CREDIT CARD FRAUD DETECTION USING PYTHON

A

MAJOR PROJECT-II REPORT

Submitted in partial fulfillment of the requirements for the

degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

By

GROUP NO.30

Ashutosh Rathore	**0187CS211041**
Shreya Singh Rathore	**0187CS211160**
Chirag Dewangan	**0187CS211055**
Ajay Meena	**0187CS211016**

Under the guidance of

Prof. Mayank Kurchuniya

(Assistant Professor)

Department of Computer Science & Engineering
Sagar Institute of Science &Technology (SISTec), Bhopal (M.P)

Approved by AICTE, New Delhi & Govt. of M.P.
Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)

June -2025

Sagar Institute of Science & Technology (SISTec), Bhopal (M.P)

Department of Computer Science & Engineering

CERTIFICATE

We hereby certify that the work which is being presented in the B.Tech. Major Project-II Report entitled **CREDIT CARD FRAUD DETECTION,** in partial fulfillment of the requirements for the award of the degree of ***Bachelor of Technology,*** submitted to the Department of **Computer Science & Engineering**, Sagar Institute of Science & Technology (SISTec)**,**Bhopal (M.P.) is an authentic record of our own work carried out during the period from Jan-2025 to Jun-2025 under the supervision of **Prof. Mayank Kurchuniya.**

The content presented in this project has not been submitted by us for the award of any other degree elsewhere.

Ashutosh Rathore	**Chirag Dewangan**	**Shreya Rathore**	**Ajay Meena**
0187CS211041	**0187CS211055**	**0187CS211160**	**0187CS211016**

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Date:

Prof. Mayank Kurchuniya	**Dr. Amit Kumar Mishra**	**Dr. D.K. Rajoriya**
Project Guide	**HOD,CSE**	**Principal**

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. D. K. Rajoriya**, **Principal, SISTec and Dr. Swati Saxena, Vice Principal SISTec** Gandhi Nagar, Bhopal for giving us an opportunity to undertake this project.

We also take this opportunity to express a deep sense of gratitude to **Dr. Amit Kumar Mishra, HOD, Department of Computer Science & Engineering** for his kindhearted support

We extend our sincere and heartfelt thanks to our guide, **Prof. Mayank Kurchuniya,** for providing us with the right guidance and advice at crucial junctures and for showing us the right way.

We are thankful to the **Project Coordinator, Prof. Deepti Jain,** who devoted her precious time in giving us the information about various aspects and gave support and guidance at every point of time.

We would like to thank all those people who helped us directly or indirectly to complete our project whenever we found ourself in any issue,

TABLE OF CONTENTS

ABSTRACT

The rapid growth of digital financial transactions has led to a significant rise in fraudulent activities, necessitating the development of efficient and accurate fraud detection systems. This project presents a machine learning-based solution for credit card fraud detection using Python. Leveraging a publicly available dataset from Kaggle, the system analyzes 284,807 transactions, of which only 0.172% are fraudulent, making the dataset highly imbalanced. The features undergo preprocessing, including normalization and dimensionality reduction using Principal Component Analysis (PCA). To address class imbalance, a random under-sampling technique is applied, resulting in a balanced dataset. Logistic regression is used as the classification model, achieving approximately 95% accuracy, high precision, strong recall, and an AUC score of 0.95. The model is further evaluated on the original imbalanced dataset to verify generalization. This project demonstrates an effective approach to identifying fraudulent transactions and highlights future improvements through advanced algorithms and enhanced feature engineering.

LIST OF ABBREVIATIONS

ACRONYM	FULL FORM
PCA	Principal Component Analysis
AUC	Area Under Curve
ROC	Receiver Operating Characteristic
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machines
KNN	K-Nearest Neighbours
HMM	Hidden Markov Models
PII	Personally Identifiable Information
WAF	Web Application Firewalls
RBAC	Role-Based Access Control

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

In today's technology-driven world, the use of credit cards has become an integral part of daily financial transactions. From e-commerce platforms and online subscriptions to retail stores and fuel stations, credit card payments offer convenience and speed. However, this increasing reliance on digital payments has opened new avenues for cybercriminals to exploit vulnerabilities in the system. As a result, credit card fraud has emerged as a major concern globally, with financial institutions, banks, and consumers facing heavy losses and legal complications.

Traditionally, fraud detection relied on manual verification and rule-based systems where predefined patterns or thresholds triggered alerts. While effective to some extent, these methods are often limited in flexibility and fail to adapt to emerging fraud strategies. Static rules quickly become outdated, especially as cybercriminals evolve and invent new ways to circumvent them. Manual checks are also resource-intensive and not scalable for systems that process millions of transactions in real time.

This is where the power of machine learning comes into play. Machine learning techniques are capable of analysing massive datasets, identifying subtle patterns, and adapting to new trends without explicit programming. By learning from historical data, machine learning models can classify transactions as either legitimate or suspicious based on the behavior and attributes of each entry. This ability to dynamically learn and detect anomalies makes machine learning a superior solution for fraud detection.

This project aims to address these challenges by implementing a credit card fraud detection system using machine learning techniques in Python. The system leverages a real-world dataset and goes through all essential stages—data analysis, preprocessing, model training, evaluation, and optimization—to detect fraudulent transactions effectively. The end goal is to build a model that not only performs well on a balanced dataset but also generalizes effectively to the original imbalanced one, simulating real-world conditions.

By combining data science practices with domain-specific insights, this project contributes to the broader effort of enhancing financial cybersecurity and building intelligent systems that safeguard user transactions. The approach and results outlined in this project can also be

extended to other fraud-prone domains like insurance, loan processing, and digital wallets, making it a valuable contribution to the financial technology space.

