

A GUI-BASED WEB INTERFACE FOR CREDIT CARD FRAUD DETECTION SYSTEM



Mini Project submitted in partial fulfillment of the requirement for the award of the
degree of

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING

Under the esteemed guidance of

Dr. K. Krishna Jyothi

Associate Professor

By

Sakshi Chandesurye 21R11A05E4

Singar Tejasvi 21R11A05E7

Undru Vardhini Venkata Sai 21R11A05F4



Department of Computer Science and Engineering

Accredited by NBA

Geethanjali College of Engineering and Technology

(UGC Autonomous)

(Affiliated to J.N.T.U.H, Approved by AICTE, New Delhi)

Cheeryal (V), Keesara (M), Medchal.Dist.-501 301.

September-2024

Geethanjali College of Engineering & Technology

(UGC Autonomous)

(Affiliated to JNTUH, Approved by AICTE, New Delhi)
Cheeryal (V), Keesara(M), Medchal Dist.-501 301.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Accredited by NBA



CERTIFICATE

This is to certify that the B.Tech Mini Project report entitled “**A GUI-Based Web Interface for Credit Card Fraud Detection System**” is a bonafide work done by **Sakshi Chandesurye (21R11A05E4), Singar Tejasvi (21R11A05E7), Undru Vardhini Venkata Sai (21R11A05F4)**, in partial fulfillment of the requirement of the award for the degree of Bachelor of Technology in “**Computer Science and Engineering**” from Jawaharlal Nehru Technological University, Hyderabad during the year 2024-2025.

Internal Guide

Dr K. Krishna Jyothi

Associate Professor

HOD – CSE

Dr. A. SreeLakshmi

Professor

Geethanjali College of Engineering & Technology

(UGC Autonomous)

(Affiliated to JNTUH Approved by AICTE, New Delhi)
Cheeryal (V), Keesara(M), Medchal Dist.-501 301.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Accredited by NBA



DECLARATION BY THE CANDIDATE

We, **Sakshi Chandesurye, Singar Tejasvi, Undru Vardhini Venkata Sai**, bearing Roll Nos. **21R11A05E4, 21R11A05E7, 21R11A05F4** hereby declare that the project report entitled “**A GUI-Based Web Interface for Credit Card Fraud Detection System**” is done under the guidance of **Dr. K. Krishna Jyothi, Associate Professor**, Department of Computer Science and Engineering, Geethanjali College of Engineering and Technology, is submitted in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering**.

This is a record of bonafide work carried out by us in **Geethanjali College of Engineering and Technology** and the results embodied in this project have not been reproduced or copied from any source. The results embodied in this project report have not been submitted to any other University or Institute for the award of any other degree or diploma.

Sakshi Chandesurye (21R11A05E4)

Singar Tejasvi(21R11A05E7)

Undru Vardhini Venkata Sai (21R11A05F4)

Department of CSE,

Geethanjali College of Engineering and Technology,

Cheeryal.

ACKNOWLEDGEMENT

We, the students of the Computer Science and Engineering Department at Geethanjali College of Engineering and Technology, would like to express our sincere gratitude to the college authorities for providing the necessary resources and facilities to successfully complete our mini project, titled “**A GUI-Based Web Interface for Credit Card Fraud Detection System.**”

We are deeply thankful to **Prof. Dr. S. Udaya Kumar**, Principal of Geethanjali College of Engineering and Technology, for his invaluable guidance and continuous encouragement throughout this project. His insightful advice and unwavering commitment to fostering a culture of innovation and learning have been instrumental in the successful completion of our work.

We also extend our heartfelt appreciation to **Dr. A. SreeLakshmi, Professor**, and Head of the Department of Computer Science and Engineering, for her dedicated support and mentorship. Her leadership and commitment to our academic and professional development have been a source of motivation and guidance throughout this project.

A special thanks to **Dr. K. Krishna Jyothi, Associate Professor**, whose technical expertise and thoughtful guidance have been essential to navigating the complex aspects of this project. Her constructive feedback and willingness to share her knowledge have greatly contributed to the technical success of our work.

We are grateful to all our mentors and faculty members for their continuous support and encouragement.

Sakshi Chandesurye (21R11A05E4)

Singar Tejasvi (21R11A05E7)

Undru Vardhini Venkata Sai (21R11A05F4)

ABSTRACT

Credit card fraud is a growing concern in today's increasingly digital economy, where millions of financial transactions occur daily across the globe. As fraudsters employ more sophisticated techniques, detecting fraudulent credit card activities has become a critical task for financial institutions to prevent losses and protect consumers from unauthorized charges. This project introduces an innovative approach to credit card fraud detection by leveraging advanced machine learning techniques within the field of data science.

The project focuses on developing a robust predictive system capable of identifying fraudulent transactions in highly imbalanced datasets, such as the dataset used, which contains only 0.172% fraudulent transactions out of 284,807 total transactions. Utilizing data preprocessing, exploratory data analysis (EDA), feature engineering, and machine learning algorithms like logistic regression, random forest, gradient boosting, decision trees, and support vector classifiers, this project aims to construct a system that effectively mitigates the risks of fraud.

To ensure accessibility, the system will feature a dual-interface deployment: a standalone Python-based Tkinter graphical user interface (GUI) for local use and a web-based platform named FraudShield Analytics, built using Flask, for online integration. Furthermore, the future scope includes real-time transaction monitoring, ensuring rapid detection and prevention of fraudulent activities, thereby enhancing security within the financial ecosystem.

By advancing fraud detection capabilities, this project not only safeguards financial assets but also plays a pivotal role in protecting consumers from daily fraudulent activities. The insights gained from this project offer valuable contributions to the ongoing fight against credit card fraud, addressing the increasing challenges of identifying fraud in massive transaction datasets.

LIST OF FIGURES/DIAGRAMS/GRAPHS

| S. No | Fig. No | Title of Figure | Page No |
|--------------|----------------|--|----------------|
| 1 | 1.1.1 | Secure Credit Card Transactions | 2 |
| 2 | 4.1.1 | System Architecture | 13 |
| 3 | 4.2.1.1 | Use case diagram | 17 |
| 4 | 4.2.2.1 | Activity diagram | 18 |
| 5 | 4.2.3.1 | Class diagram | 19 |
| 6 | 4.2.4.1 | Sequence diagram | 20 |
| 7 | 4.3.1.1 | System Design | 21 |
| 8 | 5.1.1.1 | First five Observations in the dataset | 25 |
| 9 | 5.1.1.2 | Last five Observations in the dataset | 25 |
| 10 | 5.1.2.1 | Checking Duplicates | 26 |
| 11 | 5.1.2.2 | Checking Unique values count in all the features | 26 |
| 12 | 5.1.3.1 | Distribution of Time | 27 |
| 13 | 5.1.3.2 | Distribution of Amount by Class | 28 |
| 14 | 5.1.3.3 | Class Distribution | 29 |
| 15 | 5.1.3.4 | Scatter Plot of Amount vs Time | 30 |
| 16 | 5.1.4.1 | Correlation Heatmap of Selected Features | 31 |
| 17 | 5.1.5.1 | Class Distribution Before and After Undersampling | 33 |
| 18 | 5.1.5.2 | Class Distribution Before and After Oversampling | 34 |
| 19 | 5.1.6.1.1 | Classification Report for Logistic Regression | 35 |
| 20 | 5.1.6.1.2 | Confusion Matrix and ROC Curve for Logistic Regression | 36 |
| 21 | 5.1.6.1.3 | Classification Report for Decision Tree | 37 |

| | | | |
|----|-----------|--|----|
| 22 | 5.1.6.1.4 | Confusion Matrix and ROC Curve for Decision Tree Classifier | 38 |
| 23 | 5.1.6.1.5 | Classification Report for Random Forest Classifier | 39 |
| 24 | 5.1.6.1.6 | Confusion Matrix and ROC Curve for Random Forest Classifier | 40 |
| 25 | 5.1.6.2.1 | Performance Comparision of Classifier Models | 41 |
| 26 | 5.1.6.2.2 | Saving the Random Forest Classifier Model for Fraud Detection | 41 |
| 27 | 5.1.6.1.5 | Classification Report for Random Forest Classifier | 39 |
| 28 | 7.1.1 | Home Page | 59 |
| 29 | 7.1.2 | Signup Page in Progress | 59 |
| 30 | 7.1.3 | Overview Page | 60 |
| 31 | 7.1.4 | About Page | 60 |
| 32 | 7.1.5 | Contact Page | 61 |
| 33 | 7.1.6 | Detection Page in Progress | 61 |
| 34 | 7.1.7 | Continuation of the Detection Page in Progress | 62 |
| 35 | 7.1.8 | Result Page | 62 |
| 36 | 7.2.1 | GUI Application of Credit Card Fraud Detection System | 63 |
| 37 | 7.2.2 | Final Prediction of GUI Application | 63 |
| 38 | 11.1 | Plagiarism Report | 69 |

LIST OF ABBREVIATIONS

| S. No | Abbreviation | Full Form |
|-------|--------------|--|
| 1 | EDA | Exploratory Data Analysis |
| 2 | HTML | Hypertext Markup Language |
| 3 | CSS | Cascading Style sheets |
| 4 | JS | Java Script |
| 5 | GUI | Graphical User Interface |
| 6 | ML | Machine Learning |
| 7 | SMOTE | Synthetic Minority Oversampling Technique |
| 8 | ROC | Receiver Operating Characteristic Curve |
| 9 | API | Application Programming Interface |

TABLE OF CONTENTS

| S. No | Contents | Page. No |
|--------------|---|-----------------|
| | Abstract | v |
| | List of Figures/Diagrams/Graphs | vi |
| | List of Abbreviations | viii |
| 1 | Introduction | |
| | 1.1 About the Project | 1 |
| | 1.2 Objectives of the Project | 3 |
| 2 | System Analysis | |
| | 2.1 Existing System | 4 |
| | 2.1.1 Drawbacks of the Existing System | 4 |
| | 2.2 Proposed System | 5 |
| | 2.2.1 Advantages of Proposed System | 6 |
| | 2.3 Feasibility Study | 7 |
| | 2.3.1 Details | 7 |
| | 2.3.2 Impact on environment | 7 |
| | 2.3.3 Safety | 7 |
| | 2.3.4 Ethics | 8 |
| | 2.3.5 Cost | 8 |
| | 2.3.6 Type | 8 |

| | | |
|----------|----------------------------|----|
| | 2.3.7 Standards | 9 |
| | 2.4 Scope of Project | 9 |
| | 2.5 System Configuration | 9 |
| 3 | Literature Overview | 10 |
| 4 | System Design | |
| | 4.1 System Architecture | 13 |
| | 4.1.1 Module Description | 14 |
| | 4.1.1.1 Data Acquisition | 14 |
| | 4.1.1.2 Data Preprocessing | 14 |
| | 4.1.1.3 Data Sampling | 15 |
| | 4.1.1.4 Feature Selection | 15 |
| | 4.1.1.5 Machine Learning | 15 |
| | Model | |
| | 4.1.1.6 Flask API | 16 |
| | 4.1.1.7 User Interface | 16 |
| | 4.2 UML Diagrams | 13 |
| | 4.2.1 Use Case Diagram | 17 |
| | 4.2.2 Activity Diagram | 18 |
| | 4.2.3 Class Diagram | 19 |
| | 4.2.4 Sequence Diagram | 20 |
| | 4.3 System Design | 21 |
| | 4.3.1 Modular Description | 21 |
| | 4.3.1.1 Data Acquisition | 21 |

| | | |
|----------|-------------------------------------|----|
| | 4.3.1.2 Data Preprocessing | 22 |
| | 4.3.1.3 Data Sampling | 22 |
| | 4.3.1.4 Feature Selection | 22 |
| | 4.3.1.5 Machine Learning | 23 |
| | Model | |
| | 4.3.1.6 Flask API | 23 |
| | 4.3.1.7 User Interface | 23 |
| | 4.3.2 Database Design | 24 |
| 5 | Implementation | |
| | 5.1 Implementation | 25 |
| | 5.1.1 Slicing and Dicing | 25 |
| | 5.1.2 Checking for Data Consistency | 26 |
| | 5.1.3 Exploratory Data Analysis | 27 |
| | 5.1.4 Descriptive Statistics and | 31 |
| | Correlations | |
| | 5.1.5 Data Sampling | 33 |
| | 5.1.6 Algorithms | 35 |
| | 5.1.6.1 Model Training and | 35 |
| | Evaluation | |
| | 5.1.6.2 Model Selection | 41 |
| | 5.2 Sample code | 42 |
| | 5.2.1 GUI Implementaion Code | 42 |

| | | |
|----------|---|----|
| | 5.2.2 Frontend Implementation- HTML Code | 44 |
| | 5.2.3 Flask API Code | 48 |
| | 5.2.4 Machine Learning Implementation Code | 51 |
| 6 | Testing | |
| | 6.1 Software Testing | 55 |
| | 6.1.1 Unit Testing | 55 |
| | 6.1.1.1 Model Prediction | 55 |
| | 6.1.1.2 Frontend Components | 55 |
| | 6.1.1.3 Backend Functions | 56 |
| | 6.1.2 Integration Testing | 56 |
| | 6.1.2.1 Model Integration with Backend | 56 |
| | 6.1.2.2 Frontend and Backend Interaction | 56 |
| | 6.1.3 System Testing | 56 |
| | 6.1.3.1 End to End Testing | 57 |
| | 6.1.4.1 Performance Testing | 57 |
| | 6.1.4 Blackbox Testing | 57 |
| | 6.1.5 Whitebox Testing | 57 |
| | 6.2 Test Cases | 58 |

| | | |
|-----------|--------------------------------------|----|
| 7 | Outputs | |
| | 7.1 Web Interface | 59 |
| | 7.2 GUI Interface | 63 |
| 8 | Conclusion | |
| | 8.1 Conclusion | 64 |
| | 8.2 Further Enhancements | 65 |
| 9 | Bibliography | |
| | 9.1 Books References | 66 |
| | 9.2 Websites References | 66 |
| | 9.3 Technical Publication References | 66 |
| 10 | Appendices | |
| | 10.1 Appendix A- Software Used | 67 |
| | 10.2 Appendix B- Methodologies Used | 67 |
| | 10.3 Appendix C- System Architecture | 68 |
| | 10.4 Appendix D- Testing | 68 |
| 11 | Plagiarism Report | 69 |

1. INTRODUCTION

1.1 ABOUT THE PROJECT

Credit card fraud represents a significant financial challenge that affects economies worldwide, with escalating incidents posing a threat to consumers and financial institutions alike. In 2022, approximately 3.5 billion fraudulent transactions were recorded globally, resulting in total losses estimated at \$321 billion. The prevalence of fraud has surged by 133% over the past three decades, highlighting the urgent need for effective detection and prevention mechanisms. Particularly in the United States, which accounts for only 22.9% of global credit and debit card transactions, the disproportionate financial losses underscore the critical nature of addressing this issue. As credit card usage continues to expand, so does the complexity of fraudulent techniques employed by criminals, including the creation of counterfeit cards, account theft, and card skimming during transactions.

In India and Europe, the landscape of credit card fraud is similarly alarming. With the rapid adoption of digital payment methods, fraudulent activities have surged, driven by the convenience and accessibility of online transactions. In India, the rise in digital payments has been accompanied by a notable increase in fraud cases, with reports indicating a significant jump in incidents of credit card skimming and phishing attacks. As consumers become more reliant on electronic payment methods, their vulnerability to fraud grows, necessitating robust systems for detection and prevention. Meanwhile, European countries have also seen a rise in fraud rates, particularly in cross-border transactions, where differing regulatory standards can be exploited by fraudsters.

The consequences of credit card fraud extend beyond immediate financial losses; they erode consumer trust in financial institutions and can lead to long-term repercussions for businesses. As consumers become increasingly aware of the risks associated with digital transactions, their apprehension can deter them from engaging in online shopping or utilizing credit cards, ultimately impacting economic growth. To combat this, it is essential

for financial institutions to invest in innovative technologies that enhance their fraud detection capabilities and build consumer confidence.

This project aims to address these pressing concerns by developing a sophisticated credit card fraud detection system utilizing advanced machine learning techniques. By leveraging a highly imbalanced dataset of transactions, which includes only 0.172% classified as fraudulent, the system focuses on accurately identifying suspicious activities. The integration of data preprocessing, exploratory data analysis (EDA), and feature engineering allows for the construction of a robust predictive model capable of mitigating risks associated with credit card fraud. Additionally, the deployment of this system ensures accessibility for users, enhancing the security of financial transactions. Ultimately, this project aims to fortify the defenses against credit card fraud, protecting consumers and financial institutions in an increasingly digital marketplace.

By implementing such a system, we aim not only to improve the efficiency of fraud detection but also to establish a framework that can adapt to emerging threats. Continuous learning algorithms and real-time monitoring features will empower financial institutions to stay ahead of fraudsters and minimize potential losses.



Fig 1.1.1 Secure Credit Card Transactions

1.2 OBJECTIVES OF THE PROJECT

1. To develop a sophisticated credit card fraud detection system that leverages advanced machine learning techniques to accurately identify fraudulent transactions, thereby mitigating the financial impact of fraud on consumers and financial institutions.
2. To address the challenges posed by highly imbalanced datasets, in which fraudulent transactions represent only a small fraction of total transactions, by employing tailored preprocessing and modeling strategies that enhance the detection of rare events.
3. Implement innovative algorithms designed to improve the sensitivity and specificity of fraud detection, ensuring effective differentiation between legitimate and fraudulent activities while minimizing false positives.
4. Utilize data preprocessing methods, such as resampling and feature engineering, to enrich the dataset and improve model performance, enabling effective learning from limited instances of fraud.
5. To explore the integration of real-time monitoring capabilities, allowing for immediate detection and response to fraudulent transactions, thus enhancing the overall security of financial operations.
6. To contribute to the evolving landscape of fraud prevention by providing insights and strategies that empower financial institutions to better protect their assets and customers in an increasingly digital economy.
7. Evaluate models using appropriate performance metrics tailored for imbalanced datasets, ensuring a comprehensive assessment of both sensitivity to fraud detection and precision in classifying legitimate transactions.
8. Design a user-friendly interface that facilitates seamless interaction with the fraud detection system, offering both local access via a standalone application and online integration for convenient usage across different platforms.

2. SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Current credit card fraud detection systems predominantly rely on data mining techniques to identify potentially fraudulent activities. These systems meticulously analyze transaction data to uncover patterns that may signify fraudulent behavior. Some existing solutions adopt rule-based approaches, flagging transactions that meet specific predefined criteria indicative of fraud. This method, while useful, often lacks the flexibility required to adapt to evolving fraudulent tactics.

Additionally, anomaly detection is employed by certain systems, which involves monitoring deviations from typical transaction behavior to identify suspicious activities. This approach can be effective; however, it may struggle to adapt when new fraud patterns emerge. Furthermore, some systems leverage expert knowledge and domain-specific rules derived from historical data, enabling them to identify fraudulent transactions based on prior insights and human expertise. While these strategies enhance detection capabilities, they often introduce limitations.

Despite incorporating machine learning techniques in some cases, existing systems face significant challenges in achieving high accuracy rates, particularly when dealing with highly imbalanced datasets where fraudulent transactions represent a minimal fraction of total activities. This inefficiency leads to a host of drawbacks that hinder their effectiveness.

2.1.1 DRAWBACKS OF EXISTING SYSTEM

- Inefficiency in handling highly imbalanced data, which severely hampers the detection of fraudulent events, leading to a significant number of undetected fraud cases that can result in substantial financial losses for consumers, businesses, and financial institutions alike.

- Lack of adaptability to emerging fraud techniques, limiting their effectiveness over time, as traditional systems often struggle to incorporate new patterns of fraudulent behavior without extensive manual updates and retraining.
- Limited accuracy in identifying fraudulent transactions, particularly when faced with evolving patterns of fraud, which can result in high false negative rates where genuine frauds go unnoticed while also generating false positives that inconvenience legitimate users.
- Dependence on expert knowledge, which can lead to biases and gaps in detection capabilities, as the reliance on historical fraud patterns may overlook novel tactics employed by fraudsters, rendering the system less effective.
- Lack of real-time monitoring, resulting in delayed responses to fraudulent activities that can exacerbate financial losses, as slow detection times allow fraud to escalate before any corrective actions can be implemented.
- Limited automation, requiring significant manual intervention for effective operation, which not only increases the operational burden on fraud detection teams but also introduces the potential for human error in identifying and responding to fraud cases.

