INCOME TAX FRAUD DETECTION USING AIML

A PROJECT REPORT

Submitted by,

D Soya	**-**	**20211CSE0135**
P Sruthi	**-**	**20211CSE0013**
P S Afreen	**-**	**20211CSE0101**
B Pranavi	**-**	**20211CSE0478**

Under the guidance of,

Dr. Jayanthi Kamalasekaran

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

At

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

PRESIDENCY UNIVERSITY
BENGALURU
JANUARY 2025

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

CERTIFICATE

This is to certify that the Project report **"INCOME TAX FRAUD DETECTION USING AIML"** being submitted by "Soya, Sruthi, Afreen, Pranavi" bearing roll number(s) "202211CSE0135,20211CSE0013, 20211CSE0101,20211CSE0478" in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

Dr. Jayanthi K Associate Professor School of CSE Presidency University	**Dr. Asif Mohammed H.B** Professor & HOD School of CSE&IS Presidency University

Dr. L. SHAKKEERA Associate Dean School of CSE Presidency University	**Dr. MYDHILI NAIR** Associate Dean School of CSE Presidency University	**Dr. SAMEERUDDIN KHAN** Pro-VC School of Engineering Dean -School of CSE&IS Presidency University

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **Income Tax Fraud Detection Using AIML** in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of our own investigations carried under the guidance of **Dr. Jayanthi Kamalasekaran, School of Computer Science Engineering & Information Science, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Student Name	**Roll Number**	**Signature**
D Soya Sree	20211CSE0135	
P Sruthi	20211CSE0013	
P S Afreen	20211CSE0101	
B Pranavi	20211CSE0478	

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

ABSTRACT

It is, therefore, very important and vital for a financial institute to fraud-detective financial transactions that need seriously advanced strategies to retain the security and integrity of a financial system. This work focuses on a development framework that is highly efficient in the fraud detection task by introducing various aspects: data warehousing, data visualization, information retrieval, and analytics on stream processing. An elaborate data warehousing system is deployed for handling huge volumes of transaction data and storing them in easily accessible forms such that fast and efficient access shall be provided towards its retrieval, thus enabling an in-depth examination. The framework is designed based on advanced data structures and information retrieval methods to improve precision with better fraud detection. Machine learning models are designed and optimized for the Decision Tree, Logistic Regression, Random Forest, and Naïve Bayes algorithms, all of which flexibly determine dangerous patterns and anomalies that indicate fraud against the complexity and variation in financial transaction data. Standard metrics classification includes Precision, Recall, F1-score, and AUC-ROC, all of which were computed to assess the accuracy and reliability of the performance of the system. This study explores, in more detail, practical aspects of the deployment of such systems in real financial scenarios, with regards to scalability and real-time processing. The proposed project will combine several state-of-the-art technologies and methodologies in an effort to significantly enhance the capability of financial institutions in fraud detection and prevention for improving security and building trust in financial systems. This holistic approach will offer an innovative solution to one of the most urgent challenges in the financial industry today.

ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.
We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC, School of Engineering and Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.
We express our heartfelt gratitude to our beloved Associate Deans **Dr. Shakkeera L and Dr. Mydhili Nair,** School of Computer Science Engineering & Information Science, Presidency University, and **Dr.Asif Mohammed** Head of the Department, School of Computer Science Engineering & Information Science, Presidency University, for rendering timely help in completing this project successfully.
We are greatly indebted to our guide **Dr. Jayanthi Kamalasekaran** and Reviewer **Dr. Aarif Ahamed S**, School of Computer Science Engineering & Information Science, Presidency University for her inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.
We would like to convey our gratitude and heartfelt thanks to the PIP2001 Capstone Project Coordinators **Dr. Sampath A K, Dr. Abdul Khadar A and Mr. Md Zia Ur Rahman,** Git hub coordinator **Mr. Muthuraj.**
We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

D SOYA
P SRUTHI
P S AFREEN
B PRANAVI

CHAPTER-1
INTRODUCTION

Financial fraud is a pervasive issue in financial industry globally, resulting to financial losses and weakening the trust and stability of financial institutions. As the volume and level of details in money transfers keeps on increasing, so does the sophistication of fraudulent activities. Fraud detection methods, often based on manually and rule-based systems, are increasingly inadequate in addressing the dynamic and evolving nature of fraud. This project aims to develop an advanced fraud detection framework using machine learning algorithms to classify financial transactions accurately into "Fraudulent" and "Non-fraudulent" categories. By leveraging data warehousing, data visualization, information retrieval, and stream processing analytics, the project seeks to enhance the precision and efficiency of fraud detection systems. The proposed framework incorporates a comprehensive data warehousing solution to handle and store vast amounts of transactional data, facilitating efficient data retrieval and thorough analysis. Advanced data structures and information retrieval methods are employed to improve the accuracy and speed of fraud detection algorithms. Machine learning mechanisms that includes Decision Tree, Logistic Regression, Random Forest, and Naïve Bayes, are developed to identify developments and defects that reveal acts of fraud.

1.1 Background

This section provides an overview of income tax systems, the significance of accurate tax collection, and the economic impact of tax fraud. It also highlights the growing prevalence of fraud schemes and the challenges faced by tax authorities in detecting fraudulent activities. The role of AI/ML in modern data analytics and fraud detection can be introduced here, emphasizing their potential to transform traditional methods.

1.2 Project Overview

This section summarizes the project's intent: leveraging AI/ML to enhance the detection and prevention of income tax fraud. It briefly outlines the current gaps in manual or traditional systems and the potential of AI/ML techniques to address these gaps efficiently and effectively.