1.1 ABOUT PROJECT

This project focuses on designing and implementing a machine learning-based credit card fraud detection system using Python. The key motivation behind the project is to identify fraudulent transactions from a large pool of credit card data, where only a small fraction represents actual fraud. The dataset used is the well-known Credit Card Fraud Detection dataset from Kaggle, which includes 284,807 transactions carried out by European cardholders in September 2013. The data has been anonymized using Principal Component Analysis (PCA) to preserve confidentiality.

The dataset contains 31 features: a 'Time' variable representing the seconds elapsed between each transaction and the first transaction, an 'Amount' variable indicating the transaction value, 28 anonymized features (V1 to V28), and a 'Class' label (0 for normal and 1 for fraud). A critical challenge in working with this dataset is its extreme imbalance, with fraudulent transactions making up only 0.172% of the total data—only 492 cases out of nearly 285,000. Handling such imbalance is vital to ensure that the machine learning model is not biased toward predicting the majority class.

The core model used in this project is Logistic Regression, a well-known binary classification algorithm. The dataset is split into training (80%) and testing (20%) sets using stratified sampling to maintain class proportions. The model is then trained and evaluated on the test set using various performance metrics. Key performance indicators include accuracy (~95%), precision, recall, F1-score, and AUC (Area Under Curve) of the ROC (Receiver Operating Characteristic) curve, which was approximately 0.95, indicating excellent predictive capability.

Furthermore, to ensure real-world applicability, the model is tested on the original imbalanced dataset to evaluate how well it generalizes. This is critical because in real financial systems, fraud detection models operate in naturally imbalanced environments. The project concludes with a discussion on limitations and future improvements, such as exploring SMOTE (Synthetic Minority Oversampling Technique), ensemble learning models, XGBoost, and even neural networks for improved accuracy and generalization.

1.2 PROJECT OBJECTIVE

The project is built around a well-defined set of objectives aimed at creating a robust fraud detection model that can accurately classify transactions as fraudulent or legitimate. These objectives guide the development of each stage of the project, from data acquisition to evaluation and future planning.

1.2.1. TO UNDERSTAND AND EXPLORE THE DATASET THOROUGHLY

Load the dataset using pandas and inspect its dimensions, data types, and feature descriptions. Use summary statistics and visualizations to understand the behavior of variables such as 'Amount', 'Time', and the PCA-transformed features (V1–V28). Examine the distribution of the 'Class' variable to quantify the imbalance and assess its impact on model performance. Identify and confirm the absence of missing or null values to avoid potential preprocessing issues.

1.2.2. TO PERFORM STATISTICAL AND VISUAL ANALYSIS OF TRANSACTIONS

Separate fraudulent and non-fraudulent transactions to compare their behavior. Analyze key metrics such as mean and median transaction amounts in both classes. Determine the usefulness of the 'Time' feature by checking for temporal fraud patterns. Use matplotlib and seaborn to visualize distributions, histograms, and boxplots to reveal hidden trends.

1.2.3. TO PREPROCESS AND TRANSFORM DATA FOR MODELING

Apply StandardScaler to normalize the 'Amount' variable for better convergence during model training. Drop the 'Time' column based on analysis showing its limited impact on detection. Balance the dataset using under-sampling to create a 1:1 ratio of fraud and normal transactions for training. Prepare dataframes for model input by separating features and target labels.

1.2.4. TO BUILD AND TRAIN AN EFFECTIVE CLASSIFICATION MODEL

Use the balanced dataset to train a Logistic Regression model, chosen for its interpretability and efficiency. Apply stratified splitting to ensure the class distribution is maintained across train-test sets. Train the model using the fit() method and record training time and loss, if needed. Save the trained model for future inference and real-time prediction usage.

1.2.5. TO EVALUATE MODEL PERFORMANCE USING APPROPRIATE METRICS

Use confusion matrix to observe true positives, false positives, true negatives, and false negatives. Calculate accuracy, precision, recall, and F1-score to evaluate different aspects of

performance. Plot the ROC curve and calculate AUC to assess the model's ability to distinguish between classes. Test the model against the full original dataset to verify its generalization to real-world class distributions.

1.2.6. TO IDENTIFY LIMITATIONS AND SUGGEST FUTURE ENHANCEMENTS

Discuss the drawbacks of under-sampling, such as loss of important non-fraud information. Propose oversampling methods like SMOTE to generate synthetic fraud samples. Recommend using ensemble models like Random Forest, Gradient Boosting, and XGBoost to enhance performance. Suggest deep learning techniques and advanced feature engineering, such as time-based behavior analysis and pattern recognition using neural networks.

CHAPTER 2

SOFTWARE AND HARDWARE REQUIREMENTS

The implementation of a credit card fraud detection system using machine learning involves both software tools for development and analysis, and hardware infrastructure to support the execution of computational tasks. Ensuring that the right tools and environments are available is critical for seamless development, testing, and validation of the model. This section details all the necessary components used throughout the project.

2.1. Software Requirements

The following software tools and libraries were essential for developing the machine learning pipeline:

2.1.1 Python (version 3.7 or above)

Python serves as the core programming language for the project. It is widely used in the data science and machine learning community due to its simplicity, readability, and availability of powerful libraries.

2.1.2 Jupyter Notebook / Google Colab

These interactive development environments were used for writing, running, and debugging the code.

2.1.3 pandas

A fundamental library for data manipulation and analysis. It was used to load the dataset, clean and preprocess the data, handle missing values, and explore statistical summaries.

2.1.4 Numpy

This library supports efficient numerical operations. It provides data structures like arrays and matrix operations that are crucial in data preprocessing and feature engineering.

2.1.5 scikit-learn (sklearn)

It provided tools for model building, dataset splitting, evaluation metrics, logistic regression implementation, and scaling of features using StandardScaler.

2.1.6 matplotlib

A powerful library for 2D visualizations. It was used to generate plots such as the ROC curve and histograms to understand feature distributions.

2.1.7 seaborn

Built on top of matplotlib, this library provides aesthetically pleasing visualizations. It was used for plotting the confusion matrix, heatmaps, and distribution plots for better data understanding.

