APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 6 No. 2 (2025) 680-697
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

A Random Forest and Logistic Regression Based Tool for Credit Card Fraud Detection

Zhafran Faiq Shahri[1], Nurul Azma Abdullah[1]*

[1] *Fakulti Sains Komputer dan Teknologi Maklumat,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: *azma@uthm.edu.my*
DOI: https://doi.org/10.30880/aitcs.2025.06.02.037

Article Info

Received: 21 July 2025
Accepted: 19 November 2025
Available online: 30 November 2025

Keywords

Fraud Detection, Random Forest, Logistic Regression, Credit Card, Tool

Abstract

Detecting credit card fraud is an important issue in financial transactions since it could result in large financial losses and data breaches. However, traditional rule-based systems frequently struggle with imbalanced datasets and are unable to adapt to changing fraud strategies. This project suggests a machine learning-based fraud detection tool that identifies transactions as either valid or fraudulent by using Random Forest and Logistic Regression. The Synthetic Minority Oversampling Technique (SMOTE) is employed to handle data imbalance and enhance model accuracy. Implemented as a web-based platform with Flask, the tool allows fraud analysts in banking and financial institutions to upload transaction files, analyse results, and visualize fraud patterns. The tool addresses the limitations of traditional methods and offers a practical approach to securing digital financial systems by decreasing false positives and improving fraud detection accuracy.

1. Introduction

Credit card fraud is a growing concern globally, causing significant monetary losses, data breaches, and loss of trust in digital systems [1], [2]. Traditional fraud detection methods, such as rule-based systems, are struggling to adapt to the increasing complexity of fraudulent tactics, particularly with the rise of electronic payments and online transactions. These methods rely heavily on predefined rules and manual reviews, which are often inefficient, slow, and likely to produce to errors, especially when working with imbalanced datasets where fraudulent transactions amount to only a small percentage of the total [3].

The aim of this credit card fraud detection tool is to overcome these challenges by creating a web-based fraud detection tool that makes use of machine learning methods, particularly Random Forest and Logistic Regression. The tool uses Synthetic Minority Oversampling Technique (SMOTE) to balance datasets and improve model accuracy. It offers a platform for fraud analysts to upload transaction files and classify transactions based on detected patterns, focusing on manual fraud detection rather than real-time processing. While it cannot adapt to new fraud strategies without retraining, the system fills gaps in current approaches by providing a faster, more accurate, and user-friendly solution.

The objectives of the project are to design a web-based fraud detection tool using Random Forest and Logistic Regression algorithms, to develop a model performance on imbalance data by applying SMOTE (Synthetic Minority Over-sampling Technique), and to evaluate the functionality and usability of fraud detection tool to detect fraudulent transaction. The tool is limited to pre-defined and anonymised dataset such as Kaggle Credit Card Fraud Detection Dataset, making it unsuitable for real-time detection or unfamiliar data patterns. Nevertheless, it is tailored for fraud analysts who require efficient tools to process historical transaction data.

Expected outcomes include increasing productivity through digital processing, strengthening financial transaction security, and automating the fraud detection process to minimise human error. The tool is important for banks, financial institutions, and e-commerce platforms since it offers a safe and effective method of identifying fraud, protecting assets, and preserving customer confidence. This project benefits businesses and the community at large by enhancing fraud detection systems and creating safer digital financial environments. The tool is a web-based platform that allows risk-management teams and authorised fraud analysts to safely log in, upload batches of transaction data, receive automated fraud-risk assessments using Random Forest and Logistic Regression models, and create PDF reports for compliance or auditing needs. Financial organisations' fraud analysts and compliance officers are among its primary users, however, due to its reliance on a simulated and anonymised dataset, this prototype is not suitable for direct use in live operational settings because it is unable to process or train on actual transaction records for confidentiality reasons. In conclusion, this introduction has detailed the aims of the credit card fraud detection tool, scope, and expected outcomes for the credit card fraud detection tool. The following section discusses previous studies and methods about credit card fraud detection to detect fraudulent activity.

2. Literature Review

As digital transactions increase, fraud detection has grown more important, requiring advanced techniques for preventing evolving fraud patterns [2]. This section will provide the literature reviews of credit cards fraud types, detection techniques, datasets, and existing tool. The goal of the literature review is to determine any weaknesses and gaps in the current machine learning approach.

2.1 Credit Card Fraud

As digital payments and global e-commerce have become more common, credit card transaction volumes have increased significantly, giving fraudsters more opportunities[2]. Each year, credit card fraud causes losses of billions of dollars[4]. Credit card fraud is the term used to describe illegal transactions that take advantage of system flaws to cause data breaches and monetary losses. The following paragraphs are the common fraud types that will be discussed.

Card-Not-Present (CNP) fraud occurs in transactions where the physical card is not required, such as online purchases. Fraudsters use stolen card details to make unauthorized transactions, exploiting weak authentication systems in e-commerce platforms [5], [6]. For businesses and financial organisations, this kind of fraud presents serious difficulties because the identity of cardholder cannot be physically confirmed[7].

Card-Present (CP) fraud involves the physical misuse of credit cards, often through skimming or cloning. Fraudsters duplicate card details at ATMs or POS terminals, conducting unauthorized transactions while bypassing standard security measures. Those responsible of this kind of fraud can get over standard security measures since they have the real card or a fake one.

Application fraud occurs when someone applies for a new credit card using a stolen or faked identity. This kind of fraud allows scammers to create accounts for illegal activities by taking advantage of flaws in identity verification procedures [5]. The fraudster has complete access to the card after it is issued until it is used up or the victim discovers unauthorised activity[8] .

In Account Takeover (ATO) fraud, attackers gain unauthorized access to legitimate accounts by stealing login credentials. Fraudsters manipulate account details or make unauthorized transactions, often bypassing basic security checks [6]. In order to mitigate these credit frauds, methods and techniques to detect fraudulent transaction must be well implemented which will be discuss in the next section.

2.2 Fraud Detection Techniques

Fraud detection techniques have evolved to address the increasing complexity of fraudulent activities. These methods range from traditional rule-based approaches to advanced machine learning models, each with its strengths and limitations. These methods range from traditional rule-based approaches to advanced machine learning models, each with its strengths and limitations. Rule-based systems rely on predefined rules to detect anomalies, such as flagging transactions above a certain predetermined amount. Despite being simple and interpretable, they have to adapt to evolving fraud patterns and this often results in high false positive rates [3]. Rule-based systems have limited adaptability in the fast-paced digital contexts of today due to their inability to process massive amounts of data effectively[9] .

Anomaly detection identifies suspicious transactions by identifying anomalies from normal transaction behaviour [10]. For example, a sudden high-value transaction from a different country might be tagged as an anomaly if the user regularly makes small purchases within a particular geographic area. Unexpected fraud attempts can be detected with the help of this technique [11]. These methods are flexible and adaptive but can generate false positives when legitimate behaviour varies significantly from the norm. Behavioural techniques

focus on user-specific patterns, such as spending habits or device usage. They provide a personalized approach to fraud detection but require extensive data collection which raises privacy concerns.

Ensemble methods combine multiple models to improve accuracy and robustness. Techniques like Random Forest and Gradient Boosting combine predictions to handle addressing variety of fraud patterns effectively [12]. Machine learning models, such as Logistic Regression and Neural Networks, analyse transaction data to detect fraud patterns. Machine learning models use historical transaction data to build models that classify transactions as either fraudulent or legitimate. These models are adaptable and scalable, offering significant advantages over traditional techniques. The two main categories of machine learning models used in fraud detection are supervised and unsupervised learning. Although supervised models rely significantly on the availability of labelled data, they are quite accurate when trained on high-quality datasets [13]. There are many types of machine learning to detect fraud which will be explain in the following section.

2.3 Machine Learning Algorithms for Fraud Detection

Modern fraud detection systems rely heavily on machine learning algorithms because they offer the scalability and flexibility needed to analyse big datasets and identify intricate patterns of fraudulent activity. The main fraud-detection algorithm in this credit card fraud detection tool is Random Forest, which is constructed as an "ensemble" of basic decision trees [14]. Each tree is trained on a random subset of previous transactions and a random subset of the input data such as transaction time, amount, and anonymised components V1–V28 [15]. Each tree votes on whether a new transaction is fraudulent after learning a set of "if-then" rules, such as "if amount > $1000 and time of day is late night, then flag." The majority vote across all trees makes the ultimate decision, mitigating the errors of any one tree and capturing complex, non-linear patterns that might be missed by a single model. Using cross-validation, important parameters such as number of trees, and maximum tree depth were adjusted throughout development to achieve high detection rates without excessive slowness. After being trained, the Random Forest model is stored and incorporated into the web application, enabling fast and accurate scoring of every file provided by analysts even when hundreds of transactions are being performed simultaneously [16]. The model may become computationally costly, particularly if it is trained on big, feature-rich datasets [17].

Logistic Regression is a simple, yet effective binary classification model used for distinguishing fraudulent transactions [18]. Its interpretability and ease of implementation make it a popular choice for initial fraud detection models. The chance of fraud is modelled by logistic regression as a straightforward weighted sum of the same input features that have been run through a logistic function, offering an intuitive complementary method. The algorithm determines the optimal weights that distinguish between fraudulent and legitimate transactions once the data has been scaled and balanced with SMOTE such that "Time" and "Amount" live on comparable scales. The strength with which a feature influences the prediction towards "fraud" is shown by each weight, for instance, a high positive weight on "V7" suggests that greater values of that component raise the chance of fraud. Logistic Regression is perfect for fast evaluations in resource-constrained environments because it trains and scores incredibly fast using only simple arithmetic on each record. Logistic regression is perfect for real-time detection or situations with limited resources because it is computationally inexpensive and performs well with smaller datasets [18].

