating multiple machine learning classifiers for enhanced fraud detection accuracy.
4. To combine the predictions of SVM, Random Forest, and XGBoost through a soft-voting ensemble.
5. To evaluate and validate the developed models using standard performance metrics such as accuracy, precision, recall, F1-score, and confusion matrices.
6. To build a web-based application supporting user registration, login, transaction submission, and fraud prediction.
7. To store users, transactions, and prediction results in a database for history tracking and auditing.
8. To generate fraud alerts and prediction reports for users.
9. To build a scalable and reliable framework suitable for deployment in real-time payment environments.

4 Literature survey

Machine learning techniques have been widely applied to online payment fraud detection, addressing challenges such as imbalanced datasets, accuracy, and real-time detection. Sarna et al. [1] presented a comprehensive systematic review of AI-driven fraud detection in financial networks, highlighting key models, open challenges, and future research directions.

Lee et al. [2] explored deep learning combined with Support Vector Machines and showed that the hybrid approach improves classification accuracy by capturing complex transaction patterns. Smith et al. [3] proposed a novel hybrid model that combines multiple algorithms, handles imbalanced datasets effectively, and adapts to evolving fraudulent behaviour. Wang et al. [4] applied Gradient Boosting together with Neural Networks and reported improved prediction accuracy and robustness over single-model methods.

Zhang et al. [5] enhanced fraud detection in online payment systems by integrating multiple classifiers with feature-learning methods. A comparative study of Neural Networks and SVM [6] found that neural networks achieve higher accuracy on complex data, whereas SVM performs well on smaller datasets. A real-time detection system based on hybrid machine learning models [7] reduced false positives by integrating several classifiers.

A systematic review of deep learning in payment fraud detection [8] analysed the effectiveness of deep models while noting their data-size and computational-cost challenges. Studies on Random

Forest and XGBoost for banking fraud [9] showed that XGBoost offers improved precision and recall on highly imbalanced data, and boosting-based approaches [10] improved accuracy by combining multiple weak classifiers.

The surveyed literature shows that hybrid and ensemble approaches consistently outperform single models, but many of them demand GPU resources, heavy tuning, or complex feature engineering. This motivates a lightweight yet accurate hybrid model built from classical machine learning algorithms.

5 Existing system

The existing systems for fraud detection in financial networks rely heavily on rule-based approaches that use predefined thresholds and manually defined rules, combined with basic anomaly detection on historical data. These systems are limited by their inability to adapt to evolving fraud patterns, high false-positive rates, slow detection, poor scalability with growing transaction volumes, frequent manual updates, and lack of transparency in their decisions. They are consequently inefficient for real-time monitoring and prevention in modern payment networks.

6 Proposed system

The proposed system is a machine learning-driven fraud detection solution that uses Support Vector Machine (SVM), Random Forest, XGBoost, and a Voting Classifier to detect fraudulent transactions. By learning from large transaction datasets, it identifies complex fraud patterns and classifies each incoming transaction as fraudulent or legitimate, together with a confidence score.

The framework is hybrid in two senses. First, it fuses heterogeneous supervised learners (a kernel method, a bagging ensemble, and a boosting ensemble) through soft voting. Second, its architecture is designed to be extended with unsupervised anomaly detectors such as Isolation Forest and Autoencoders for novel fraud, and with an expert-defined rule layer for known fraud types. The implemented prototype in this paper realizes the supervised soft-voting ensemble.

The system is organized around a **modular architecture** in which preprocessing, model prediction, storage, and alerting are independent components. The major capabilities include user registration and login, transaction input, ML-based prediction, model information and accuracy display, fraud alerts, prediction history, and report viewing. The key advantages are:

- High accuracy with fewer false positives and false negatives
- Real-time fraud detection
- Adaptability through retraining on new data
- Scalability to large transaction volumes
- Reduced operational cost through automation

7 System architecture

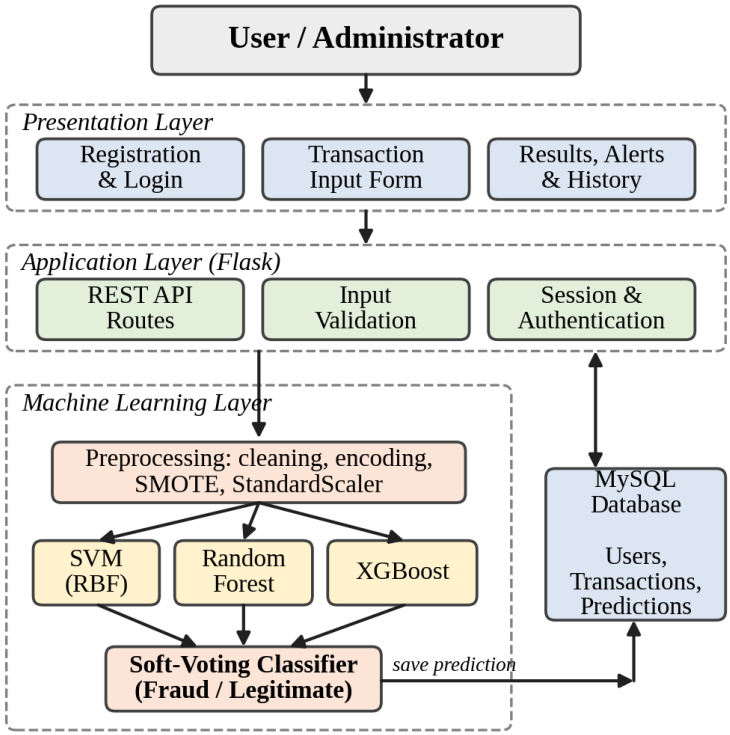


Figure 1. Overall system architecture of the hybrid AI model for online payment fraud detection.

The architecture consists of a presentation layer, an application layer, a machine learning layer, and a data layer. The user interacts through a Bootstrap-based web interface; requests are handled by Flask REST endpoints, which pass validated inputs to the preprocessing pipeline and then to the trained hybrid model. Prediction results are returned to the user and simultaneously stored in the MySQL database.

8 Technology stack

8.1 Frontend

- HTML5
- CSS3
- JavaScript
- Bootstrap (responsive layouts)

8.2 Backend

- Python 3.x
- Flask (REST API development)
- Joblib (model and scaler persistence)

8.3 Database

MySQL, hosted through the XAMPP server, is used as the persistent database for user records, transaction details, and prediction results. It supports quick retrieval of past transactions, enabling fraud tracking and system logging.

8.4 Machine learning libraries

- NumPy and Pandas for data handling
- Scikit-learn for SVM, Random Forest, Voting Classifier, scaling, and metrics
- XGBoost for gradient boosting
- Imbalanced-learn (SMOTE) for class balancing
- Matplotlib and Seaborn for visualization
- PyTorch (Torch) for extension to deep and autoencoder models

8.5 Development and hardware environment

Visual Studio Code is used as the IDE, and the system runs on Windows 7/8/10 or Linux. The minimum hardware requirements are an Intel Core i3 (or equivalent) processor, 8 GB RAM or more for faster model training, and 128 GB HDD/SSD storage.

9 Dataset and preprocessing

The models are trained on a publicly available online transaction dataset containing more than six million records. Each record contains the transaction time step (1 step = 1 hour, range 0–743), transaction type, transaction amount, originating account balance before and after the transaction, destination account balance before and after the transaction, and a binary fraud label. Fraudulent records form only a tiny fraction of the data.