1.2.1 Current Industry Challenges

Discusses challenges faced by tax authorities and financial institutions, including:

- **Data Volume:** Handling large datasets generated by millions of taxpayers.
- **Complex Fraud Techniques:** Use of sophisticated methods by fraudsters to evade detection.
- **Resource Constraints:** Limited manpower and time for manual audits.
- **False Positives:** Challenges in distinguishing legitimate anomalies from fraudulent activity.

1.2.2 Problem Statement

Despite advancements, traditional fraud detection techniques often fail to adapt to evolving fraud patterns, leading to revenue losses and inefficient resource utilization. The project aims to address the problem of accurately and efficiently identifying income tax fraud by utilizing

AI/ML models that can analyse large datasets, identify anomalies, and predict fraud patterns in real time.

1.3 Project Objectives

- Develop a machine learning-based framework to detect income tax fraud effectively.
- Improve detection accuracy while minimizing false positives.
- Automate the analysis of large tax datasets to save time and resources.
- Adaptively learn new fraud techniques to keep detection systems up-to-date.
- Provide actionable insights to tax authorities to improve policy enforcement.

1.4 Project Scope

- **In Scope:**
    - Development and implementation of AI/ML models for fraud detection.
    - Use of historical tax data for training and validation.
- **Out of Scope:**
    - Legal or policy enforcement actions post-detection.
    - Addressing non-digital fraud methods (e.g., cash-based fraud).

1.5 Document Organization

- **Section 1:** Introduction, including background, project overview, objectives, and scope.
- **Section 2:** Literature review on income tax fraud and AI/ML applications in fraud detection.
- **Section 3:** Methodology, covering data collection, model selection, and system architecture.
- **Section 4:** Results and discussion, presenting key findings from the model's performance.
- **Section 5:** Conclusion and suggestion for future research.
-

CHAPTER-2
LITERATURE SURVEY

2.1 Overview of AI/ML Technology

AI and ML have revolutionized data analysis and decision-making processes across various domains, including fraud detection. Key aspects to cover:

- **Artificial Intelligence:** A division of computer science aimed on creating computers that resemble human intelligence, such as reasoning, problem solving, and learning.
- Machine Learning: It is a branch of AI that utilizes algorithms based on data to create predictions and judgments. Discuss types:

- **Supervised Learning:** Used for both regression and classification tasks., such as identifying fraudulent patterns in tax filings.

    - **Unsupervised Learning:** Detecting anomalies in tax data without labeled examples.
    - **Reinforcement Learning:** Improving fraud detection systems through iterative feedback.
- **Key Techniques in Fraud Detection:**
    - Anomaly detection for spotting irregular patterns.
    - Predictive modeling to identify taxpayers with high fraud risk.
    - Feature engineering to capture meaningful attributes (e.g., income vs. expenses).

2.2 Related Research

A review of academic and industry contributions to fraud detection using AI/ML:

- **Key Studies:**
    - Research on classification algorithms like Logistic Regression, Support Vector Machines, and Neural Networks in fraud detection.
    - Uses of deep learning techniques, like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), to analyse complex datasets.
- **Case Studies:**
    - Implementation of fraud detection systems in banking (e.g., credit card fraud) and its parallels to income tax fraud.
    - Studies demonstrating the success of hybrid approaches combining multiple algorithms.
- **Challenges in Research:**
    - Difficulty in accessing high-quality, labelled datasets for training models.
    - Balancing precision and recall to minimize false positives.

2.3 Technology Stack Analysis

An in-depth look at tools and frameworks that enable fraud detection:

- **Programming Languages:**
    - Python: Widely used for its libraries like NumPy, Pandas, and Scikit-learn.
    - R: Preferred for statistical modelling and analysis.
- **ML Frameworks:**
    - TensorFlow and PyTorch for deep learning model development.
    - Scikit-learn for classic ML techniques.
- **Data Handling:**
    - Apache Spark and Hadoop for distributed processing of large datasets.
    - SQL and NoSQL databases for structured and unstructured tax data storage.
- **Visualization Tools:**
    - Tableau and Matplotlib for presenting model insights to stakeholders.
- **Cloud Services:**
    - AWS SageMaker, Azure AI, and Google AI for scalable, cloud-based ML solutions.

2.4 Knowledge Base Systems (KBS)

KBS enhance AI/ML models by incorporating domain-specific knowledge:

- **Rule-based Systems:** Encoding tax laws and regulations into decision-making logic to flag potential violations.
- **Hybrid Systems:** Combining rule-based logic with ML to improve detection accuracy.
- **Use Cases in Tax Fraud:**
    - Systems that automatically compare reported income against third-party financial records.
    - Incorporating historical fraud cases into the knowledge base to improve predictions.

2.5 Integration Approaches

Strategies for incorporating AI/ML models into tax authority systems:

- **Data Integration:**
    - Consolidating data from multiple sources like tax filings, financial institutions, and audits.
- **Workflow Design:**
    - Preprocessing and cleaning raw tax data for ML consumption.
    - Automating data pipelines for real-time fraud detection.
- **System Compatibility:**
    - Designing APIs to interface ML models with legacy systems.

- **Security and Privacy:**
    - Ensuring encryption of sensitive taxpayer data.
    - Compliance with data protection regulations like GDPR.
- **Deployment Models:**
    - Batch processing for retrospective analysis.

2.6 Gaps in Current Research

Identifying shortcomings in existing approaches to highlight opportunities for innovation:

- **Dynamic Fraud Patterns:** Limited ability of current models to adapt to evolving fraud techniques.
- **Bias in Data:** Algorithms trained on biased data may lead to unfair targeting of specific demographics.
- **Scalability Issues:** Challenges in deploying AI/ML systems on a national scale with millions of taxpayers.
- **Explainability:** Difficulty in interpreting black-box ML models, reducing trust among tax authorities.