Decision Tree models create a flowchart-like structure to classify transactions based on specific rules. The ability of decision trees to handle both numerical and categorical data is one of their main advantages. While easy to interpret, they are prone to overfitting, making them less reliable for complex fraud patterns [19]. K-Nearest Neighbours (KNN) classifies transactions by comparing them with the 'k' nearest data point When it comes to fraud detection, KNN is especially helpful in spotting transactions that resemble known fraudulent ones [20]. It is intuitive and effective for smaller datasets but becomes computationally expensive as the dataset grows [20].

Neural Networks are highly accurate for detecting intricate fraud patterns due to their ability to model non-linear relationships. They are excellent at identifying intricate, non-linear correlations, which is especially useful in situations involving fraud detection where the patterns are complex and subtle [7]. However, they require significant computational resources and large datasets for optimal performance. Datasets are essential in training and evaluating fraud detection models. Anonymised transaction data with labels for fraudulent and lawful transactions can be found in popular datasets such as the IEEE-CIS Fraud Detection.

2.4 Dataset

To train and assess fraud detection models, a dataset is necessary. Anonymised transaction data with labels for fraudulent and lawful transactions can be found in a popular dataset such as the Kaggle Credit Card Fraud Detection Dataset [15]. Only numerical input variables that have undergone PCA transformation are included. However, the original features and additional context for the data cannot be provided because of confidentiality concerns. Features V1 until V28 are the principal components obtained with PCA. 'Time' and 'Amount' are the only features that have not been transformed through PCA. The dataset includes credit card transactions performed by European cardholders in September 2013 [15]. Out of the 284,807 transactions in this dataset, 492

were fraudulent and took place over the course of two days. The positive class, or frauds, make up 0.172% of all transactions, making the dataset extremely unbalanced with a small percentage of transactions being fraudulent, which results in biassed model predictions [15] [21]. This problem is addressed by methods such as the Synthetic Minority Oversampling Technique (SMOTE), which improves model training and accuracy by creating synthetic samples of minority classes. Despite these initiatives, creating reliable fraud detection systems continues to be significantly hampered by the need for balanced datasets. In order to understand more about credit card fraud detection, there have to be related works that uses these fraud detection techniques and machine algorithms which will be discuss in the following part.

2.5 Comparative of Existing Tools

Existing fraud detection tools, such as FIFO Falcon [22], SAS Fraud Management [22] and PayPal In-House System [23], highlight the strengths of advanced fraud detection methods. However, they face challenges in cost, complexity, and limited scalability. These limitations underscore the need for a more efficient, adaptable, and user-friendly solution, which this project aims to provide.

Table 1 *The Comparison of Existing Tools about Credit card Fraud Detection along with the Proposed Tool*

Feature \ Tools	FICO Falcon Fraud Manager	SAS Fraud Management	PayPal In-House System	Proposed Tool
Platform	Cross-platform	Multi-platform	Exclusive to PayPal	Web-Based Platform
Detection Approach	Machine learning and rule-based	Predictive analytics and rule-based	Machine learning (supervised & unsupervised)	Machine Learning (Random Forest and Logistic Regression)
Data Source	Global fraud data, internal banking data	Multi-channel	PayPal transaction data	Kaggle credit card fraud dataset
Real-Time Detection	Yes	Yes	Yes	No (file upload only)
Adaptability	Customizable rules for institution need	Multi-channel customizable	High, continuous feedback	Restricted to fixed dataset analysis
Cost and Accessibility	High, suited for large organization	High, suited for large organizations	Not available externally	Low-cost, open to various organizations
Ease of Use	Complex, requires training	Complex, requires technical knowledge	Streamline for PayPal use	Simple interface, designed for financial analyst
Result Visualization	Customizable dashboard	Advanced analytics reports	Internal dashboard, live alerts	Visual output with fraud score

Comparing fraud detection tools reveals their unique advantages. For big businesses with lots of resources, FICO Falcon [22] and SAS Fraud Management [24] provide sophisticated real-time detection with machine learning. PayPal's in-house technology offers a customised, adaptable solution for its platform[23]. The suggested tool, on the other hand, is an affordable, approachable option suited for smaller companies. Although it does not have real-time capabilities, it allows for efficient historical fraud investigation using Random Forest and Logistic Regression on static datasets like Kaggle, making it accessible to analysts with no technical background. The following methodology shows the phases in order to implement the proposed credit card tool.

3. Methodology

To guarantee effective project progression, the methodology includes flexible and iterative practices inside an Agile development framework. Planning, Design, Development, Testing, Deployment, and Review are the six main stages of the process. Every phase encourages flexibility and ongoing development while concentrating on activities meant to accomplish the goals of the project. The visualization of the Agile methodology process is shown on Fig. 1.

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 680-697

Fig. 1 *Agile Methodology Phases* [25]

The objectives, scope, and deliverables of the project are established during the planning phase. The primary goal is to develop a fraud detection tool that uses machine learning to identify fraudulent transactions. Important tasks include choosing technologies such as Python, Flask and Scikit-learn, evaluate datasets, and outlining tasks like interface design, model training, and data preprocessing. A product backlog is used to prioritise these tasks, guaranteeing attention to important features and directing the development process.

Before development starts, the design phase concentrates on producing a well-organised system and user interface. Data preprocessing, machine learning model execution, and the user interface are important elements. Designing the system architecture, specifying how the frontend, backend, and models interact, and producing wireframes for UI features like file uploads and result visualisation are among the tasks. Workflow diagrams show how models are run, data is preprocesses, and results are visualised. System architecture and interface design are done with tools like Lucidchart and Figma, which guarantee that the developer has a clear plan for both functionality and user experience.

System components are developed and integrated repeatedly during the development phase. Building the data preprocessing module, which includes activities like feature normalisation, data cleaning, and applying SMOTE for class imbalance, is the main goal of the first iteration. Using a processed dataset, machine learning models such as Random Forest and Logistic Regression are trained in the following iterations. Additionally, frontend development is done, creating the user interface for file uploads and displays of fraud classification. Scikit-learn is used for model training, and Flask is used for backend development. A functional test version of the tool is produced at the conclusion of this phase, which marks the complete integration of the essential components tool.

The testing phase makes sure the tool is accurate, dependable, and easy to use. Unit testing validates individual components like the data preparation pipeline, while integration testing ensures smooth communication between the frontend, backend, and models. Performance testing uses metrics like precision, recall, and F1-score to assess accuracy, with tools like Scikit-learn for model evaluation and PyTest for backend testing. Based on the goals of the tool, the findings of the literature review, and the advice of project supervisor, the user needs were established with a particular focus on batch-mode CSV uploads, automatic machine learning scoring, secure account management, and PDF reporting. Unit testing was conducted by the developer and supervisor, while integration testing was carried out jointly by the fellow students acting as test users.

Additionally, informal User Acceptance Testing (UAT) was conducted during development. The supervisor and fellow students evaluated the interface layout, report generation, and data upload functions. Since domain experts such as fraud analysts were not available due to confidentiality concerns and limited institutional access, IT personnel with knowledge of machine learning were engaged to evaluate the technical aspects of the fraud detection tool. They assessed the reliability, consistency, and correctness of the machine learning implementation. While expert-level fraud validation could not be performed, these tests helped confirm that the system met its design objectives. Formal UAT involving real-world fraud analysts will be considered in the future when access to live transaction environments is available.

Features like real-time monitoring and live input of data are outside the current prototype, but the scope of the system includes a web-based, batch-mode platform with five modules which are File Upload, Data Preprocessing, Machine-Learning Detection, Detection Results and Reporting. Once confidentiality and compliance criteria are satisfied, the intended users are the fraud analysts which will link to actual transaction feeds. By the end of this phase, the tool is confirmed to be functional and ready for deployment.

The deployment phase involves delivering the tool to users and ensuring it meets their needs. The system can be set up locally or on a cloud server, accessible on a web browser. User guides and tutorials are provided to help users navigate the tool. Preliminary feedback is gathered to identify areas for improvement, such as adding

features or addressing usability issues. The result is an accessible tool, user manuals, and initial input to guide future enhancements.

The goal of the review phase is to make sure the tool stays current and functional after deployment. This phase involves maintaining an eye on the functionality of the tool, fixing problems like bugs, and making adjustments in response to user input. The tool is updated to improve its functionality, including by supporting more datasets or adding new capabilities. Tracking user interactions and pinpointing areas for development can be done with tools like Google Analytics. Python is used to add new features and improve the backend of the system. Further details in the analysis and design of the project will be discussed in the following section.

4. System Analysis and Design

System analysis and design is an important phase in software development because it translates user requirements into an organised framework for system implementation. This section makes sure that the tool is effective, scalable, and capable of dealing with the complexity of modern fraud detection by combining a thorough analysis of requirements with a clear design strategy, laying the groundwork for the upcoming implementation phase.