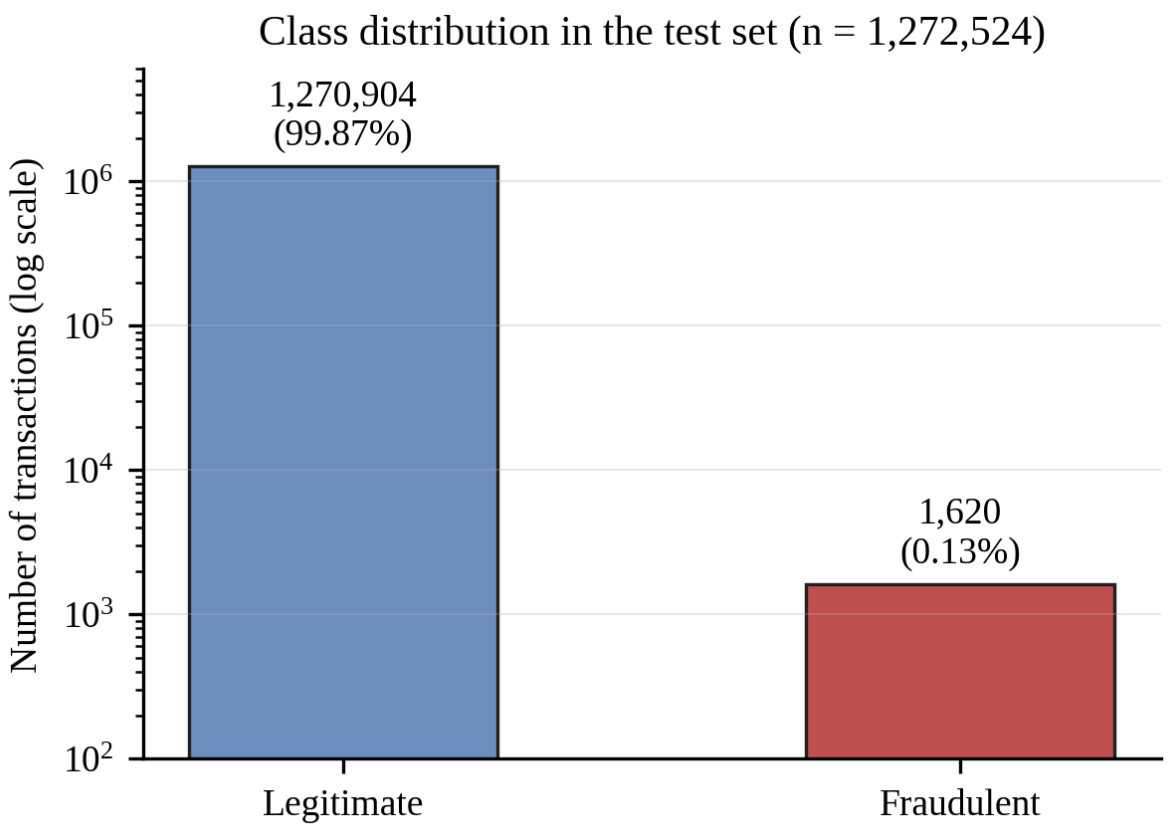


Figure 2. Class distribution of fraudulent and legitimate transactions in the dataset.

The preprocessing module performs the following operations:

1. Data cleaning — removal of missing values, duplicates, and irrelevant fields.
2. Feature encoding — categorical transaction types are converted into numeric form.
3. Train–test split — the data is split into training and testing sets (70:30 split with a fixed random state in the reference implementation).
4. Imbalance handling — SMOTE generates synthetic minority samples in the training set only.
5. Feature scaling — StandardScaler is fitted on the training data and applied to both training and test sets.

The processed dataset is then passed to the machine learning module for model training.

10 System modules

The fraud detection system is divided into five modules:

1. **User Interface Module** — an interactive web interface for registration, login, entry of transaction details, and display of prediction results, fraud status, and past transactions.
2. **Data Preprocessing Module** — cleaning, encoding, scaling, and SMOTE balancing of the dataset.
3. **Machine Learning Model Module** — the core module that trains and executes SVM, Random Forest, XGBoost, and the Voting Classifier.

4. **Database Storage and Retrieval Module** — storage of user records, transactions, and predictions in MySQL for retrieval and logging.
5. **Fraud Alert and Reporting Module** — generation of an alert whenever a transaction is predicted as fraudulent, and access to fraud analysis and reports.

11 Machine learning models

11.1 Support vector machine (SVM)

SVM constructs an optimal separating hyperplane in a transformed feature space. A Radial Basis Function (RBF) kernel with probability estimates enabled is used so that its output can participate in soft voting. SVM performs well on smaller, high-dimensional datasets.

11.2 Random Forest

Random Forest is a bagging ensemble of decision trees trained on bootstrapped samples with random feature selection. The implementation uses 150 estimators. It is robust to noise and handles non-linear relationships and outliers well.

11.3 XGBoost

XGBoost is an optimized gradient boosting algorithm that builds trees sequentially, each correcting the errors of the previous ones. It is evaluated with the logarithmic loss metric and is known for speed, regularization, and strong performance on imbalanced tabular data.

11.4 Hybrid soft-voting classifier

The hybrid model combines SVM, Random Forest, and XGBoost in a Voting Classifier using soft voting. Instead of counting class votes, soft voting averages the predicted class probabilities of the base models and selects the class with the highest averaged probability. This produces a more accurate and reliable final prediction than any individual model and leverages the strengths of multiple algorithms.

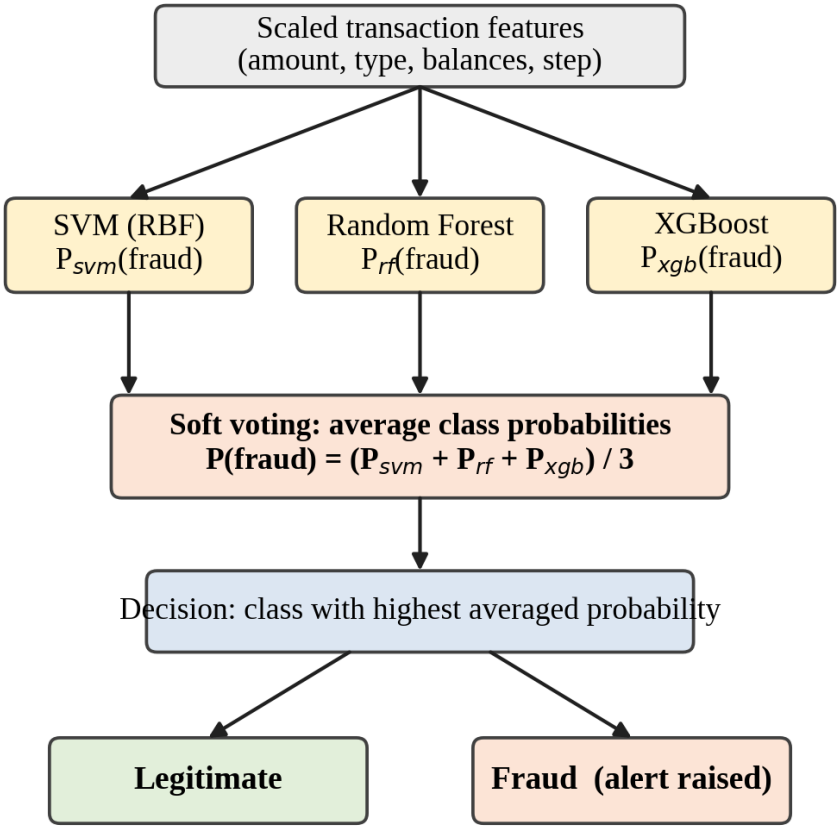


Figure 3. Hybrid soft-voting classifier combining SVM, Random Forest, and XGBoost.

12 Implementation

The system is implemented in the following stages:

1. **Project setup** — creating the Python environment, installing Flask, Pandas, NumPy, Scikit-learn, XGBoost, and imbalanced-learn, and organizing folders for models, routes, templates, static files, and database configuration. A MySQL database stores users, transactions, and predictions.
2. **Model definition** — Python implementations of SVM, Random Forest, XGBoost, and the Voting Classifier that analyze features such as transaction amount, transaction type, and sender and receiver balances.
3. **Input handling** — HTML forms collect the transaction amount, type, and old/new account balances. Every field is validated so that only correctly formatted data reaches the model.
4. **Backend logic** — Flask view functions receive inputs, apply scaling and encoding, obtain the prediction from the trained model, return the result, and save it to MySQL.
5. **Templates and routes** — Bootstrap templates for login, registration, fraud prediction, model information, and prediction history, connected using Flask app.route() decorators.
6. **Execution** — the database is initialized and the Flask server is started, enabling all pages to be tested locally.

The algorithm of the proposed hybrid model is summarized below.

Algorithm 1. Hybrid fraud detection

1. Load the transaction dataset and separate features X and label y.
2. Split into training and test sets.
3. Apply SMOTE to the training set to balance classes.
4. Fit StandardScaler on the training set; transform training and test sets.
5. Train SVM (RBF), Random Forest (150 trees), and XGBoost on the balanced training data.
6. Build a soft-voting classifier from the three trained models and fit it.
7. For each new transaction, scale the input and obtain the averaged class probabilities.
8. Output Fraud if the fraud probability is the highest; otherwise output Legitimate.
9. Store the transaction, prediction, and confidence score in the database and raise an alert if fraud is predicted.

The trained hybrid model and scaler are serialized with Joblib and loaded by a Flask /predict endpoint that accepts transaction data in JSON format and returns the fraud prediction.

13 System design diagrams

The behaviour of the system is modelled using standard UML and data-flow diagrams.

13.1 Activity diagram

The activity diagram shows the flow from system access, through registration or login, to the dashboard, where the user can view data, upload a dataset, view results, or run fraud detection before logging out.