These limitations highlight the necessity for a more robust fraud detection system that can effectively handle imbalanced data while providing accurate and timely identification of fraudulent transactions. The proposed model aims to address these challenges by utilizing advanced machine learning techniques, ensuring improved accuracy and user accessibility through a friendly interface.

2.2 PROPOSED SYSTEM

The proposed system enhances fraud detection accuracy using advanced machine learning techniques, moving beyond traditional rule-based methods. It preprocesses transaction data for analysis and employs exploratory data analysis (EDA) to uncover patterns, correlations, and trends, helping the model differentiate between legitimate and fraudulent transactions.

A key feature of the system is its ability to construct robust predictive models capable of handling highly imbalanced datasets, where fraudulent transactions make up only a small fraction of the total data. By employing advanced machine learning algorithms and techniques, such as feature engineering and tailored model training, the system effectively identifies fraudulent behavior. The model is designed to balance sensitivity and specificity, ensuring it captures rare fraudulent events while minimizing false positives. The integration of these machine learning techniques significantly enhances the system's fraud detection capabilities compared to existing approaches.

Furthermore, the system is designed to be user-friendly and accessible through two primary interfaces: a Python-based Tkinter graphical user interface (GUI) for local use and a web-based interface using Flask for online integration. This dual-interface approach ensures flexibility and ease of use for different types of users.

2.2.1 ADVANTAGES OF PROPOSED SYSTEM

- The system leverages advanced machine learning algorithms to improve the accuracy and reliability of fraud detection. It effectively identifies fraudulent transactions even in highly imbalanced datasets, minimizing the risk of undetected fraud and reducing false positives.
- The system utilizes exploratory data analysis (EDA) and feature engineering to uncover hidden patterns in transaction data, improving the accuracy of fraud detection. This ensures a more comprehensive approach to identifying suspicious activities.
- A robust predictive model is built to detect fraudulent transactions in large datasets, maintaining high sensitivity and precision. This tailored model effectively handles the challenges of imbalanced data.
- With both a Python-based Tkinter GUI for local usage and a Flask-powered web interface for online integration, the system provides easy-to-navigate platforms. These interfaces allow users to interact with the fraud detection system efficiently, making it accessible for various use cases across different environments.

2.3 FEASIBILITY STUDY

2.3.1 DETAILS

This project on credit card fraud detection is categorized under the domain of financial technology and machine learning. It focuses on utilizing advanced algorithms to identify fraudulent transactions within highly imbalanced datasets, where legitimate transactions far outnumber fraudulent ones. The primary goal is to develop an accurate and efficient system capable of detecting rare fraudulent activities by analyzing transaction patterns. This system is particularly relevant in today's digital economy, where credit card usage continues to grow, and the need for secure financial transactions is paramount. The project leverages data preprocessing techniques, exploratory data analysis (EDA), and tailored machine learning models to enhance fraud detection, with a user-friendly interface for both local and web-based platforms.

2.3.2 IMPACT ON ENVIRONMENT

Operating primarily in a digital environment, the project has a minimal direct environmental impact. However, it indirectly supports sustainability by preventing fraudulent transactions and reducing financial losses, which promotes responsible use of digital infrastructure. Furthermore, the integration of both web-based and local interfaces decreases the need for paper-based reporting and manual intervention, leading to reduced resource consumption. By relying on digital technology, the system minimizes the demand for physical resources related to fraud prevention measures, thereby avoiding higher energy consumption and environmental waste associated with outdated hardware systems.

2.3.3 SAFETY

The proposed system emphasizes strong password management to enhance security for user accounts. By enforcing complex password requirements, such as a mix of uppercase letters, lowercase letters, numbers, and special characters, it minimizes the risk of unauthorized access. Additionally, login access is designed to be secure, ensuring that only authenticated

users can access sensitive financial data. These measures collectively contribute to the protection of users' financial information, significantly reducing the likelihood of data breaches and unauthorized access.

2.3.4 ETHICS

Ethical considerations for this fraud detection system focus on maintaining transparency and ensuring privacy. The project adheres to strict data protection policies to secure sensitive financial information, guaranteeing that user data is only used for fraud detection purposes. It operates with transparency in how data is collected and processed, ensuring that users are informed about the use of their transaction data. The system is designed to avoid biases in fraud detection by using a balanced approach in model training, ensuring that all transactions are fairly assessed, regardless of origin or user demographics.

2.3.5 COST

The cost analysis for the project encompasses essential components such as development, deployment, and maintenance. Development expenses primarily include the contributions of developers, and since the project utilizes open-source tools, infrastructure costs are minimized, focusing mainly on local deployment without extensive server requirements. Investment in security measures is crucial for effectively safeguarding user information. Additionally, ongoing maintenance, user support, and updates to the fraud detection algorithms are integral to the budget, ensuring the system remains efficient in detecting new fraud patterns. Overall, a comprehensive financial assessment will facilitate informed decision-making throughout the project's lifecycle.

2.3.6 TYPE

The project utilizes a machine learning-based approach integrated with a Flask API for web deployment and a Tkinter-based local GUI for offline access. Python serves as the core programming language, with libraries such as scikit-learn and pandas being employed to build, train, and evaluate the fraud detection models. This technology stack ensures

efficient processing of large transaction datasets and supports seamless interaction between the front-end and back-end components.

2.3.7 STANDARDS

The project follows the Agile SDLC model, emphasizing iterative development and continuous feedback. It begins with gathering requirements focused on fraud detection needs and progresses through iterative cycles of development, testing, and refinement. This approach allows for regular updates and improvements based on feedback, ensuring the system aligns with user needs and adapts to evolving fraud techniques.

2.4 SCOPE OF THE PROJECT

The scope of this project covers the design, development, and deployment of a credit card fraud detection system that handles imbalanced data effectively. The system includes data preprocessing steps, exploratory data analysis (EDA), and model training, with a focus on improving the detection of rare fraudulent events. By analyzing transaction data patterns, the system is equipped to detect anomalies and suspicious behavior. The project aims to enhance the accuracy of fraud detection, reduce false positives, and provide an intuitive interface for users to interact with the system efficiently.

2.5 SYSTEM CONFIGURATION

Software Required:

1. Operating System-Windows/Mac/Linux
2. Python 3.x with Jupyter Notebook
3. Spyder IDE

Hardware Required:

1. GPU integrated Laptop/Desktop
2. RAM: 4GB or Higher
3. Hard Disk: 50 GB

3. LITERATURE OVERVIEW

The rise of digital transactions has led to an alarming increase in credit card fraud, necessitating innovative approaches for its detection. This literature survey reviews various studies that explore machine learning techniques and methodologies for credit card fraud detection, emphasizing their relevance to this project. As fraudsters continually adapt their tactics, traditional detection methods often fall short, highlighting the need for advanced analytical frameworks. Recent research has demonstrated the effectiveness of machine learning algorithms, such as logistic regression, decision trees, and ensemble methods, in identifying fraudulent patterns within large and imbalanced datasets. By leveraging these techniques, this project aims to enhance detection capabilities, ensuring a robust defense against evolving fraudulent activities in the financial sector. Through a thorough examination of existing literature, we aim to identify best practices and innovative solutions that can inform the development of a predictive model tailored to the unique challenges of credit card fraud detection.

1. ANDREA DAL POZZOLO, OLIVIER CAELEN, YANN-AËL LE BORGNE, SERGE WATERSCHOOT, GIANLUCA BONTEMPI. “LEARNED LESSONS IN CREDIT CARD FRAUD DETECTION FROM A PRACTITIONER PERSPECTIVE, EXPERT SYSTEMS WITH APPLICATIONS.”

This study highlighted the effectiveness of machine learning models in detecting fraudulent transactions and emphasized the importance of ensemble methods and feature engineering for improving accuracy. The findings suggest that adaptive learning strategies significantly enhance the performance of fraud detection systems. By integrating these advanced techniques, our project aims to develop a robust predictive model capable of continuously learning and adapting to new fraud patterns, thereby improving overall security in financial operations. Additionally, the study underlined the importance of handling imbalanced datasets to ensure that fraud detection models are both precise and scalable. These insights will guide our model's approach to data sampling and real-time detection improvements.

2. OGWUELEKA, FRANCISCA NONYELUM. “DATA MINING APPLICATION IN CREDIT CARD FRAUD DETECTION SYSTEM.”

This research explored the use of data mining techniques in building an efficient credit card fraud detection system. Various algorithms, including decision trees and neural networks, were evaluated for their effectiveness, highlighting their respective strengths and limitations in real-world applications. The study concluded that combining these algorithms could offer a more robust fraud detection framework. Drawing from these insights, our project will implement similar methodologies to improve the accuracy and reliability of the fraud detection system. Additionally, this approach aims to balance the trade-offs between speed and precision, ensuring timely detection without sacrificing accuracy. By incorporating advanced data preprocessing techniques, the system will be better equipped to handle the complexities of real-time fraud detection.

3. DAHEE CHOI AND KYUNGHO LEE. “MACHINE LEARNING BASED APPROACH TO FINANCIAL FRAUD DETECTION PROCESS IN MOBILE PAYMENT SYSTEM.”

This study tackles the rising issue of mobile payment fraud, highlighting the need for accurate detection methods using both supervised and unsupervised machine learning techniques. The proposed model incorporates data preprocessing, sampling, feature selection, and classification and clustering algorithms. Verification is performed at each phase to assess the model's effectiveness. The preprocessing stage includes data cleaning, correlation analysis, transformation, integration, and reduction. The sampling phase uses random over-sampling and under-sampling methods to evaluate the dataset. Feature extraction employs filter-based methods, leading to a clustering process that generates a training and validation set for classification. By applying supervised algorithms to clustering results, the model achieves improved predictive accuracy, validated through precision and recall rates using the F1 measure. These insights can enhance our project's credit card fraud detection strategies, ensuring a robust response to emerging fraud patterns.

4. LAWRENCE BORAH, SALEENA B, PRAKASH B. “CREDIT CARD FRAUD DETECTION USING DATA MINING TECHNIQUES.”

This study outlines a comprehensive approach to credit card fraud detection by integrating a payment gateway's functionalities with advanced data mining methods. It emphasizes the importance of collecting pertinent credit card and transaction details, which are processed by a Fraud Detection Program utilizing effective classification algorithms, specifically the random forest algorithm. The findings highlight the ability of this model to evaluate cardholder data alongside transaction information, significantly enhancing the accuracy of fraud detection. By delivering a final judgment of each transaction, whether fraudulent or not, to the payment gateway administrator, the system aims to provide merchants with a reliable UI report. This research informs our project's implementation of similar data mining techniques, ensuring a robust and efficient fraud detection mechanism that adapts to emerging patterns of fraudulent activity. Additionally, the incorporation of real-time monitoring capabilities is expected to further improve the system's responsiveness to suspicious transactions, thus minimizing potential financial losses for merchants.

5. AMANZE B.C., AND ONUKWUGHA C.G. “DATA MINING APPLICATION IN CREDIT CARD FRAUD DETECTION SYSTEM.”

This study highlights how data mining methodologies enable banks to identify individuals committing credit card fraud and predict potential defaults. By analyzing customer behavior and reliability, data mining techniques facilitate comprehensive profiling and risk ranking of bank branches concerning credit card fraud. To achieve this, relevant data can be sourced from credit risk information service databases, which contain crucial details related to credit card fraud. This includes key attributes such as cardholder identity, the branch or bank location where the credit card was issued, and the locations where transactions are made. Furthermore, the application of these techniques can significantly enhance the accuracy of fraud detection systems, allowing for timely interventions. Ultimately, implementing robust data mining strategies can lead to more effective risk management and reduced financial losses for banks.

4. SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

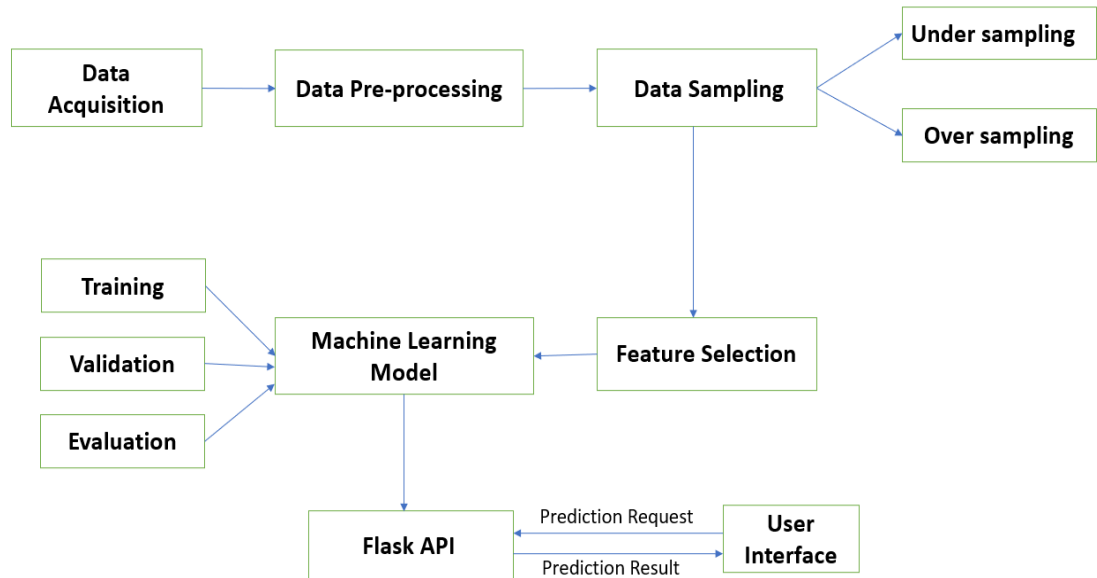


Fig 4.1.1 System Architecture

A system architecture diagram provides a clear and structured representation of the various components and processes involved in developing a credit card fraud detection system. It outlines the flow of data, starting with the dataset, which serves as the foundation of the project. The dataset undergoes rigorous data preprocessing steps, including cleaning, handling missing values, and normalizing transaction data, to ensure its quality and consistency. Next, data sampling techniques, such as undersampling and oversampling, are applied to address the imbalance in the dataset, ensuring that fraudulent and non-fraudulent transactions are appropriately represented for model training. Following this, feature selection is performed to identify the most relevant attributes that contribute to the identification of fraudulent activities, thus optimizing the efficiency and accuracy of the machine learning model.

The heart of the system involves training, validation, and evaluation of various machine learning models, such as logistic regression, decision trees, random forests, and other classifiers, to detect fraudulent transactions effectively. Once the model is trained and fine-tuned, it is integrated with a Flask API, which serves as the bridge between the backend machine learning model and the user-facing interface. The API allows the system to process incoming transaction data in real-time and make fraud detection predictions. Finally, the user interacts with the system through a web-based interface, where the predictions and insights are displayed, enabling immediate action to prevent fraudulent transactions.

4.1.1 MODULE DESCRIPTION

The seamless integration of these modules enables real-time monitoring and analysis, allowing for quick responses to potential fraud incidents. Furthermore, the modular design ensures scalability and adaptability, making it easier to incorporate new features and algorithms as fraud detection methodologies evolve.

4.1.1.1 DATA ACQUISITION

This module is responsible for collecting and curating a comprehensive dataset relevant to credit card transactions. Data can be sourced from various repositories, including public datasets from Kaggle or financial institutions. The dataset should include a wide range of features, such as transaction amounts, transaction timestamps ensuring that it is representative of real-world scenarios. Quality and diversity in the dataset are crucial to enhancing the model's performance.

4.1.1.2 DATA PREPROCESSING

In this module, the collected data undergoes a series of preprocessing steps to ensure its quality and suitability for analysis. This includes cleaning the data by removing duplicates, handling missing values, and normalizing features to bring them to a common scale. Additionally, exploratory data analysis (EDA) is performed to uncover patterns,

relationships, and insights within the dataset, allowing for better understanding of the data distribution and potential factors influencing fraud. Visualization techniques such as histograms, box plots, and correlation matrices are employed during this phase to identify trends and anomalies. This module lays the groundwork for effective data analysis by ensuring that the dataset is consistent, well-structured, and insightful, ultimately enhancing the model's predictive performance.

4.1.1.3 DATA SAMPLING

To address the class imbalance commonly found in credit card fraud detection datasets, this module implements data sampling techniques. Undersampling reduces the number of instances from the majority class, while oversampling increases instances of the minority class to achieve a more balanced dataset. Techniques like SMOTE (Synthetic Minority Over-sampling Technique) can be utilized to generate synthetic samples for the minority class, thus enhancing the model's ability to recognize fraudulent transactions. This balance is critical for improving model performance and reducing bias in predictions.

4.1.1.4 FEATURE SELECTION

In this module, relevant features are identified and selected based on their significance in predicting fraud. Various techniques, such as filter methods, wrapper methods, and embedded methods, can be applied to evaluate the importance of each feature. The goal is to retain only the most informative features while eliminating redundant or irrelevant data that could complicate the model. This step not only improves model accuracy but also reduces computational overhead, making the training process more efficient.

4.1.1.5 MACHINE LEARNING MODEL

This module focuses on the development and fine-tuning of machine learning models tailored for fraud detection. Different algorithms, such as logistic regression, decision trees, and random forests, will be trained on the preprocessed dataset. The model's performance is evaluated using validation techniques to ensure its robustness. Metrics such as accuracy,

precision, recall, and F1 score are employed to assess the model's effectiveness in detecting fraudulent transactions. Continuous iteration and optimization will be performed to achieve the best results. Finally, the best-performing model will be selected for deployment, ensuring it is well-suited to adapt to the evolving landscape of fraud tactics.

4.1.1.6 FLASK API

The Flask API module serves as the intermediary between the trained machine learning model and the end users. It enables seamless communication by handling user requests and passing relevant data to the fraud detection model for analysis. The API is responsible for receiving transaction details, processing them through the model, and returning the fraud detection results in real time. This module ensures that the system is accessible and user-friendly, facilitating easy integration with frontend applications. Additionally, the API is designed to manage authentication and authorization, ensuring that only authorized users can access sensitive transaction data. It will also provide logging capabilities for monitoring system performance and user interactions, enabling quick identification and resolution of any issues that may arise.

4.1.1.7 USER INTERFACE

The User Interface module encompasses the interface through which merchants and stakeholders interact with the fraud detection system. Users can input transaction data via a web interface named "Fraud Shield Analytics" or a Tkinter-based GUI, receive real-time feedback on potential fraud, and view detailed reports of the fraud detection outcomes. This module aims to provide a smooth user experience, ensuring that users can easily understand the system's findings and make informed decisions based on the results provided. The Tkinter GUI will offer an intuitive layout, allowing users to input data effortlessly and receive instant feedback in a visually appealing manner. The web interface is designed to be user-friendly, enabling easy navigation and quick access to fraud detection results. Regular user feedback will also be collected to inform future enhancements and optimizations of the system, helping to ensure that it remains aligned with user needs and expectations.