CHAPTER-3
RESEARCH GAPS OF EXISTING METHODS

3.1 Analysis of Current AI/ML Systems

Despite the rapid advancement of Artificial intelligence and machine learning (ML) in detecting income tax fraud, several limitations persist:

- Data Quality and Availability:
  - Current AI systems rely heavily on structured datasets. However, income tax fraud often involves unstructured or semi-structured data, such as handwritten receipts, vague transaction descriptions, or incomplete financial records.
  - Limited access to comprehensive datasets due to privacy concerns hinders robust model training.
- Model Interpretability:
  - Many AI systems, like deep learning models, are treated as "black boxes." Tax agencies and auditors often struggle to trust their predictions due to a lack of explainability.
- 3Adaptability:
  - Models often fail to adapt to new fraudulent schemes. Fraudsters continuously evolve their tactics, making static models ineffective.

3.2 Customer Support Integration Challenges

- Scalability:
  - Integrating fraud detection models into real-time customer support systems (e.g., tax helplines) is challenging due to the high volume of queries. Models need to analyze customer data while maintaining operational speed.
- False Positives:
  - Misclassifications can lead to unnecessary scrutiny, frustrating legitimate taxpayers.
- Personalization Issues:
  - Current systems often lack personalization, failing to address customer-specific tax compliance issues during interactions.
- Lack of Multichannel Support:
  - AI systems often struggle with seamless integration across multiple channels (e.g., web portals, chatbots, and email).

3.3 Technical Implementation Gaps

- Insufficient Feature Engineering:

- Models fail to account for subtle behavioral patterns in financial transactions, such as seasonal tax filings or industry-specific anomalies.

- Limited Use of Advanced Techniques:

- Emerging methods like graph-based fraud detection and reinforcement learning remain underutilized in tax domains.

- Integration with Legacy Systems:

- Many tax agencies use outdated software. AI/ML solutions often face compatibility issues when deployed in these environments.

- Latency in Decision-Making:

- Real-time detection models require high computational power, leading to delays in decision-making for complex cases.

3.4 User Experience Gaps

- Usability for Non-Technical Users:

- Current tools are not user-friendly for tax officers with minimal AI expertise. Complex dashboards or jargon-heavy reports reduce usability.

- Transparency and Feedback:

- Taxpayers flagged for potential fraud often lack clarity on why their filings were flagged, leading to trust issues.

- Cross-Cultural Challenges:

- AI systems are often designed with assumptions that may not apply universally, ignoring linguistic and cultural nuances in tax systems.

3.5 Fraud Detection-Specific Limitations

- Limited Focus on Rare Patterns:

- Fraudulent cases are rare, leading to imbalanced datasets. Models tend to prioritize non-fraudulent patterns, missing sophisticated fraud.

- Overreliance on Historical Data:

- Models trained on outdated fraud patterns are less effective against new tactics.

- Manual Validation Bottleneck:

- High reliance on human auditors for model validation slows the process and increases operational costs.

- Collusion Challenges:

- AI struggles to identify complex collusion frauds involving multiple parties, as they require relational analysis beyond individual data points.

3.6 Proposed Solutions

- Enhanced Data Techniques:
    - Incorporate multi-modal data (e.g., text, images, and financial graphs) to improve fraud detection accuracy.
    - Leverage synthetic data generation to address data scarcity while maintaining privacy.
- Explainable AI (XAI):
    - Implement models that provide interpretable insights, enabling tax officers to understand fraud detection logic.
- Adaptive Learning Models:
    - Use reinforcement learning to allow models to adapt to evolving fraud tactics.
- Graph Analytics:

  - Apply graph-based methods to detect relational anomalies, particularly in cases of collusion or linked accounts.
- Integration Frameworks:
  - Develop APIs and middleware to seamlessly integrate AI systems with legacy tax agency infrastructures.
- User-Centric Designs:
  - Introduce intuitive interfaces for tax officers, with customizable reports and guided workflows.
- Real-Time Monitoring Systems:

- Use high-speed computing technologies, such as edge AI, to enable real-time fraud detection with minimal latency.

CHAPTER-4
PROPOSED MOTHODOLOGY

4.1 System Architecture

The proposed Framework of systems comprises three primary layers:

- Data Input Layer:

- Sources: Income tax filings, transaction data, bank statements, and digital receipts.
- Input: Structured and unstructured data, including scanned documents and user-provided information**.**

- Processing Layer:

- Core Components: Natural language processing (NLP) modules, machine learning algorithms, and a centralized knowledge base.
- Functionality: Extract, analyze, and classify data for potential fraud detection.

- Output Layer:

- Dashboards: Provides actionable insights and fraud likelihood scores for auditors.
- Alerts: Generates real-time alerts for high-risk cases.

The architecture follows a modular design for scalability, ensuring it can handle increasing data volumes and adapt to new fraud patterns.