2.2. Hardware Requirements

Though the project can run on standard hardware, certain minimum and recommended specifications help ensure faster execution and model training.

2.2.1 Processor (CPU)

A minimum of Intel i3 or AMD Ryzen 3 processor is required. For better performance, especially when dealing with large datasets or training complex models, Intel i5, i7, or equivalent processors are recommended.

2.2.2 RAM (Memory)

At least 4 GB RAM is needed to run basic model training and data analysis. However, for better performance, especially while working on cloud platforms or running visualizations and model evaluations, 8 GB or more is highly recommended.

2.2.3 Internet Connection

A stable internet connection is required to download datasets from Kaggle, install Python libraries, use Google Colab, and access documentation or external resources. Cloud-based training also depends heavily on good bandwidth.

2.2.4 Input Devices

A standard keyboard and mouse setup is required for programming and navigation through development environments. Precision in input devices helps speed up the development process.

CHAPTER 3
PROBLEM STATEMENT

3.1 RISE IN CREDIT CARD FRAUD

With the exponential growth of online payments and digital transactions, credit card fraud has become a significant and persistent threat to global financial security. Fraudsters use stolen credentials, phishing techniques, and advanced cyber tactics to make unauthorized transactions. These fraudulent activities result in not only financial losses but also erosion of customer trust and legal consequences for financial institutions.

3.2 CHALLENGES IN DETECTING FRAUD

Fraudulent transactions are often designed to appear similar to legitimate user behavior, making them difficult to isolate using traditional rule-based systems. Static rules are unable to keep up with the rapidly changing fraud patterns, leading to delayed responses or overlooked attacks. Furthermore, such systems often raise false alarms, causing inconvenience to customers whose genuine transactions get flagged unnecessarily.

3.3 HIGHLY IMBALANCED DATA

In fraud detection datasets, legitimate transactions vastly outnumber fraudulent ones, with fraud cases often constituting less than 0.2% of the total data. This extreme imbalance can mislead machine learning models into predicting only the majority class, thereby ignoring the minority class completely. As a result, models might achieve high accuracy but fail to detect actual fraud cases effectively.

3.4 HIGHLY IMBALANCED DATANEED FOR INTELLIGENT SOLUTIONS

There is an urgent need for adaptive and intelligent fraud detection systems that can learn from historical data and continuously improve with time. Machine learning algorithms can analyze large volumes of transaction data, extract patterns, and detect subtle anomalies indicative of fraudulent behavior. Such models can be trained to make real-time decisions, minimizing both false negatives and false positives.

3.5 PROJECT AIM

This project focuses on building a machine learning model to detect credit card fraud using the logistic regression algorithm. The model is trained on a publicly available dataset from Kaggle, containing anonymized European transaction data. Key tasks include handling data imbalance, feature scaling, training and testing the model, and evaluating its effectiveness through metrics like accuracy, precision, recall, F1-score, and AUC.

3.6 FUTURE SCOPE AND GENERALIZATION

The system is designed not only to work on a balanced dataset but also to generalize well on the original imbalanced dataset, ensuring practical applicability. The methodology used can be extended to larger datasets or integrated into real-time fraud detection systems. Future improvements could involve the use of more complex models like Random Forests or neural networks, as well as the incorporation of advanced techniques like SMOTE and ensemble learning.

CHAPTER 4
LITERATURE SURVEY

From [1], this paper presents valuable insights from a real-world practitioner's viewpoint on designing and implementing credit card fraud detection systems. The authors analyze the limitations of conventional fraud detection techniques and propose practical strategies to overcome challenges such as data imbalance, delayed labelling, and adaptive fraud behavior. Their study emphasizes the importance of model evaluation using cost-based metrics instead of accuracy alone, and they advocate for incremental learning approaches to handle concept drift in fraud patterns. This research contributes real-world knowledge to bridge the gap between academic methods and industry practices.

From [2], this study performs a comprehensive comparison of several machine learning algorithms—such as decision trees, support vector machines (SVM), neural networks, and logistic regression—for detecting fraudulent credit card transactions. The authors analyze the performance of each algorithm using precision, recall, and ROC metrics on a real dataset. They found that no single classifier consistently outperforms the others across all metrics, and stress the importance of selecting models based on the specific objectives of fraud detection (e.g., minimizing false positives vs. detecting more frauds). Their comparative framework provides a strong foundation for selecting suitable algorithms in future projects.

From [3], this paper introduces a novel fusion-based methodology combining the Dempster–Shafer theory of evidence with Bayesian learning to improve fraud detection accuracy. The approach integrates multiple indicators and sources of information, allowing the system to make more robust and reliable decisions under uncertainty. The hybrid model demonstrates improved accuracy and resilience in detecting fraud compared to traditional single-model systems. This research highlights the value of multi-strategy fusion for complex fraud detection tasks.

From [4], this research focuses on the critical role of feature engineering in enhancing the performance of machine learning models for fraud detection. The authors propose a set of derived features based on historical spending behavior, transaction frequency, and user patterns. By incorporating these features into their models, they achieve significant improvements in precision and recall. The paper concludes that carefully crafted features can significantly outperform more complex models trained on raw data, thus establishing feature engineering as a cornerstone of effective fraud detection pipelines.

From [5], Fiore and colleagues explore the application of Generative Adversarial Networks (GANs) to address the issue of class imbalance in fraud detection datasets. GANs are used to generate synthetic fraudulent transactions to augment the minority class during training. Their experiments show that this approach enhances model learning and improves classification performance, particularly recall, on imbalanced datasets. This work demonstrates the innovative use of deep learning for data augmentation in highly skewed classification problems.

From [6], This study demonstrates the effectiveness of ensemble learning, particularly AdaBoost combined with a majority voting strategy, in detecting credit card fraud. By aggregating multiple weak learners, the authors achieve a strong classifier that significantly outperforms traditional single-algorithm models. Their results indicate improved detection rates and reduced false positives. This research supports the adoption of ensemble methods as a robust solution for fraud detection tasks where precision and adaptability are critical.