4.1 System Analysis and Design

System requirement analysis determines what the tool requires to perform properly, ensuring it satisfies the expectations of users and stakeholders. This section describes the functional and non-functional requirements. These features are critical for developing a reliable and successful credit card fraud detection tool. Table 2 shows the list of five functional requirements.

Table 2 *Functional Requirements*

Modules	Functional
File Upload Module	Users can upload transaction data files in formats such as CSV or XLSX.
Data Preprocessing	Cleans data, normalises features, and uses SMOTE to deal with imbalanced datasets.
Fraud Detection	Uses Random Forest and Logistic Regression to determine transactions as fraudulent or real.
Results Visualization	Displays fraud detection results, such as fraud probabilities and performance metrics.
Report Generation	Creates downloadable reports summarising detection findings and fraud analysis insights.

Functional requirements focus on tasks the system must perform to meet the needs of user. Key modules include file upload for transaction data, fraud detection using machine learning, result visualization for interpretation, and secure user authentication. Table 2 details these components and their role in achieving the objectives of the project. Table 3 shows the four categories of non-functional requirements.

Table 3 *Non-Functional Requirements*

Modules	Non-Functional
Usability	Provides a user-friendly interface for fraud analysts with no technical knowledge.
Performance	Ensures that uploaded files are processed within one minute and that detection accuracy remains over 90%.
Scalability	Manages uploaded transaction files in batch mode to process datasets of different sizes without experiencing performance decrease during regular usage.
Security	Enables secure file uploads, prevents unauthorised access, and safeguards important transaction data.

Table 3 details the performance, security, and usability standards for the tool. Non-functional requirements define the quality characteristics and limitations of the system which ensures precision, reliability, scalability, security, and user-friendliness of the tool. Key factors include response time and data handling efficiency. In the following section elaborate the system analysis of the proposed tool

4.2 System Analysis

This section examines the proposed project using object-oriented programming principles. It uses visual representations like use case diagrams, UML class diagrams, and flowcharts to describe the way the system functions and structure. These tools provide a clear picture of the processes and interactions of the tool.

4.2.1 Use Case

The use case diagram provides a high-level view of user interactions with the credit card fraud detection tool, focusing on key features. The primary user actions and system reactions to those actions are highlighted in this diagram, which provides a simple and understandable representation of the functionality of the tool. This simple representation outlines the primary functions of the tool, as shown in Fig. 2.

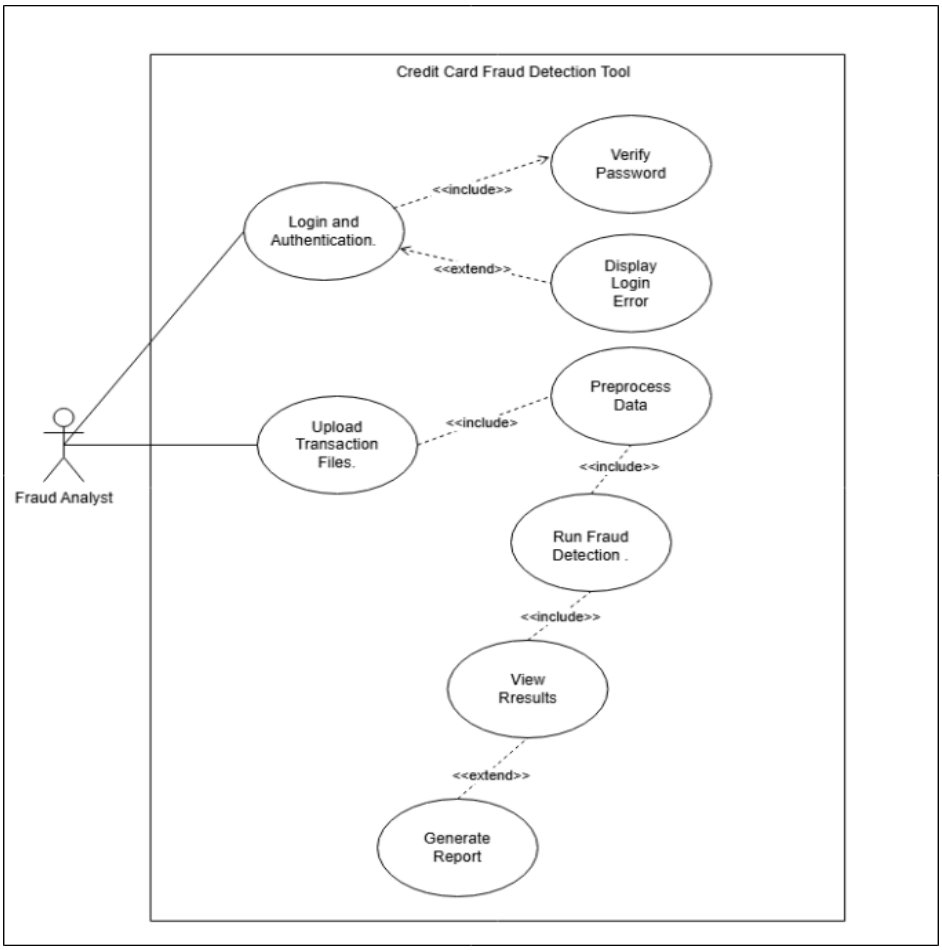


Fig. 2 *Use Case Diagram for Credit Card Fraud Detection Tool*

Fig. 2 shows the use case diagram for the proposed tool which identifies the fraud analyst as the main actor and includes use cases such as login, uploading transaction data, preprocessing, running fraud detection, viewing results, and generating reports. The diagram highlights dependencies, such as preprocessing before detection and report generation after viewing results.

4.2.2 Class Diagram

The UML Class Diagram offers a thorough overview of the structure of the tool, illustrating important classes along with their attributes, functions, and relationships. In this section, the class diagram illustrates the object-oriented framework interactions between the various parts of the credit card fraud detection tool. Fig. 3 displays the class diagram of the tool.

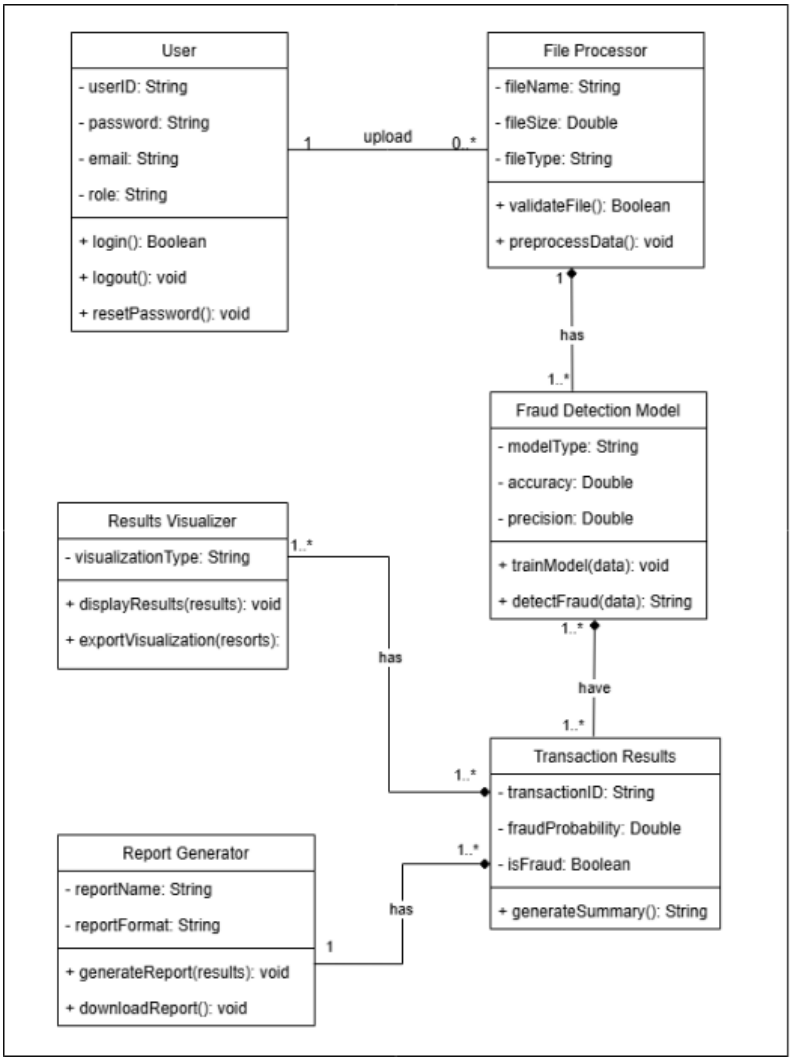


Fig. 3 *UML Class Diagram for Credit Card Fraud Detection Tool*

In Fig. 3, User, FileProcessor, FraudDetectionModel, ResultVisualizer, and ReportGenerator are main classes. The User class controls role-based access and authentication, while the FileProcessor handles data preprocessing. Using machine learning, the FraudDetectionModel class conducts the fundamental fraud analysis, and the ResultVisualizer presents the findings. The results are compiled into reports by the ReportGenerator. These classes cooperate in a scalable, modular, and maintainable way. To visualise the process, the following flowchart will be discussed in more detail.

4.2.3 Flowchart

The workflow from user contact to fraud detection and reporting is illustrated in the flowchart, which shows the steps in the credit card fraud detection tool. The decision-making process, dependencies, and data flow across the system are all shown in the flowchart. The flowchart guarantees a smooth transition between system elements, resulting in efficient fraud detection and satisfied users.