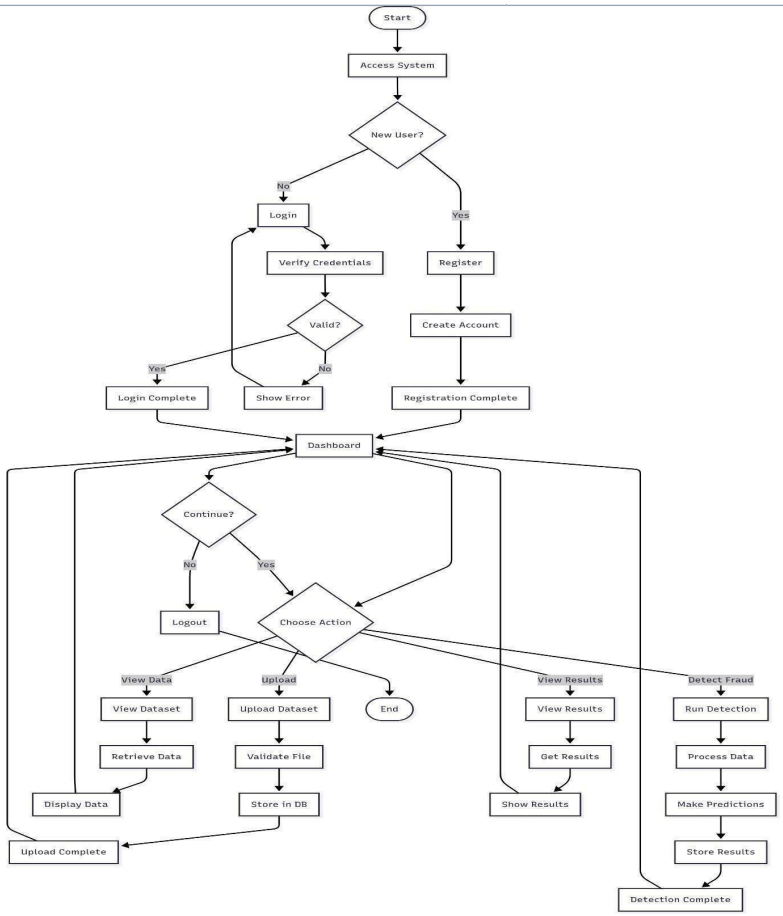


Figure 4. Activity diagram of the fraud detection system.

13.2 Use case diagram

The key actors are the **User**, who submits transactions and receives results, and the **Admin**, who manages user accounts, reviews fraud reports, and monitors system performance. Major use cases are Submit Transaction, View Fraud Detection Results, Manage User Accounts, and Generate Reports.

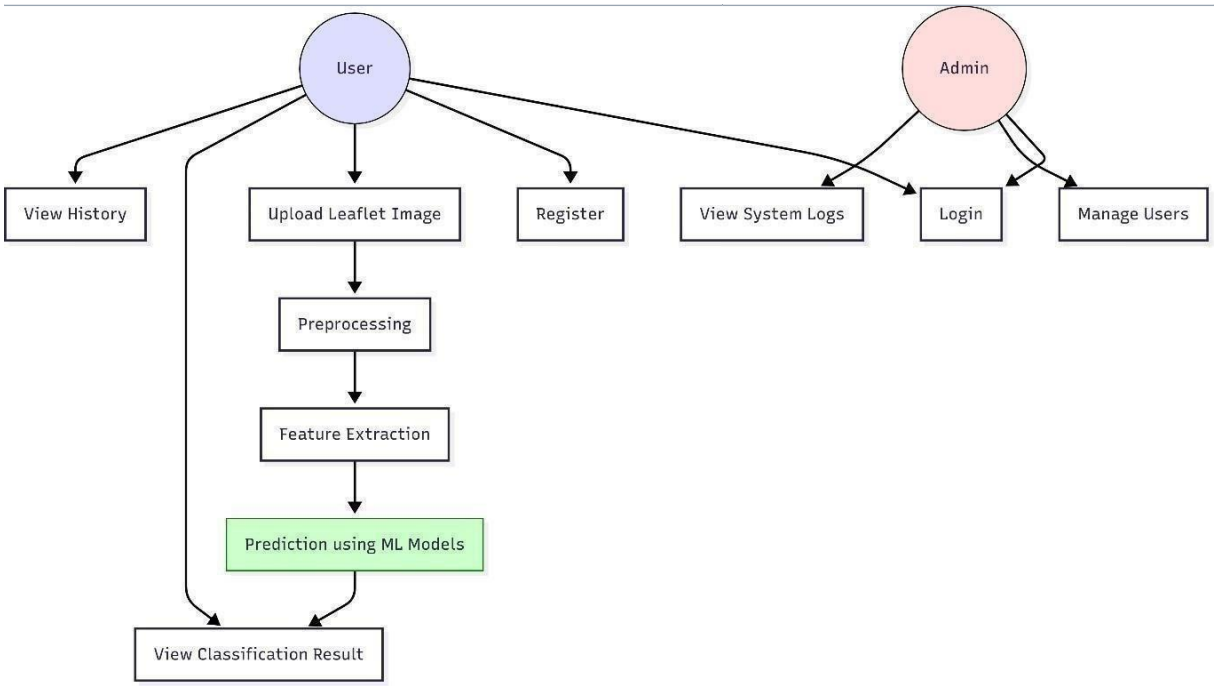


Figure 5. Use case diagram showing User and Admin interactions.

13.3 Data flow diagram

The data flow diagram shows the processes Authenticate User, Process Transaction, Analyze for Fraud, and Generate Reports, with data stores for users, transactions, ML models, and fraud results.

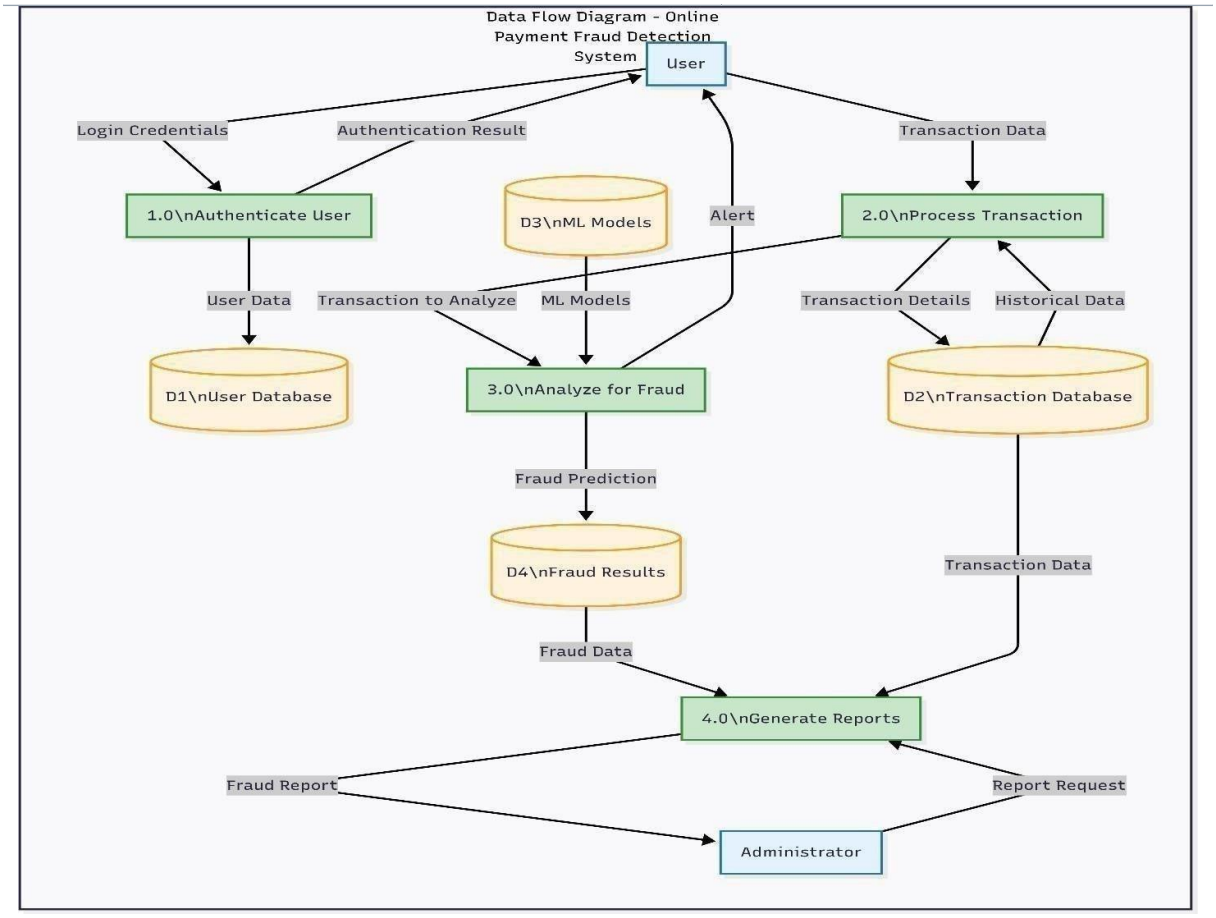


Figure 6. Data flow diagram of the online payment fraud detection system.