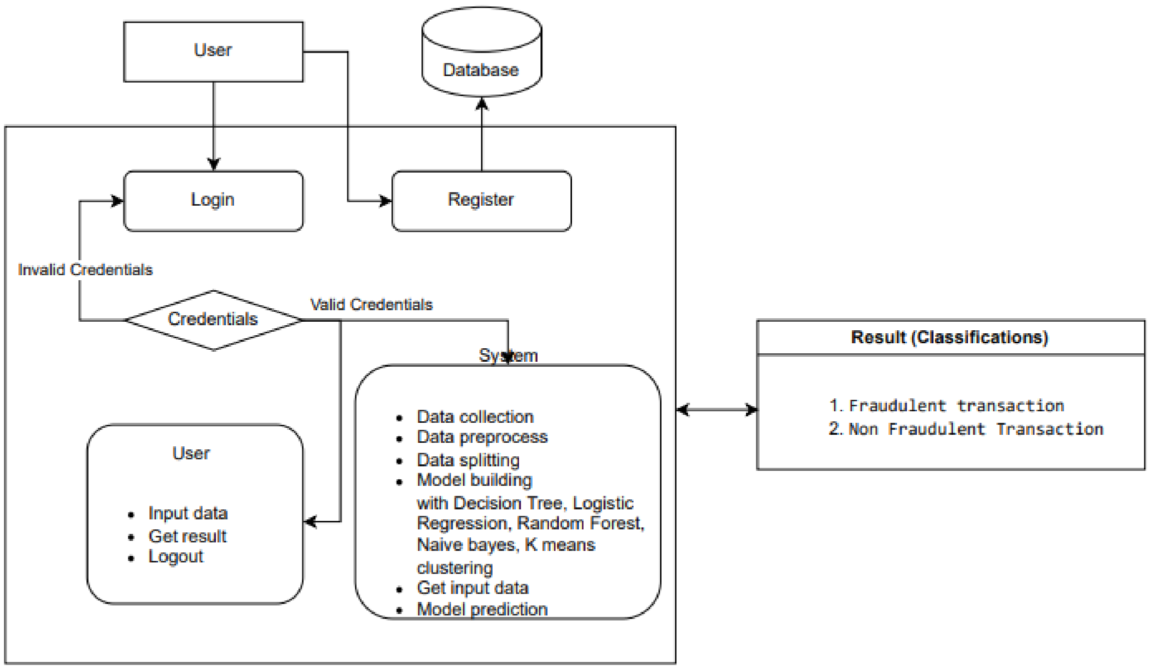


Figure 4.1: Architecture

4.2 Natural Language Processing (NLP) Implementation

NLP has a important role play in analyzing unstructured text data, such as:

- Document Parsing:

- Extract information from scanned documents, tax filings, and email communications.
- Utilize optical character recognition (OCR) combined with NLP for text extraction**.**

- Semantic Analysis:

- Identify suspicious language patterns, such as ambiguous transaction descriptions or discrepancies in justifications.

- Entity Recognition:

- Detect and link entities, such as taxpayers, businesses, and third-party organizations, to uncover relationships and anomalies.

- Contextual Analysis:

- Use sentiment analysis and contextual NLP models to identify attempts to manipulate tax claims.

4.3 Knowledge Base Architecture

The knowledge base acts as the central repository for domain-specific tax knowledge, including:

- Regulatory Rules and Policies:

- Encodes tax laws, compliance thresholds, and fraud indicators.

- Case Histories:

- Stores past fraud cases to enable pattern recognition and model refinement.

- Ontology-Based Structure:

- Represents the relationships between entities like taxpayers, income sources, and transactions.

- Continuous Updates:

- Implements a feedback loop to incorporate new regulations and fraud schemes.

4.4 Machine Learning Components

The machine learning module includes:

- Feature Engineering:

- Extracts key features such as income anomalies, unexplained deductions, or repetitive transactions.

- Supervised Learning Models:

- Utilizes labeled datasets to train fraud detection models.
- Algorithms: Random Forest, Gradient Boosting Machines, and Neural Networks.

- Unsupervised Learning Models:

- Detects outliers and novel fraud patterns using clustering techniques like DBSCAN or isolation forests.

- Model Optimization:

- Employs hyperparameter tuning and ensemble methods for improved accuracy.

4.5 Integration Framework

To ensure seamless deployment, the integration framework includes:

- API-Based Integration**:**

- Connects the fraud detection system with existing tax agency systems.

- Cloud Deployment:

- Enables scalable processing and data storage.

- Real-Time Processing**:**

- Leverages streaming platforms (e.g., Apache Kafka) for real-time fraud detection.

- Cross-Platform Compatibility:

- Ensures compatibility with web-based and mobile systems for remote accessibility.

4.6 Security Implementation

Given the sensitive nature of tax data, robust security measures are essential:

- Data Encryption:

- end-to-end encryption for data is implemented during the rest and in transit.

- Access Controls:

- Role-based access management to control sensitive data access to prevent unauthorized access.

- Fraud Prevention Measures:

- Deploys secure logging mechanisms to detect and prevent insider threats.

- Compliance with Standards:

- Adheres to data protection regulations such as GDPR and ISO 27001.

4.7 Testing Strategy

The system's efficacy will be validated through rigorous testing:

- Unit Testing:

- Ensures individual modules, such as NLP parsers and ML models, function as intended.

- Integration Testing:

- Tests the interoperability of various system components, including APIs and data pipelines.

- Performance Testing:

- system's ability to handle large data volumes and real-time processing is measured.

- User Acceptance Testing (UAT):

- Involves tax auditors to examine the system's usability and accuracy in real-world scenarios.

- Adversarial Testing:

- Simulates fraud tactics to assess the robustness of fraud detection mechanisms.