From [7], Zhang and co-authors propose HOBA, a new methodology for feature engineering that integrates deep learning techniques to extract high-level features from transaction data. The approach leverages both manual and automated feature extraction processes to build a comprehensive input set for classification models. Their deep learning architecture achieves higher accuracy and better generalization on unseen data. The study demonstrates how deep learning can be effectively combined with domain-specific feature knowledge to boost fraud detection performance.

From [8], the authors tackle the problem of realistic modeling for fraud detection by considering the temporal nature of transactions and concept drift. They introduce a novel learning strategy based on adaptive sliding windows to capture time-dependent patterns and evolving fraud behaviors. Their method outperforms static classifiers in long-term performance and adapts well to changing fraud trends. This work is crucial for designing systems that remain effective in dynamic, real-world settings.

From [9], this conference paper presents a comparative study of several machine learning techniques applied to the task of credit card fraud detection. Algorithms such as decision trees, logistic regression, SVM, and k-nearest neighbors (KNN) are evaluated on a benchmark dataset. The authors discuss model performance in terms of detection accuracy, training time, and interpretability. Their findings serve as a useful guide for selecting appropriate models based on use-case priorities like speed or transparency.

From [10], Lucas and colleagues propose an automated feature engineering framework based on Hidden Markov Models (HMMs) to capture temporal patterns in transaction sequences. The multi-perspective approach enables the extraction of behavioral features from both the user and the transaction history. Their results show that this technique enhances fraud detection performance without manual feature construction. This paper contributes to the growing interest in automated machine learning (AutoML) for fraud detection applications.

From [11], this book is considered a foundational resource for understanding the core principles of data mining and machine learning. It covers various techniques such as classification, clustering, association analysis, and anomaly detection, all of which are applicable to fraud detection. The authors provide in-depth explanations of algorithms like decision trees, SVM, neural networks, and ensemble models. Particularly relevant is the section on outlier and rare event detection, which directly aligns with the problem of identifying fraudulent credit card transactions. The book also discusses real-world applications, performance metrics, and practical issues in data preprocessing and model evaluation.

From [12], this book offers a rigorous, mathematical approach to statistical learning, which is central to machine learning-based fraud detection. It covers both supervised and unsupervised learning techniques, including logistic regression, discriminant analysis, neural networks, and boosting. The book provides valuable insights into the bias-variance trade-off, regularization methods, and model selection strategies, which are crucial for building robust fraud detection models. Its deep theoretical grounding makes it ideal for understanding the nuances of model performance in the presence of imbalanced and noisy datasets.

CHAPTER 5

SOFTWARE REQUIREMENTS SPECIFICATION

5.1 FUNCTIONAL REQUIREMENTS

5.1.1 DATA PREPROCESSING

The system should load and clean transaction data, handle missing values, and scale features for model training.

5.1.2 FRAUD DETECTION MODEL

The system should implement and train a machine learning model (e.g., logistic regression) to classify transactions as fraudulent or legitimate.

5.1.3 IMBALANCED DATASET HANDLING

The system should apply techniques like under-sampling or SMOTE to balance the dataset before model training.

5.1.4 MODEL EVALUATION

The system should evaluate the model's performance using metrics such as accuracy, precision, recall, F1-score, and AUC.

5.1.5 REAL-TIME PREDICTION

The system should allow for real-time prediction of fraud status for incoming credit card transactions.

5.1.6 ALERTS AND NOTIFICATIONS

The system should generate alerts or notifications for detected fraudulent transactions.

5.2 NON-FUNCTIONAL REQUIREMENTS

5.2.1 PERFORMANCE

The system should make predictions in real-time with low latency.

5.2.2 SCALABILITY

The system should be scalable to handle increasing amounts of transaction data over time.

5.2.3 ACCURACY

The system should achieve high accuracy in detecting fraud while minimizing false positives.

5.2.4 RELIABILITY

The system should be robust and dependable, with minimal downtime or errors.

5.2.5 SECURITY

The system should ensure secure handling of sensitive transaction data, complying with industry standards.

5.2.6 USABILITY

The system should have a user-friendly interface for monitoring and interpreting results, such as fraud detection alerts and model performance metrics.

5.3 PERFORMANCE

5.3.1 REAL-TIME FRAUD DETECTION CAPABILITY

The system should process individual transactions in less than 1 second. The response time must remain consistent under normal and peak loads.

5.3.2 MODEL ACCURACY

It must maintain at least 95% accuracy on a balanced dataset and it should minimize false positives to prevent legitimate transactions from being blocked.

5.3.3 HIGH THROUGHOUT

It should handle at least 200–300 transactions per second. The architecture should support parallel processing if needed.

5.3.4 OPTIMIZED RESOURCE USAGE

Memory consumption should remain under 1 GB during inference. CPU usage should not exceed 60% in normal conditions.

5.3.5 EFFICIENT DATA HANDLING

Large datasets should be processed in under 10 seconds during preprocessing. Batch operations should be optimized using vectorized functions and libraries like NumPy.

5.3.6 CONSISTENT AUC SCORE

AUC score should consistently be ≥ 0.90 for model validation and it should be monitored after each model update to ensure performance consistency.

5.4 SECURITY

5.4.1 ENCRYPTED DATA TRANSMISSION

All communication between users, databases, and servers must be conducted over secure HTTPS connections using SSL/TLS encryption protocols. APIs and endpoints must enforce encrypted channels to prevent man-in-the-middle (MITM) attacks or data sniffing during transmission.

5.4.2 DATA MASKING AND ANONYMIZATION

Personally identifiable information (PII) such as credit card numbers, CVV codes, user IDs, and billing addresses must be either masked or removed before being logged or used in training. Use of hashed or tokenized identifiers is required in both the training dataset and during real-time prediction.