13.4 Sequence diagram

The sequence diagram depicts the ordered interaction among the User, System, Database, and ML Model for registration, login, dataset upload, prediction, and result retrieval.

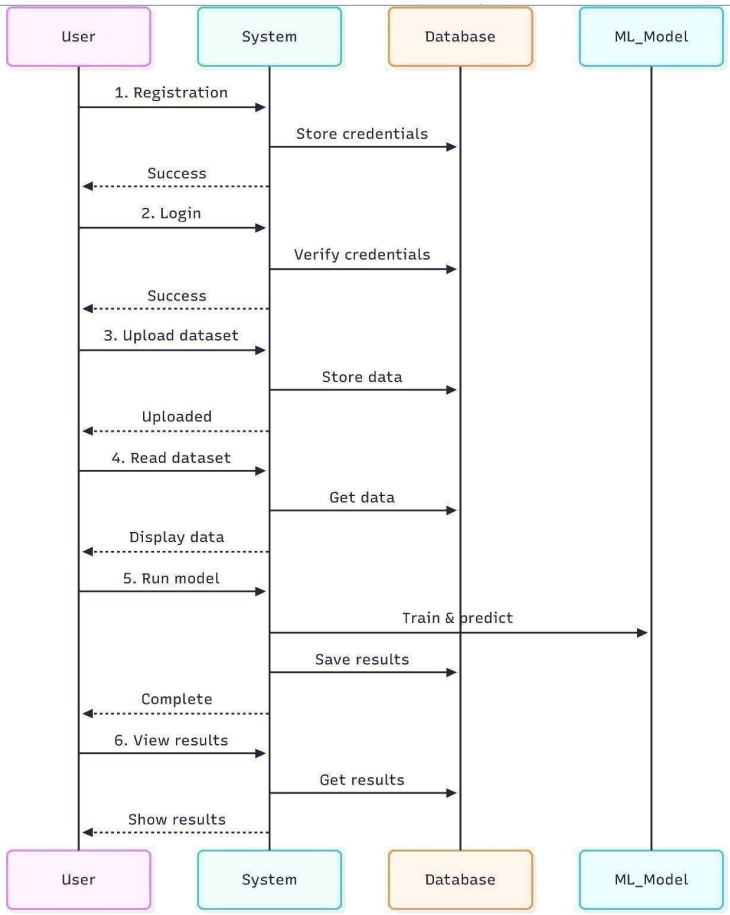


Figure 7. Sequence diagram of user, system, database, and ML model interaction.

14 User interface design

The interface follows a clean, responsive Bootstrap design with a consistent navigation bar (Home, About, Model, Fraud Detection, Prediction History, Logout), clear form labels, and a dedicated result panel. The main screens are described below.

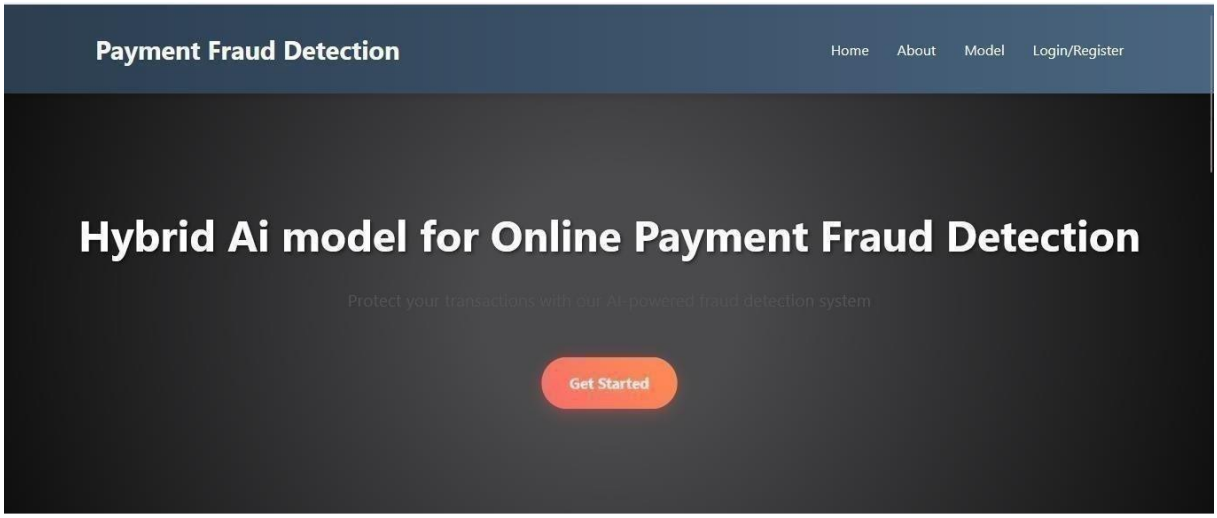


Figure 8. Homepage of the payment fraud detection system.

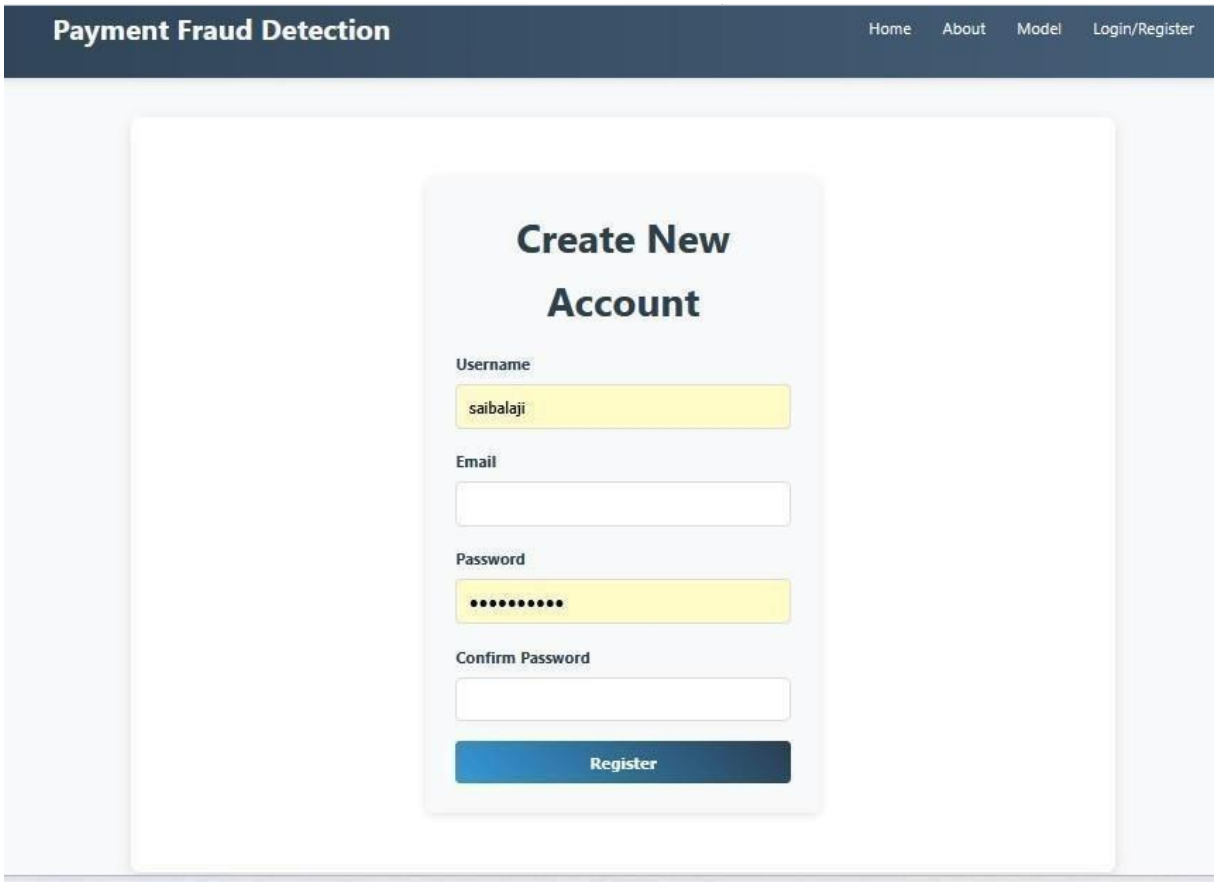


Figure 9. User registration page.