4.2 UML DIAGRAMS

4.2.1 USECASE DIAGRAM

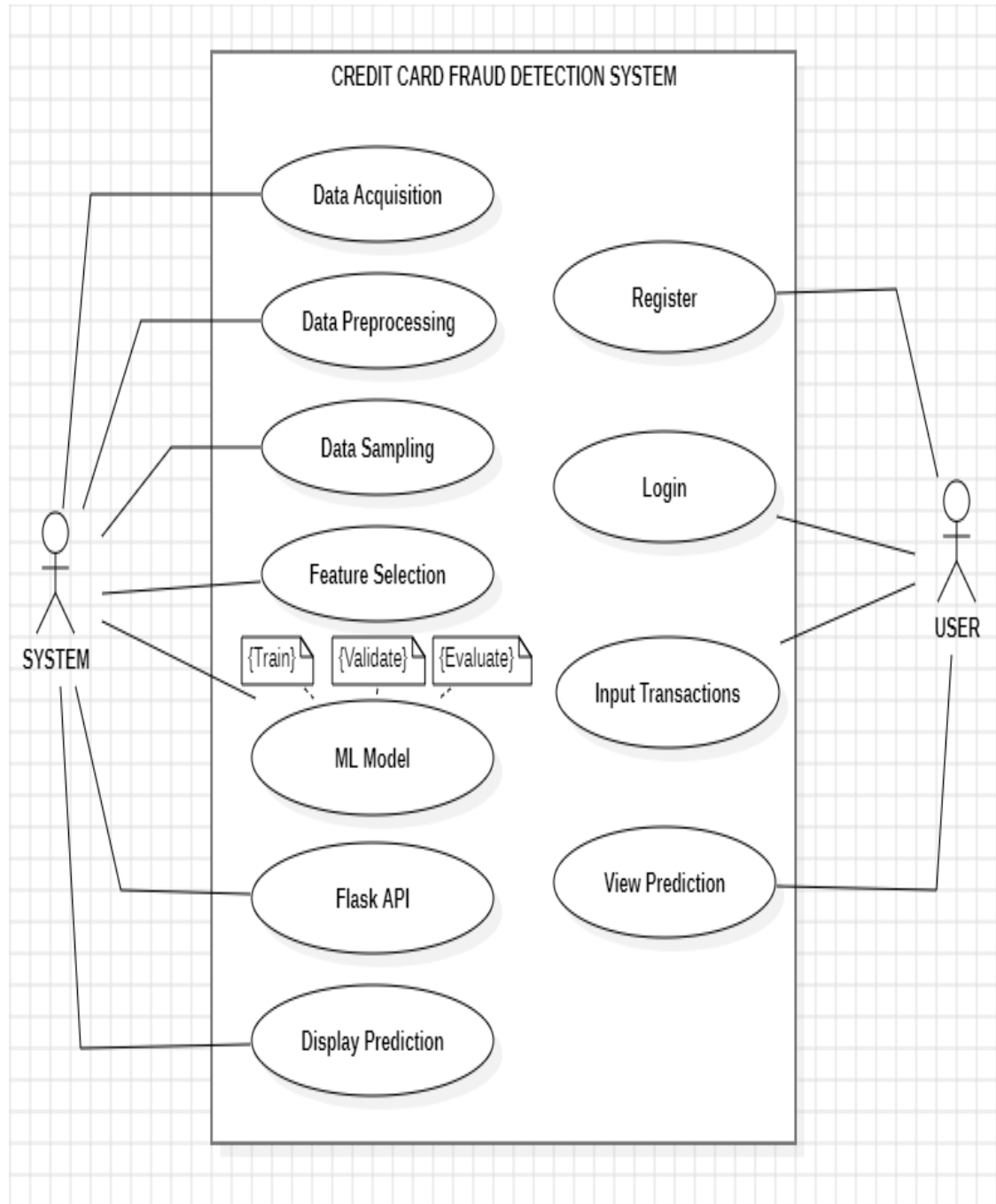


Fig 4.2.1.1 Use Case diagram

4.2.2 ACTIVITY DIAGRAM

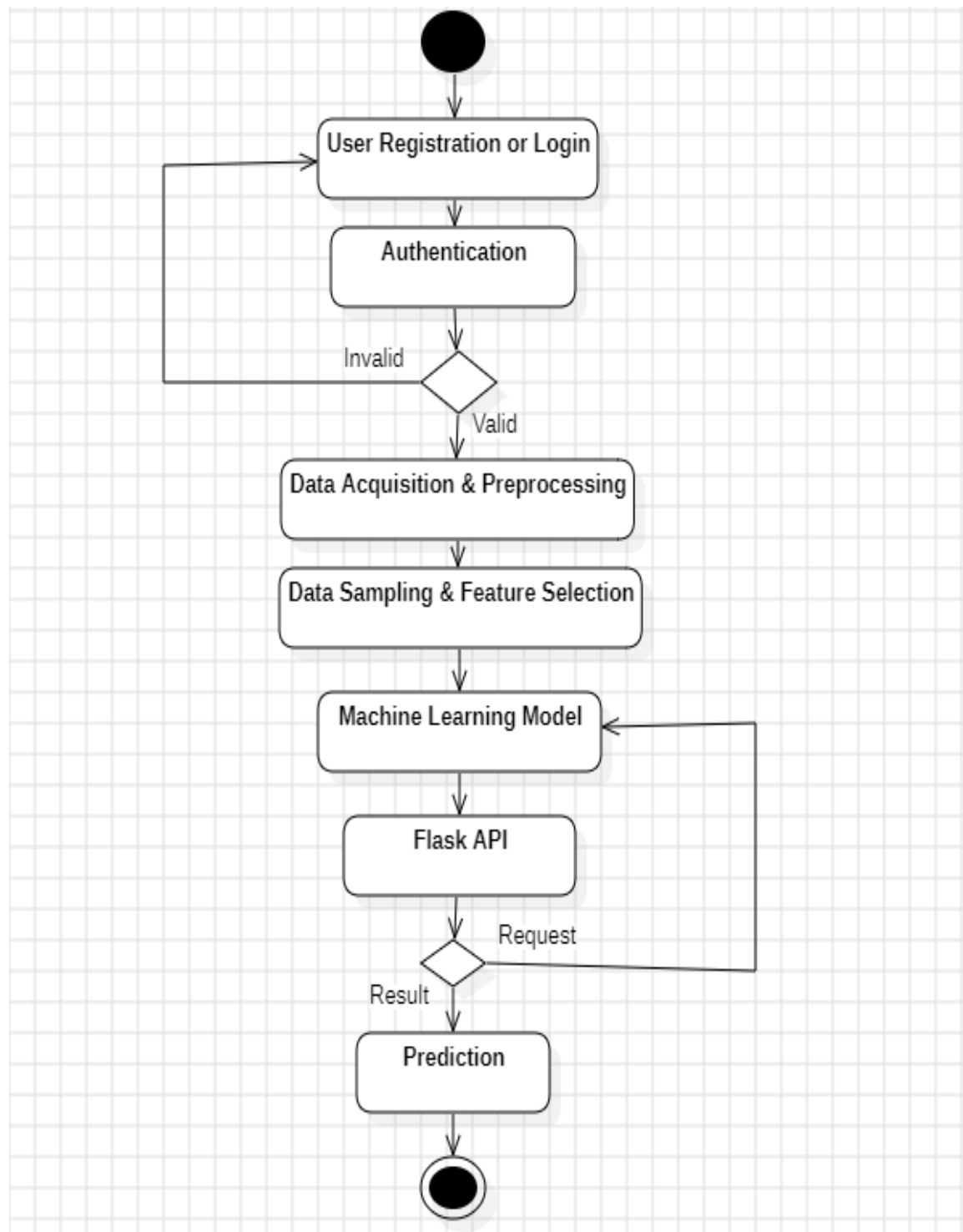


Fig 4.2.2.1 Activity diagram

4.2.3 CLASS DIAGRAM

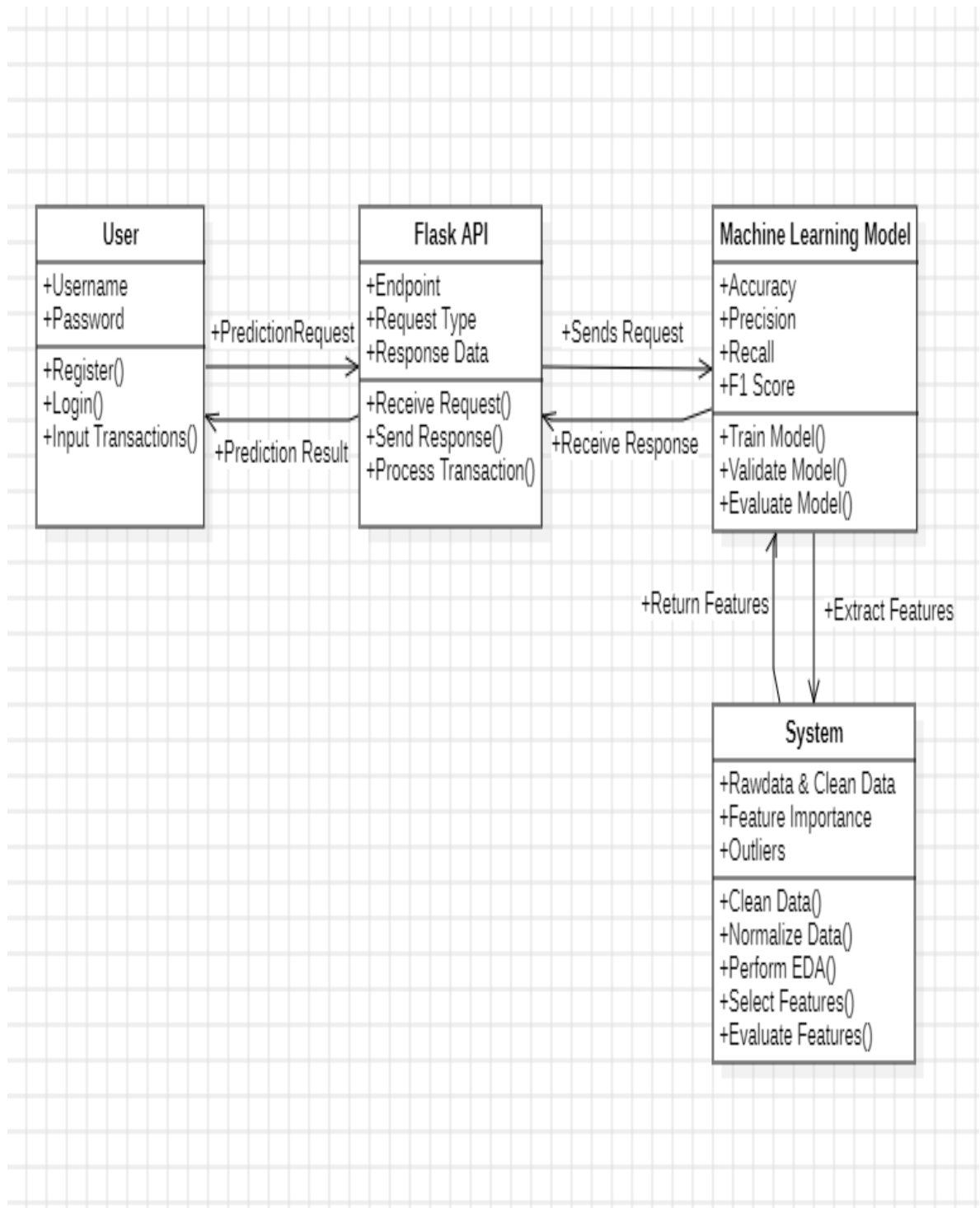


Fig 4.2.3.1 Class diagram

4.2.4 SEQUENCE DIAGRAM

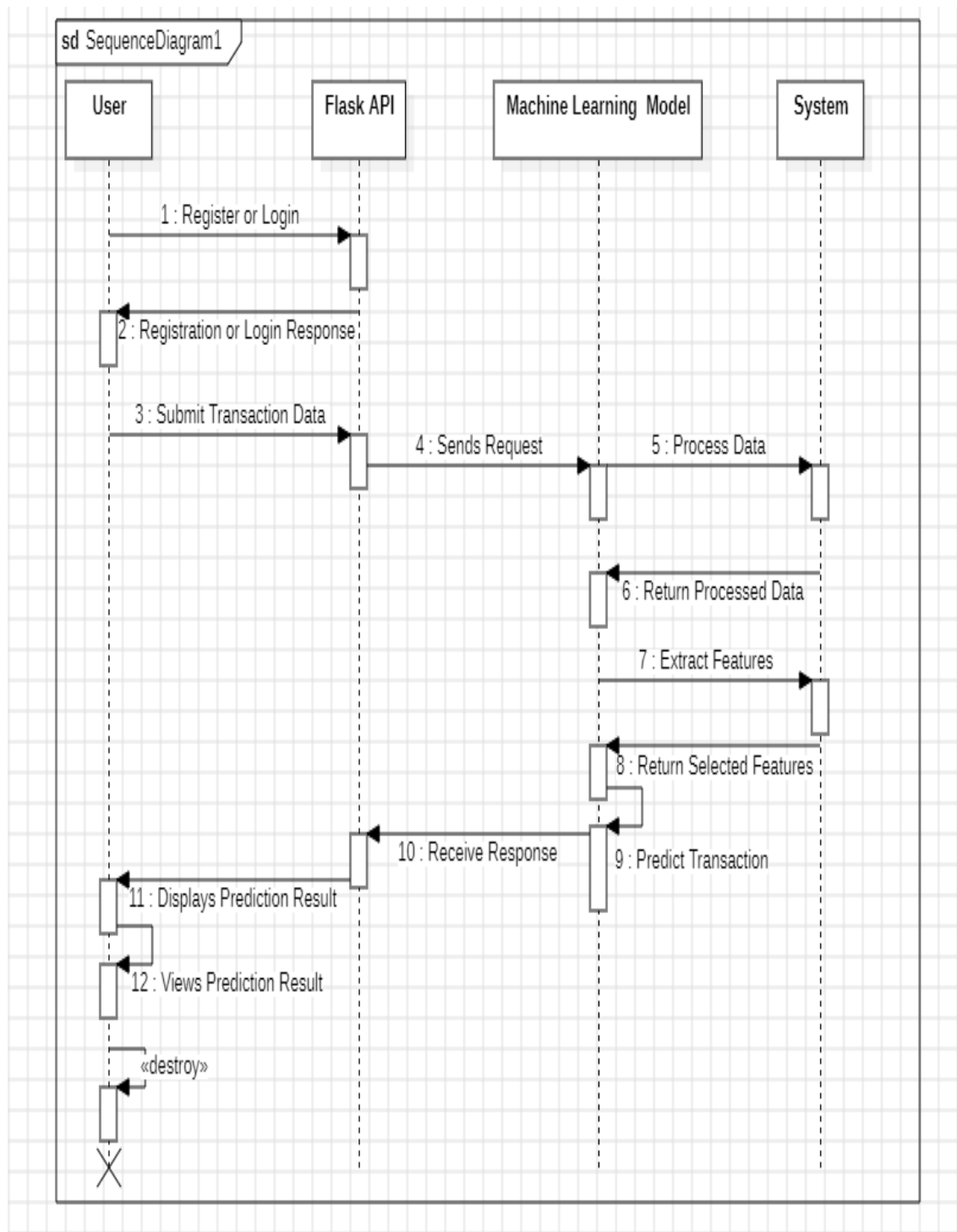


Fig 4.2.4.1 Sequence diagram

4.3 SYSTEM DESIGN

4.3.1 MODULAR DESIGN

The system design for the credit card fraud detection project is structured into distinct modules, each responsible for specific functionalities. This modular approach enhances maintainability, scalability, and clarity within the system architecture. The following modules outline the key components of the design:

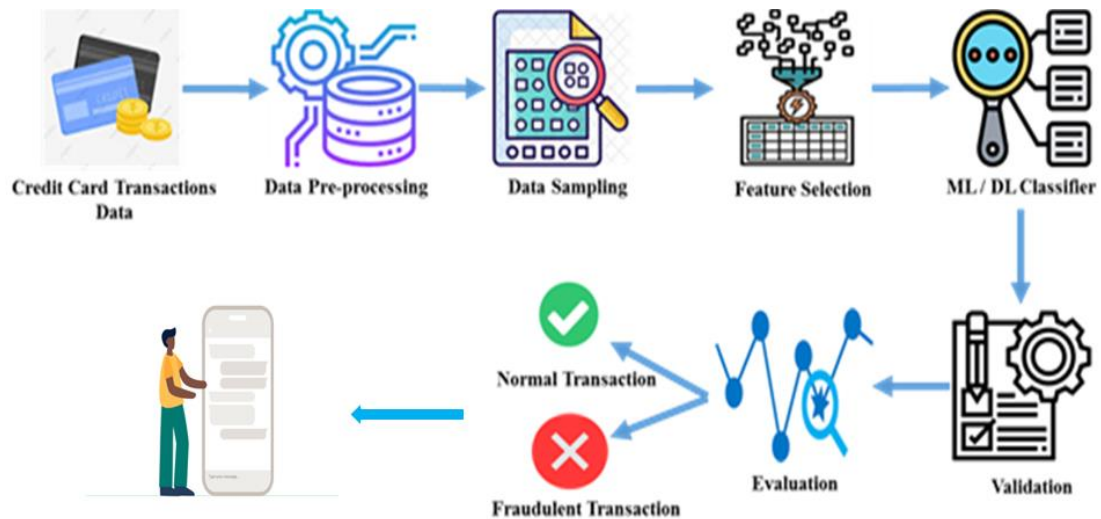


Fig 4.3.1.1 System Design

4.3.1.1 DATA ACQUISITION

This module is dedicated to the collection and curation of a comprehensive dataset pertinent to credit card transactions. Data sources include public repositories such as Kaggle and various financial institutions. The dataset encompasses a broad spectrum of features, including transaction amounts and timestamps, ensuring it accurately reflects real-world scenarios. The quality and diversity of the dataset are vital for enhancing the performance of the subsequent models. Additionally, ensuring the dataset is up-to-date with the latest transaction patterns and fraud trends helps the model remain relevant and effective in detecting emerging fraud schemes.

4.3.1.2 DATA PREPROCESSING

In this module, the collected dataset undergoes critical preprocessing steps to enhance its quality and ensure it is suitable for analysis. Key activities include cleaning the data by removing duplicates, addressing missing values, and normalizing features to a common scale. Additionally, exploratory data analysis (EDA) is conducted to uncover patterns and relationships within the dataset, facilitating a deeper understanding of data distribution and factors influencing fraud. Visualization techniques such as histograms, box plots, and correlation matrices are employed to identify trends and anomalies. This module lays a solid foundation for effective data analysis, ensuring the dataset is consistent, well-structured, and insightful, which ultimately boosts the predictive performance of the model.

4.3.1.3 DATA SAMPLING

To tackle the class imbalance prevalent in credit card fraud detection datasets, this module implements data sampling techniques. Undersampling reduces the number of instances in the majority class, while oversampling increases instances in the minority class to achieve a balanced dataset. Techniques like SMOTE (Synthetic Minority Over-sampling Technique) may be utilized to generate synthetic samples for the minority class, enhancing the model's ability to identify fraudulent transactions. Achieving this balance is critical for improving model performance and minimizing bias in predictions.

4.3.1.4 FEATURE SELECTION

This module focuses on identifying and selecting relevant features that significantly contribute to fraud prediction. Various techniques including filter methods, wrapper methods, and embedded methods are applied to evaluate the importance of each feature. The objective is to retain the most informative features while eliminating redundant or irrelevant data that could complicate the model. This process not only enhances model accuracy but also reduces computational overhead, resulting in a more efficient training process. Additionally, feature selection helps improve the model's interpretability, enabling stakeholders to better understand which factors are driving fraud detection decisions.

4.3.1.5 MACHINE LEARNING MODEL

This module is dedicated to the development and fine-tuning of machine learning models specifically tailored for fraud detection. Various algorithms, such as logistic regression, decision trees, and random forests, are trained on the preprocessed dataset. Model performance is evaluated using validation techniques to ensure robustness, employing metrics such as accuracy, precision, recall, and F1 score to assess effectiveness in detecting fraudulent transactions. Continuous iteration and optimization are performed to achieve optimal results, with the best-performing model selected for deployment, ensuring adaptability to evolving fraud tactics.

4.3.1.6 FLASK API

The Flask API module serves as the communication bridge between the trained machine learning model and end users. It efficiently handles user requests and transmits relevant data to the fraud detection model for analysis. The API is responsible for receiving transaction details, processing them through the model, and returning real-time fraud detection results. Designed for accessibility and user-friendliness, this module facilitates easy integration with frontend applications. Additionally, the API manages authentication and authorization to ensure that only authorized users can access sensitive transaction data. Logging capabilities are incorporated to monitor system performance and user interactions, allowing for quick identification and resolution of any emerging issues.

4.3.1.7 USER INTERFACE

The User Interface module provides platforms for merchants and stakeholders to interact with the fraud detection system. Users can input transaction data via a web interface, "Fraud Shield Analytics" or a Tkinter-based GUI, receiving real-time fraud feedback and detailed detection reports. Designed for ease of use, the Tkinter GUI offers a simple layout for quick data input and feedback, while the web interface ensures smooth navigation and fast access to results. User feedback will be collected to inform future improvements, keeping the system aligned with user needs.