In Fig. 4, the user uploads transaction data after logging in, and it is then verified and preprocesses for analysis. The Random Forest and Logistic Regression techniques are used to classify the data. A straightforward interface shows the results, together with fraud measures and probabilities. Additionally, a report summarising the detection results can be generated by the user. The design of the system will be explained in the next section.

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 680-697

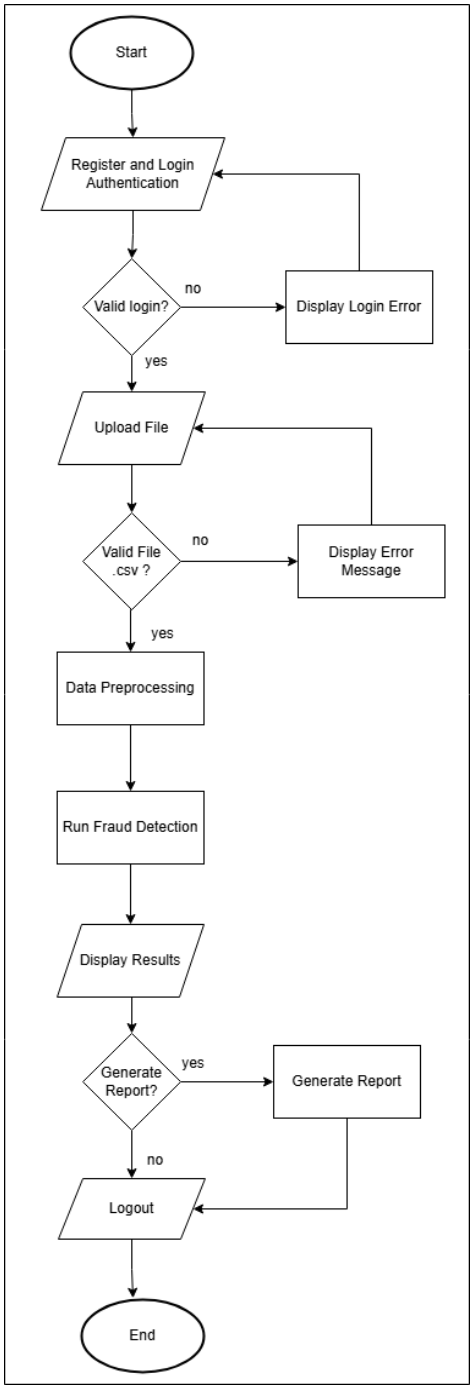


Fig. 4 *Flowchart for Credit Card Fraud Detection Tool*

4.3 System Design

System design focuses on the high-level architecture and interaction between system components. The high-level architecture and how system components interact are the main topics of system design. This section makes sure the credit card fraud detection tool is well-structured and follows object-oriented programming concepts with its modular design. Scalability, maintainability, and ease of integration are all considered in the design. Fig. 5 shows the architecture design of the proposed Credit Card Fraud Detection Tool.

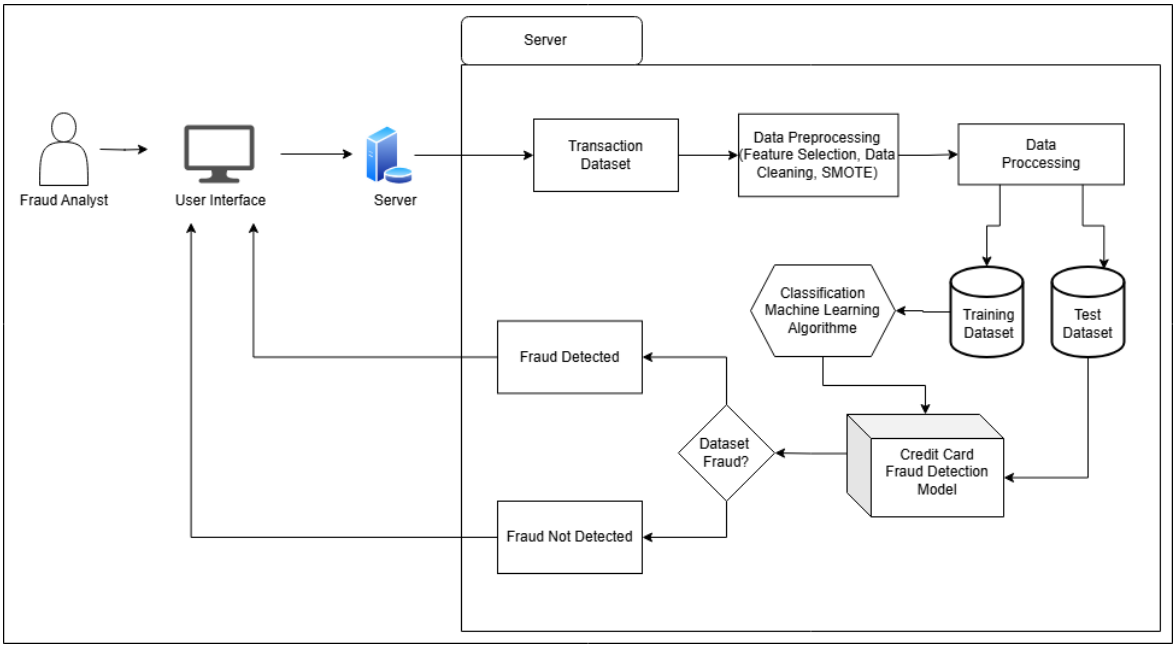
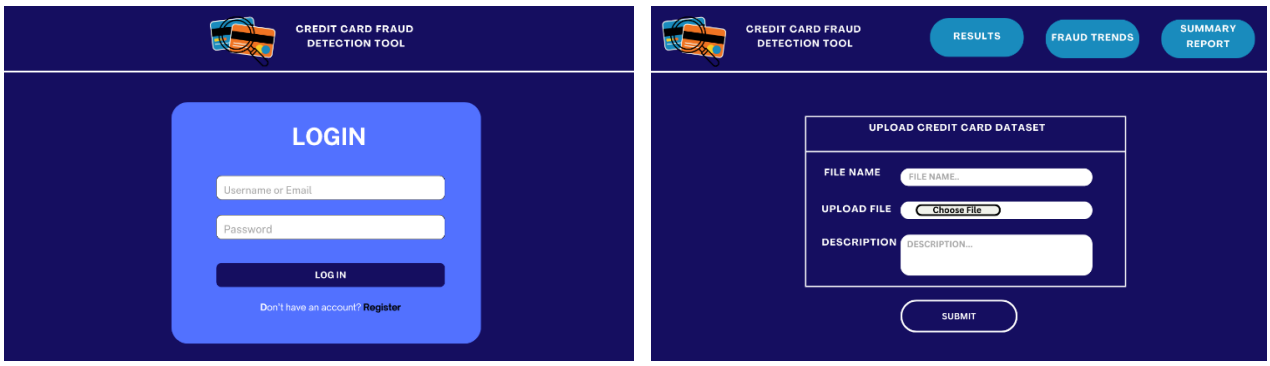


Fig. 5 *System Architecture for Credit Card Fraud Detection Tool*

The end-to-end architecture of the fraud-detection tool is shown in Fig. 5. Using the web interface, a fraud analyst first uploads a transaction file to the Flask server, which is currently an anonymised, simulated CSV of credit card details. Before being divided into training and test subsets, the "Transaction Dataset" is pre-processed on the server such as feature selection, data cleaning, and SMOTE oversampling. A "Credit Card Fraud Detection Model" is created by training a machine-learning algorithm such as Random Forest and Logistic Regression on the training set and validating it on the test set. The model assigns a score to each record when the analyst uploads new data, passing it through a decision node that marks it as either "Fraud Detected" or "Fraud Not Detected." The results are then returned to the user interface for examination. The prototype can only flag patterns it has learnt, not actual, live fraud events, because it uses totally anonymised, simulated data. It will take further model validation and legally compliant access to actual transaction streams to modify the system to identify true fraud.

4.4 Interface design

The credit card fraud detection tool consists of multiple pages designed for a smooth user experience. The design includes sections for essential features including file uploads, report generation, result visualization, and authentication. Every page has a consistent style and easy navigation to help users navigate the system effectively. These pages provide the full functionality of the tool, from login to report generation. The following figures are the example of the interface of the proposed Credit Card Fraud Detection Tool.



(a) **(b)**

Fig. 6 *Interface design of Credit Card Craud Detection (a) Login Page; (b) File Upload Page*

In Fig. 6 shows The Login Page which ensures secure access with username and password input, while the Register Page allows new users to create accounts. After login, users can upload transaction data on the File Upload Page which provides an easy-to-use interface for file selection. The next figure will show the next process of the interface after File Upload Page.

(a) **(b)**

Fig. 7 *Interface design of Credit Card Craud Detection (a) View Results Page; (b) Generate Report Page*

In Fig. 7 shows The View Results Page which displays the analysis results, showing whether transactions are legitimate or fraudulent. The Visualization Results Page presents graphical representations for better understanding, and the Generate Report Page enables users to create, delete and save reports. The results and discussion of the implementation of the project will be explain in the next section.