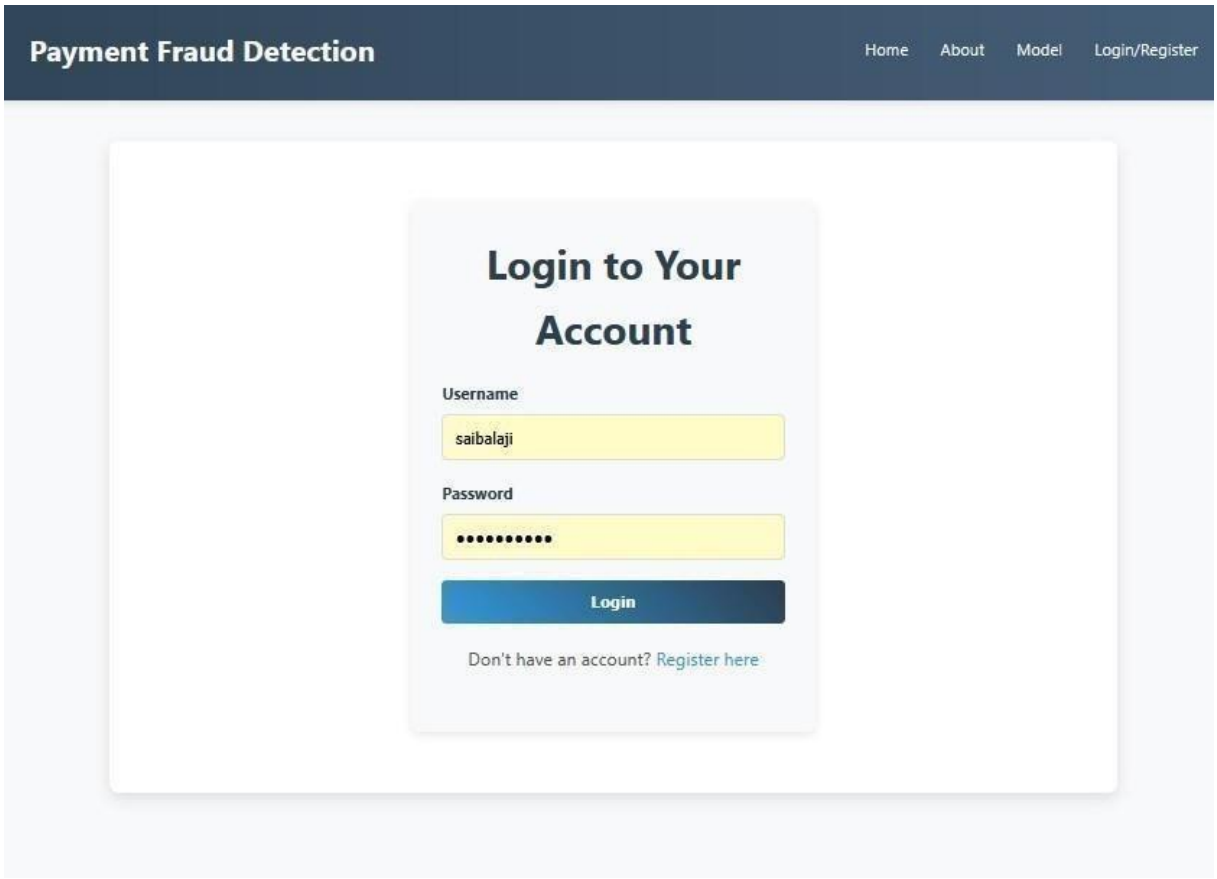


Figure 10. Login page.

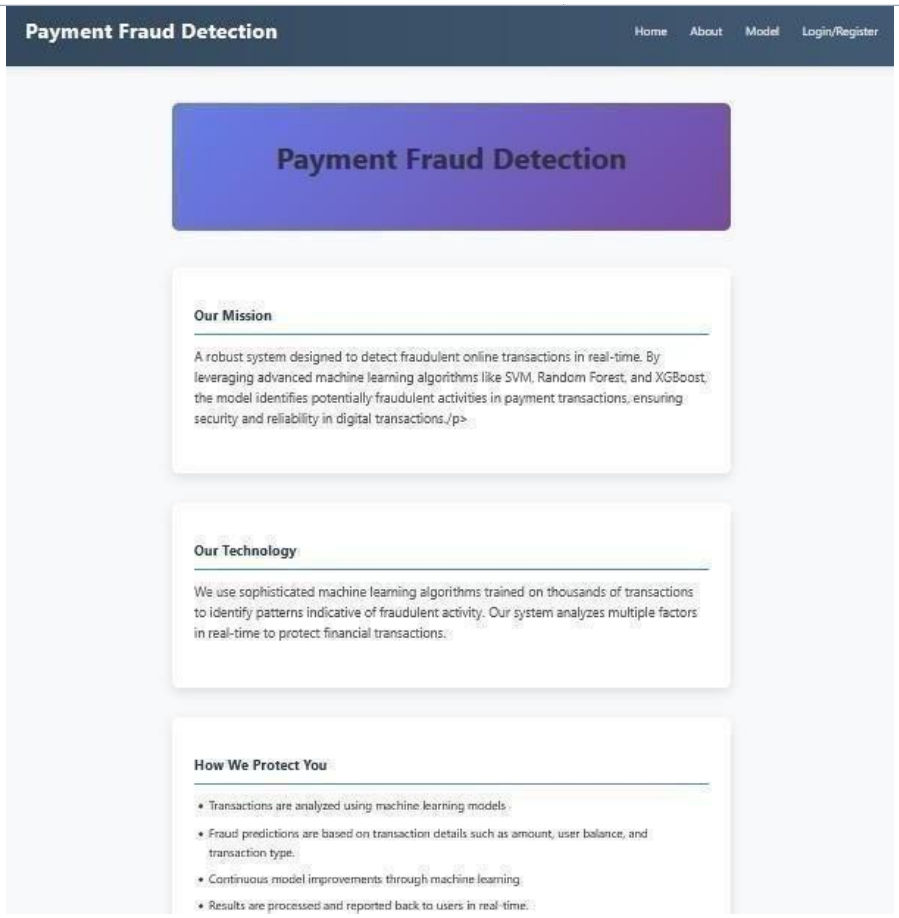


Figure 11. About page – system overview.

After login, the user can access the **Transaction Input Form**, which collects the time step, transaction amount, original and new balances, destination balances before and after the transaction, and the transaction type. The **Model page** lets the user select a model (SVM, Random Forest, XGBoost) and displays its accuracy and a short description. The **Prediction History** page lists the date and time, prediction, confidence, and model used for previous predictions, with filtering and pagination.

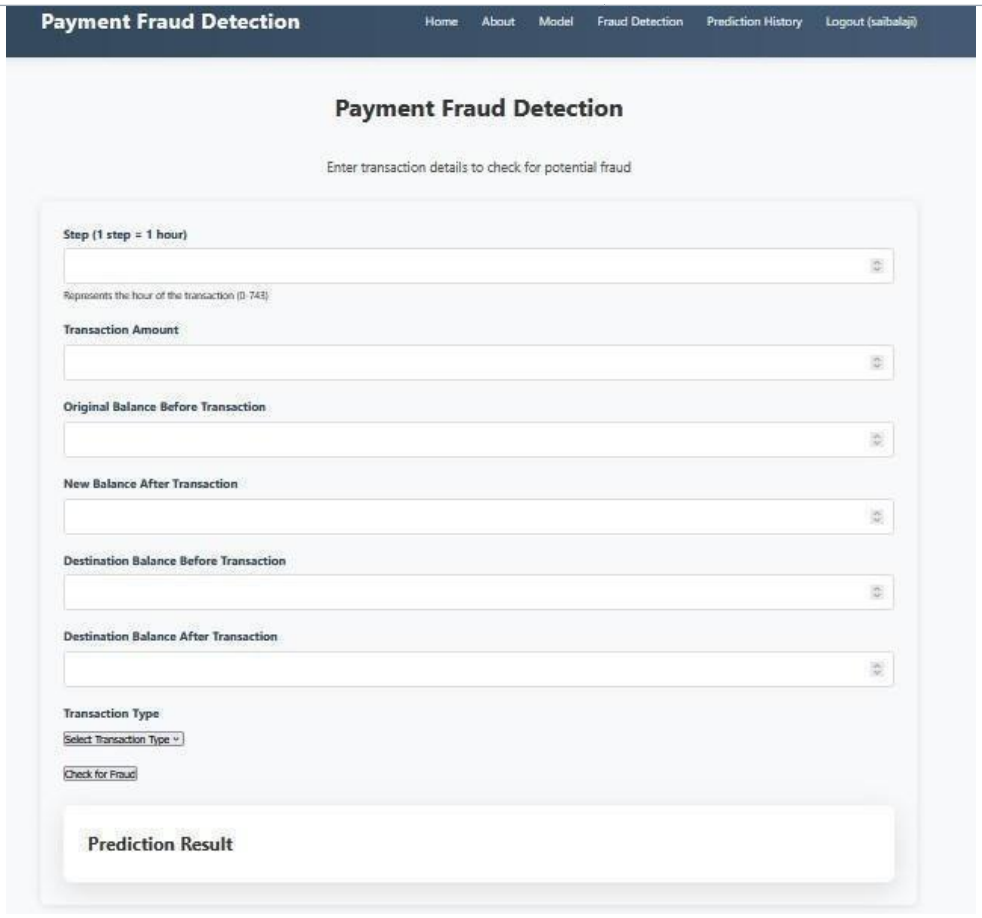


Figure 12. Transaction input form – fraud detection module.