4.3.2 DATABASE DESIGN

The database design for the credit card fraud detection system is structured to handle a large volume of transactional data while maintaining efficiency and accessibility for analysis. The dataset contains transactions made by European cardholders, spanning two days and comprising 284,807 transactions, out of which 492 are fraudulent. Given the class imbalance, where fraudulent transactions account for only 0.172% of the total, the design emphasizes efficient data retrieval and processing for both the minority and majority classes.

Brief description of the Dataset

| S.no | Feature Name | Description |
|------|---------------------|---|
| 1 | Time | The elapsed time (in seconds) between each transaction and the first transaction in the dataset. |
| 2 | Amount | The transaction amount, which can also serve for cost-sensitive learning. |
| 3 | V1-V28 Transactions | Principal components derived from a PCA (Principal Component Analysis) transformation, preserving confidentiality by not revealing the original features. |
| 4 | Class | The target variable indicating whether a transaction is fraudulent (1) or not (0). |

The database design is optimized for handling a large, imbalanced dataset, facilitating quick data access for model training and prediction tasks. This structure enables fast and efficient querying, allowing machine learning models to seamlessly access and process the relevant transaction data for fraud detection.

5. IMPLEMENTATION

5.1 IMPLEMENTATION

5.1.1 SLICING AND DICING

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 |
|---|------|-----------|-----------|----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 0 | 0.0 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 | 0.239599 | 0.098698 | 0.363787 | 0.090794 |
| 1 | 0.0 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.078803 | 0.085102 | -0.255425 | -0.166974 |
| 2 | 1.0 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.791461 | 0.247676 | -1.514654 | 0.207643 |
| 3 | 1.0 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 | 0.237609 | 0.377436 | -1.387024 | -0.054952 |
| 4 | 2.0 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.592941 | -0.270533 | 0.817739 | 0.753074 |

Fig 5.1.1.1 First five Observations in the dataset

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 |
|--------|----------|------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|----------|-----------|
| 284802 | 172786.0 | -11.881118 | 10.071785 | -9.834783 | -2.066656 | -5.364473 | -2.606837 | -4.918215 | 7.305334 | 1.914428 | 4.356170 |
| 284803 | 172787.0 | -0.732789 | -0.055080 | 2.035030 | -0.738589 | 0.868229 | 1.058415 | 0.024330 | 0.294869 | 0.584800 | -0.975926 |
| 284804 | 172788.0 | 1.919565 | -0.301254 | -3.249640 | -0.557828 | 2.630515 | 3.031260 | -0.296827 | 0.708417 | 0.432454 | -0.484782 |
| 284805 | 172788.0 | -0.240440 | 0.530483 | 0.702510 | 0.689799 | -0.377961 | 0.623708 | -0.686180 | 0.679145 | 0.392087 | -0.399126 |
| 284806 | 172792.0 | -0.533413 | -0.189733 | 0.703337 | -0.506271 | -0.012546 | -0.649617 | 1.577006 | -0.414650 | 0.486180 | -0.915427 |

Fig 5.1.1.2 Last five Observations in the dataset

Observation

1. The dataset includes numerical features V1 to V28, derived from PCA, while 'Time' and 'Amount' remain untransformed.
2. The response variable 'Class' is binary, with a value of 1 indicating a fraudulent transaction and 0 indicating a legitimate one.

5.1.2 CHECKING FOR DATA CONSISTENCY

1. Duplicates found

```
#Checking for the Duplicate Values in the Dataframe  
creditcard_df.duplicated().sum()  
  
9144
```

Fig 5.1.2.1 Checking duplicates

2. Unique Values

```
V1      275663  
V2      275663  
V3      275663  
V4      275663  
V5      275663  
V6      275663  
V7      275663  
V8      275663  
V9      275663  
V10     275663  
V11     275663  
V12     275663  
V13     275663  
V14     275663  
V15     275663  
V16     275663  
V17     275663  
V18     275663  
V19     275663  
V20     275663  
V21     275663  
V22     275663  
V23     275663  
V24     275663  
V25     275663  
V26     275663  
V27     275663  
V28     275663  
Amount  32767  
Class    2  
dtype: int64
```

Fig 5.1.2.2 Checking unique values count in all the features

Observation

1. The dataset contains 9,144 duplicate records, which may indicate that certain transactions are repeated. These duplicates should be handled appropriately during data preprocessing, as they could affect the model's performance by introducing bias or redundancy.
2. Most features (V1 to V28) have 275,663 unique values, while Time has 124,592, Amount has 32,767, and Class has 2 unique values, indicating high feature variability and binary classification for fraud detection.

5.1.3 EXPLORATORY DATA ANALYSIS

1. Distribution of Transaction Time

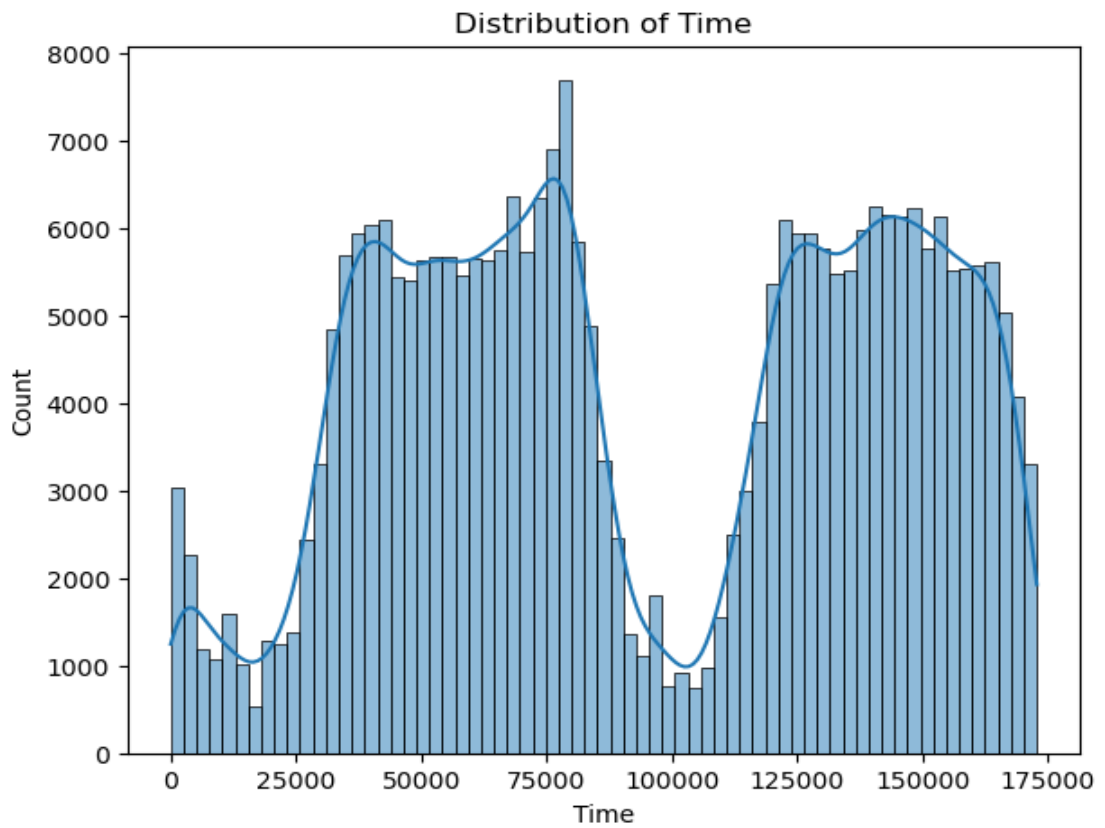


Fig 5.1.3.1 Distribution of Time

Observation

1. The plot shows two distinct peaks, indicating that transaction activity is concentrated in two main time windows during the 48-hour period.
2. There are noticeable dips between the peaks, suggesting lower transaction activity during certain times.
3. The distribution is bimodal, likely reflecting periods of higher activity, possibly linked to working hours or specific time zones in Europe.
4. The largest spike occurs around 75,000 seconds (approximately 21 hours), possibly indicating a surge in transaction activity during this specific period.

2. Distribution of Amount by Fraud and Non-Fraud Classes

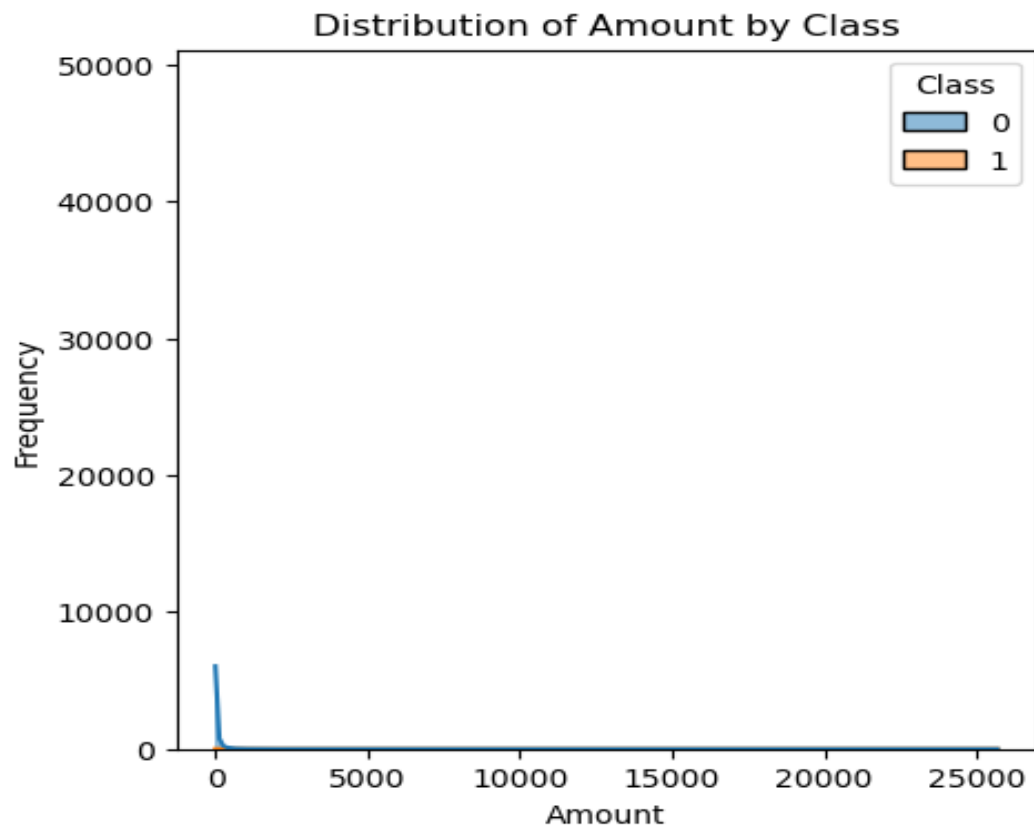


Fig 5.1.3.2 Distribution of Amount by Class

Observation

1. The majority of transactions, both fraudulent (Class 1) and non-fraudulent (Class 0), involve small amounts, as indicated by the high frequency near the lower end of the amount axis.
2. Larger transaction amounts are relatively rare and spread across the dataset, with only a few instances surpassing 10,000, and these are predominantly non-fraudulent transactions.

3. Distribution of Class

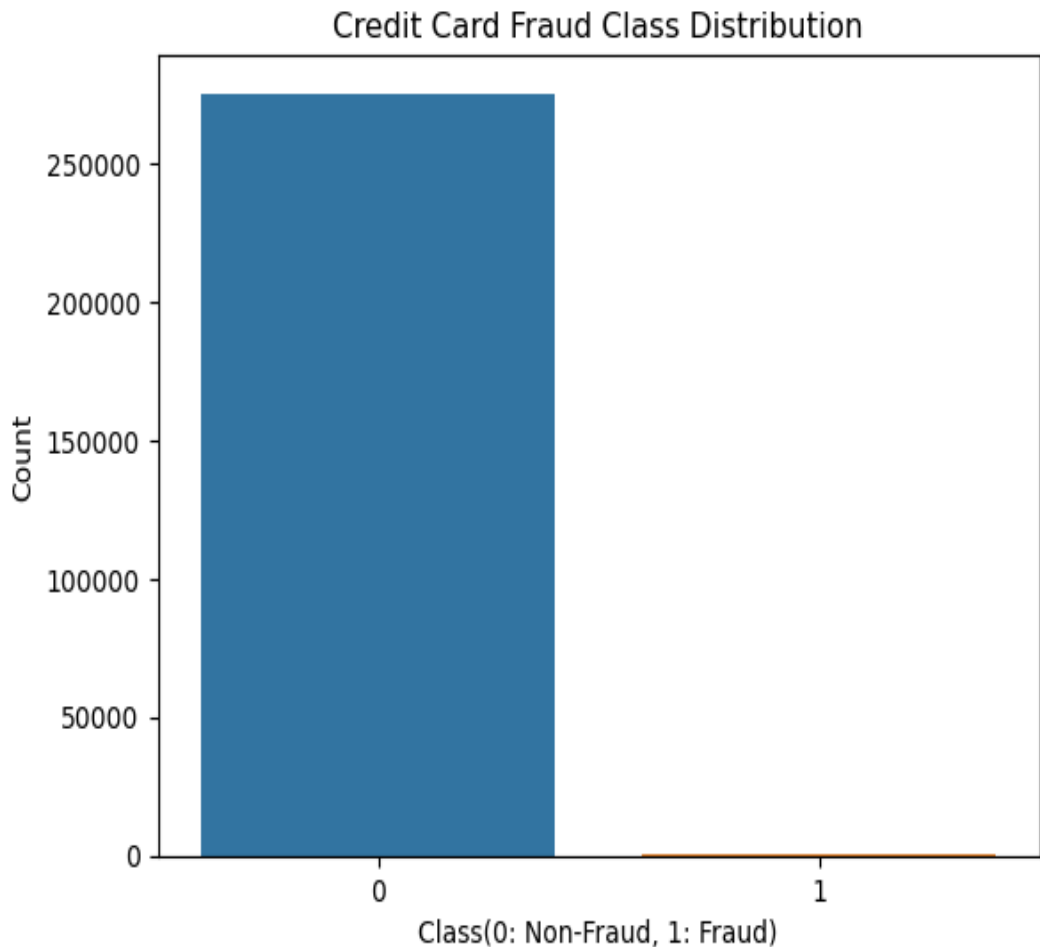


Fig 5.1.3.3 Class Distribution

Observation

1. The dataset is highly imbalanced, with a significantly higher number of non-fraudulent transactions (Class 0) compared to fraudulent ones (Class 1), indicating that fraud cases are rare.
2. This imbalance suggests that special techniques, such as resampling or using anomaly detection methods, may be necessary to effectively model the data and detect fraudulent transactions.

4. Analysis of Transaction Amounts Against Time

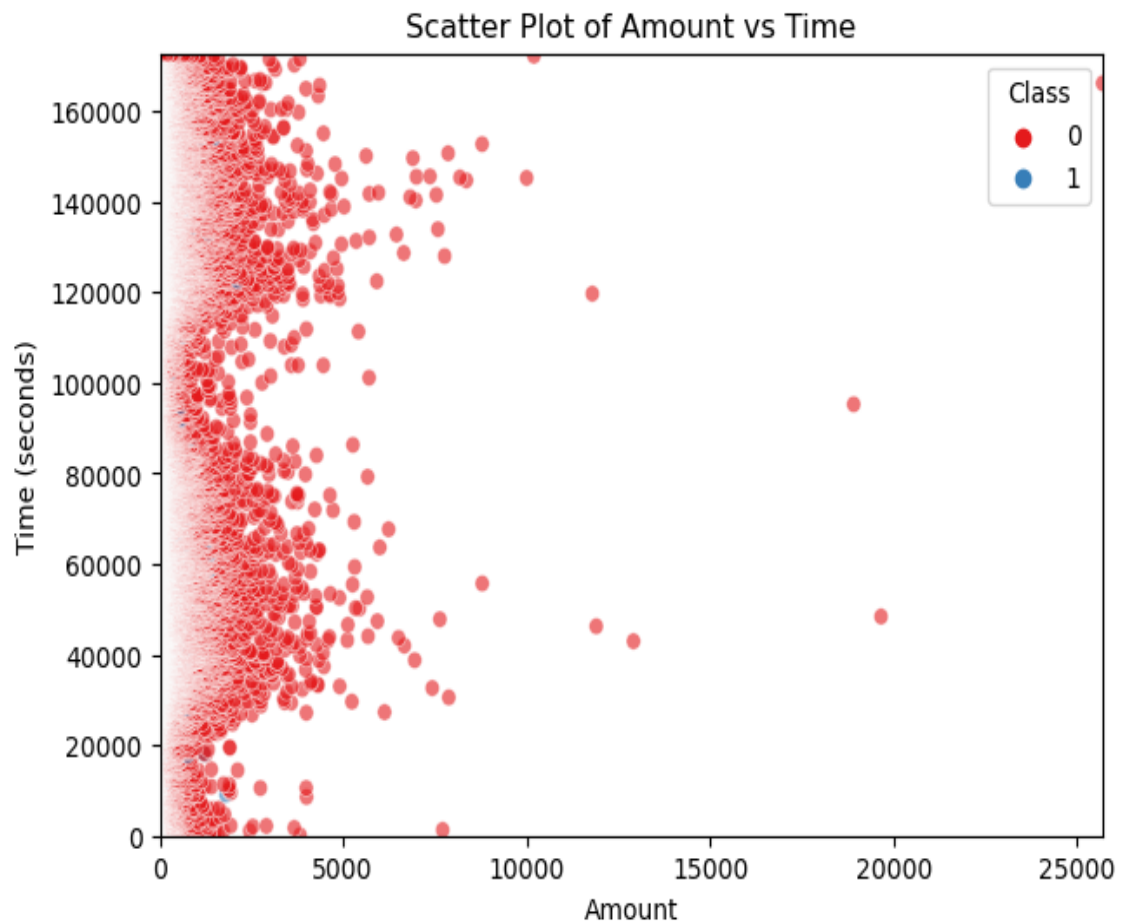


Fig 5.1.3.4 Scatter Plot of Amount vs Time

Observation

1. The majority of transactions cluster at lower amounts suggesting that most transactions are of small value, which is typical in credit card transactions.
2. The Time axis shows a wide range, with transactions occurring at various times throughout the observed period. However, there are no clear patterns indicating specific times that are more likely to have fraudulent transactions.
3. The plot illustrates a significant class imbalance, with many more non-fraudulent transactions (Class 0) compared to fraudulent ones (Class 1). This highlights the need for careful handling of class distribution in modeling to avoid bias.