5.4.3 ROLE-BASED ACCESS CONTROL (RBAC)

The system should implement fine-grained access control where different roles (Admin, Developer, Analyst, Auditor) are assigned specific permissions. For example, Admins can modify system settings and retrain models, Analysts can view predictions and dashboards, and Auditors can access logs only.

5.4.4 ACCESS LOGS AND AUDITING

The system must log all user actions including login/logout events, data uploads, model updates, and API calls, along with timestamps and IP addresses. Logs should be stored in secure servers, accessible only to admin-level users or auditors.

5.4.5 SECURITY MONITORING AND INTRUSION PREVENTION

Use of firewalls and Web Application Firewalls (WAFs) to monitor incoming traffic and prevent injection attacks, DDoS attacks, and brute-force attempts.

5.5 USABILITY

5.5.1 USER-FRIENDLY INPUT INTERFACE

A form or upload panel to accept CSV or JSON input formats. Real-time validation of uploaded data structure.

5.5.2 CLEAR PREDICTION OUTPUT

Each transaction is labeled as “Fraud” or “Normal” with a confidence score. Include tooltips or hover info to explain prediction results.

5.5.3 PERFORMANCE DASHBOARD

It displays metrics like precision, recall, and accuracy using visual aids. It should allow selection of timeframes (e.g., last 7 days, monthly) for performance trends.

5.5.4 ALERT SYSTEM

Fraud detection alerts should be shown in red or highlighted boxes. Users can export flagged transactions for review.

5.5.5 MINIMAL LEARNING CURVE

Navigation must follow a logical flow with labeled buttons and menus. It helps sections or walkthrough tutorials should be included for first-time users.

5.5.6 COMPREHENSIVE DOCUMENTATION

It includes step-by-step setup and usage guide. It also explains terms like AUC, Precision, and Recall in a glossary section.