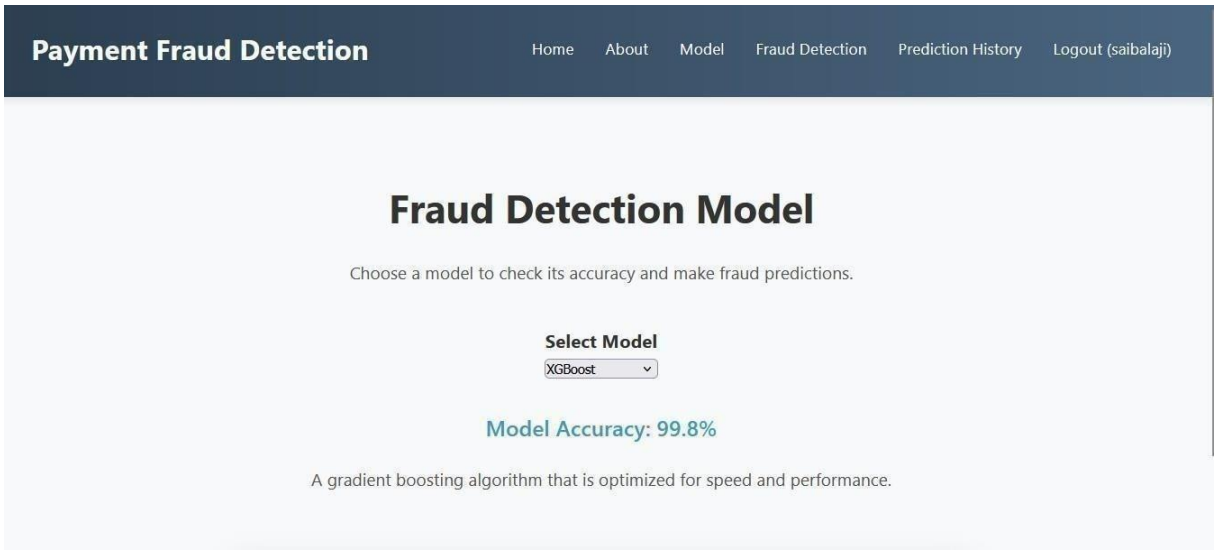


Figure 13. Fraud detection model selection interface.

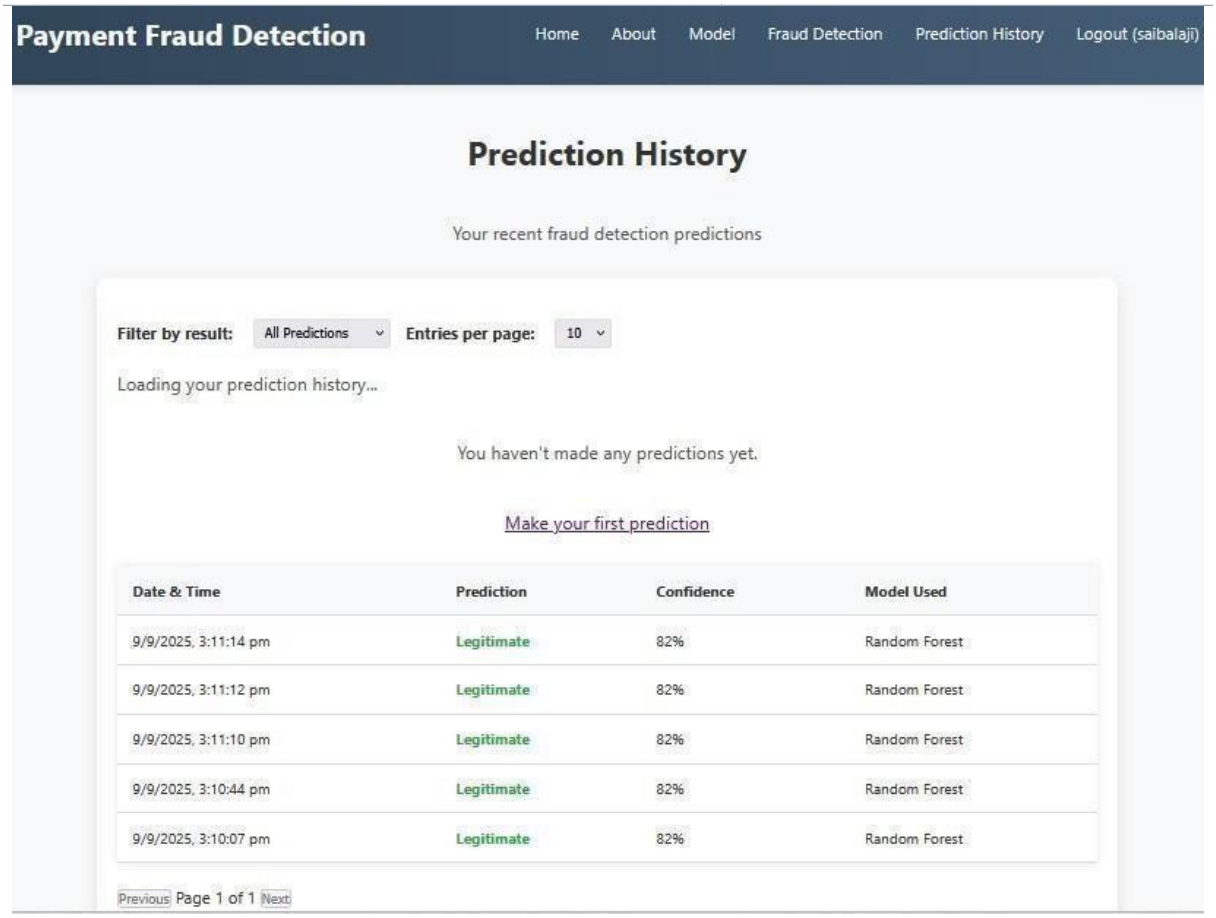


Figure 14. Prediction history page.

15 Security and data handling

Because the system processes user credentials and financial transaction information, security is an integral design concern.

15.1 Authentication

Users must register and log in before accessing prediction and history features; invalid credentials are handled with appropriate error messages.

15.2 Input validation

Each input field is validated so that only correctly formatted and legitimate data is processed, preventing invalid or malicious inputs from affecting predictions.

15.3 Data storage

User records, transactions, and predictions are stored in a MySQL database, and past results are retrieved per user so that histories remain separated.

15.4 Deployment security

For cloud deployment (AWS, Heroku, or DigitalOcean), production configuration such as allowed hosts, database credentials, and static file handling must be updated, and SSL, load balancers, and environment variables should be used.

16 Testing and validation

Testing verifies that every component works correctly and that the integrated system meets its functional and non-functional requirements. Each module — authentication, preprocessing, fraud prediction, transaction storage, and history retrieval — was tested independently and as part of the integrated application. Unit testing covered preprocessing, model training, and authentication, while integration testing verified communication between the backend, ML models, and database.

Functional testing confirmed that registration, login, fraud prediction, transaction history, and fraud alerts operate according to specification. **Performance testing** evaluated execution speed, inference time, and response time to confirm that decisions are made in near real time. The executed test cases are given in Table 1.

Table 1. Test cases and results.

No	Test	Input	Expected Output	Result
1	User Registration	Name, Email, Password	Registration successful message	Pass
2	User Login	Email and Password	Login success or error message	Pass
3	Fraud Prediction	Type, amount, balances	Fraud / Legitimate prediction	Pass
4	Transaction Storage	Details + prediction	Data saved in database	Pass
5	Prediction History	User ID	Accurate history displayed	Pass

17 Results and findings

The models were evaluated using accuracy, precision, recall, F1-score, and confusion matrices. Because fraud is rare, accuracy alone is misleading; fraud-class precision, recall, and F1-score are therefore emphasized.

