International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

Peer Reviewed, Referred & Indexed Journal

Research Paper

Credit Card Fraud Detection System Using Django and Machine Learning

Morri Sai Pravallika

Reg. No. 9240115

saipravalliika785@gmail.com

Department of Master of Computer Applications

SIR C R REDDY COLLEGE (AUTONOMOUS)

Eluru, Andhra Pradesh, India

Under the guidance of Mehaboob Karishma Assistant Professor

mehaboob.karishma@gmail.com

Abstract*—The Credit Card Fraud Detection System is designed to identify and prevent fraudulent credit card transactions using Machine Learning algorithms integrated with the Django framework. The project detects unusual transaction behaviour based on parameters such as transaction amount, merchant category, and transaction country. Fraudulent activities have become increasingly complex, and traditional detection systems often fail to identify evolving fraud patterns; this work therefore leverages data analytics and machine learning to improve the accuracy and speed of fraud detection. The system uses classification algorithms such as Logistic Regression, Decision Tree, and Random Forest, trained on a dataset containing both legitimate and fraudulent transactions, to predict the probability of a transaction being fraudulent in real time, with transactions exceeding a configured probability threshold flagged as suspicious. A Django web interface allows users and administrators to view flagged transactions, train models, and test new data, with role-based access control that supports three roles: Administrator, Analyst, and Reviewer, each with specific permissions. The trained model is serialized using Joblib or Pickle and integrated with Django for real-time prediction, and the application is structured in a presentation–application–data layered architecture. By combining machine learning with web-based delivery and role-based security, the project demonstrates a secure, efficient, and user-friendly platform that reduces false positives, improves detection accuracy, and automates the monitoring process while ensuring data privacy and scalability for real-world use.*

Keywords*—Credit Card Fraud Detection; Machine Learning; Django; Logistic Regression; Decision Tree; Random Forest; Role-Based Access Control; Real-Time Prediction.*

I. INTRODUCTION

Credit-card fraud detection is a major concern for financial institutions and consumers worldwide. With the rapid growth of online transactions and digital payments, fraudulent activities have become more sophisticated,

causing significant financial losses. The Credit Card Fraud Detection System aims to automatically detect potentially fraudulent transactions before they cause harm, by analysing transaction characteristics and identifying patterns indicative of fraud.

The system uses a machine-learning-based approach that takes into account factors such as transaction amount, country, time, and merchant category. These factors are analysed to predict whether a transaction is legitimate or fraudulent, and a Django web interface is provided through which users can log in and test transactions while administrators monitor flagged activities. The key objective is to build a secure, efficient, and user-friendly fraud-detection platform that can continuously learn from data; by applying machine-learning models, the system reduces false positives, improves detection accuracy, and automates the monitoring process.

The project also emphasises the integration of role-based access, in which different users (Administrator, Analyst, and Reviewer) have separate permissions to view, train, or analyse data, ensuring secure and organised management of operations. From an implementation perspective, the project covers the end-to-end workflow—from data preprocessing, model training, and evaluation to web integration, role-based authentication, and transaction management—so that every user must be registered and authenticated before accessing the system, ensuring data security and accountability.

II. LITERATURE SURVEY

Fraud detection in credit-card transactions has been an active research area for many years due to the increasing volume of digital payments and the corresponding rise in cybercrime. Early systems were based on predefined rules such as transaction limits, unusual login behaviour, or geographic mismatches; while simple to implement, they were inflexible, failed to detect new fraud patterns, and produced many false positives. With the advance of machine learning, models such as Logistic Regression, Decision Trees, Random Forest, and Support Vector Machines were widely applied, learning patterns from historical labelled data and improving detection accuracy by treating fraud detection as a binary classification problem; ensemble methods such as Random Forest are often preferred because they are robust on imbalanced data and reduce overfitting compared with individual classifiers.

Broader work on data mining and machine learning—such as the foundational texts by Han, Kamber, and Pei and by Witten et al.—establishes the data-mining concepts and tools relevant to building such systems, and practical machine-learning references by Géron, Raschka and Mirjalili, and Brownlee describe the modern Python ecosystem (scikit-learn, pandas, TensorFlow) that is used for implementation. McKinney's pandas-focused work on Python data analysis provides the data-wrangling foundations, and Goodfellow, Bengio, and Courville's deep-learning text frames more advanced extensions to neural-network-based fraud detection. The Django documentation and scikit-learn documentation describe the web-framework and ML libraries that are integrated to deliver the system as a usable web application. Across the literature, the prevailing pattern is to combine classical ML classifiers with a web-based delivery channel and role-based security, which is the approach adopted in this work.