CHAPTER 6
SOFTWARE DESIGN

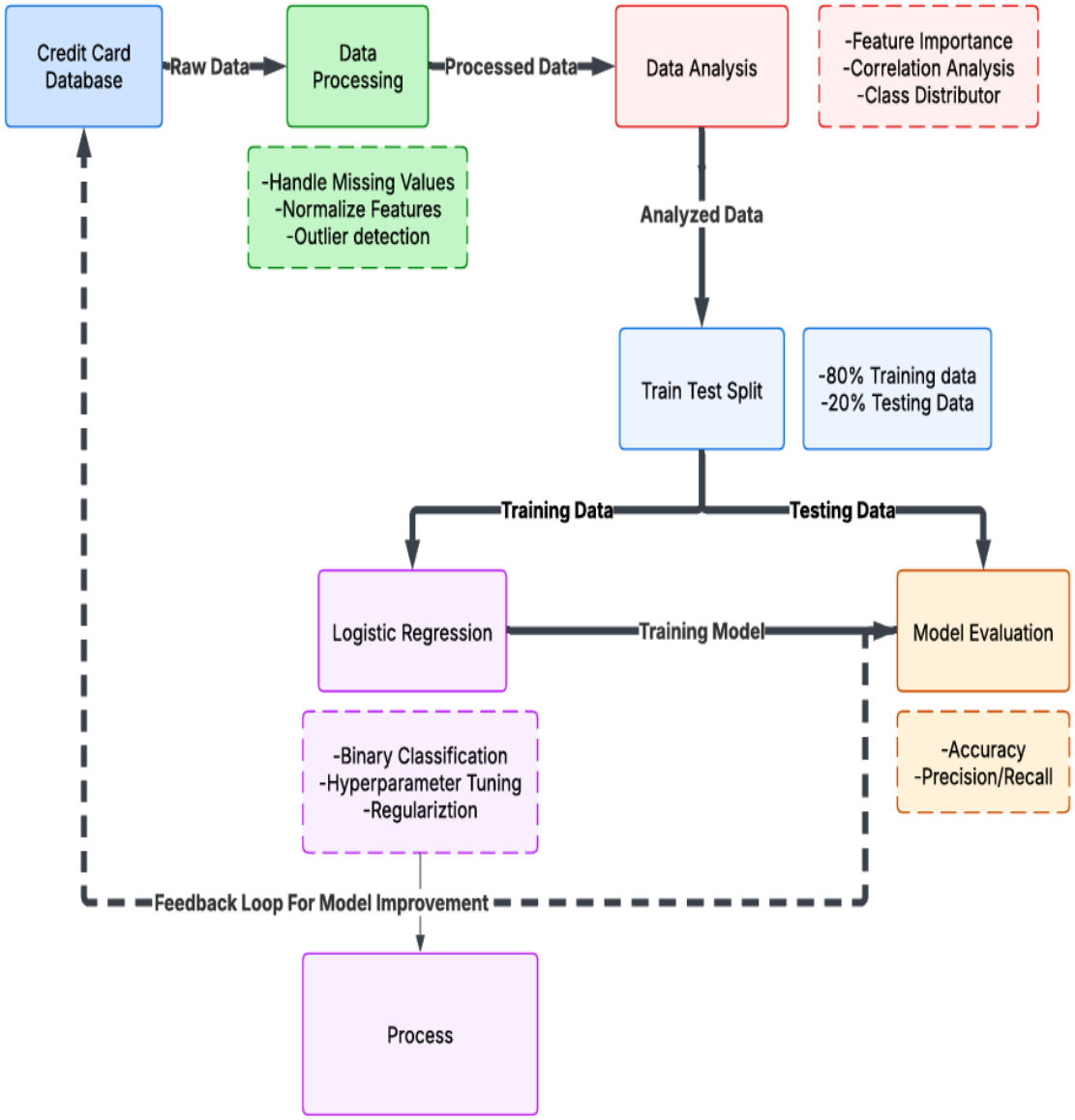

Figure 6.1: Data Flow Diagram

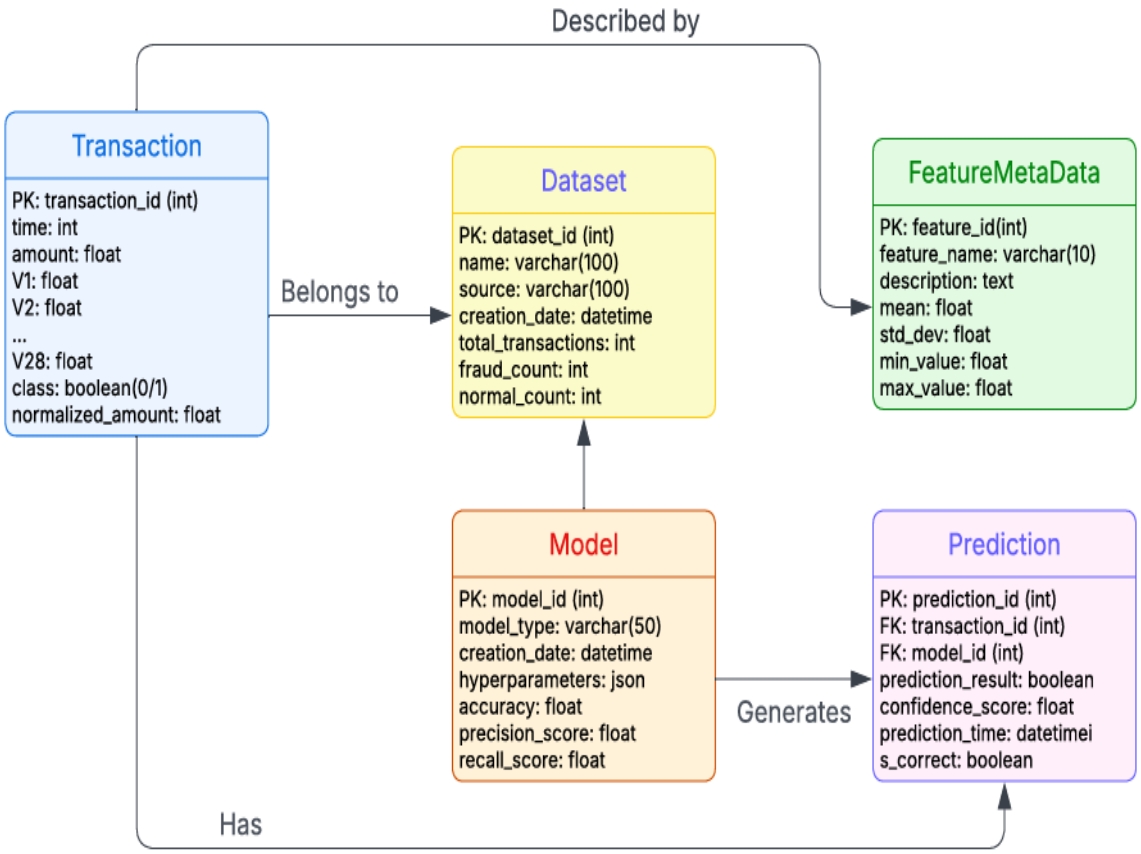

Figure 6.2: ER Diagram

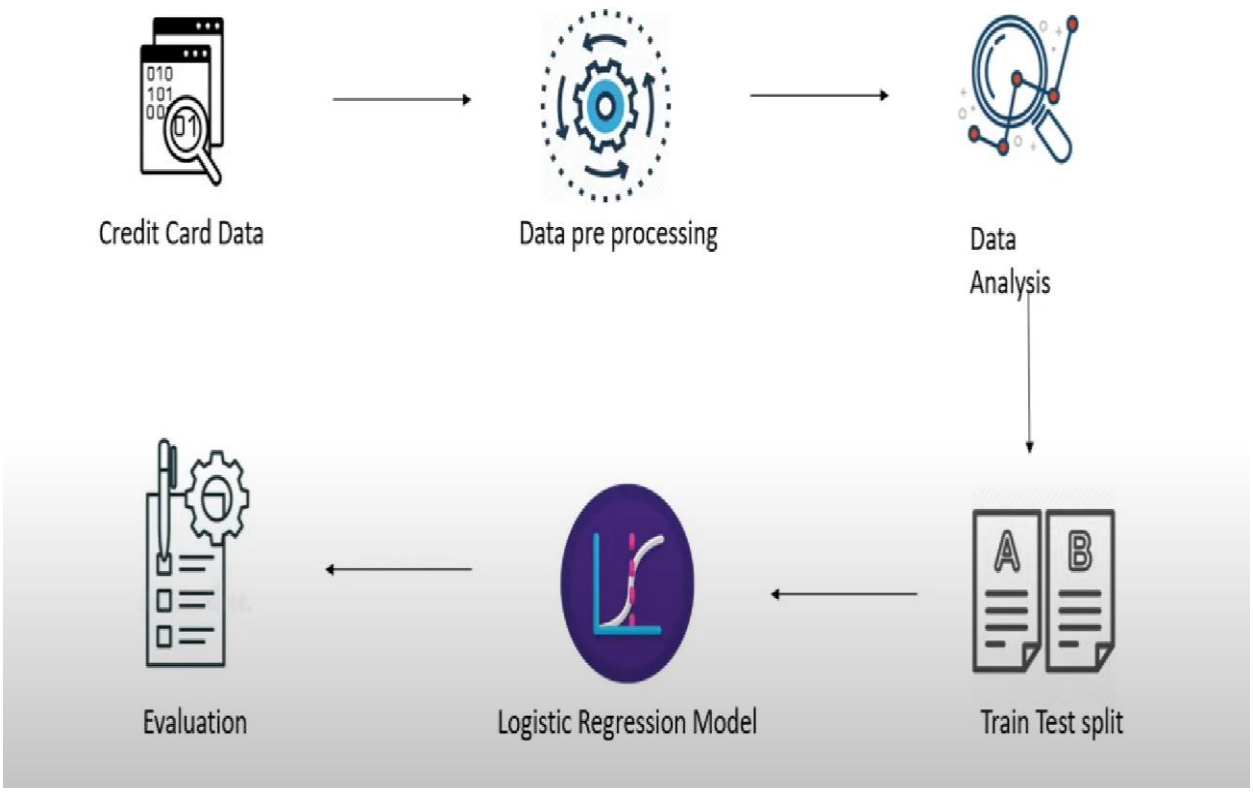

Figure 6.3: Architecture

CHAPTER 7
OUTPUT SCREENS

7.1 OUTPUT SCREENS

Figure 7.1.1: Homepage

Figure 7.1.2: Credit Cards Dataset

Figure 7.1.3: Dataset Evaluation

Figure 7.1.4: Legitimate Transaction Result

CHAPTER 8
DEPLOYMENT

8.1 OVERVIEW

Deploying a credit card fraud detection model in a real-world scenario involves converting the machine learning prototype into a usable service or product that can interact with other systems and end-users. Although this project was developed and tested in a research or academic environment (such as Jupyter Notebook), the following deployment plan outlines how the model can be transitioned into a functional and scalable fraud detection tool.

8.2 DEPLOYMENT GOALS

The primary objective of deploying the credit card fraud detection system is to make the trained machine learning model accessible and usable in real-time applications. The deployment aims to provide a reliable interface where incoming credit card transactions can be analyzed instantly to identify potentially fraudulent activity. It is essential that the deployed system ensures fast and accurate predictions, enabling financial institutions or businesses to take immediate action when fraud is detected. Additionally, the deployment should support scalability to handle high volumes of transaction data efficiently without compromising performance.

Another key goal is to maintain the security and privacy of sensitive financial information during the transaction and detection processes. The system must be built with proper encryption and authentication mechanisms to protect user data and prevent unauthorized access. Furthermore, the deployment should include tools for monitoring the model's performance in a live environment, allowing for the detection of issues such as model drift or increasing false positives. Over time, the model should be adaptable to new types of fraudulent patterns by enabling periodic retraining with fresh data. Overall, the deployment phase bridges the gap between model development and practical application, making the system valuable and actionable in real-world scenarios.

8.3 USER INTERFACE

The user interface (UI) for the credit card fraud detection system is designed to be simple, intuitive, and functional. It allows users—such as analysts or system administrators—to input transaction details manually or upload batch data for fraud analysis. The UI displays the

prediction results clearly, highlighting whether a transaction is classified as fraudulent or legitimate. For enhanced usability, the interface may include features such as a dashboard to view flagged transactions, summary statistics, and alerts for high-risk activity. Tools like basic web interface using HTML and JavaScript can be used to build the UI, providing an accessible front end for interacting with the backend model.

8.4 MONITORING AND MAINTENANCE