5.1.4 DESCRIPTIVE STATISTICS AND CORRELATIONS

1. Correlation Heat Map of input variables

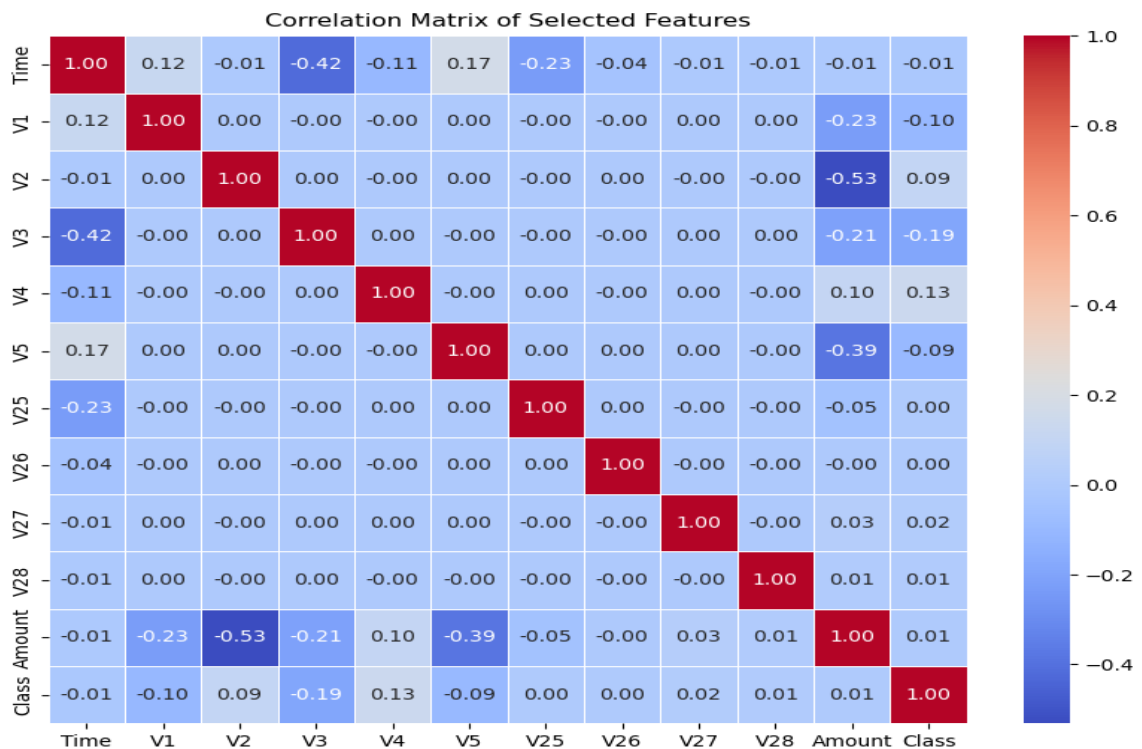


Fig 5.1.4.1 Correlation Heatmap of Selected Features

Observation

1. There is a strong negative correlation between the transaction amount and V2, indicating that as the transaction amount increases, V2 values significantly decrease. This suggests an inverse relationship between these two features.
2. A similar negative correlation is observed between Amount and V3, where larger transaction amounts correspond to lower V3 values, reinforcing the pattern of decreasing feature values with increasing amounts.
3. The correlation between transaction amount and the classification of transactions as fraudulent or legitimate is very low (0.01), indicating that the amount of a transaction has little influence on whether it is flagged as fraud or not.
4. There is a moderate positive correlation between the Class label (fraud or non-fraud) and V4, suggesting that higher V4 values are slightly associated with a higher likelihood of a transaction being classified as fraudulent.
5. A positive correlation is also noted between Class and V2, indicating that higher V2 values may have a weak association with fraudulent transactions, although the relationship is not very strong.
6. The correlation between Time and Class is virtually non-existent, showing that the time at which a transaction occurs has no significant impact on whether it is classified as fraudulent or legitimate.
7. A small negative correlation between V1 and Class suggests that higher V1 values are slightly more associated with legitimate transactions rather than fraudulent ones.
8. A very small positive correlation exists between V27 and Class, indicating a weak relationship where higher V27 values have a negligible association with fraudulent transactions.

These correlations provide critical insights into how different transaction features interact with each other and with fraud classification, helping to inform feature selection and model development for fraud detection.

5.1.5 DATA SAMPLING

1. Class Distribution Before and After Undersampling

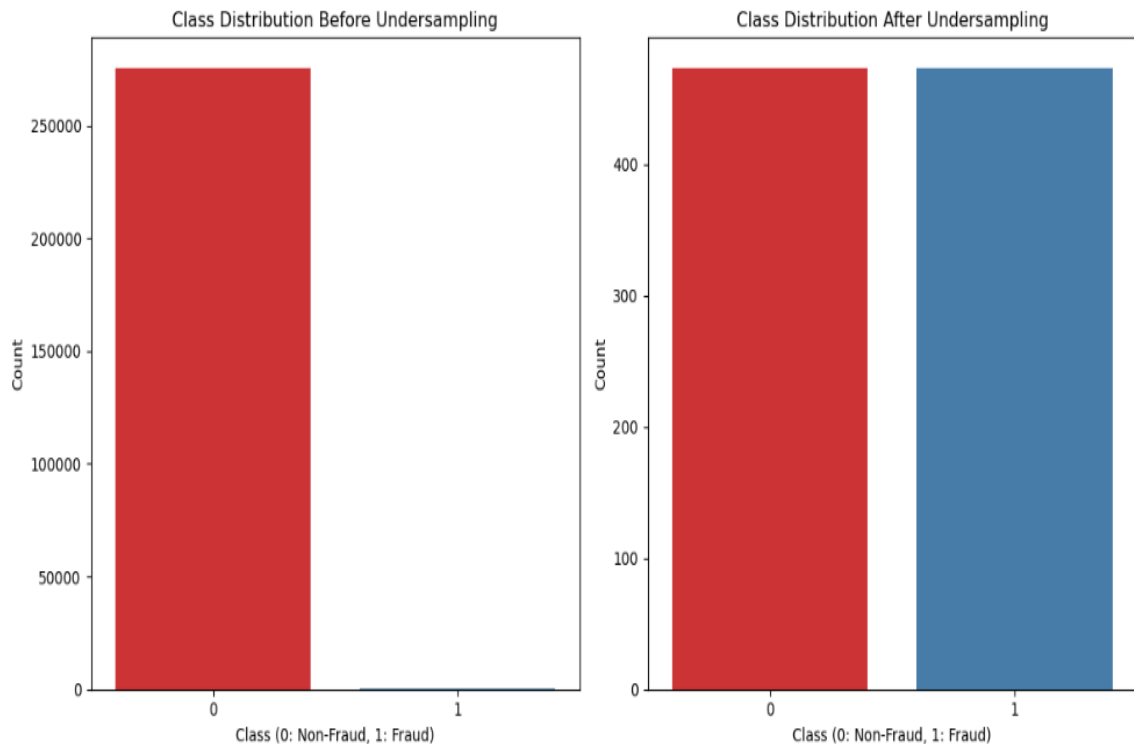


Fig 5.1.5.1 Class Distribution Before and After Undersampling

Observation

1. The original dataset is highly imbalanced, featuring 275,190 non-fraudulent (Class 0) transactions against only 473 fraudulent (Class 1) transactions, which can skew model predictions toward the majority class.
2. Post-undersampling, both classes are balanced at 473 transactions each. While this addresses the imbalance, it risks losing important data from the majority class.
3. To preserve valuable information while ensuring balance, oversampling techniques like SMOTE could be employed for the minority class. This would help in better training of the model without losing critical transaction details from the original data.

2. Class Distribution Before and After Oversampling

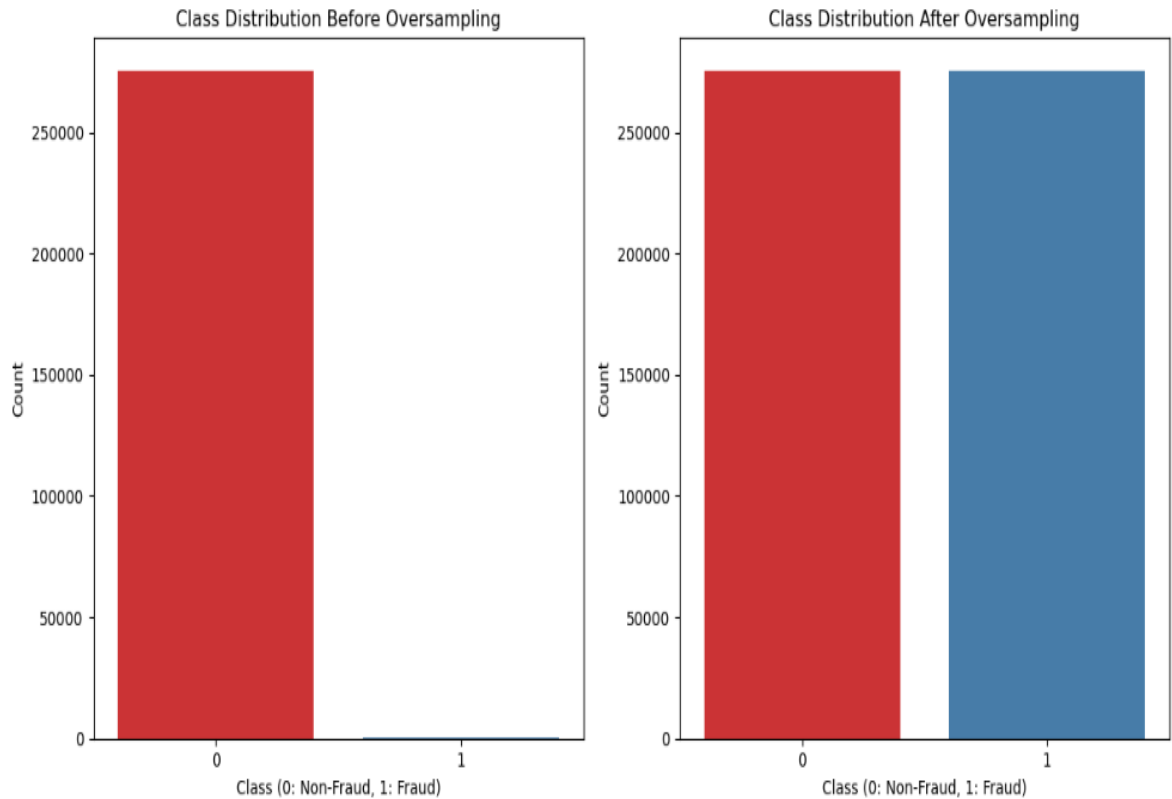


Fig 5.1.5.2 Class Distribution Before and After Oversampling

Observation

1. The undersampled dataset contains 473 instances each of non-fraud (Class 0) and fraud (Class 1), promoting an equitable training process and reducing potential bias from class imbalance.
2. After applying the SMOTE technique, both classes increased to 275,190 instances each, further balancing the dataset and allowing the model to learn effectively from an equal number of examples.
3. This balanced representation is crucial for developing a robust fraud detection model, as it improves performance and generalizability by providing sufficient training data for both classes.

5.1.6 ALGORITHMS

5.1.6.1 MODEL TRAINING AND EVALUATION

The following algorithms are used to predict Credit Card Fraud

1. Logistic Regression
2. Decision Tree Classifier
3. Random Forest Classifier

1. Logistic Regression

| Classification Report for Logistic Regression: | | | | |
|--|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 0.92 | 0.97 | 0.95 | 55073 |
| 1 | 0.97 | 0.92 | 0.94 | 55003 |
| accuracy | | | 0.95 | 110076 |
| macro avg | 0.95 | 0.95 | 0.95 | 110076 |
| weighted avg | 0.95 | 0.95 | 0.95 | 110076 |

Fig 5.1.6.1.1 Classification Report for Logistic Regression

Observation

1. Class 0 (non-fraudulent transactions) has a precision of 0.92, meaning 92% of predicted non-fraudulent transactions were accurate, while class 1 (fraudulent transactions) has a precision of 0.97, indicating that 97% of predicted fraudulent transactions were correct.
2. Class 0's recall is 0.97, and class 1's is 0.91, indicating high detection rates for non-fraudulent and slightly lower for fraudulent transactions.
3. Class 0's F1-score is 0.95, and class 1's is 0.94, reflecting a strong balance between precision and recall.
4. The model achieves 0.94 overall accuracy, with room for improvement, particularly in reducing false negatives.

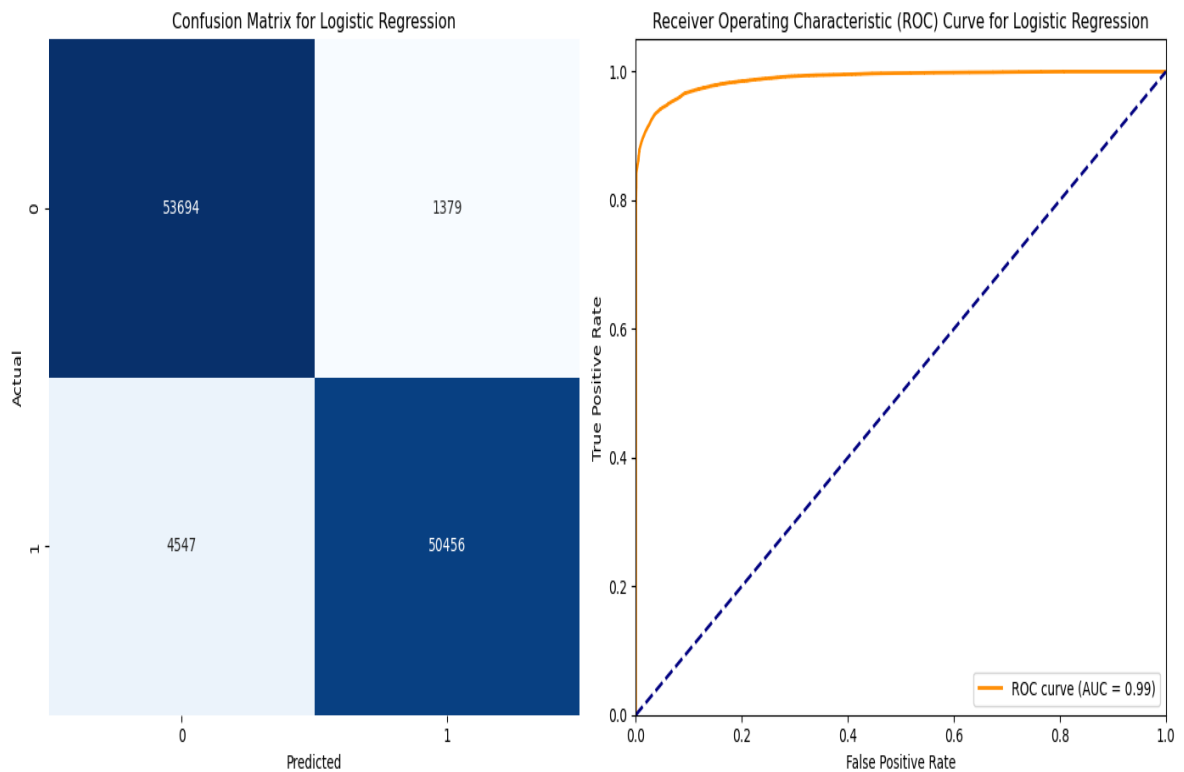


Fig 5.1.6.1.2 Confusion Matrix and ROC Curve for Logistic Regression

Observation

1. The model correctly predicted 50,456 non-fraudulent transactions (True Negatives) and 53,694 fraudulent transactions (True Positives).
2. There were 1,379 instances where the model incorrectly flagged non-fraudulent transactions as fraudulent (False Positives). Additionally, 4,547 fraudulent transactions were misclassified as non-fraudulent (False Negatives).
3. The ROC curve shows strong performance with an AUC (Area Under the Curve) of 0.99, indicating that the model is excellent at distinguishing between fraudulent and non-fraudulent transactions.
4. Additionally, the ROC curve's proximity to the top-left corner indicates a low false positive rate while maintaining a high true positive rate, further demonstrating the model's effectiveness in accurately classifying transactions.

2. Decision Tree Classifier

| Classification Report for Decision Tree Classifier: | | | | |
|---|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 0.9992 | 0.9973 | 0.9983 | 55073 |
| 1 | 0.9973 | 0.9992 | 0.9983 | 55003 |
| accuracy | | | 0.9983 | 110076 |
| macro avg | 0.9983 | 0.9983 | 0.9983 | 110076 |
| weighted avg | 0.9983 | 0.9983 | 0.9983 | 110076 |

Fig 5.1.6.1.3 Classification Report for Decision Tree

Observation

1. The precision for class 0 (Non Fraudulent transactions) is 0.9992, indicating that 99.92% of predicted non-fraudulent transactions were correct. For class 1 (Fraudulent transactions), the precision is slightly lower at 0.9973, demonstrating that 99.73% of predicted fraudulent transactions were accurate.
2. The F1-scores are 0.9983 for both classes, highlighting an excellent balance between precision and recall. This underscores the model's effectiveness in accurately classifying both non-fraudulent and fraudulent transactions.
3. The overall accuracy of the model is 0.9983, suggesting that 99.83% of all transactions were classified correctly. This exceptionally high accuracy reflects the model's robustness in distinguishing between fraudulent and non-fraudulent transactions.
4. The support for both classes is nearly equal, with 55,073 instances for Non Fraud and 55,003 for Fraud. This balanced distribution indicates that the model was trained on a well-represented dataset, significantly enhancing its performance in identifying both classes effectively.

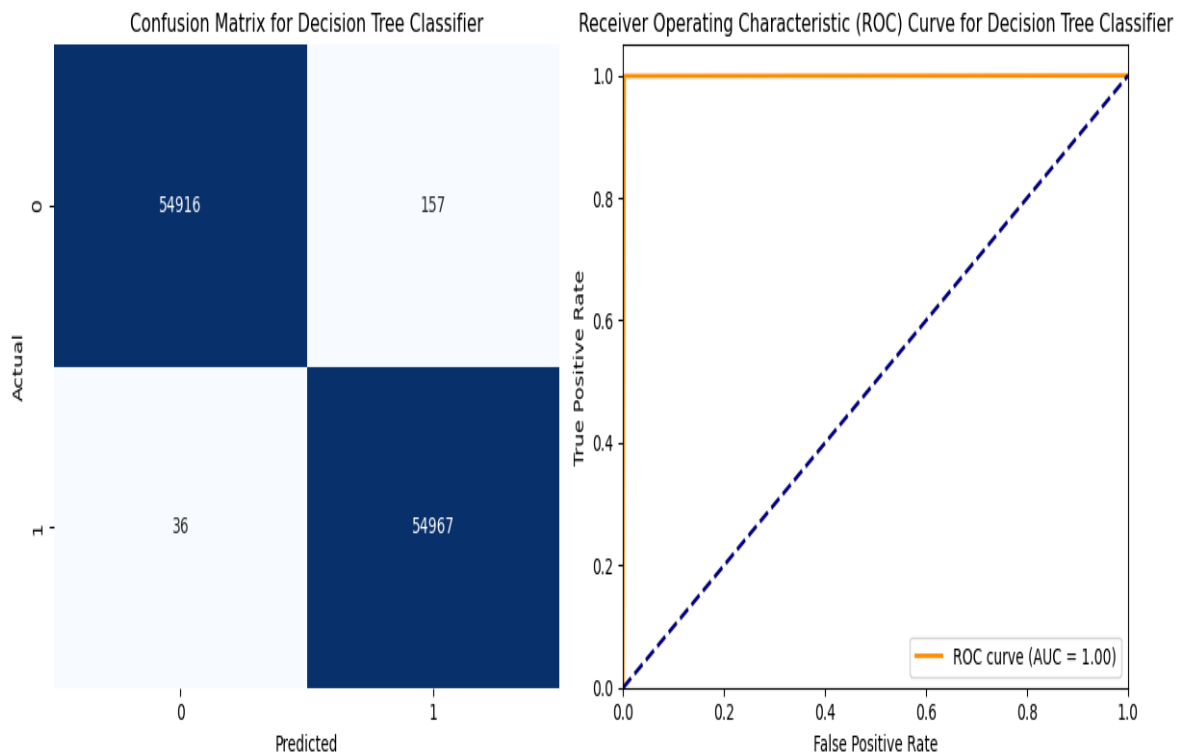


Fig 5.1.6.1.4 Confusion Matrix and ROC Curve for Decision Tree Classifier

Observation

1. The model correctly predicted 54,967 non-fraudulent transactions (True Negatives) and 54,916 fraudulent transactions (True Positives).
2. There were 157 instances where the model incorrectly flagged non-fraudulent transactions as fraudulent (False Positives). Additionally, 36 fraudulent transactions were misclassified as non-fraudulent (False Negatives).
3. The ROC curve shows strong performance with an AUC (Area Under the Curve) of 1.00, indicating that the model is excellent at distinguishing between fraudulent and non-fraudulent transactions.
4. Additionally, the ROC curve's proximity to the top-left corner indicates a low false positive rate while maintaining a high true positive rate, further demonstrating the model's effectiveness in accurately classifying transactions.