TABLE I. REPRESENTATIVE TECHNIQUES FOR FRAUD DETECTION

S.No	Approach	Technique / Note	Limitation / Note
1	Rule-based systems	Predefined thresholds and rules	Inflexible; high false positives
2	Logistic Regression	Linear classification baseline	Interpretable; limited non-linearity
3	Decision Tree	Rule-based ML; non-linear	Can overfit single trees
4	Random Forest	Ensemble of decision trees	Robust; common production choice
5	SVM	Margin-based classifier	Sensitive to kernel / parameters
6	Deep learning	Neural networks for complex patterns	Higher computational cost

III. EXISTING SYSTEM AND PROPOSED SYSTEM

A. Existing System

Existing credit-card fraud-detection systems mainly rely on rule-based engines and manual verification by analysts. Rule-based engines use predefined thresholds (for example, transaction amount, country mismatch, or unusual transaction time) and are easy to set up but fail to adapt to new fraud strategies, generating high false-positive rates and often missing more sophisticated patterns. Manual verification is labour-intensive and not scalable, and traditional systems also lack integrated web interfaces, automated retraining, and role-based access for different stakeholders such as administrators, analysts, and reviewers.

Limitations of the existing system:

- Inflexible rule-based engines cannot adapt to new fraud patterns.
- Manual verification is labour-intensive and not scalable.
- High false-positive rates and missed sophisticated frauds.
- Lack of integrated web access and automated retraining.
- No role-based access control for distinct stakeholders.

B. Proposed System

The proposed system provides an efficient and automated solution for detecting fraudulent credit-card transactions using machine-learning algorithms integrated with the Django web framework. It analyses transaction details, identifies suspicious patterns, and alerts administrators or reviewers about potential frauds, predicting whether a transaction is legitimate or fraudulent based on behavioural and transactional factors such as amount, merchant category, and country. The trained machine-learning model (for example, Decision Tree or Random Forest) is integrated into the Django web application to perform real-time predictions for new transactions, and the system is built from multiple modules including user registration and authentication, role-based access control, model training, fraud prediction, and flagged-transaction review, with three main user roles—Administrator, Analyst, and Reviewer—each having specific access privileges.

Advantages of the proposed system:

- ML-based detection adapts to evolving fraud patterns.

- Reduces false positives and improves detection accuracy.
- Real-time prediction via integrated Django web application.
- Role-based access (Administrator / Analyst / Reviewer).
- Automated training and retraining with uploaded datasets.
- Layered architecture supporting scalability and security.

IV. SYSTEM DESIGN AND ARCHITECTURE

A. Requirements and Roles

Functionally, the system must support user registration and login, role-based access control, dataset upload, model training and retraining, fraud prediction for new transactions, review of flagged transactions, and basic analytics. Three user roles are defined. The Administrator has the highest level of access: registered and verified admins manage users, assign and update roles, control access to sensitive features, upload CSV files to train or retrain the model, and supervise overall operations. The Analyst is a registered user who focuses on data processing and model operations such as data preparation and analysis. The Reviewer is responsible for validating flagged transactions and can log in to access flagged data; this separation ensures that only authorised reviewers can access sensitive fraud data, preventing misuse.

B. Layered Architecture

The system is organised in a layered architecture comprising a presentation layer, an application layer, and a data layer. The presentation layer, built with HTML, CSS, and JavaScript integrated with Django templates, provides a clean and intuitive web interface in which users register, log in, check transactions, upload datasets, and view flagged fraud cases. The application layer implements role-based access, dataset handling, model training and prediction, and the review workflow. The data layer stores users, roles, datasets, model artefacts, predictions, and flagged transactions, accessed through Django's database models and ORM.

C. Workflow

An administrator uploads a labelled CSV dataset; the system preprocesses it (handling missing values, encoding categorical variables such as merchant category and country, and normalising numerical values) and splits it into training and testing sets; a classifier (Logistic Regression, Decision Tree, or Random Forest) is trained and evaluated, with the trained model saved using Joblib or Pickle. When a new transaction is submitted, the Django application loads the serialised model and predicts the probability of fraud; transactions exceeding the probability threshold are flagged for review; reviewers then validate flagged transactions, and analysts process data and retrain the model as new examples accumulate.