Monitoring and maintenance are essential to ensure the fraud detection system remains accurate, reliable, and up to date. After deployment, the system should be continuously monitored for performance metrics such as prediction accuracy, false positive rate, and response time. Logs of all transactions and model decisions should be recorded for audit and analysis. Periodic retraining of the model with new data is recommended to adapt to evolving fraud patterns. Additionally, security updates, bug fixes, and infrastructure maintenance should be regularly performed to keep the system stable and secure in a production environment.

8.5 DEPLOYMENT WORKFLOW SUMMARY

The deployment workflow for the credit card fraud detection system begins with training the machine learning model and saving both the model and the preprocessing pipeline using tools like joblib or pickle. Once the model is finalized, a backend application is developed using frameworks such as Flask or FastAPI to serve the model through a REST API. This backend is then containerized using Docker to ensure consistent deployment across environments. The Docker container is deployed to a cloud platform such as AWS, Google Cloud, or Azure, enabling scalable and remote access to the prediction service. Optionally, a simple front-end interface or dashboard is created to allow users to input transaction data and view results in real time. Finally, the system is set up for continuous monitoring and periodic maintenance to track performance, manage logs, and retrain the model with new data as needed, ensuring the deployment remains robust and effective over time.

REFERENCES

JOURNALS/ RESEARCH PAPERS

1. Dal Pozzolo, A., Caelen, O., Le Borgne, Y. A., Waterschoot, S., & Bontempi, G. (2014). "Learned lessons in credit card fraud detection from a practitioner perspective."
2. Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). "Data mining for credit card fraud: A comparative study."
3. Panigrahi, S., Kundu, A., Sural, S., & Majumdar, A. K. (2009). "Credit card fraud detection: A fusion approach using Dempster–Shafer theory and Bayesian learning."
4. Bahnsen, A. C., Aouada, D., Stojanovic, A., & Ottersten, B. (2016). "Feature engineering strategies for credit card fraud detection."
5. Fiore, U., De Santis, A., Perla, F., Zanetti, P., & Palmieri, F. (2019). "Using generative adversarial networks for improving classification effectiveness in credit card fraud detection."
6. Randhawa, K., Loo, C. K., Seera, M., Lim, C. P., & Nandi, A. K. (2018). "Credit card fraud detection using AdaBoost and majority voting."
7. Zhang, X., Han, Y., Xu, W., & Wang, Q. (2021). "HOBA: A novel feature engineering methodology for credit card fraud detection with a deep learning architecture."
8. Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2018). "Credit card fraud detection: A realistic modeling and a novel learning strategy."
9. Varmedja, D., Karanovic, M., Sladojevic, S., Arsenovic, M., & Anderla, A. (2019). "Credit card fraud detection-machine learning methods."
10. Lucas, Y., Portier, P. E., Laporte, L., He-Guelton, L., Caelen, O., Granitzer, M., & Calabretto, S. (2020). "Towards automated feature engineering for credit card fraud detection using multi-perspective HMMs."