3. Random Forest Classifier

| Classification Report for Random Forest Classifier: | | | | | |
|---|-----------|--------|----------|---------|--|
| | precision | recall | f1-score | support | |
| 0 | 1.00 | 1.00 | 1.00 | 55073 | |
| 1 | 1.00 | 1.00 | 1.00 | 55003 | |
| accuracy | | | 1.00 | 110076 | |
| macro avg | 1.00 | 1.00 | 1.00 | 110076 | |
| weighted avg | 1.00 | 1.00 | 1.00 | 110076 | |

Fig 5.1.6.1.5 Classification Report for Random Forest Classifier

Observation

1. The precision for both classes (non-fraudulent and fraudulent transactions) are 1.00, indicating that all predicted instances for each class are correct. This suggests the model has excellent performance in identifying both fraudulent and non-fraudulent transactions without any misclassifications.
2. The recall for both classes (non-fraudulent and fraudulent transactions) is 1.00, indicating that every instance predicted for each class is accurate. This demonstrates that the model performs exceptionally well in identifying both fraudulent and non-fraudulent transactions without any misclassifications.
3. The F1-scores for both classes are also 1.00, reflecting a perfect balance between precision and recall. This indicates that the model not only predicts accurately but also captures all relevant instances, which is crucial in fraud detection scenarios.
4. The model achieves an overall accuracy of 1.00, meaning it correctly classifies all instances in the test dataset. This balanced distribution indicates that the model was trained on a well-represented dataset, enhancing its performance in identifying both classes effectively.

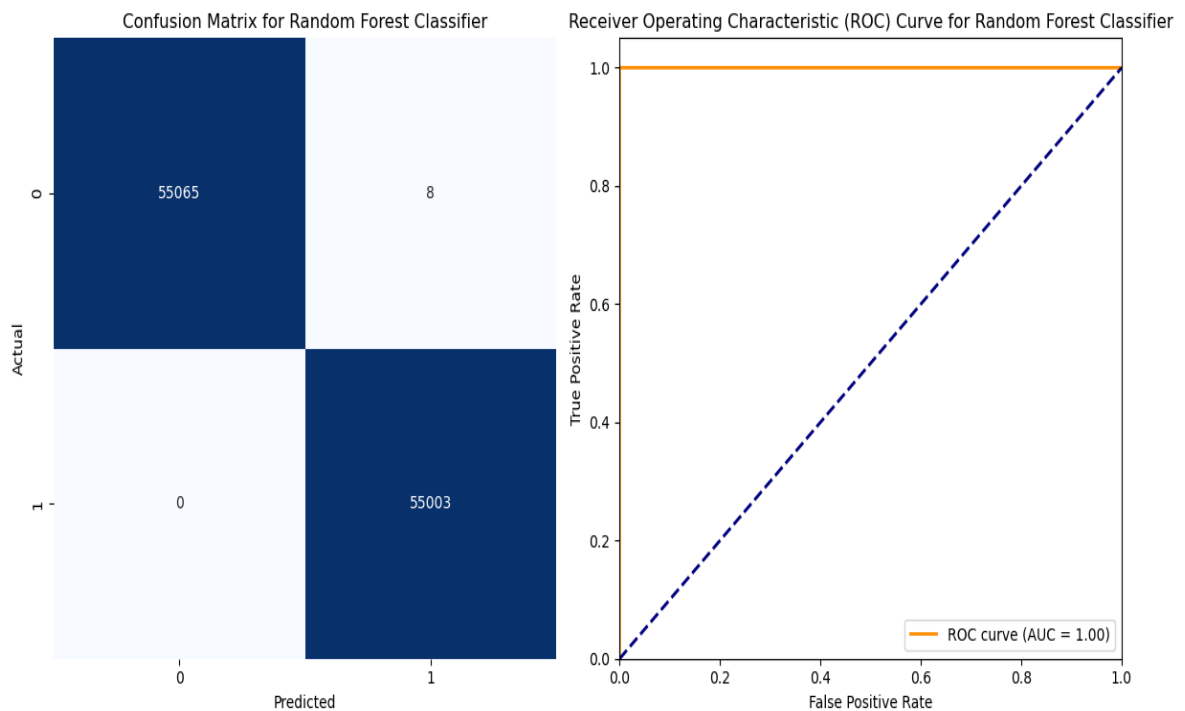


Fig 5.1.6.1.6 Confusion Matrix and ROC Curve for Random Forest Classifier

Observation

1. The confusion matrix shows that the model correctly classified 55,065 non-fraudulent transactions and 55,003 fraudulent transactions. There were only 8 false positives, indicating a very low rate of misclassification for non-fraudulent transactions, which highlights the model's reliability.
2. The ROC curve illustrates that the model achieved an AUC (Area Under the Curve) of 1.00, indicating perfect separation between fraudulent and non-fraudulent transactions. This suggests that the Random Forest Classifier is highly effective in distinguishing between the two classes.
3. The near-perfect recall for both classes indicates that the model can identify nearly all actual fraudulent and non-fraudulent transactions, highlighting its effectiveness in fraud detection. The Random Forest Classifier shows exceptional performance and robustness, making it an ideal choice for real-world fraud detection systems.

5.1.6.2 MODEL SELECTION

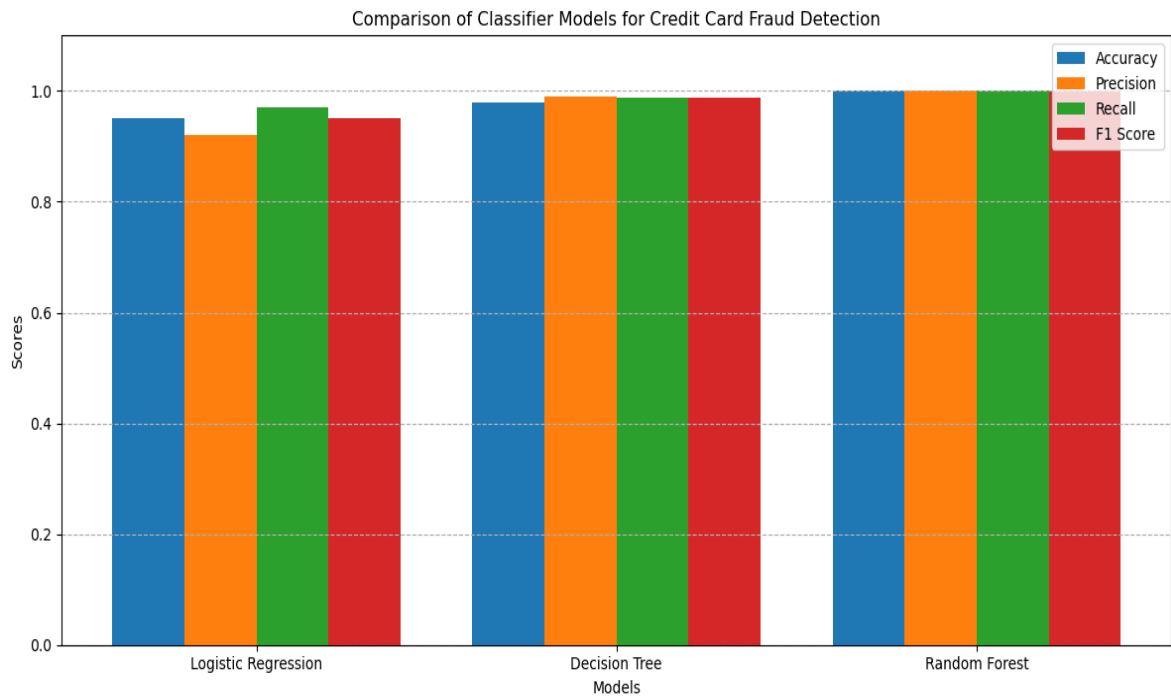


Fig 5.1.6.2.1 Performance Comparison of Classifier Models

Observation:

The Random Forest Classifier is the best model for predicting credit card fraud, achieving an accuracy of 1.00 across all metrics. The confusion matrix shows correct classifications of 55,065 non-fraudulent transactions and 55,003 fraudulent transactions, with only 8 false positives. Therefore, we save the model for fraud detection.

```
RandomForestClassifier  
RandomForestClassifier()
```

Fig 5.1.6.2.2 Saving the Random Forest Classifier Model for Fraud Detection

5.2 SAMPLE CODE

5.2.1 GUI IMPLEMENTATION CODE

```
def show_entry_fields():

    values = [float(entry.get()) for entry in entries]

    model = joblib.load('Credit_Card_Model')

    y_predict = model.predict([values])

    if y_predict == 0:

        result = "Normal Transaction"

    else:

        result = "Fraud Transaction"

    messagebox.showinfo("Credit Card Fraud Detection System", f"Final Prediction:

    {result}")

    engine = pyttsx3.init()

    engine.say(f"The transaction is predicted to be a {result}")

    engine.runAndWait()

master = tk.Tk()

master.title("Credit Card Fraud Detection System")

tk.Label(master, text="Credit Card Fraud Detection System", bg="black", fg="white",
```

```

width=50).grid(row=0, column=0, columnspan=2)

entries = []

for i in range(1, 30):

    tk.Label(master, text=f"Enter Value of V{i}").grid(row=i, column=0)

    entry = tk.Entry(master)

    entry.grid(row=i, column=1)

    entries.append(entry)

button_frame = tk.Frame(master)

button_frame.grid(row=30, column=0, columnspan=2, pady=10)

predict_button = tk.Button(button_frame, text='PREDICT', command=show_entry_fields,

width=10)

predict_button.pack(pady=5, padx=10, ipadx=10, ipady=5)

window_width = master.winfo_reqwidth()

window_height = master.winfo_reqheight()

position_right = int(master.winfo_screenwidth() / 2 - window_width / 2)

position_down = int(master.winfo_screenheight() / 2 - window_height / 2)

master.geometry("+{ }+{ }".format(position_right, position_down))

master.mainloop()

```


5.2.2 FRONTEND IMPLEMENTATION- HTML CODE

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Detection - FraudShield Analytics</title>

    <link rel="stylesheet" href="{{ url_for('static', filename='style2.css') }}">

</head>

<body>

    <div class="main">

        <div class="navbar">

            <div class="icon">

                <h2 class="logo">FraudShield Analytics</h2>

            </div>

            <div class="menu">

                <ul>

                    <li><a href="{{ url_for('home') }}">HOME</a></li>

                    <li><a href="{{ url_for('about') }}">ABOUT</a></li>

                    <li><a href="{{ url_for('overview') }}">OVERVIEW</a></li>

                    <li><a href="{{ url_for('predict') }}">DETECTION</a></li>

                    <li><a href="{{ url_for('contact') }}">CONTACT</a></li>

                </ul>

            </div>

        </div>

    </div>
```

```

<div class="search">
    <input class="srch" type="search" name="" placeholder="Type To text">
    <a href="https://www.google.com/"><button class="btn">Search</button></a>
</div>
</div>
<div class="content">
    <div class="center">
        <form id="predictionForm" class="form" action="{{ url_for('predict') }}"
        method="POST">
            <h2>Enter the Transactions</h2>
            <input type="text" id="v1" name="v1" placeholder="Enter value of v1">
            <br><br>
            <input type="text" id="v2" name="v2" placeholder="Enter value of v2">
            <br><br>
            <input type="text" id="v3" name="v3" placeholder="Enter value of v3">
            <br><br>
            <input type="text" id="v4" name="v4" placeholder="Enter value of v4">
            <br><br>
            <input type="text" id="v5" name="v5" placeholder="Enter value of v5">
            <br><br>
            <input type="text" id="v6" name="v6" placeholder="Enter value of v6">
            <br><br>
            <input type="text" id="v7" name="v7" placeholder="Enter value of v7">
            <br><br>

```

<input type="text" id="v8" name="v8" placeholder="Enter value of v8">

<input type="text" id="v9" name="v9" placeholder="Enter value of v9">

<input type="text" id="v10" name="v10" placeholder="Enter value of v10">

<input type="text" id="v11" name="v11" placeholder="Enter value of v11">

<input type="text" id="v12" name="v12" placeholder="Enter value of v12">

<input type="text" id="v13" name="v13" placeholder="Enter value of v13">

<input type="text" id="v14" name="v14" placeholder="Enter value of v14">

<input type="text" id="v15" name="v15" placeholder="Enter value of v15">

<input type="text" id="v16" name="v16" placeholder="Enter value of v16">

<input type="text" id="v17" name="v17" placeholder="Enter value of v17">

<input type="text" id="v18" name="v18" placeholder="Enter value of v18">

<input type="text" id="v19" name="v19" placeholder="Enter value of v19">


```
<input type="text" id="v20" name="v20" placeholder="Enter value of v20">
<br><br>
<input type="text" id="v21" name="v21" placeholder="Enter value of v21">
<br><br>
<input type="text" id="v22" name="v22" placeholder="Enter value of v22">
<br><br>
<input type="text" id="v23" name="v23" placeholder="Enter value of v23">
<br><br>
<input type="text" id="v24" name="v24" placeholder="Enter value of v24">
<br><br>
<input type="text" id="v25" name="v25" placeholder="Enter value of v25">
<br><br>
<input type="text" id="v26" name="v26" placeholder="Enter value of v26">
<br><br>
<input type="text" id="v27" name="v27" placeholder="Enter value of v27">
<br><br>
<input type="text" id="v28" name="v28" placeholder="Enter value of v28">
<br><br>
<input type="text" id="v29" name="v29" placeholder="Enter value of v29">
<br><br>
<button type="submit" class="btnn">Predict</button>
</form>
</div>
</div>
```

</div>

</body>

</html>

5.2.3 FLASK API CODE

```
from flask import Flask, render_template, request, url_for, session, redirect
import pickle
import numpy as np
filename = 'credit-card-model.pkl'
model = pickle.load(open('credit-card-model.pkl', 'rb'))
app = Flask(__name__)
app.secret_key = 'C16'
@app.route('/')
def home():
    return render_template('home.html')
@app.route('/about')
def about():
    if 'loggedin' in session:
        return render_template('about.html')
    return redirect(url_for('home'))
@app.route('/overview')
def overview():
    if 'loggedin' in session:
        return render_template('overview.html')
    return redirect(url_for('home'))
@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':
        v1 = float(request.form['v1'])
```

```

v2= float(request.form['v2'])
v3= float(request.form['v3'])
v4= float(request.form['v4'])
v5= float(request.form['v5'])
v6 = float(request.form['v6'])
v7 = float(request.form['v7'])
v8 = float(request.form['v8'])
v9 = float(request.form['v9'])
v10 = float(request.form['v10'])
v11 = float(request.form['v11'])
v12 = float(request.form['v12'])
v13= float(request.form['v13'])
v14= float(request.form['v14'])
v15= float(request.form['v15'])
v16= float(request.form['v16'])
v17= float(request.form['v17'])
v18 = float(request.form['v18'])
v19 = float(request.form['v19'])
v20 = float(request.form['v20'])
v21 = float(request.form['v21'])
v22 = float(request.form['v22'])
v23 = float(request.form['v23'])
v24 = float(request.form['v24'])
v25 = float(request.form['v25'])
v26 = float(request.form['v26'])
v27 = float(request.form['v27'])
v28 = float(request.form['v28'])
v29 = float(request.form['v29'])
data =np.array([[v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12,v13,v14,v15,v16,v17,
v18,v19,v20,v21,v22,v23,v24,v25,v26,v27,v28,v29]])

```

```

        my_prediction = model.predict(data)
        return render_template('result.html', prediction=my_prediction[0])
    if 'loggedin' in session:
        return render_template('detection.html')
    return redirect(url_for('home'))
@app.route('/result/<prediction>')
def result(prediction):
    return render_template('result.html', prediction=prediction)
@app.route('/contact')
def contact():
    if 'loggedin' in session:
        return render_template('contact.html')
    return redirect(url_for('home'))
@app.route('/signup')
def signup():
    return render_template('signup.html')
@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']
    if username == 'admin' and password == 'password':
        session['loggedin'] = True
        return redirect(url_for('overview'))
    return render_template('home.html', error='Invalid username or password.')
@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    return redirect(url_for('home'))
if __name__ == '__main__':
    app.run(debug=True)

```

5.2.4 MACHINE LEARNING IMPLEMENTATION CODE

1.Data Acquisition and Data Preprocessing

```
#Reading the creditcard.csv file into DataFrame
creditcard_df=pd.read_csv('creditcard.csv')
pd.options.display.max_columns=None
#Displaying the First 5 rows of the dataset
creditcard_df.head()
#Displaying the Last 5 rows of the dataset
creditcard_df.tail()
#Displaying the Unique values of input variables
unique_values = creditcard_df.nunique()
print(unique_values)
#Information about shape of the dataset, datatypes, null values and memory requirements
creditcard_df.info()
#Displaying the number of rows and columns of the dataset
print("Number of Rows:",creditcard_df.shape[0])
print("Number of Columns:",creditcard_df.shape[1])
#The sum of missing values for each column in the DataFrame
creditcard_df.isnull().sum()
#Checking for the Duplicate Values in the Dataframe
creditcard_df.duplicated().sum()
#Eliminating the duplicate Values in the Dataframe
creditcard_df=creditcard_df.drop_duplicates()
#Count of the number of occurrences for each unique value in the 'Class' column used to
understand the distribution of classes in a binary classification problem
creditcard_df['Class'].value_counts()
#Visualizing the distribution of Time which provides insights into the frequency of time
plt.figure(figsize=(15,12))
plt.subplot(2,2,2)
```



```

sns.histplot(creditcard_df['Time'], kde=True)
plt.title("Distribution of Time")
plt.show()
#Visualizing the distribution of Amount by Class
plt.figure(figsize=(5,5))
sns.histplot(creditcard_df,x='Amount',hue='Class',multiple='stack',kde=True)
plt.xlabel('Amount')
plt.ylabel('Frequency')
plt.title('Distribution of Amount by Class')
plt.show()

```

2. Undersampling

```

#Storing Normal transactions in 'normal' variable and Fraud transactions in 'fraud' variable
normal=creditcard_df[creditcard_df['Class']==0]
fraud=creditcard_df[creditcard_df['Class']==1]
#A sample of 473 randomly selected instances is taken from the 'normal' DataFrame, and
the resulting DataFrame is assigned to the variable 'normal_sample'.
normal_sample=normal.sample(n=473)
normal_sample.shape
#A new DataFrame 'newcreditcard_df' is created by concatenating a sample of normal
transactions with the entire fraud transactions DataFrame
newcreditcard_df=pd.concat([normal_sample,fraud],ignore_index=True)
newcreditcard_df['Class'].value_counts()

```

3. Oversampling

```

#Store the feature matrix in vector 'X' and response in vector 'Y'
X=creditcard_df.drop('Class',axis=1)
Y=creditcard_df['Class']
#Performing oversampling on the features X and Y
X_result,Y_result=SMOTE().fit_resample(X,Y)

```

```
Y_result.value_counts()
```

4. Feature Selection

```
# Remove the 'Time' column from the features
```

```
X_result = X_result.drop(columns=['Time'])
```

```
# Display the updated shape of the feature set
```

```
print("Updated feature set shape:", X_result.shape)
```

```
#Splitting the dataset into the train and test sets
```

```
X_train,X_test,Y_train,Y_test=train_test_split(X_result,Y_result,test_size=0.2,random_state=42)
```

5. Logistic Regression

```
#Training the Model using Logistic Regression
```

```
log.fit(X_train,Y_train)
```

```
#Predicting the Model using Logistic Regression
```

```
Y_predict1=log.predict(X_test)
```

```
# Classification report
```

```
print("Classification Report for Logistic Regression:")
```

```
print(classification_report(Y_test, Y_predict1))
```

6. Decision Tree Classifier

```
#Training the Model Using Decision Tree Classifier
```

```
dt.fit(X_train,Y_train)
```

```
#Predicting the Model Using Decision Tree Classifier
```

```
Y_predict2=dt.predict(X_test)
```

```
#Classification Report
```

```
print("Classification Report for Decision Tree Classifier:")
```

```
print(classification_report(Y_test, Y_predict2))
```

7. Random Forest Classifier

```
#Training the model using Random Forest Classifier
rf.fit(X_train,Y_train)

#Predicting the model using Random Forest Classifier
Y_predict3=rf.predict(X_test)

# Classification Report
print("Classification Report for Random Forest Classifier:")
print(classification_report(Y_test, Y_predict3))
```

8. Model Selection

```
final_accuracydata=pd.DataFrame({'Models':['LR','DT','RF'], "Accuracy":
                                [accuracy_score(Y_test,Y_predict1)*100,
                                accuracy_score(Y_test,Y_predict2)*100,
                                accuracy_score(Y_test,Y_predict3)*100]})

final_accuracydata
```

9. Save the Best Model

```
#Saving the Model Random Forest Classifier
rf1.fit(X_result,Y_result)