V. SYSTEM IMPLEMENTATION

A. Technology Stack

TABLE II. TECHNOLOGY STACK

Component	Technology / Tool
Programming Language	Python
Web Framework	Django
Frontend	HTML, CSS, JavaScript

Component	Technology / Tool
	(Django templates)
ML Library	scikit-learn
Data Handling	pandas, NumPy
Algorithms	Logistic Regression, Decision Tree, Random Forest
Model Serialization	Joblib / Pickle
Access Control	Role-based: Administrator / Analyst / Reviewer

B. Implementation Details

The implementation is developed in Python with Django as the web framework. Scikit-learn provides the machine-learning algorithms, while pandas and NumPy support data handling and numerical computation. The training pipeline ingests an uploaded CSV file, preprocesses the data, trains a classifier on a labelled dataset of legitimate and fraudulent transactions, and evaluates the model on a held-out set. The chosen model—often Random Forest because of its robustness on imbalanced data—is serialised with Joblib or Pickle and integrated into the Django application for real-time prediction; the web interface accepts new transaction details and returns a fraud probability, flagging transactions that exceed the configured threshold for review.

C. Roles and Security

Security is provided through Django's authentication system and a role-based access model. Administrators manage users and trigger model training; Analysts handle dataset upload and processing; Reviewers see only the flagged transactions they are responsible for validating. Registration and login ensure that only authorised users can access sensitive fraud data, preventing misuse and supporting accountability across the workflow.

VI. SYSTEM TESTING AND RESULTS

The system was validated through functional testing of its core modules: user registration and authentication, role-based access enforcement, dataset upload, model training, fraud prediction with probability threshold, flagged-transaction listing, and reviewer validation. Integration testing confirmed that the preprocessing, ML model, and Django application interact correctly end-to-end. The reported testing confirmed that the system behaves as expected, training the models, producing predictions, flagging transactions above the configured threshold, and routing them to reviewers without errors.

TABLE III. TESTING SUMMARY

Test Level	Focus	Outcome
Functional testing	Auth, RBAC, upload, train, predict, flag, review	Behaved as expected
Model testing	Classifier output and probability threshold	Produced valid predictions

International Journal of Engineering Research and Science & Technology

Test Level	Focus	Outcome
Integration testing	Preprocessing + model + Django flow	Behaved as expected

A. Observed Results

The implemented system performs end-to-end fraud detection: it accepts uploaded datasets, trains and evaluates Logistic Regression, Decision Tree, and Random Forest classifiers, integrates the chosen model into the Django web application via Joblib/Pickle, and provides real-time predictions with probability-threshold-based flagging. Role-based access cleanly separates administration, analysis, and review responsibilities. The source describes these outcomes qualitatively; although evaluation metrics such as accuracy, precision, recall, and F1-score are appropriate for this task, no specific numeric values are stated in the source, so none are asserted here, and real-world performance depends on dataset quality and the balance of legitimate-to-fraudulent samples.

Representative screenshots from the prototype implementation:

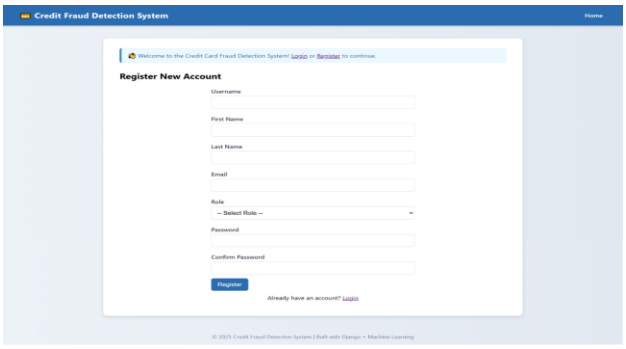

Fig. 1. User registration and role-based login.

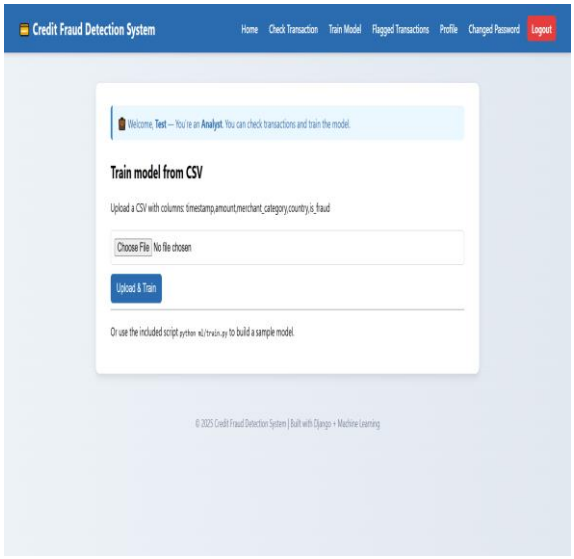

Fig. 2. Dataset upload and model training (admin).