17.1 Full test-set evaluation

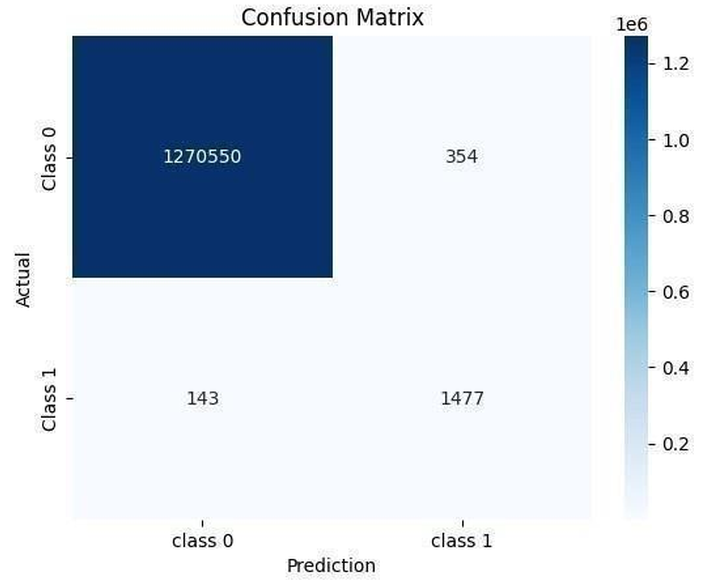
On the full test set of 1,272,524 transactions (1,270,904 legitimate and 1,620 fraudulent), Random Forest correctly identified 1,477 of 1,620 fraud cases (recall 0.91) with 354 false positives (precision 0.81, F1 0.86). XGBoost detected 1,591 of 1,620 fraud cases (recall 0.98) at the cost of 1,464 false positives (precision 0.52, F1 0.68), achieving an overall accuracy of 99.88%.

Table 2. Confusion matrix values (full test set).

Model	TN	FP	FN	TP	Accuracy
Random Forest	1,270,550	354	143	1,477	99.96%
XGBoost	1,269,440	1,464	29	1,591	99.88%

Table 3. Fraud-class metrics (full test set).

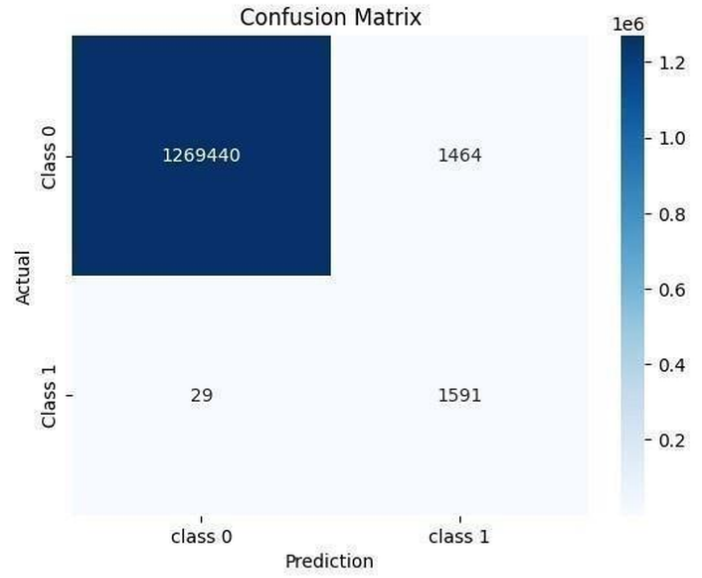
Model	Precision (Fraud)	Recall (Fraud)	F1 (Fraud)
Random Forest	0.81	0.91	0.86
XGBoost	0.52	0.98	0.68



Random Forest Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1270904
1	0.81	0.91	0.86	1620
accuracy			1.00	1272524
macro avg	0.90	0.96	0.93	1272524
weighted avg	1.00	1.00	1.00	1272524

Figure 15. Random Forest confusion matrix and classification report.



Accuracy Score for XGBClassifier : 0.9988267411852351

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1270904
1	0.52	0.98	0.68	1620
accuracy			1.00	1272524
macro avg	0.76	0.99	0.84	1272524
weighted avg	1.00	1.00	1.00	1272524

Figure 16. XGBoost confusion matrix and classification report.

17.2 Comparison of individual models and voting classifier

In a further evaluation on a 100,000-record test subset (99,866 legitimate and 134 fraudulent transactions), all four models achieved very high accuracy, as shown in Table 4. XGBoost obtained the highest individual accuracy of 99.928%, followed by Random Forest (99.911%), the Voting Classifier (99.906%), and SVM (99.878%). In terms of fraud-class behaviour, SVM and Random Forest showed perfect precision but low recall (0.09 and 0.34), whereas XGBoost offered a better balance (precision 0.91, recall 0.51, F1 0.66).

Table 4. Model comparison on 100,000-record test subset.

Model	Accuracy	Prec. (Fraud)	Recall (Fraud)	F1 (Fraud)
SVM	99.878%	1.00	0.09	0.16
Random Forest	99.911%	1.00	0.34	0.50
XGBoost	99.928%	0.91	0.51	0.66
Voting Classifier	99.906%	—	—	—

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

           0       1.00      1.00      1.00     99866
           1       1.00      0.09      0.16       134

    accuracy                           1.00    100000
   macro avg       1.00      0.54      0.58    100000
weighted avg       1.00      1.00      1.00    100000

Random Forest Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     99866
           1       1.00      0.34      0.50       134

    accuracy                           1.00    100000
   macro avg       1.00      0.67      0.75    100000
weighted avg       1.00      1.00      1.00    100000

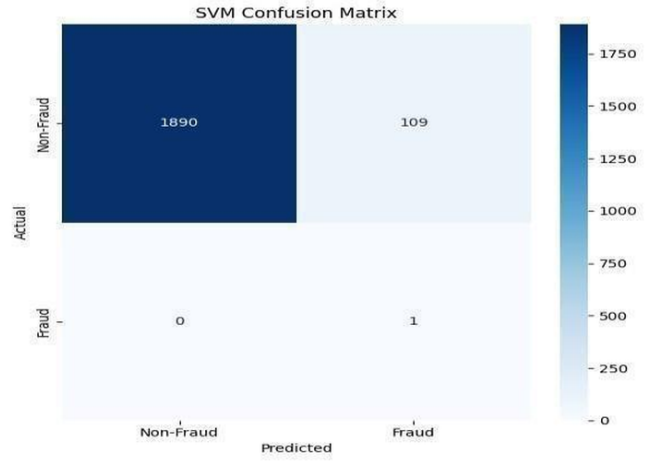
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     99866
           1       0.91      0.51      0.66       134
...
SVM Accuracy: 0.99878
Random Forest Accuracy: 0.99911
XGBoost Accuracy: 0.99928
Voting Classifier Accuracy: 0.99906
```

Figure 17. Comparison of model performance metrics (SVM, Random Forest, XGBoost, Voting Classifier).

17.3 SVM on small evaluation sample

An initial SVM evaluation on a small sample of 2,000 transactions produced an accuracy of 94.55%, with recall of 0.97 and macro precision of 0.50. Only one fraudulent transaction was present in this sample, so the result is indicative only and is not used for model comparison.




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

           0       1.00      0.95      0.97      1999
           1       0.01      1.00      0.02         1

    accuracy                           0.95      2000
   macro avg       0.50      0.97      0.49      2000
weighted avg       1.00      0.95      0.97      2000

SVM Confusion Matrix:
 [[1890  109]
 [   0    1]]
SVM Accuracy: 0.9455
SVM Precision: 0.5045
SVM Recall: 0.9727
SVM F1 Score: 0.4950
```

Figure 18. SVM confusion matrix and classification report.

17.4 Efficiency, reliability, and scalability

Core functions — login, transaction submission, prediction, and history retrieval — completed within short response times during testing, and no errors or crashes were observed across repeated test iterations, including edge-case and outlier transactions. Training on a dataset of more than six million transactions indicates that the approach can manage large data volumes. Large-scale stress and load testing was outside the current scope and is identified as future work.

Table 5. Comparison with existing fraud detection models.