#Loading the Pre-trained Model Random Forest Classifier
model=joblib.load('Credit_Card_Model')

#The RandomForestClassifier model is saved using the Joblib library and stored with the
filename "Credit_Card_Model".
joblib.dump(rf1,"Credit_Card_Model")
```

6. TESTING

6.1 SOFTWARE TESTING

Software testing is a crucial aspect of the software development lifecycle, ensuring that the Credit Card Fraud Detection System functions correctly and meets specified requirements. It helps identify and rectify defects, bugs, or issues. Testing can be broadly categorized into two types: manual testing, which involves executing test cases manually, and automated testing, which leverages tools to automate the testing process.

6.1.1 UNIT TESTING

Unit Testing involves testing individual components or modules of the Credit Card Fraud Detection System in isolation to ensure each part functions correctly. The following steps are followed in unit testing

6.1.1.1 MODEL PREDICTION

The machine learning model used for predicting fraudulent transactions is tested with various inputs to verify its accuracy

1. Correctly classified transaction data from the training dataset.
2. Transaction data not present in the training dataset to check generalization.
3. Edge cases, such as low-quality data, to test the model's robustness.

6.1.1.2 FRONTEND COMPONENTS

Each user interface element, such as buttons, forms, and input fields, is tested to ensure they function as expected

1. The 'Predict' button triggers the prediction process correctly.
2. Input fields for transaction values validate user entries correctly.
3. The homepage and other static pages load properly without errors.

6.1.1.3 BACKEND FUNCTIONS

The backend logic, including user authentication, data processing, and model interaction, is tested

1. Validation of user login and registration processes.
2. Handling of input data and storage on the server.
3. Interaction between the backend and the machine learning model for prediction.

6.1.2 INTEGRATION TESTING

Integration Testing focuses on ensuring that different modules of the system work correctly when combined. The following steps are followed in integration testing

6.1.2.1 MODEL INTEGRATION WITH BACKEND

The integration between the machine learning model and the backend server is tested to ensure that uploaded transaction data is correctly processed and predictions are returned without issues

1. Checking the flow from data input to prediction display.
2. Ensuring that the backend correctly handles model errors or timeouts.

6.1.2.2 FRONTEND AND BACKEND INTERACTION

The communication between the frontend user interface and the backend server is tested

1. Verifying that user inputs on the frontend trigger the correct backend processes.
2. Ensuring that backend responses are properly rendered on the frontend.

6.1.3 SYSTEM TESTING

System Testing involves testing the entire system as a whole to verify that it meets the specified requirements. This phase ensures that all integrated components work together seamlessly and perform the intended functions in real-world scenarios. The testing process

includes both functional and non-functional aspects, such as usability, performance, and security. The following steps are followed in system testing

6.1.3.1 END TO END TESTING

The complete workflow is tested, starting from user registration and login to data input, prediction, and display of results. This ensures that the system performs all intended functions correctly when used by an end-user.

6.1.3.2 PERFORMANCE TESTING

The system's performance is tested under various conditions to assess its response time, load handling, and scalability

1. Testing the prediction response time to ensure it is within acceptable limits.
2. Stress testing the system with multiple concurrent users to verify that it can handle high traffic without crashing or slowing down.

6.1.4 BLACKBOX TESTING

Black Box Testing evaluates the system's functionality without examining the internal code structure. The following steps are followed in black box testing

1. Testing the user interface for usability and user experience.
2. Validating the outputs for given inputs based on the specifications without looking into the internal implementation.

6.1.5 WHITEBOX TESTING

White Box Testing focuses on the internal workings of the code to ensure that all logic and algorithms work as intended. The following steps are followed in white box testing

1. Verifying the correctness of algorithms used for data processing and prediction.
2. Testing individual functions and methods within the code to ensure all paths are executed correctly.

6.2 TEST CASES

| S. No | Test Case Description | Expected Result | Test Result |
|-------|---|---|-------------|
| 1 | Verify successful user registration and login with incorrect credentials | 1. User account is created successfully, redirecting to the login page. 2. Attempting to log in with incorrect credentials displays an error message for invalid username or password. | Success |
| 2 | Verify prediction with valid transaction data and test form submission with missing required fields | The system returns a prediction for valid inputs and shows an error message if required fields are missing. | Success |
| 3 | Test the interaction between frontend and backend API | The system sends the data to the Flask API, and the API returns a prediction response without errors, displayed appropriately on the frontend. | Success |
| 4 | Test interaction between the Flask API and the model | The Flask API correctly receive transaction data, pass it to the machine learning model, and return a valid prediction response without errors. | Success |
| 5 | Test API response time for valid transaction data | The Flask API should respond within acceptable time limits | Success |

7. OUTPUTS

7.1 WEB INTERFACE

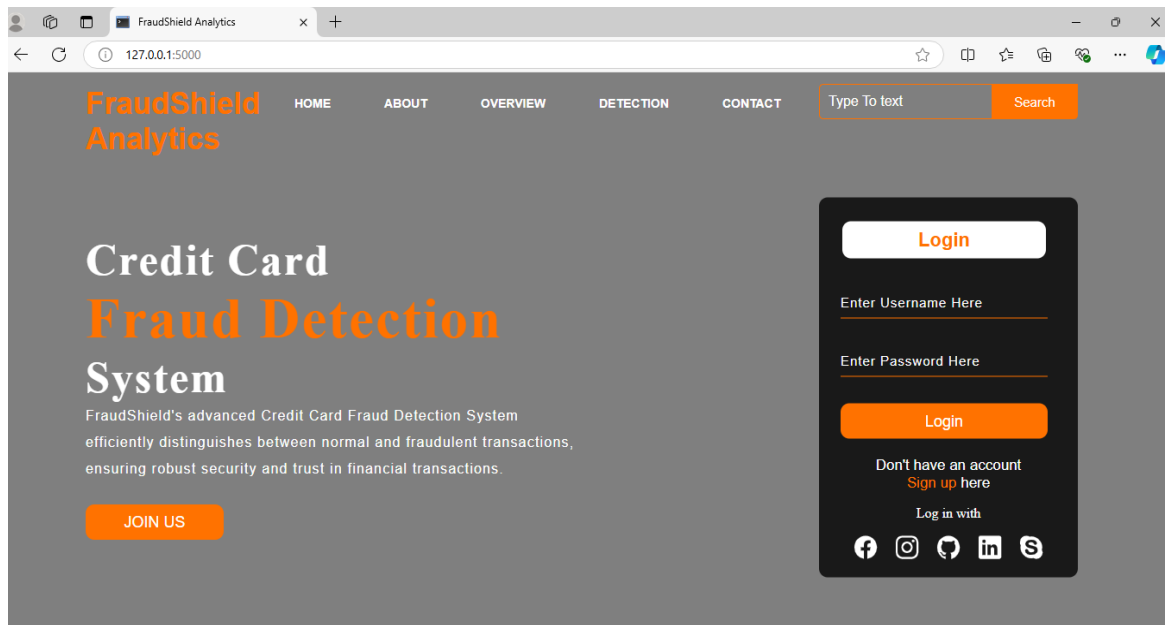


Fig 7.1.1 Home Page

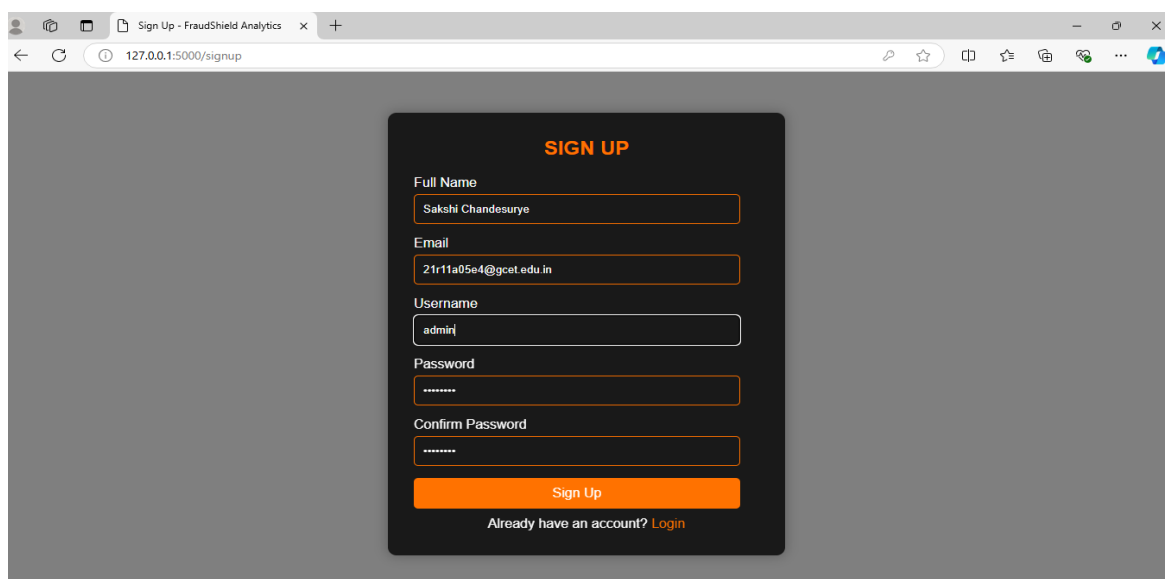


Fig 7.1.2 Signup Page in progress

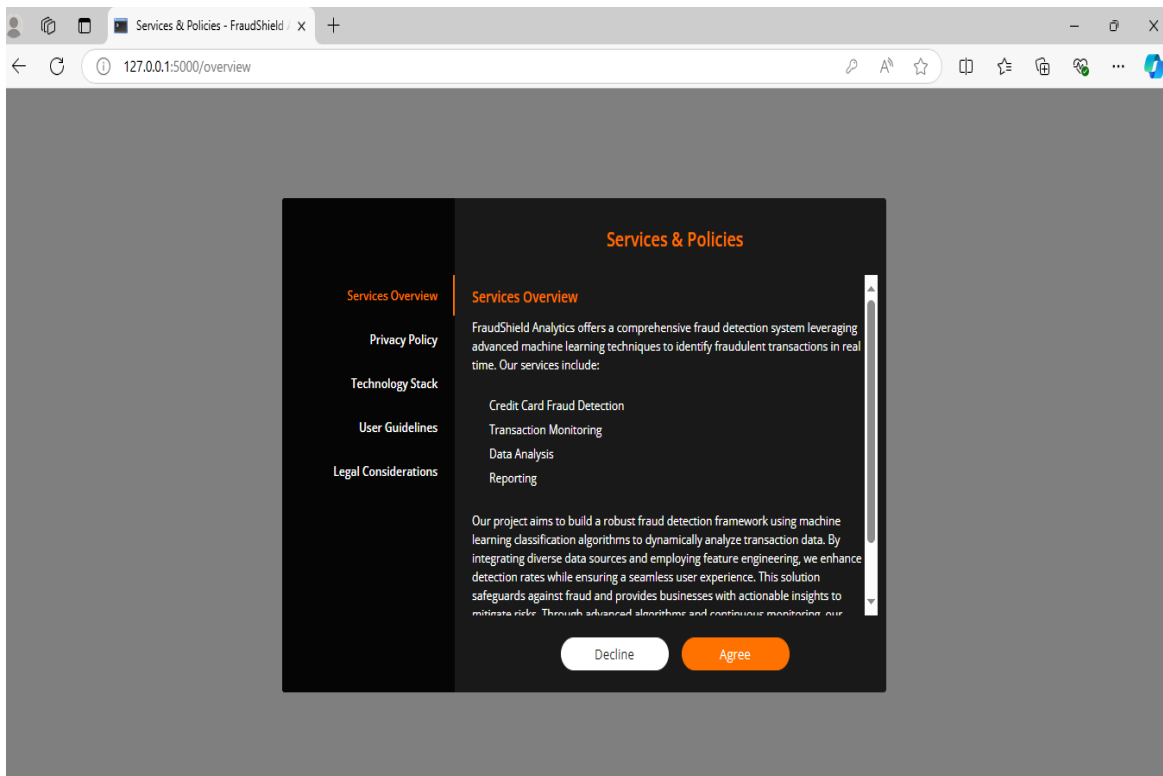


Fig 7.1.3 Overview Page

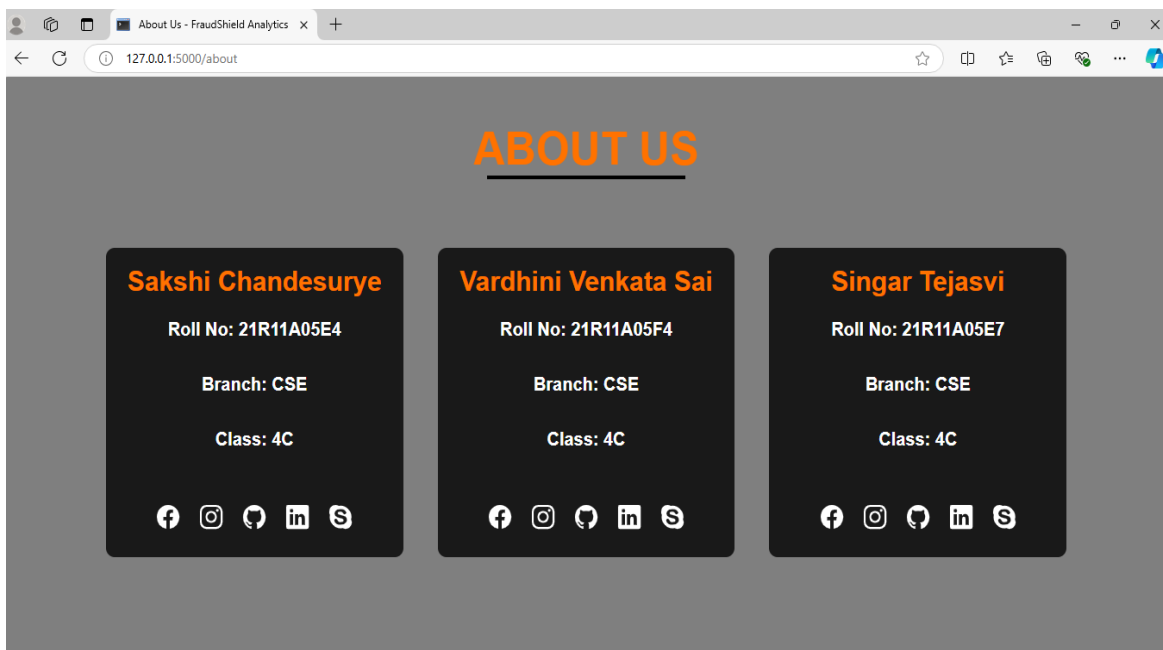


Fig 7.1.4 About Page

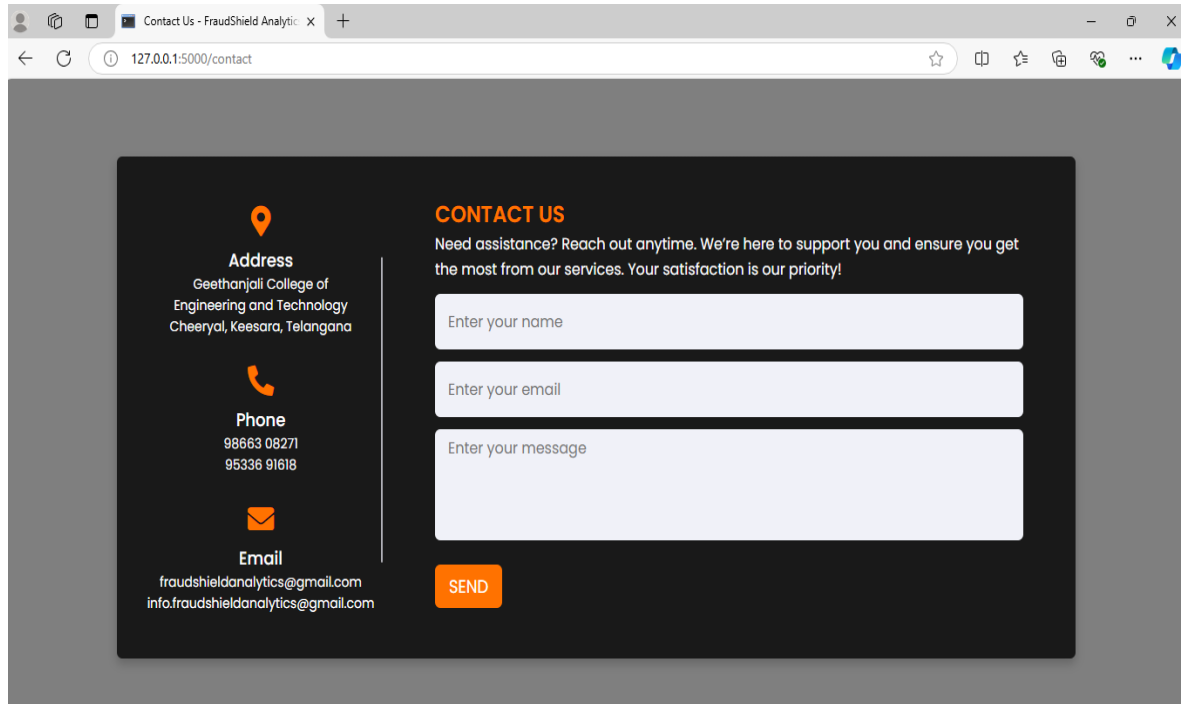


Fig 7.1.5 Contact Page

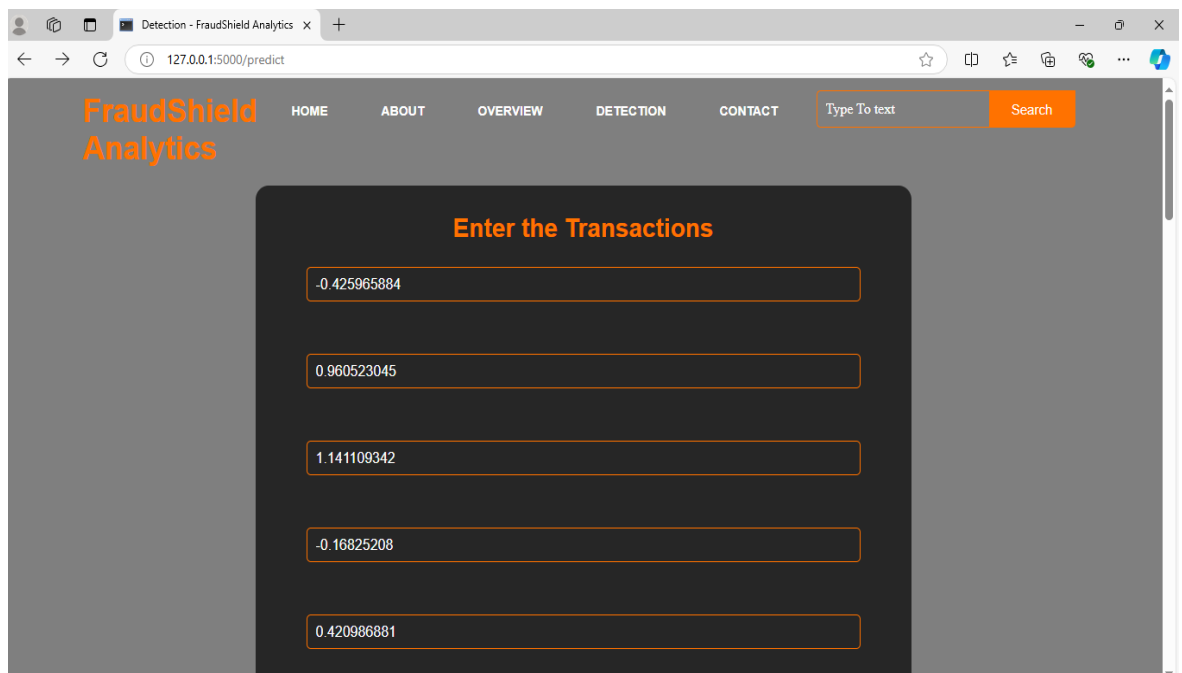


Fig 7.1.6 Detection Page in Progress

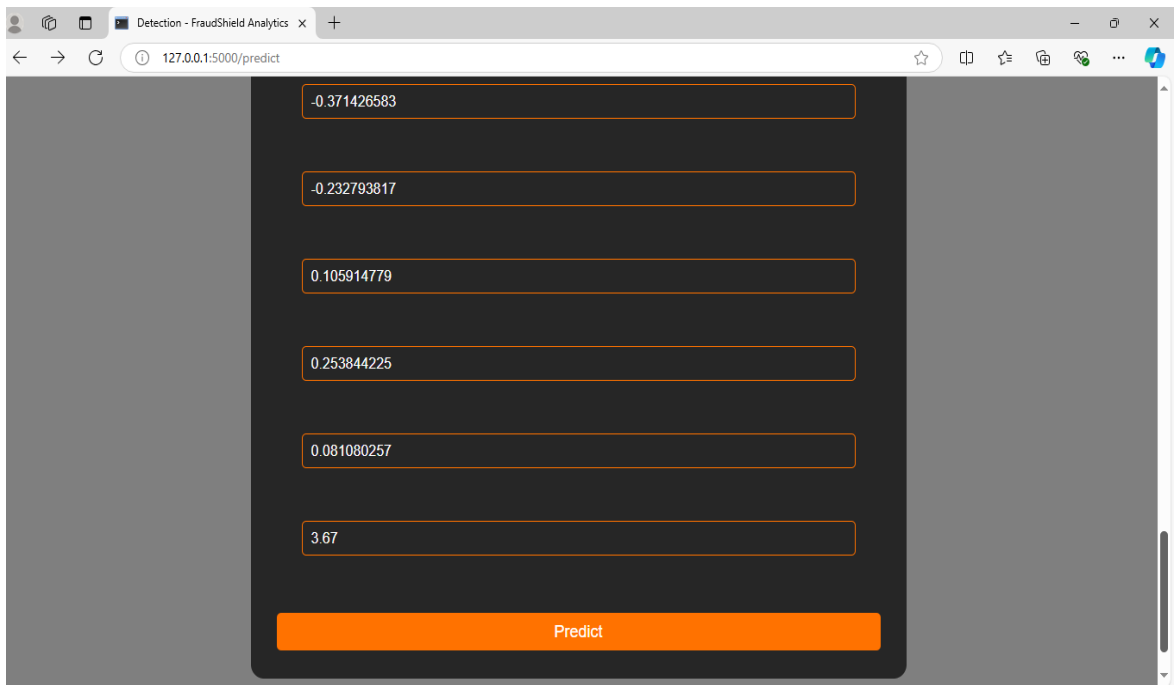


Fig 7.1.7 Continuation of the Detection Page in Progress

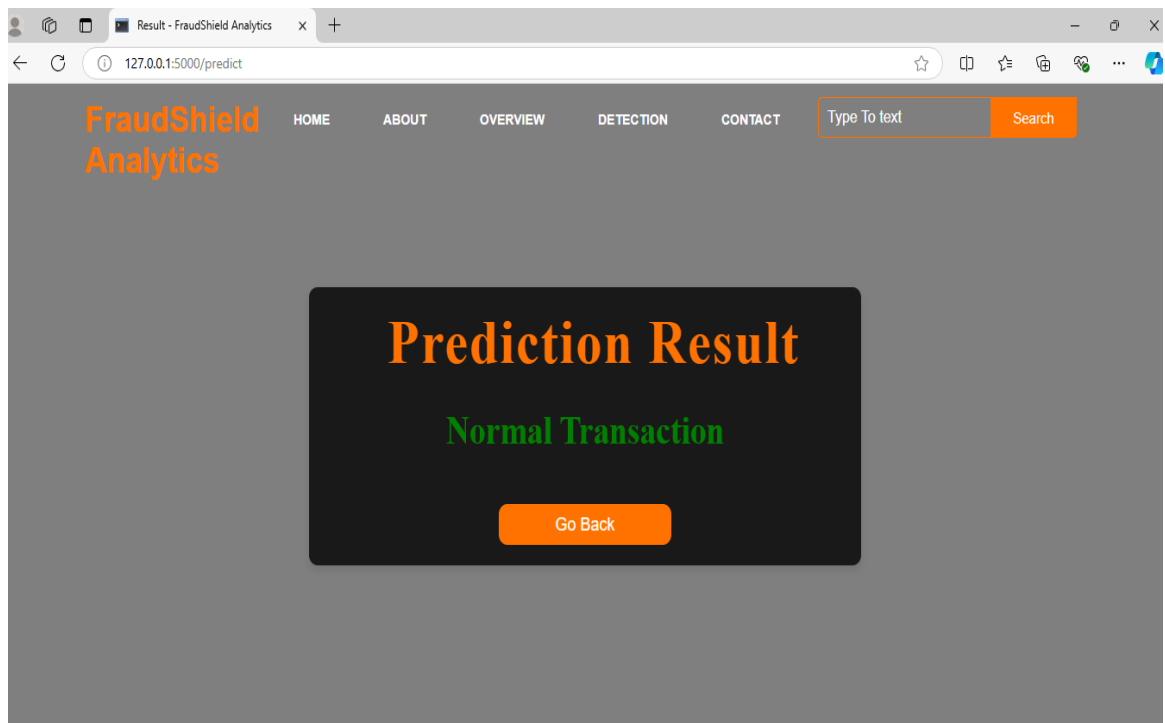
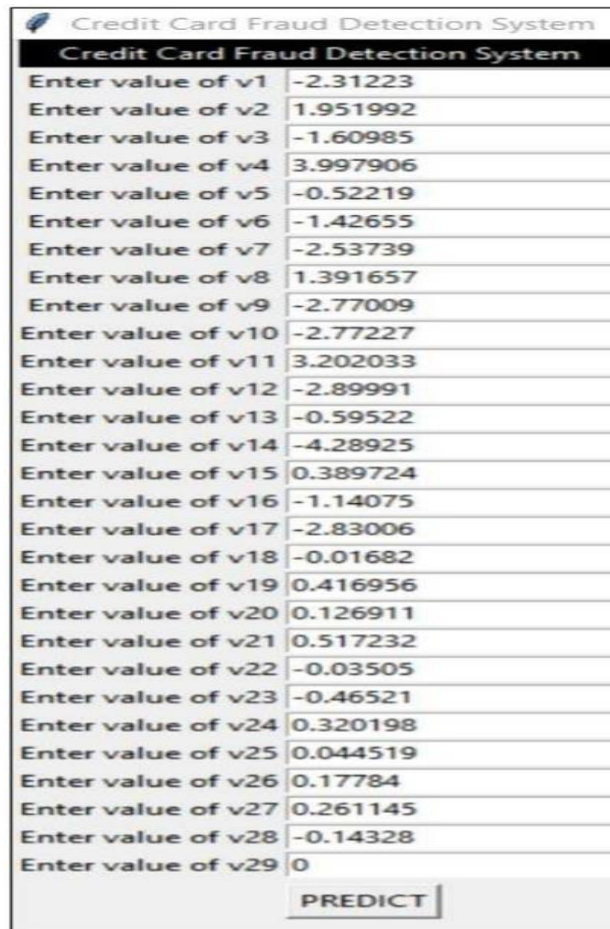


Fig 7.1.8 Result Page

7.2 GUI INTERFACE



Credit Card Fraud Detection System

Credit Card Fraud Detection System

| | |
|--------------------|----------|
| Enter value of v1 | -2.31223 |
| Enter value of v2 | 1.951992 |
| Enter value of v3 | -1.60985 |
| Enter value of v4 | 3.997906 |
| Enter value of v5 | -0.52219 |
| Enter value of v6 | -1.42655 |
| Enter value of v7 | -2.53739 |
| Enter value of v8 | 1.391657 |
| Enter value of v9 | -2.77009 |
| Enter value of v10 | -2.77227 |
| Enter value of v11 | 3.202033 |
| Enter value of v12 | -2.89991 |
| Enter value of v13 | -0.59522 |
| Enter value of v14 | -4.28925 |
| Enter value of v15 | 0.389724 |
| Enter value of v16 | -1.14075 |
| Enter value of v17 | -2.83006 |
| Enter value of v18 | -0.01682 |
| Enter value of v19 | 0.416956 |
| Enter value of v20 | 0.126911 |
| Enter value of v21 | 0.517232 |
| Enter value of v22 | -0.03505 |
| Enter value of v23 | -0.46521 |
| Enter value of v24 | 0.320198 |
| Enter value of v25 | 0.044519 |
| Enter value of v26 | 0.17784 |
| Enter value of v27 | 0.261145 |
| Enter value of v28 | -0.14328 |
| Enter value of v29 | 0 |

PREDICT

Fig 7.2.1 GUI Application of Credit Card Fraud Detection System

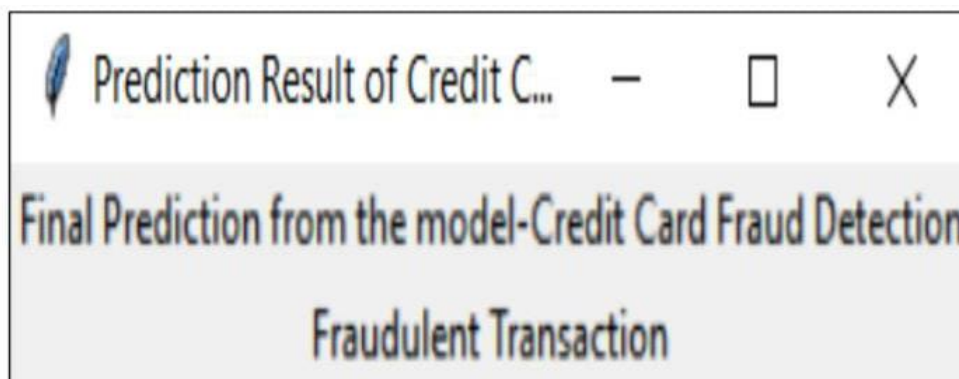


Fig 7.2.2 Final Prediction of GUI Application

8. CONCLUSION

8.1 CONCLUSION