5. Results and Discussion

This section demonstrates how each criterion was met in practice and establishes the framework for assessing the effectiveness of the tool, filling the gap between the design objectives and the functional prototype. This section shows that the credit card fraud detection tool is more than just code, it also fulfils its promise to help analysts identify suspicious transactions by demonstrating the implementation options and suggesting a testing strategy.

5.1 Implementation

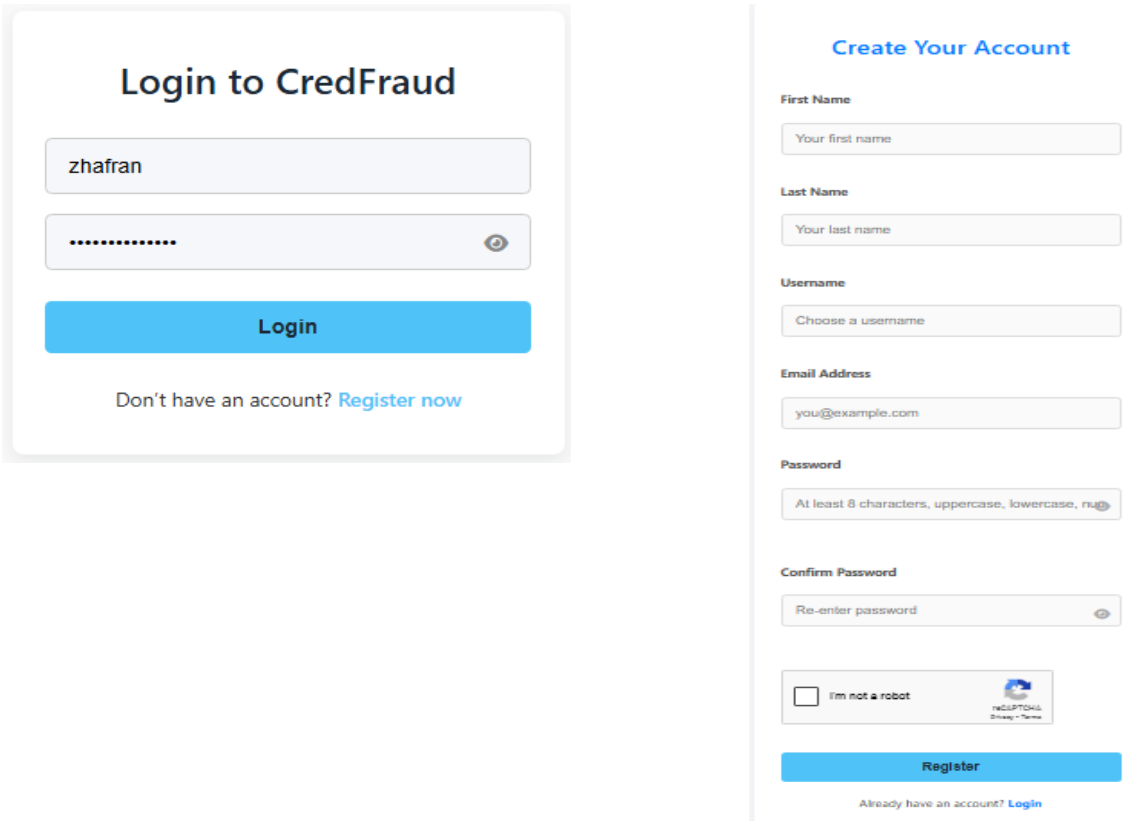
The Flask framework in Python was used to create a browser-based application. All dependencies, including Flask for templating and routing, scikit-learn and imbalanced-learn for machine learning, pandas for data handling, Matplotlib and xhtml2pdf for reporting, and security extensions like Flask-WTF and Flask-Limiter, were installed in a specific virtual environment for reproducibility. A MySQL database handles persistent storage, and db_config.py centralized connection parameters such that user accounts, transaction uploads, and prediction results are stored in organized tables without the need for hard-coded login credentials.

Analysts are guided through the registration process using CAPTCHA and password-strength checks, login process using secure sessions and hashed-password verification, profile updating, password changes, and account termination using user management procedures. Security measures, such as rate-limiting of critical routes, Cross-Site Request Forgery (CSRF) tokens on all forms, HTTP Strict Transport Security (HSTS) headers on all responses, and cookies designated Secure and HTTPOnly that preserve session data and user credentials

The upload module stores incoming CSV files, puts their contents into Pandas for preprocessing, and verifies that the files have the requisite columns, size restrictions, MIME type checks, and a.csv extension. "Time" and "Amount" are transformed using a StandardScaler, and the unusual fraud class is balanced by SMOTE oversampling prior to the inference application.

Pre-trained Random Forest and Logistic Regression models are loaded through Joblib at startup, together with the scaler that has been fitted. To generate per-record fraud labels and probability scores, incoming transaction batches are transformed into DataFrames, scaled, and scored by both classifiers. The predictions are combined to provide summary statistics, which are then stored for display in the database.

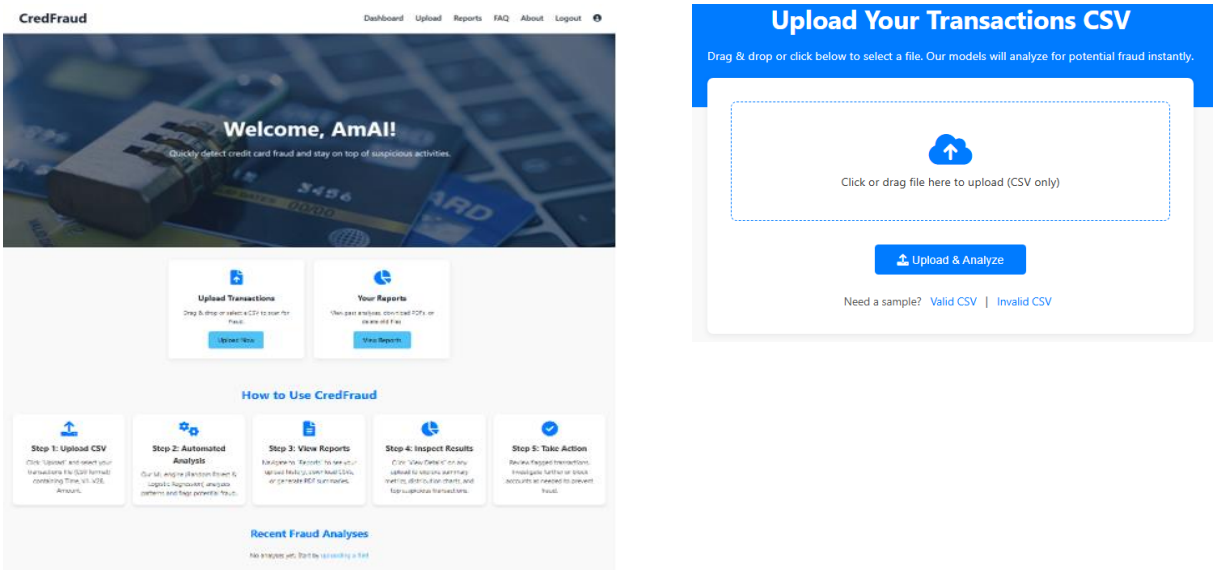
Matplotlib creates pie charts that display the percentage of fraudulent and valid transactions after deduction is finished. These images, along with upload information and a table of records that have been detected, are then translated into an HTML template and transformed into a formatted PDF report using xhtml2pdf. The report history of users contains a list of generated PDFs that can be downloaded. The following figures are the results of this implementation during the development of the credit card fraud detection tool.



(a) **(b)**

Fig. 8 *Interface of Credit Card Craud Detection (a) Login Page; (b) Registration Page*

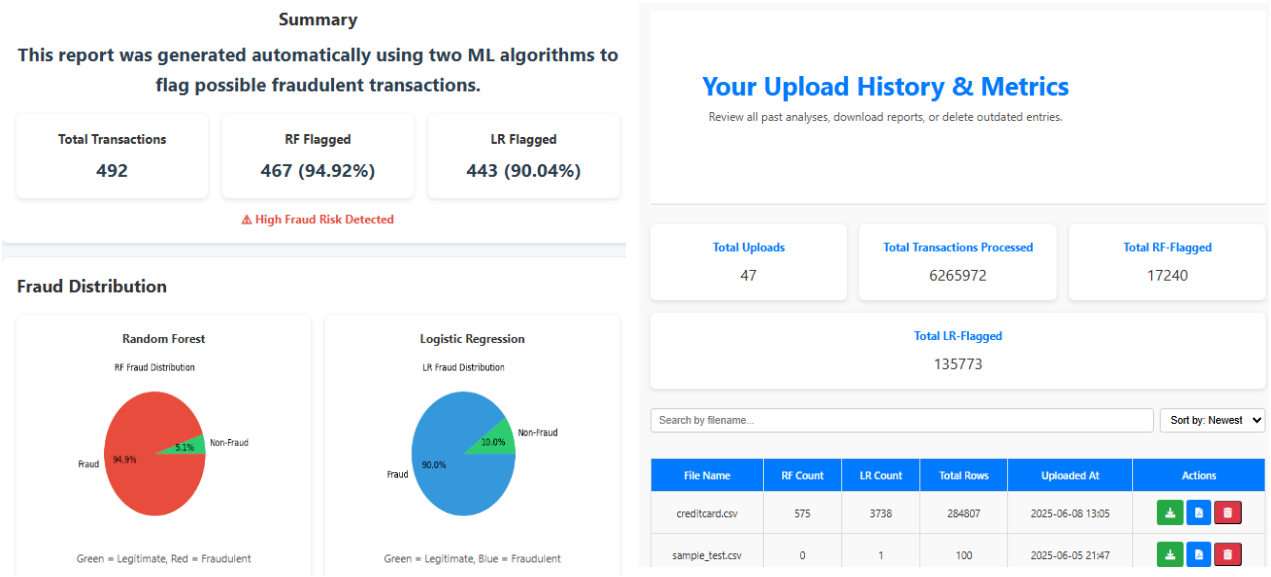
Fig. 8 illustrate the two main point of entry for analysts. To implement credential verification and secure sessions, the login form includes fields for the username and password, a clear "Login" button, and a link to the registration page. To prevent automated sign-ups, the registration page asks new users to input their first and last names, a unique username, and an email address. They must then select and verify a password that satisfies complexity requirements and complete a Google reCAPTCHA challenge. Only authentic, human users can establish accounts and log in to the fraud-detection tool thanks to the integrated validation in both interfaces, which also connect to CSRF protection and secure-cookie settings on the backend. The next figure is about the interface that analyst will see once logged in their account.