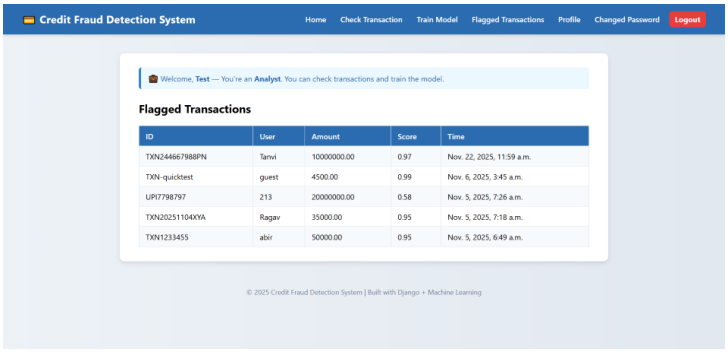

Fig. 3. Transaction submission and fraud-probability prediction.

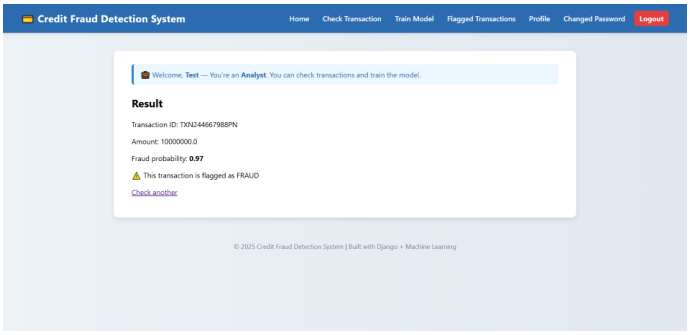

Fig. 4. Flagged-transaction review interface.

VII. CONCLUSION AND FUTURE SCOPE

The Credit Card Fraud Detection System successfully demonstrates how Machine Learning and Django-based web development can be integrated to create an intelligent, secure, and user-friendly fraud-detection platform. The project achieves its main objective—to identify fraudulent credit-card transactions by analysing key transaction attributes such as amount, merchant type, and country of origin. Compared with traditional rule-based and manual approaches, the system replaces inflexible rules with adaptive, data-driven classifiers (Logistic Regression, Decision Tree, and Random Forest), integrates the chosen model with a Django web interface for real-time prediction and management, and adds clear role-based access control for Administrator, Analyst, and Reviewer users. The layered architecture, model-serialisation step, and probability-threshold flagging together provide a practical workflow for both training and operating the system, and functional testing confirmed correct behaviour across the registration, training, prediction, and review modules.

Several enhancements would make the system more robust, scalable, and applicable in real-world banking environments. Larger and more diverse transaction datasets, together with techniques for class imbalance such as resampling or cost-sensitive learning, would improve detection accuracy and reduce false positives. Advanced models such as Gradient Boosting and neural networks (and, where appropriate, sequence models for transaction streams) can further improve performance, and explainable-AI techniques would clarify which features drove a flag, supporting compliance and analyst trust. Real-time streaming with technologies such as Apache Kafka, deployment on cloud platforms with autoscaling, integration with bank or payment-gateway systems, and stronger security controls such as two-factor authentication and audit logging are key directions for production adoption. Continuous learning pipelines that retrain the model as new examples accumulate would keep the system effective as fraud patterns evolve.

REFERENCES

[1] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. Sebastopol, CA, USA: O'Reilly Media, 2019.

[2] S. Raschka and V. Mirjalili, Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2. Birmingham, U.K.: Packt Publishing, 2019.

[3] J. Brownlee, Machine Learning Mastery with Python: Understand Your Data, Create Accurate Models, and Work Projects End-to-End. Machine Learning Mastery, 2018.

[4] W. McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2017.

[5] Django Software Foundation, "Django Documentation," 2024. [Online]. Available: https://docs.djangoproject.com/

[6] Scikit-learn Developers, "Scikit-learn Documentation," 2024. [Online]. Available: https://scikit-learn.org/

[7] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

[8] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, Data Mining: Practical Machine Learning Tools and Techniques, 4th ed.

Burlington, MA, USA: Morgan Kaufmann, 2016.

[9] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, 3rd ed. Burlington, MA, USA: Morgan Kaufmann (Elsevier), 2012.

[10] L. Breiman, “Random Forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.