Model / Study	Accuracy	Precision (Fraud)	Recall (Fraud)	F1 (Fraud)
Proposed Hybrid Model	99.87–99.91%	High (up to 1.00)	Improved	Balanced
Stacking Ensemble (2024–2025)	99.92%	99.93%	99.7%	99.95%
Hybrid CNN–LSTM (2024)	99.7–99.8%	98%	99%	99%
XGBoost + RF Hybrid (2023)	99.9%	99.2%	99.5%	99.3%
GA-Optimized RF (2023)	99.96%	99.40%	99.80%	99.60%

When compared with existing models, the proposed hybrid model offers a strong balance between accuracy and real-time performance. Although several recent research models achieve slightly higher accuracy, they often require complex architectures, GPU support, or heavy feature engineering. In contrast, the proposed model uses lightweight classical machine learning techniques to deliver high precision, improved recall, and a stable F1-score without expensive computation, making it faster and

easier to deploy in real-world online payment fraud detection systems. In particular, the stacking ensemble is more complex, the hybrid CNN–LSTM requires a GPU and deep learning, the XGBoost + RF hybrid needs heavy tuning, and the GA-optimized RF relies on complex feature engineering.

18 Discussion

Rule-based systems depend on fixed conditions, single-model ML systems depend on one view of the data, and deep-learning systems depend on heavy computation. The proposed approach connects these concerns by combining diverse classical learners with balanced training data and a lightweight web deployment.

The experiments show why a hybrid approach is valuable. SVM and Random Forest achieved perfect fraud-class precision but missed many fraud cases at default settings, while XGBoost detected more fraud at the cost of more false alarms. Soft voting mitigates such individual weaknesses by averaging probabilities, yielding stable overall accuracy. The trade-off between precision and recall is central in practice: a missed fraud (false negative) causes direct financial loss, whereas a false alert (false positive) causes customer inconvenience. The operating threshold of the ensemble can therefore be tuned to the risk appetite of the institution.

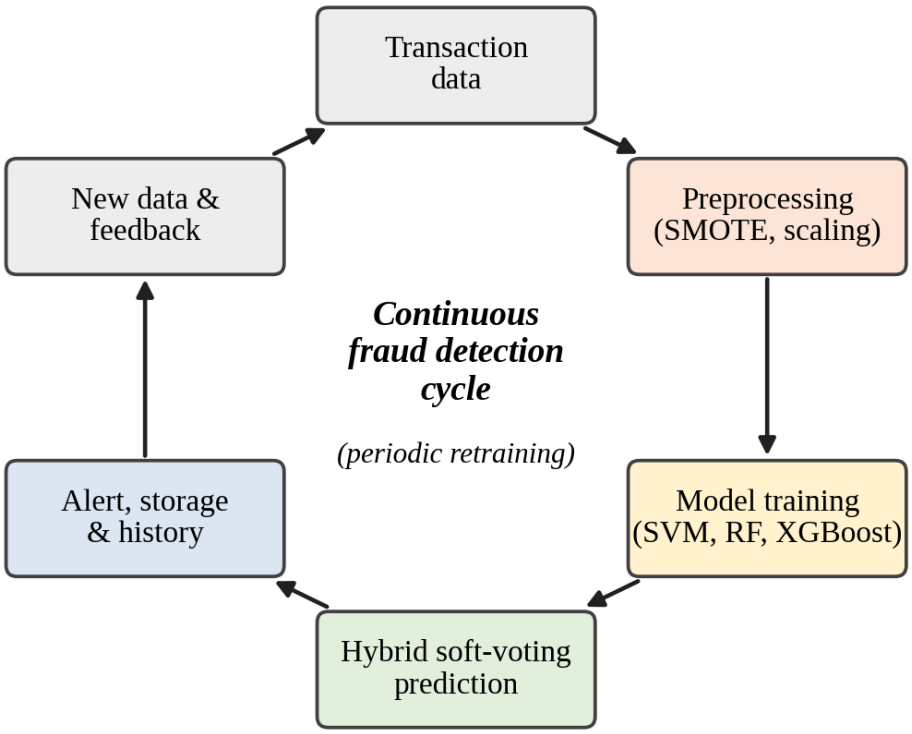


Figure 19. Continuous fraud detection cycle: data → training → prediction → feedback → retraining.

The cycle allows the platform to evolve as new transactions and fraud tactics appear, rather than treating every prediction as an isolated event.

19 Novelty and contribution

19.1 Lightweight hybrid ensemble

A soft-voting fusion of SVM, Random Forest, and XGBoost delivers high accuracy without GPU or deep-learning infrastructure.

19.2 Imbalance-aware pipeline

SMOTE and feature scaling are integrated into a reproducible preprocessing pipeline for highly skewed transaction data.

19.3 End-to-end deployable system

The work goes beyond model training to provide registration, login, prediction, confidence display, history, and alerting in one web application.

19.4 Multi-model transparency

Users can select a model and view its accuracy, and every prediction records the model used and its confidence.

19.5 Extensible architecture

The modular design allows unsupervised anomaly detectors, rule layers, and explainability tools to be added without redesigning the system.

20 Limitations

1. Fraud-class recall of some models is low at default settings because of extreme class imbalance.
2. SMOTE generates synthetic samples that may not perfectly represent real fraud behaviour.
3. The system was trained on a publicly available dataset; performance on live production data may differ.
4. Large-scale stress and load testing has not yet been performed.
5. The current prototype realizes the supervised ensemble; unsupervised and rule-based layers are planned extensions.
6. Model predictions require monitoring for concept drift as fraud tactics change.
7. Soft-voting decisions are less transparent than simple rules unless explainability tools are added.

21 Future enhancements

Potential future enhancements include:

- Deep learning models such as LSTM and CNN architectures for complex fraud patterns
- Real-time detection using streaming data-processing frameworks
- Concept drift detection and automatic model retraining
- Unsupervised anomaly detection using Isolation Forest and Autoencoders
- An expert-defined rule layer for known fraud types

- Explainable AI using SHAP and LIME
- Cloud deployment with SSL, load balancing, and monitoring
- Threshold tuning and cost-sensitive learning to balance precision and recall

These enhancements can be incorporated without changing the fundamental modular architecture.

22 Conclusion

The Hybrid AI Model for Online Payment Fraud Detection demonstrates the capability of machine learning techniques to identify fraudulent transactions with high accuracy and reliability. By integrating SVM, Random Forest, XGBoost, and a soft-voting classifier, the system achieves robust detection performance, reduces false positives, and improves detection of high-risk transactions.

The backend, developed using Flask and supported by a MySQL database, provides fast processing, secure data handling, and reliable storage, while the web interface lets users submit transactions, receive instant predictions, and review their history. The use of SMOTE and feature scaling addresses the severe class imbalance of fraud data.

The system therefore offers a scalable, accurate, and efficient fraud detection solution suitable for real-world deployment in online payment platforms. It enhances transaction security, improves user trust, and contributes to minimizing financial risk for both users and businesses. Its modular design provides a foundation for deep learning, real-time streaming, concept drift handling, and explainable AI in future work.

Acknowledgments

The authors would like to express their sincere gratitude to the Department of Computer Science and Engineering, Rajiv Gandhi Institute of Technology, Bengaluru, for providing the academic environment and resources required to carry out this project. The authors thank the Principal, the Head of the Department, the project coordinators, and the guide, Mrs. Bhagyashri Wakde, for their constant encouragement and technical guidance. The authors also acknowledge the support of the faculty, colleagues, and families who contributed to the successful completion of this work.

References

[1] Sarna NJ, Rithen FA, Jui US, Belal S, Amin A, Oishee TK, et al. AI driven fraud detection models in financial networks: A comprehensive systematic review. *IEEE Access*. 2025;13:141204–141228. https://doi.org/10.1109/ACCESS.2025.3596060

[2] Lee GR, Zhang TM, Liu HY. Fraud detection in online payments using deep learning and SVM models. *IEEE Transactions on Neural Networks*. 2022;12(4):125–134.

[3] Smith HR, Roberts PL, Gonzalez RA. A novel hybrid model for online payment fraud detection using machine learning. *Journal of Machine Learning Research*. 2021;21(7):345–360.

[4] Wang Y, Liu Q, Chen X. Financial fraud detection using gradient boosting and neural networks. *IEEE Access*. 2020;9:8975–8983.

[5] Zhang ZF, Liu X, Zhao YJ. Improving fraud detection in online payment systems using hybrid machine learning models. *Journal of Computational and Applied Mathematics*. 2021;40:233–245.

[6] S. SRAB. A comparative study of neural networks and SVM for financial fraud detection. *Journal of Financial Services Technology*. 2019;45(3):128–142.

[7] G. FHP. Real-time fraud detection for digital transactions using hybrid machine learning models. *International Journal of Artificial Intelligence*. 2020;24:321–334.

[8] L. ACZ. Deep learning for fraud detection in payment systems: Systematic review. *International Journal of Machine Learning & Cybernetics*. 2021;22:101–112.

[9] P. JGK. Fraud detection in banking transactions using random Forest and XGBoost. *Journal of Financial Engineering*. 2022;11(4):452–463.

[10] P. APR, V. SR. Boosting methods for online payment fraud detection. *Journal of Financial Risk Management*. 2020;30:211–225.