(a) **(b)**

Fig. 9 *Interface of Credit Card Craud Detection (a) Dashboard Page; (b) Upload Page*

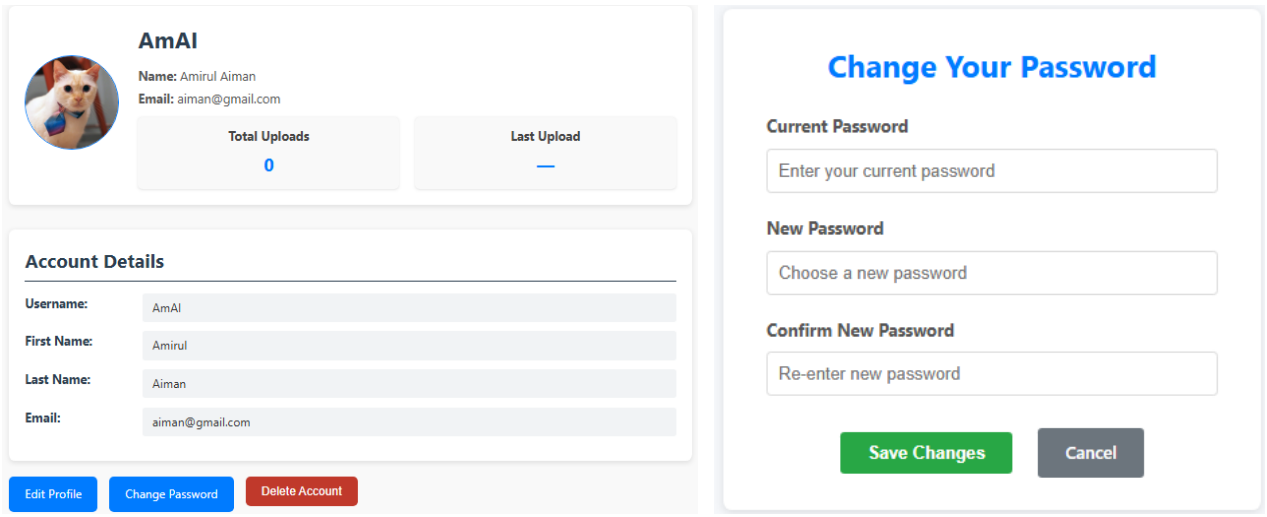
In Fig. 9 consists of the dashboard and upload pages which user will enter once logged in. In order to keep work front and centre, the Dashboard welcomes each user by name as well as a detailed guide through the entire process from CSV submission to result inspection and follow-up actions. Recent analyses appear to urge analyst to upload a file. When users click "Upload Transactions," they are taken to the Upload page, where a sizable drag-and-drop area that only accepts legitimate CSV files with explicit size and schema requirements as well as links to sample templates. Selecting "Upload & Analyse" submits the file, causing the ML engine to preprocess, score, and return results to the dashboard for review in a simple way. After uploading, user will gain access to the results of the analysis which will be explain in Fig. 10.



(a) **(b)**

Fig. 10 *Interface of Credit Card Craud Detection (a) Analysis Page; (b) Report Page*

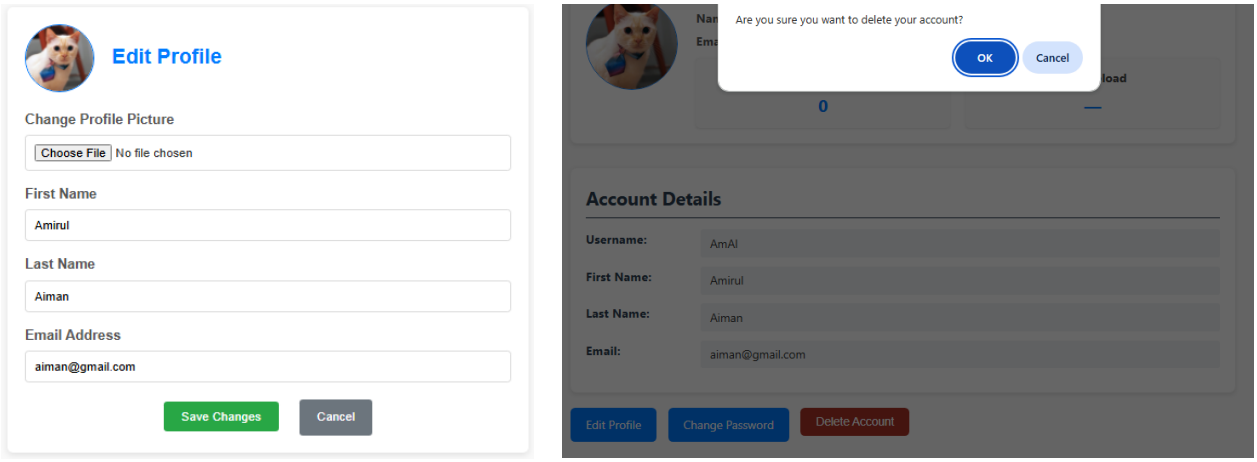
Fig. 10 displays the analysis page and report page of the tool once user have gain access to their account. All post-upload insights are gathered in one view on the analysis page. A sortable "Top 10 Suspicious Transactions" table identifies the most high-risk records, while summary cards show the total number of transactions and the percentage or number of flagged by each model. Adjacent doughnut charts show the fraud and the legitimate split for Random Forest and Logistic Regression. For compliance, presentation, or preservation needs, a "Download Full PDF Report" button then exports these precise components such as metadata, summary metrics, charts, and the whole list of flagged transactions into a clear, print-ready document that replicates the on-screen analysis. Analyst can also view their profile that is demonstrated in the following figures.



(a) **(b)**

Fig. 11 *Interface of Credit Card Craud Detection (a) Profile Page; (b) Edit Profile Page*

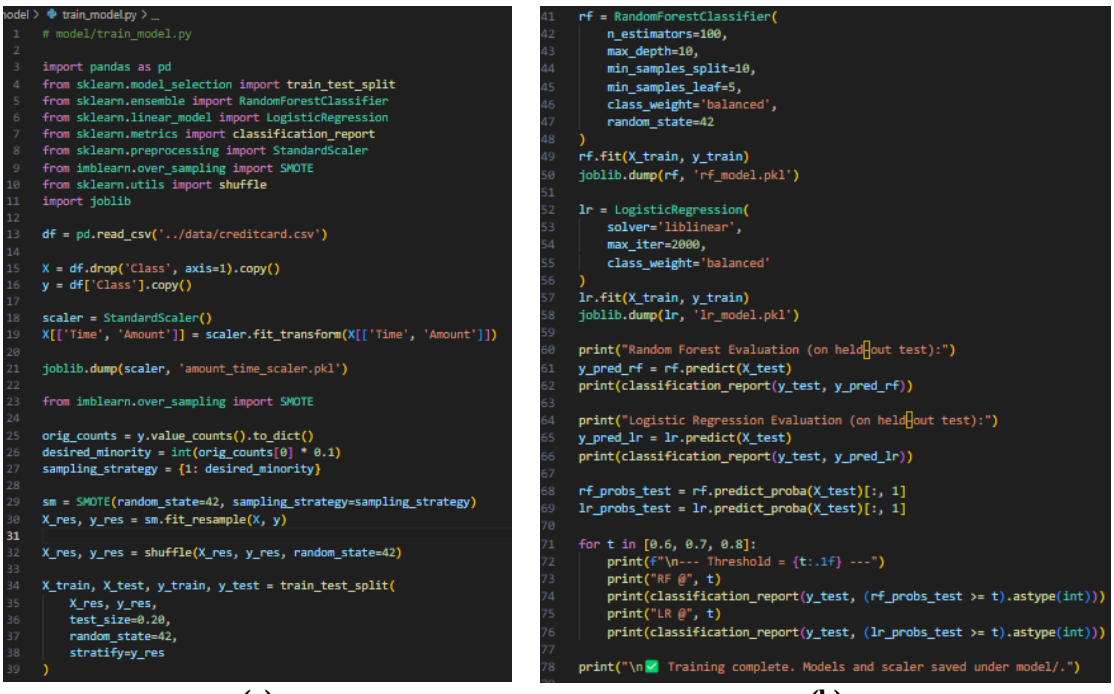
Fig. 11 shows a comprehensive picture of the user account through the Profile page, which also shows their name, email address, avatar, and recent activity metrics such the number of uploads and the most recent login. From here, buttons let users start the Change Password process or change non-sensitive information. After confirming the current password, the Change Password page itself asks for a new one, which must be typed twice, according to the same difficulty guidelines as were in place at registration. A successful update instantly invalidates existing sessions to guarantee account security, and users are guided by clear validation messages if the current password is wrong or if the new entries do not match or meet strength requirements. Besides changing password, analyst can also edit or delete their profile as shown in the next figures.