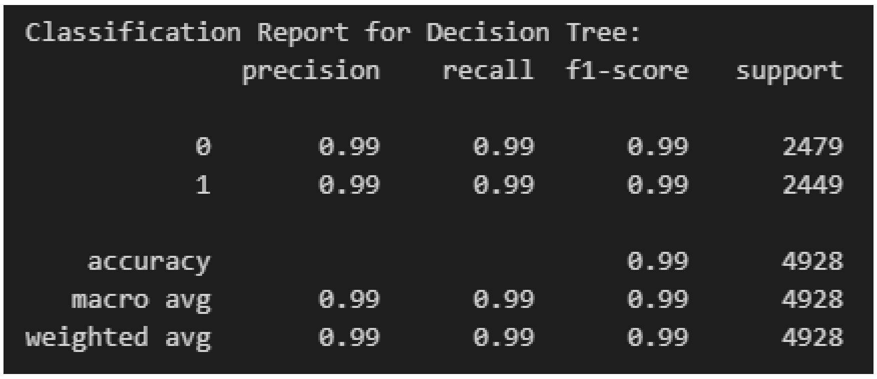
CHAPTER-5
METHODOLOGY

5.1 K-Means clustering

K-Means Clustering is an unsupervised algorithm for machine learning. It segregates a dataset into a small set of non-overlapping, discontinuous clusters. The main purpose of K-Means is to get high variance between the clusters and low variance inside the cluster.
K-Means Clustering is an unsupervised learning methodology that segments the dataset into K non-overlapping clusters. This is done by generating K centroids first, either randomly or by some heuristic. By distributing each data point to its nearest centroid, the K clusters are formed. The centroids are updated by taking a mean of every point in a cluster iteratively. This process of assignment and updating centroids continuously repeats until convergence-for example, the centroids do not change significantly -or when a maximum number of iterations is attained. The sum of squares inside a cluster decreases during the algorithm's process of making the points inside each cluster as close as possible to one another. K-Means is sensitive to the initial placement of centroids. Techniques such as K-Means++ are usually used for better initialization. It is one of the most common choices for clustering problems because it is inexpensive and scalable to very large datasets.

5.2 Decision Tree

A logistic regression is mostly a supervised learning algorithm used for classification problems or problems that can have two answers. Its name may have "relapse" in it, but, in general, it's an order calculation rather than a relapse one. Logistic regression is a form of statistical analysis which predicts the probability of a binary outcome based on one or more predictor factors. The procedure begins with the fitting of data to a linear equation, which is essentially a weighted sum of input features. The logistic function, also called the sigmoid function, then serves to convert this linear combination into a probability value between 0 and 1. Because of the logistic function, its output satisfies the requirements for binary classification, as it is bounded within the output's desired domain. Maximum likelihood estimation, which estimates the parameters that maximize the likelihood of seeing the provided data, is used to estimate the model parameters (weights). Based on a threshold, usually 0.5, the probabilities produced by the finished model can be transformed into class predictions. Regularization methods like as L1 (Lasso) and L2 (Ridge) can be used to avoid overfitting and increase the robustness of the model.



```
Classification Report for Decision Tree:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      2479
           1       0.99      0.99      0.99      2449

    accuracy                           0.99      4928
   macro avg       0.99      0.99      0.99      4928
weighted avg       0.99      0.99      0.99      4928
```

Figure 5.1: Classification report for Decision Tree

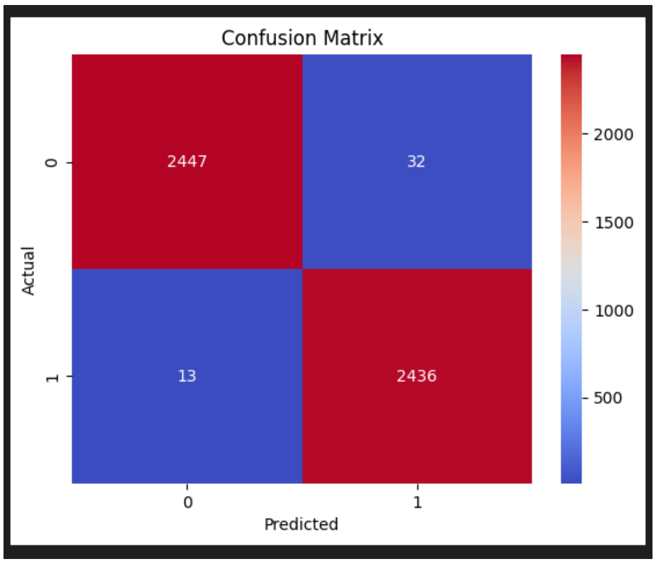
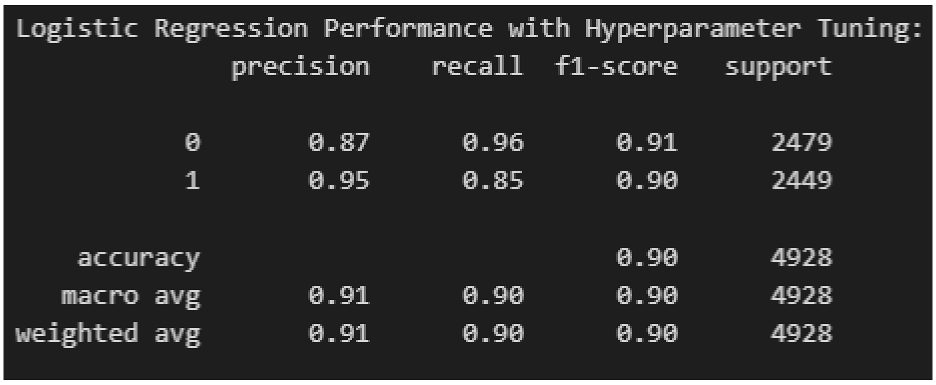


Figure 5.2: Confusion matrix for Decision Tree

5.3 Logistic Regression

A supervised learning algorithm called logistic regression is mostly used for binary classification problems. In spite of the fact that it has "relapse" in its name, it is in a general sense an order calculation as opposed to a relapse one. A method of statistical analysis for binary classification called logistic regression estimates the likelihood of a binary outcome depending on one or more predictor factors. The procedure begins by fitting the data, which is a weighted sum of the input features, to a linear equation. The logistic function, also known as the sigmoid function, then serves to convert this linear combination into a probability value between 0 and 1. The output is acceptable for binary classification because the logistic function guarantees that it is bounded. Maximum likelihood estimation, which determines the parameters that maximize the likelihood of seeing the provided data, is used to estimate the model parameters (weights). Based on a threshold, usually 0.5, the probabilities produced by the finished model can be transformed into class predictions. Regularization methods like as L1 (Lasso) and L2 (Ridge) can be applied to prevent overfitting and improve model robustness.



```
Logistic Regression Performance with Hyperparameter Tuning:
              precision    recall  f1-score   support

           0       0.87      0.96      0.91      2479
           1       0.95      0.85      0.90      2449

    accuracy                           0.90      4928
   macro avg       0.91      0.90      0.90      4928
weighted avg       0.91      0.90      0.90      4928
```