BOOKS

11. Han, J., Kamber, M., & Pei, J. (2012). Data Mining: Concepts and Techniques (3rd Edition). Morgan Kaufmann.
12. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd Edition). Springer.

PROJECT SUMMARY

About Project

Title of the project	Credit Card Scam Detection using Python
Semester	8th
Members	4 members
Team Leader	Ashutosh Rathore
Describe role of every member in the project	Chirag Dewangan (Coding) Ashutosh Rathore (Data Cleaning, Presentation) Shreya Singh Rathore (Documentation) Ajay Meena (Reference, Research Paper)
What is the motivation for selecting this project?	The motivation behind selecting this project is the growing threat of credit card fraud in today's digital economy. With the rapid increase in online transactions, detecting fraudulent activity has become critically important. Machine learning provides a powerful solution to identify suspicious patterns in large datasets. This project aims to contribute towards secure financial systems using intelligent automation.
Project Type **(Desktop Application, Web Application, Mobile App, Web)**	Standalone Desktop Application

Tools &Technologies

Programming language used	Python
Compiler used **(with version)**	Jupyter Notebook
IDE used **(with version)**	Jupyter Notebook
Front End Technologies **(with version, wherever Applicable)**	
Back End Technologies **(with version, wherever applicable)**	
Database used **(with version)**	

Software Design& Coding

Is prototype of the software developed?	Yes
SDLC model followed (Waterfall, Agile, Spiral etc.)	Waterfall
Why above SDLC model is followed?	The Waterfall model is followed because it provides a clear and linear development structure, ideal for projects with well-defined requirements. It allows thorough documentation and phase-wise progress tracking. Since the dataset and objectives were fixed, there was minimal need for changes during development. This model ensured systematic completion of each phase before proceeding to the next.
Justify that the SDLC model mentioned above is followed in the project.	The Waterfall model is justified in this project as each phase was executed in a defined sequence—from requirement gathering to model deployment. Data was analyzed first, followed by preprocessing, model building, evaluation, and final reporting. No phase was revisited once completed, aligning with the Waterfall model's structured flow. This ensured clarity, discipline, and smooth project progression.
Software Design approach followed (Functional or Object Oriented)	The Functional design approach was followed in this project. The implementation focused on structured code using functions for data preprocessing, model training, and evaluation. It was suitable for this task-oriented and script-based machine learning workflow.
Name the diagrams developed (According to the Design approach followed)	Data Flow Diagram, Circuit Diagram, Architecture Diagram
In case Object Oriented approach is followed, which of the OOPS principles are covered in design?	
No. of Tiers (example 3-tier)	1
Total no. of front-end pages	1
Total no. of tables in database	
Database in which Normal Form?	
Are the entries in database encrypted?	
Front end validations applied (Yes / No)	No
Session management done (in case of web applications)	
Is application browser compatible (in case of web applications)	
Exception handling done (Yes / No)	No

Commenting done in code (Yes / No)	Yes
Naming convention followed (Yes / No)	
What difficulties faced during deployment of project?	
Total no. of Use-cases	
Give titles of Use-cases	

Project Requirements

MVC architecture followed (Yes / No)	No
If yes, write the name of MVC architecture followed (MVC-1, MVC-2)	
Design Pattern used (Yes / No)	
If yes, write the name of Design Pattern used	
Interface type (CLI / GUI)	
No. of Actors	1
Name of Actors	User 1
Total no. of Functional Requirements	6
List few important non-Functional Requirements	Reliability, Usability, Scalability, Security, Maintainability, Availability

Testing

Which testing is performed? (Manual or Automation)	Yes
Is Beta testing done for this project?	Yes

Write project narrative covering above mentioned points

This project, titled "Credit Card Fraud Detection Using Machine Learning", is developed as a Desktop Application using Python. The Waterfall SDLC model is followed, ensuring a clear, linear flow through requirement analysis, data preprocessing, model building, evaluation, and deployment. The motivation behind this project is the rising cases of credit card fraud and the need for intelligent systems to detect anomalies effectively. The project adopts a Functional design approach, structuring the code into reusable functions for each task. The implementation uses logistic regression on a balanced dataset, achieving high accuracy and AUC. It is a data-driven project with fixed objectives and no iterative development required, which suits the Waterfall model. All phases were systematically completed and documented, ensuring a disciplined and organized development process.

Ashutosh Rathore	0187CS211041
Shreya Singh Rathore	0187CS211160
Chirag Dewangan	0187CS211055
Ajay Meena	0187CS211016

Guide Signature
(Prof. Mayank Kurchuniya)

APPENDIX-1 GLOSSARY OF TERMS

A

Algorithm The machine learning model used in this project is Logistic Regression, chosen for its efficiency and interpretability in binary classification tasks such as fraud detection.

Accuracy The model achieved an approximate accuracy of 95% on the balanced dataset, demonstrating strong performance in identifying both fraudulent and non-fraudulent transactions.

AUC Score The Area Under the ROC Curve (AUC) was found to be 0.95, indicating excellent ability of the model to distinguish between fraud and non-fraud cases.

D

Dataset The dataset used is the Credit Card Fraud Detection dataset from Kaggle, containing 284,807 transactions with only 492 fraud cases, making it highly imbalanced.

F

Features The dataset includes 31 features: 28 anonymized PCA-transformed features (V1–V28), along with 'Time', 'Amount', and 'Class'. 'Time' represents the time elapsed since the first transaction, while 'Amount' indicates the transaction value.

P

Precision The model showed high precision, meaning that it effectively avoided falsely classifying normal transactions as fraud.

Preprocessing Preprocessing involved standardizing the 'Amount' feature using StandardScaler and dropping the 'Time' feature due to its limited relevance. The data was then prepared for modeling by splitting and balancing it through random under-sampling

V

Visualization Visualization was done to understand data distribution and model performance. Seaborn and Matplotlib were used to plot class distribution, confusion matrix, and ROC curve. These visualizations helped in evaluating accuracy, precision, and recall effectively.