(a) **(b)**

Fig. 12 *Interface of Credit Card Craud Detection (a) Edit Profile Page; (b) Delete Confirmation*

Fig. 12 shows The Edit Profile page that allows analysts to change their email address, first and last names, and avatar image in a single form with pre-filled sections for ease of use and validation checks to make sure the information is correct before saving. To prevent accidental or unauthorised account removal, the Delete Account feature is accessible through a highly labelled button on the profile page. Once users confirm or cancel, the system securely deletes all user data, terminates the session, and reroutes to the public landing page. The next is the discussion about the backend process on how the tool detect fraudulent transaction using Random Forest and Logistic Regression algorithm.



```python
# model/train_model.py

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE
from sklearn.utils import shuffle
import joblib

df = pd.read_csv('../data/creditcard.csv')

X = df.drop('Class', axis=1).copy()
y = df['Class'].copy()

scaler = StandardScaler()
X[['Time', 'Amount']] = scaler.fit_transform(X[['Time', 'Amount']])

joblib.dump(scaler, 'amount_time_scaler.pkl')

from imblearn.over_sampling import SMOTE

orig_counts = y.value_counts().to_dict()
desired_minority = int(orig_counts[0] * 0.1)
sampling_strategy = {1: desired_minority}

sm = SMOTE(random_state=42, sampling_strategy=sampling_strategy)
X_res, y_res = sm.fit_resample(X, y)

X_res, y_res = shuffle(X_res, y_res, random_state=42)

X_train, X_test, y_train, y_test = train_test_split(
    X_res, y_res,
    test_size=0.20,
    random_state=42,
    stratify=y_res
)
```

(a)

```python
rf = RandomForestClassifier(
    n_estimators=100,
    max_depth=10,
    min_samples_split=10,
    min_samples_leaf=5,
    class_weight='balanced',
    random_state=42
)
rf.fit(X_train, y_train)
joblib.dump(rf, 'rf_model.pkl')

lr = LogisticRegression(
    solver='liblinear',
    max_iter=2000,
    class_weight='balanced'
)
lr.fit(X_train, y_train)
joblib.dump(lr, 'lr_model.pkl')

print("Random Forest Evaluation (on hold-out test):")
y_pred_rf = rf.predict(X_test)
print(classification_report(y_test, y_pred_rf))

print("Logistic Regression Evaluation (on hold-out test):")
y_pred_lr = lr.predict(X_test)
print(classification_report(y_test, y_pred_lr))

rf_probs_test = rf.predict_proba(X_test)[:, 1]
lr_probs_test = lr.predict_proba(X_test)[:, 1]

for t in [0.6, 0.7, 0.8]:
    print(f"\n--- Threshold = {t:.1f} ---")
    print("RF @", t)
    print(classification_report(y_test, (rf_probs_test >= t).astype(int)))
    print("LR @", t)
    print(classification_report(y_test, (lr_probs_test >= t).astype(int)))

print("\n✅ Training complete. Models and scaler saved under model/.")
```

(b)

Fig. 13 *Coding for model training process using Random Forest and Logistic Regression (a) Data Loading, Scaling, SMOTE Resampling, and Train/Test Split ; (b) Model Instantiation, Training, Serialization, and Evaluation*

Fig. 13 shows the training script that imports the raw credit card transactions into a panda DataFrame, isolates the fraud label from the features, and uses a stored scaler to standardise the "Time" and "Amount" columns so that preprocessing at detection is consistent. After that, it uses SMOTE to up-sample the minority fraud class to 10% of the majority, shuffles and stratifies the data into an 80/20 train/test split, and fits two classifiers. The script produces precision, recall, and F1-scores on the held-out test set, along with threshold-based analyses at 0.6, 0.7, and 0.8 to show how sensitivity adjustments affect fraud detection performance. Both models and the scaler are serialised with Joblib for usage in the web application. The result of this process is shown in the next figure.

```
Random Forest Evaluation (on held-out test):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.99      0.97      0.98      5686

    accuracy                           1.00     62550
   macro avg       0.99      0.98      0.99     62550
weighted avg       1.00      1.00      1.00     62550

Logistic Regression Evaluation (on held-out test):
              precision    recall  f1-score   support

           0       0.99      0.98      0.98     56864
           1       0.79      0.92      0.85      5686

    accuracy                           0.97     62550
   macro avg       0.89      0.95      0.92     62550
weighted avg       0.97      0.97      0.97     62550
```

Fig. 14 *Evaluation Results of the trained Random Forest model and Logistic Regression model*

Fig. 14 demonstrate the results of the trained Random Forest model and Logistic Regression model. With precision, recall, and F1 at or above 0.97 for both classes and 100% total accuracy, the Random Forest gets near-perfect results. This means that it accurately flags nearly all fraudulent transactions without mistakenly labelling legal ones. The Logistic Regression also does well (97% accuracy), but it misses some fraud cases in exchange for fewer false positives, as evidenced by its lower recall (0.79) on the fraud class. Although the extraordinarily high scores, particularly on Random Forest, indicate that real-world validation is necessary to rule out overfitting, these results show that SMOTE balance and feature scaling have allowed both classifiers to learn the distinctive patterns of fraud efficiently. Next section will be testing the credit card tool for user application.

5.2 Testing

Although no domain experts such as fraud analysts were available to evaluate the effectiveness of the tool in detecting actual fraud since the tool was trained on a simulated, anonymized dataset rather than actual transaction records, user acceptance testing was conducted with the assistance of IT personnel familiar with machine learning. These individuals were able to assess the implementation, behavior, and reliability of the machine learning models used in the system. The project supervisor and fellow students also participated in evaluating the interface, usability, and workflow. While the absence of domain experts limited the ability to validate the effectiveness in real fraud scenarios, the feedback from technically skilled testers helped verify that the system operates as intended and that the machine learning models function correctly. Future work will involve formal testing with fraud analysts once access to appropriate environments and real transactions data is granted.

Table 4 *Functional Test Summary*

Test Case ID	Feature	Expected Outcome	Actual Outcome	Status
FT-01	Registration	A new account has been created, redirected to Login.	The browser shows "Registration successful" once a new user record is added to the users database and takes user to the login page.	Pass
FT-02	Login	Access to the Dashboard is granted with valid credentials.	The dashboard loads with the customised greeting "Welcome, (username)" after the user enters their login and password correctly and sets a session cookie.	Pass
FT-03	Upload well-formed CSV	File approved, summary cards are shown on the analysis page.	CSV file upload was saved to /uploads, a new entry with the accurate timestamp was added to the uploads table, Summary cards are displayed on the analysis page such as Total: 1000 and RF-flagged: 10.	Pass
FT-04	Upload CSV missing column	Schema invalid" is displayed, there is no database insert.	When a CSV file is uploaded without an amount, the page reads "CSV schema invalid: missing 'Amount'" in red, the uploads table shows no new entries.	Pass
FT-05	Download PDF report	Downloads of PDF files with charts and a flagged-transaction table	After selecting "Download Full PDF Report," the browser downloads the PDF file," The PDF opens with a header, two pie charts, and a table of the suspicious transactions.	Pass
FT-06	Change password	The password has been changed; a new one is needed for the next login.	"Password changed successfully" is displayed after entering the old password correctly and matching the new one, logging out and back in with the new password is successful.	Pass

Table 4 demonstrates the summary of the functionality test. User registration, secure login, CSV upload validation, PDF report creation, and password changes are all important features that were thoroughly tested in both error-prone and typical scenarios. All six functional tests went according to plan, although incorrect inputs resulted in clear error messages and no unexpected side effects, valid inputs such as account creation, dashboard access, file analysis, and report download generated the proper results.

Table 5 *Integration Test Summary*

Scenario ID	Workflow	Steps	Status
IT-01	End-to-end upload → analysis → report.	1. Log in 2. Upload CSV 3. View Analysis. 4. Download PDF	Summary cards and charts were shown on the analysis page, selecting "Download Full PDF Report" caused the browser to download the report in pdf file
IT-02	Rejection of an invalid file.	1. Log in 2. Upload CSV missing “Amount” column 3. Observe error handling	The upload page displayed the red error "CSV schema invalid: missing 'Amount'", the uploads table included no new records, and the page stayed there.
IT-03	Session timeout.	1. Log in 2. Remain idle for more than 30 minutes. 3. Click Dashboard.	The browser was returned to the landing page; dashboard access is unavailable until credentials are re-entered.
IT-04	Profile update flow.	1. Log in. 2. Navigate to Profile page 3. Edit first name and email 4. Save changes	When the profile page refreshed, the name and email fields were updated, and the accompanying user table entries were updated. There were no errors visible.