Figure 5.3: Classification report for Logistic Regression

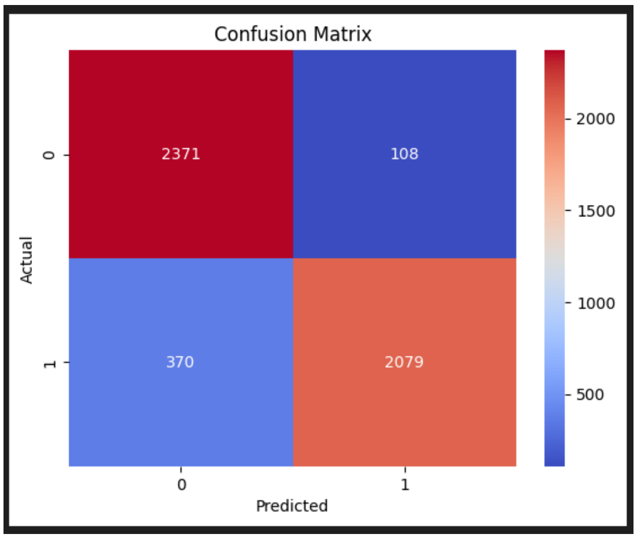
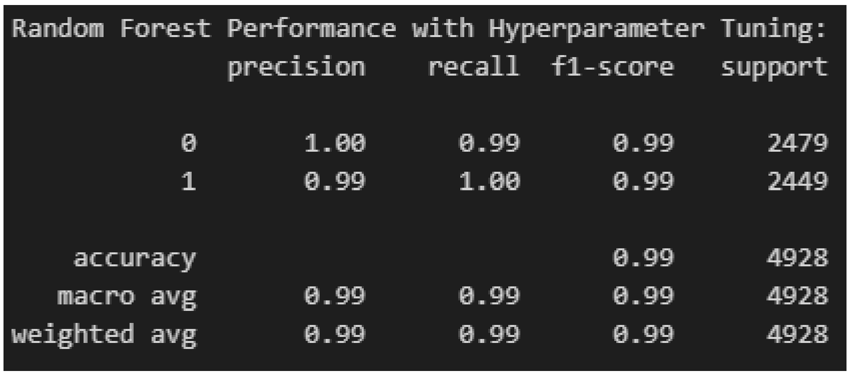


Figure 5.4: Confusion matrix for Logistic Regression

5.4 Random Forest

A supervised learning algorithm that works well for both classification and regression tasks is Random Forest. It capabilities as an outfit technique by accumulating the expectations of a few choice trees to improve exactness and vigor.

During training, the Random Forest ensemble learning approach generates several decision trees and gives the mode of their classifications or the mean prediction for regression tasks. The methodology begins by creating numerous bootstrap samples from the original dataset. For each sample, a decision tree is built by selecting a random subset of features at each split, promoting diversity among the trees. This random feature selection and bootstrapping process ensure that the trees are de-correlated, reducing the risk of overfitting. Once all trees are trained, the Random Forest aggregates their predictions through majority voting (for classification) or averaging (for regression). The final model benefits from the collective wisdom of the ensemble, providing high accuracy, robustness to noise, and better generalization compared to individual trees.



```
Random Forest Performance with Hyperparameter Tuning:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      2479
           1       0.99      1.00      0.99      2449

    accuracy                           0.99      4928
   macro avg       0.99      0.99      0.99      4928
weighted avg       0.99      0.99      0.99      4928
```

Figure 5.5: Classification for Random Forest

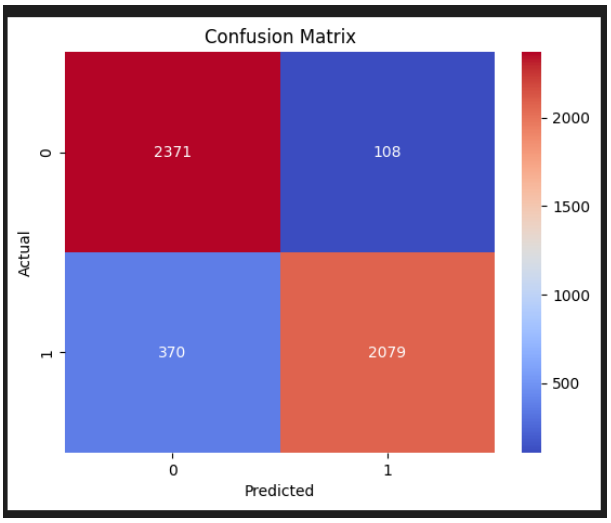
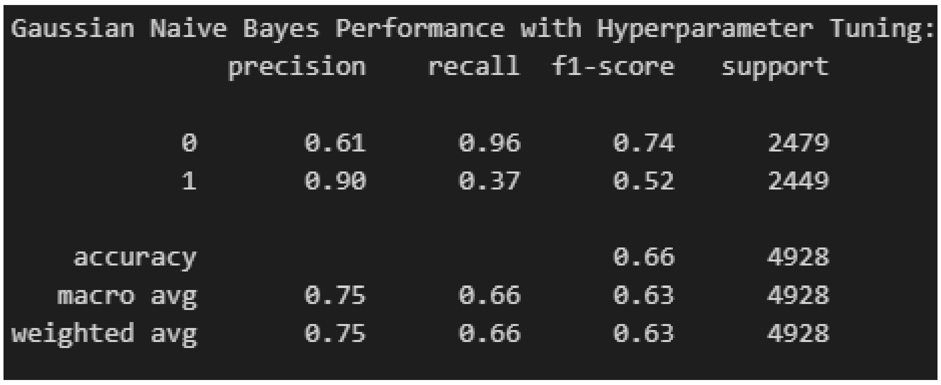


Figure 5.6: Confusion matrix for Random Forest

5.5 Gaussian Naive Bayes

Gaussian Naive Bayes is a probabilistic classification method founded on Bayes' Theorem. It operates under the assumption that the features used for making predictions are conditionally independent, given the class label. This simplification, termed the "naive" assumption, allows for straightforward probability calculations, despite the fact that real-world features often exhibit some degree of correlation.