In conclusion, this project successfully developed a sophisticated credit card fraud detection system utilizing advanced machine learning techniques to address the rising challenge of fraud in digital transactions. By focusing on a highly imbalanced dataset where only a small fraction of transactions were classified as fraudulent, the project implemented a comprehensive approach that included data preprocessing, exploratory data analysis (EDA), and innovative feature engineering. Among the various models evaluated, the Random Forest Classifier emerged as the most effective for predicting credit card fraud, achieving an impressive accuracy of 1.00 across all metrics. This highlights its ability to reliably distinguish between legitimate and fraudulent transactions, significantly enhancing the system's overall performance.

The application of machine learning algorithms significantly enhanced the system's ability to accurately identify fraudulent activities while minimizing false positives. Throughout the development process, the project evaluated various algorithms, ultimately revealing the effectiveness of tailored models designed to navigate the complexities of fraud detection. The user-friendly interfaces, consisting of both a Python-based Tkinter GUI and a web application built with Flask, ensure that the system is accessible for diverse user needs, promoting widespread adoption.

Furthermore, this project underscores the importance of continuous improvement and adaptation in fraud detection technologies, as the landscape of digital transactions evolves. By integrating real-time monitoring capabilities and leveraging continuous learning algorithms, the proposed system is well-equipped to respond to emerging threats, enhancing the overall security of financial operations. The insights gained from this project contribute to a broader understanding of fraud prevention strategies, empowering financial institutions to better protect their assets and customers in an increasingly digital economy.

8.2 FURTHER ENHANCEMENTS

This project lays a strong foundation for a credit card fraud detection system, but several enhancements can be implemented to improve its accuracy, usability, and overall impact

1. Developing a real-time monitoring feature that continuously analyzes transaction data could significantly enhance fraud detection capabilities. By implementing a system that flags suspicious activities as they occur, financial institutions can respond more quickly to potential threats, reducing the risk of financial loss. This could involve integrating machine learning models that adapt to incoming data in real-time, enabling dynamic learning from evolving fraud patterns.
2. Introducing adaptive learning algorithms that evolve with the changing landscape of fraud techniques can enhance the system's resilience. By continuously retraining the models on new data, the system can remain effective against emerging fraud patterns, ensuring long-term reliability.
3. Expanding the system's accessibility by developing mobile applications for both Android and iOS platforms would allow users to interact with the fraud detection system on-the-go. This would enable users to receive real-time alerts and insights, further enhancing the security of their transactions.

By focusing on these enhancements, the credit card fraud detection system can evolve into a more robust, user-friendly, and adaptive solution, ultimately contributing to the ongoing fight against financial fraud in an increasingly digital economy.

9. BIBILOGRAPHY

9.1 BOOKS REFERENCES

1. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
2. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.
3. Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media.
4. Fernández, A., Galar, M., García, S., & Herrera, F. (2018). Learning from Imbalanced Data Sets. Springer.

9.2 WEBSITE REFERNCES

1. Scikit-learn: Machine Learning in Python. <https://scikit-learn.org>
2. Flask Documentation. <https://flask.palletsprojects.com>
3. Pandas Documentation. <https://pandas.pydata.org>
4. Matplotlib Documentation. <https://matplotlib.org>

9.3 TECHNICAL PUBLICATION REFERENCES

1. Andrea Dal Pozzolo, Olivier Caelen, Yann-Aël Le Borgne, Serge Waterschoot, Gianluca Bontempi. “Learned lessons in credit card fraud detection from a practitioner perspective, Expert Systems with Applications.”
2. Ogwueleka, Francisca Nonyelum. “Data mining application in credit card fraud detection system.”
3. Dahee Choi and Kyungho Lee. “Machine learning based approach to financial fraud detection process in mobile payment system.”
4. Lawrence Borah, Saleena B, Prakash B. “Credit card fraud detection using data mining techniques.”
5. Amanze B.C. and Onukwugha C.G. “Data mining application in credit card fraud detection system.

10. APPENDICES

10.1. APPENDIX A- SOFTWARE USED

1. **Python:** The main programming language used to develop the machine learning model, process data, and implement backend functionality.
2. **Scikit-learn, Pandas, Numpy:** Libraries used for implementing machine learning algorithms, data manipulation, and numerical processing for credit card fraud detection.
3. **Flask:** A lightweight web framework used for building the web interface, enabling users to interact with the system and receive predictions.
4. **HTML/CSS/JavaScript:** Technologies employed for front-end development, ensuring the web application is interactive and responsive.
5. **Jupyter Notebook:** Used for model experimentation, data analysis, and presenting results in a structured manner.

10.2. APPENDIX B- METHODOLOGIES USED

1. **Random Forest Classifier:** The primary machine learning algorithm used for predicting credit card fraud, known for its high accuracy and ability to handle imbalanced datasets.
2. **Data Preprocessing:** Includes handling missing values, encoding categorical features, and scaling numerical values for better model performance.
3. **Imbalanced Dataset Techniques:** Strategies like oversampling the minority class and using metrics like AUC-ROC to evaluate model performance on the imbalanced dataset.
4. **Exploratory Data Analysis (EDA):** Visualizing transaction data, identifying patterns of fraud, and understanding feature correlations to improve model interpretability.

5. **Web Development:** Flask was used to integrate the machine learning model with a user-friendly web interface. HTML, CSS, and JavaScript were used for front-end design.

10.3. APPENDIX C- SYSTEM ARCHITECHTURE

1. **Credit Card Transactions:** The dataset consists of transaction records with features like transaction amount, location, and time. Each transaction is labeled as fraudulent or non-fraudulent.
2. **Data Processing:** Prepares the submitted transaction data for model prediction, ensuring it is in the correct format.
3. **Training and Test Split:** The dataset was divided into training and test sets to evaluate model performance on unseen data. A stratified split was used to maintain class distribution.
4. **Handling Class Imbalance:** Techniques like SMOTE (Synthetic Minority Oversampling Technique) were applied to balance the class distribution and improve model performance.
5. **Client-Server Model:** The web-based application follows a client-server architecture, where the client (browser) interacts with the Flask server to submit transaction data and receive fraud predictions.
6. **User Interface:** Handles user inputs, such as transaction data, and displays fraud detection results. The interface is built using HTML, CSS, and JavaScript.

10.7. APPENDIX D- TESTING

1. **Test Cases:** Several test cases were created to validate system functionality, including unit, integration, black-box, and white-box testing. These ensured seamless module interaction, prediction accuracy, and input validation across various transaction scenarios.
2. **Result Analysis:** Predictions were compared against actual transaction outcomes to evaluate the model's accuracy, precision, recall, and F1-score, leading to further model optimization based on the analysis.

11. PLAGIARISM REPORT

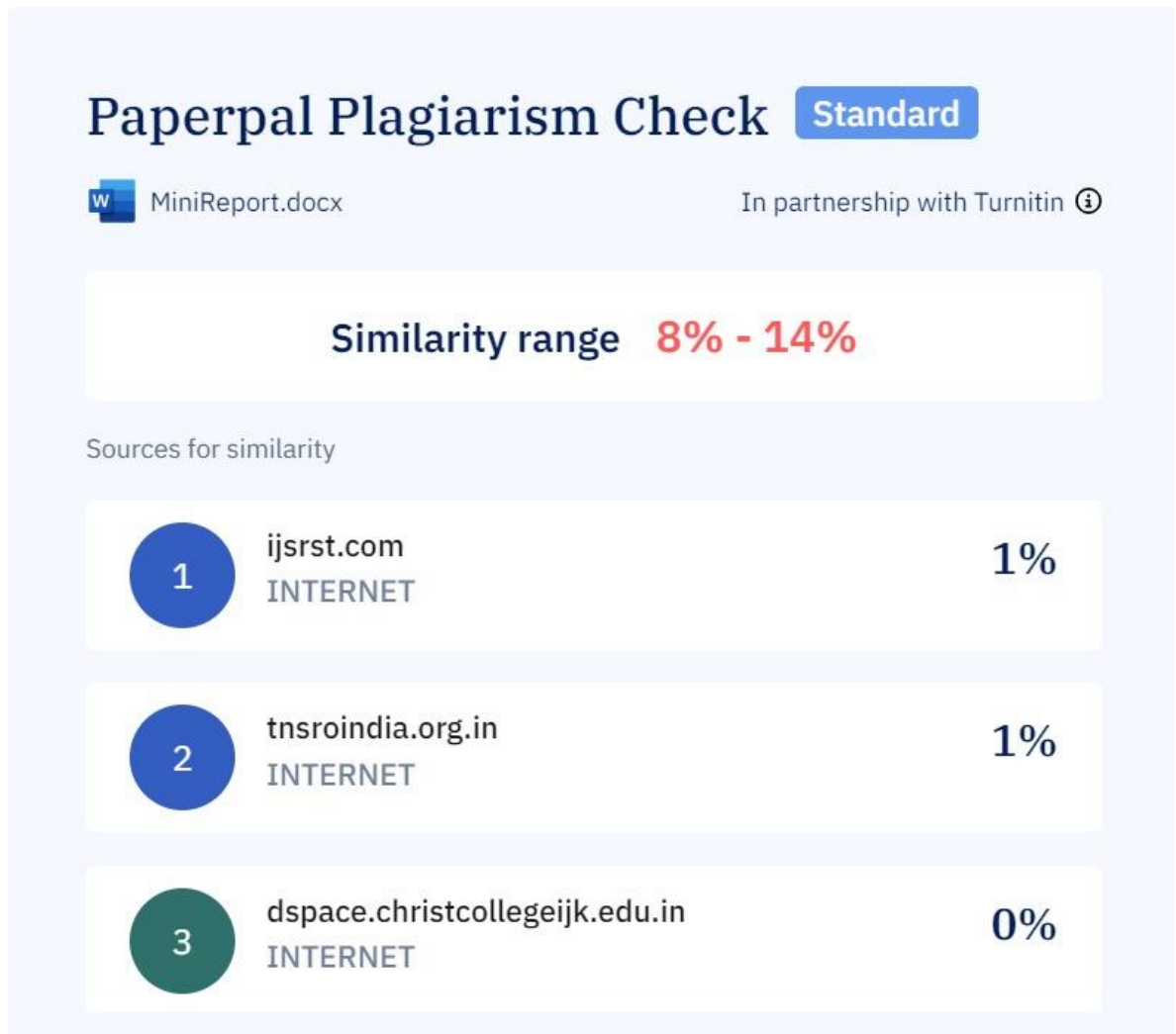


Fig 11.1 Plagiarism Report