Table 5 documents four end-to-end integration scenarios that verify the compatibility of modules of the tool. Situation After logging in, uploading a legitimate CSV file, viewing the analysis, and getting the PDF report, IT-01 leads users through the entire process, while IT-02 makes sure that corrupted files are detected and handled without crashing. IT-03 confirms that access is terminated and tests session timeout by lingering for more than 30 minutes, whereas IT-04 confirms that profile modifications are instantly recorded and reflected. Every scenario worked, proving that every part of the application from managing front-end forms to processing back-end data to creating reports works in accordance with actual user experiences. The conclusion of the credit card fraud detection tool will be explained in the following section to know whether the tool achieve its purpose and discuss further enhancement of this this project.

6. Conclusion

The credit card fraud detection tool has successfully achieved all the objectives outline in this project. Analysts can now safely register, upload transaction data, and evaluate results without the need for specialised software due to a fully browser-based fraud detection application developed with Flask. To make sure that uncommon fraud cases were appropriately identified, two machine-learning models, Random Forest and Logistic Regression were trained using SMOTE-balanced data. Both classifiers achieved precision and recall rates above 90% on held-out tests. Ultimately, a systematic validation procedure that included functional, integration, and informal user-acceptance testing verified that every feature functions as planned and is user-friendly, from CSV upload to PDF report creation.

Several improvements can bring the tool closer to production readiness in the future. For future work, the models will be retrained using legally obtained, privacy-preserving real transaction data, including live transaction datasets if attainable, to address the limitations of anonymized and simulated datasets. This will help the model to learn from real fraud patterns and spending behavior. Although the current accuracy is high, future improvements will focus on making the model more reliable by using better features and trying other methods like gradient-boosted trees, autoencoders, and tuning the model settings automatically. Adding dashboard filters, configurable alarms, and role-based access control, as well as moving from batch uploads to near-real-time scoring through message queues and microservices, will increase responsiveness and align with organizational operations. The system will continue to be dependable, safe, and simple to manage as it grows due to containerization, CI/CD pipelines, and cloud deployment with autoscaling and monitoring.

In conclusion, this credit card fraud detection tool shows that it is possible to provide robust, machine learning-driven fraud detection using a straightforward online interface. Strong model performance, extensive security measures, and well-designed reporting are all combined in this tool to provide a strong foundation for practical implementation. With the following improvements and domain-expert validation, it can develop into a production-grade solution that aids financial institutions in more efficiently identifying and combating fraud.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Z. F. Shahri, N. A. Abdullah;* ***data collection:*** *Z. F. Shahri, N. A. Abdullah;* ***analysis and interpretation of results:*** *Z. F. Shahri, N. A. Abdullah;* ***draft manuscript preparation:*** *Z. F. Shahri, N. A. Abdullah. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] R. Raman, V. Kumar, B. G. Pillai, D. Rabadiya, R. Divekar, and H. Vachharajani, "Detecting Credit Card Fraud: A Comparative Analysis of KNN, Random Forest, and Logistic Regression Methods," in *2nd IEEE International Conference on Data Science and Information System, ICDSIS 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/ICDSIS61070.2024.10594698.

[2] S. Nami and M. Shajari, "Cost-sensitive payment card fraud detection based on dynamic random forest and k-nearest neighbors," *Expert Syst Appl*, vol. 110, pp. 381–392, Nov. 2018, doi: 10.1016/j.eswa.2018.06.011.

[3] A. Khanum, K. S. Chaitra, B. Singh, and C. Gomathi, "Fraud Detection in Financial Transactions: A Machine Learning Approach vs. Rule-Based Systems," in *Proceedings of the 2nd International Conference on*

Intelligent and Innovative Technologies in Computing, Electrical and Electronics, ICIITCEE 2024, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/IITCEE59897.2024.10467759.

[4] V. Ghai and S. S. Kang, "Role of Machine Learning in Credit Card Fraud Detection," in *Proceedings - 2021 3rd International Conference on Advances in Computing, Communication Control and Networking, ICAC3N 2021*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 939–943. doi: 10.1109/ICAC3N53548.2021.9725540.

[5] P. Tomar, S. Shrivastava, and U. Thakar, "Ensemble Learning based Credit Card Fraud Detection System," in *2021 5th Conference on Information and Communication Technology, CICT 2021*, Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/CICT53865.2020.9672426.

[6] A. RB and S. K. KR, "Credit card fraud detection using artificial neural network," *Global Transitions Proceedings*, vol. 2, no. 1, pp. 35–41, Jun. 2021, doi: 10.1016/j.gltp.2021.01.006.

[7] T. T. Nguyen, H. Tahir, M. Abdelrazek, and A. Babar, "Deep Learning Methods for Credit Card Fraud Detection.". arXiv preprint. https://doi.org/10.48550/arXiv.2012.03754

[8] W. Hilal, S. A. Gadsden, and J. Yawney, "Financial Fraud: A Review of Anomaly Detection Techniques and Recent Advances," May 01, 2022, *Elsevier Ltd*. doi: 10.1016/j.eswa.2021.116429.

[9] E. Kurshan, H. Shen, and H. Yu, "Financial Crime & Fraud Detection Using Graph Computing: Application Considerations & Outlook," in *2020 Second International Conference on Transdisciplinary AI (TransAI)*, IEEE, Sep. 2020, pp. 125–130. doi: 10.1109/TransAI49837.2020.00029.

[10] P. Chatterjee, D. Das, and D. B. Rawat, "Digital twin for credit card fraud detection: opportunities, challenges, and fraud detection advancements," *Future Generation Computer Systems*, vol. 158, pp. 410–426, Sep. 2024, doi: 10.1016/j.future.2024.04.057.

[11] L. Ruff *et al.*, "A Unifying Review of Deep and Shallow Anomaly Detection," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, May 2021, doi: 10.1109/JPROC.2021.3052449.

[12] N. Thockchom, M. M. Singh, and U. Nandi, "A novel ensemble learning-based model for network intrusion detection," *Complex & Intelligent Systems*, vol. 9, no. 5, pp. 5693–5714, Oct. 2023, doi: 10.1007/s40747-023-01013-7.

[13] P. K. Sadineni, "Detection of Fraudulent Transactions in Credit Card using Machine Learning Algorithms," in *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, IEEE, Oct. 2020, pp. 659–660. doi: 10.1109/I-SMAC49090.2020.9243545.

[14] M. A. Ganaie, M. Tanveer, P. N. Suganthan, and V. Snasel, "Oblique and rotation double random forest," *Neural Networks*, vol. 153, pp. 496–517, Sep. 2022, doi: 10.1016/j.neunet.2022.06.012.

[15] M. L. Gambo, A. Zainal, and M. N. Kassim, "A Convolutional Neural Network Model for Credit Card Fraud Detection," in *2022 International Conference on Data Science and Its Applications, ICoDSA 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 198–202. doi: 10.1109/ICoDSA55874.2022.9862930.

[16] V. R. Sonwane, S. Zanje, S. Yenpure, Y. Gunjal, Y. Kulkarni, and R. Yeole, "Advanced Machine Learning Techniques for Credit Card Fraud Detection: A Comprehensive Study," in *2024 5th International Conference on Smart Electronics and Communication (ICOSEC)*, IEEE, Sep. 2024, pp. 1978–1981. doi: 10.1109/ICOSEC61587.2024.10722667.

[17] G. Lei, S. Su, and W. Liao, "Classification of credit card holders based on random forest algorithm," in *2021 The 5th International Conference on Machine Learning and Soft Computing*, New York, NY, USA: ACM, Jan. 2021, pp. 29–32. doi: 10.1145/3453800.3453806.

[18] X. Dong, "Loan Default Prediction based on Machine Learning (LightGBM Model)," *BCP Business & Management*, vol. 25, pp. 457–468, Aug. 2022, doi: 10.54691/bcpbm.v25i.1857.

[19] D. Shah and L. Kumar Sharma, "Credit Card Fraud Detection using Decision Tree and Random Forest," *ITM Web of Conferences*, vol. 53, p. 02012, 2023, doi: 10.1051/itmconf/20235302012.

[20] D. Rzayeva Kapital Bank OJC and S. Malekzadeh, "A Combination of K-Nearest Neighbor and Deep Neural Networks for Credit Card Fraud Detection."

[21] H. Huang, B. Liu, X. Xue, J. Cao, and X. Chen, "Imbalanced credit card fraud detection data: A solution based on hybrid neural network and clustering-based undersampling technique," *Appl Soft Comput*, vol. 154, Mar. 2024, doi: 10.1016/j.asoc.2024.111368.

[22] "FICO® Falcon® Fraud Manager | Falcon Fraud Manager | FICO." Accessed: Dec. 31, 2024. [Online]. Available: https://www.fico.com/en/products/fico-falcon-fraud-manager

[23] "Fraud Protection Advanced | PayPal US." Accessed: Dec. 31, 2024. [Online]. Available: https://www.paypal.com/us/enterprise/fraud-protection-advanced

[24] P. Mookiah, I. Holmes, J. Watkins, and T. O' Connell, "A Real-time Solution for Application Fraud Prevention." [Online]. Available: https://www.sas.com/en_us/software/fraud-management.html

[25] "What Is Agile Methodology? (A Beginner's Guide) [2024] • Asana." Accessed: Dec. 19, 2024. [Online]. Available: https://asana.com/resources/agile-methodology