Based on Bayes' Theorem, the Naïve Bayes classifier is probabilistic and believes that a feature's presence in a class is independent of other features. The process involves estimating the likelihood of each feature given each class and the prior probability of each class from the training data. The posterior probability of each class for a particular case is calculated by summing these probabilities. The class with the highest posterior probability is chosen by the algorithm to be the predicted class. Naive Bayes works well in many real-world scenarios, especially when dealing with big datasets and text classification tasks, despite its simplicity and strong independence assumption. It is trained in processing both binary and multiclass classification problems and is computationally efficient, requiring just one pass over the training data.



```
Gaussian Naive Bayes Performance with Hyperparameter Tuning:
              precision    recall  f1-score   support

           0       0.61      0.96      0.74      2479
           1       0.90      0.37      0.52      2449

    accuracy                           0.66      4928
   macro avg       0.75      0.66      0.63      4928
weighted avg       0.75      0.66      0.63      4928
```

Figure 5.7: Classification report for Gaussian Naïve Bayes

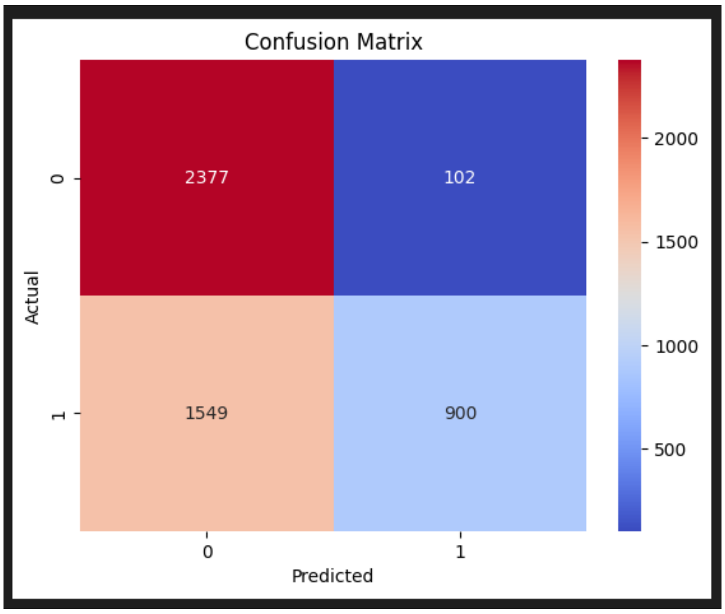


Figure 5.8: Confusion matrix for Gaussian Naïve Bayes

5.6 Summary

This project employs several machine learning algorithms for fraud detection. Decision Trees create a model based on simple decision rules inferred from data features A logistic function can be utilized in logistic regression to express the likelihood of a binary outcome. For better accuracy and perseverance, Random Forest constructs several decision trees and combines their forecasts. Naïve Bayes determines the likelihood of classes based on feature independence using Bayes' Theorem. Data gets separated into small clusters using K-Means Clustering, which avoids within-cluster variance. Each algorithm contributes uniquely to detecting fraudulent transactions, enhancing the system's overall performance and reliability.

CHAPTER-6
SYSTEM DESIGN

6.1 Input Design:

Raw data which in an information system is processed to produce results is referred to as input. At the design of a system, developers need to consider input devices such as PCs, MICRs
(Magnetic Ink Character Recognition), OMRs (Optical Mark Recognition), among others. This is because the quality of the system's output largely depends on the quality of its input.
Good input forms and screens are characterized by some of the following:
• They have specific functions, like storing, recording, and retrieving information.
• They ensure proper entry of data that is accurate and complete.
• They are simple, clear, and easy to understand.
• They guide the user's eye to the appropriate areas; they maintain consistency and are clear.
All of this can be achieved by putting into practice the universal design principles, such as:
• Knowing what type of inputs the system requires.
• Understanding how end users interact with the various components of forms and screens.

6.1.1 Objectives of Input Design:

The key goals of input design are as follows:
• To design efficient data entry and input processes.
• To minimize the amount of input required.
• To design documents or procedures that capture data at source.
• To design input records, data entry screens, and user interface screens.
• To incorporate validation checks and robust input controls.

6.2 Output Design:

The main task of any system is the design of output. At this stage, the need for the required output is identified, and the output controls and prototype report layouts needed are considered.

6.2.1 Objectives of Output Design:

The following are the objectives of output design:
•To design an output that serves the required purpose and minimize the unwanted output.
•Output design: To create a design of output that can meet the requirements of the end user.
•Providing the right amount of output.
•To format the output in suitable form and send it to the correct person.
•Available in time to support good decisions

6.3 UML DIAGRAMS

Unified Modelling Language shortcut UML. It is an agreed general-purpose modelling language within the Object-oriented Software Engineering, area.
It is a controlled standard, and was created by, the Object Management Group.
a Meta-model and a notation. It may eventually be part of, or affiliated with, UML too.
Unified Modelling Language is the universal language allowing the specification, Visualization, Construction and documentation of the software system artefacts as well as the business modelling

and other non-software systems.

UML represents a body of established engineering best practices that have invariably been effective in modeling complex, large systems.

UML is one of the significant issues of object -oriented software design and software design process. The UML is mainly graphical notations used to specify design of software projects.

6.3.1 CLASS DIAGRAM

In software engineering, a class in Unified Modelling Language, or shortly UML, is a static structure diagram. This model indicates the structure of a system, with classes of the system, their properties, and operations or methods of the class and the relationship between classes, that describes which classes have information.

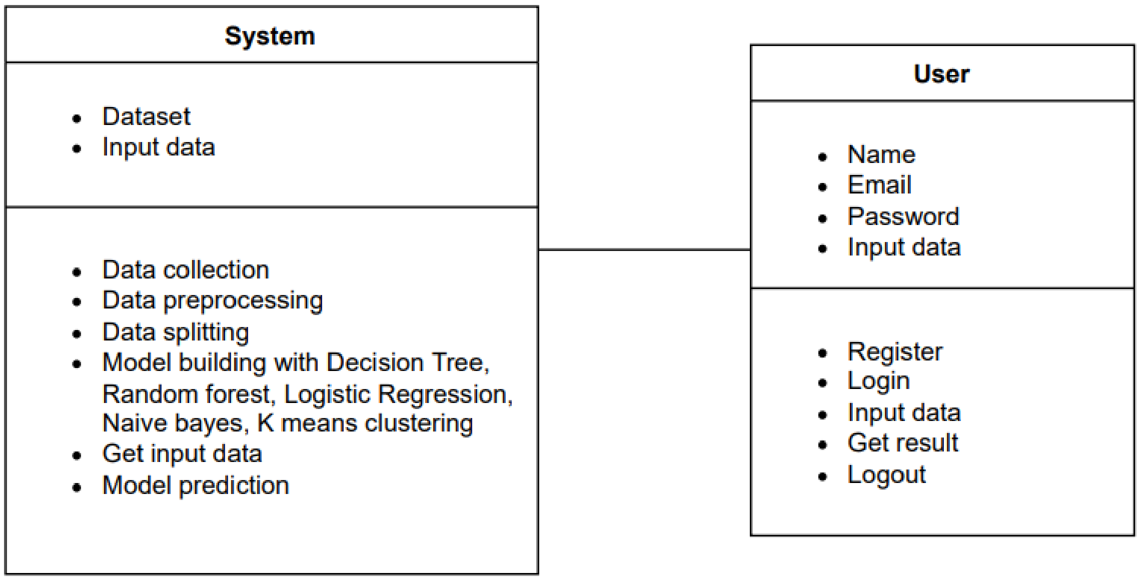


Figure 6.1: Class Diagram

6.3.2 USE CASE DIAGRAM

A use case is a behavioral diagram, which is defined and built by a use case analysis. Main aim is to provide a visual Presentation of functional description of the system in terms of actors' intentions. (described by use cases), and the relationships between use cases.

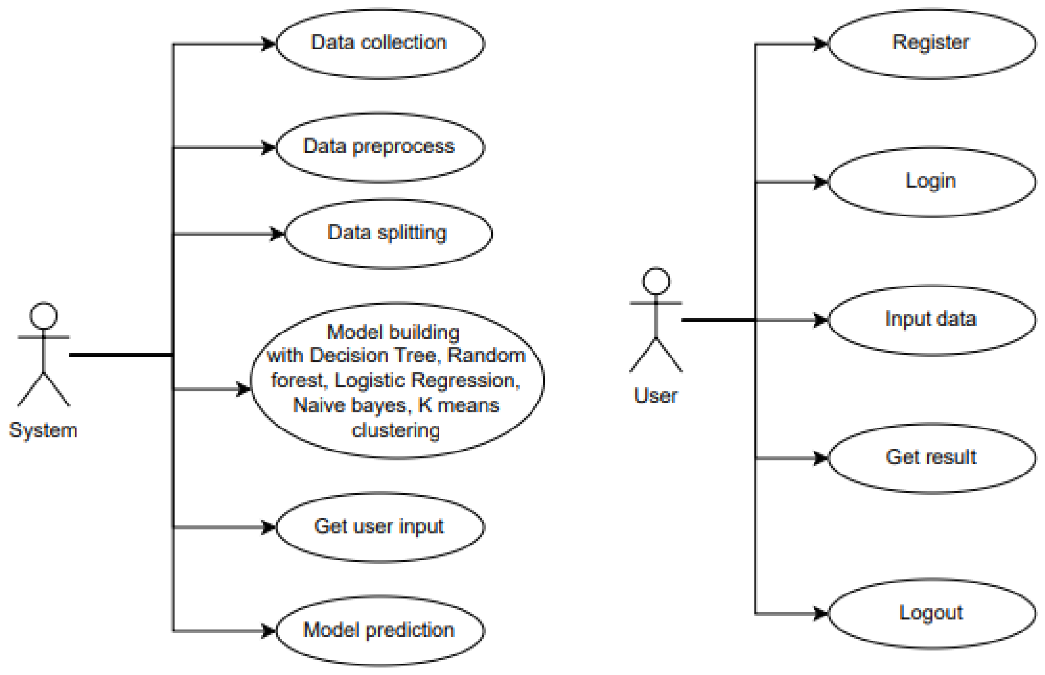


Figure 6.2: Use Case Diagram

6.3.3 SEQUENCE DIAGRAM

In UML,an interaction diagram that depicts how processes interoperations occur in what sequence. It is a process of building a of a Message Sequence Chart. Sometimes Sequence diagram called event diagram, event scenario, and timing diagrams

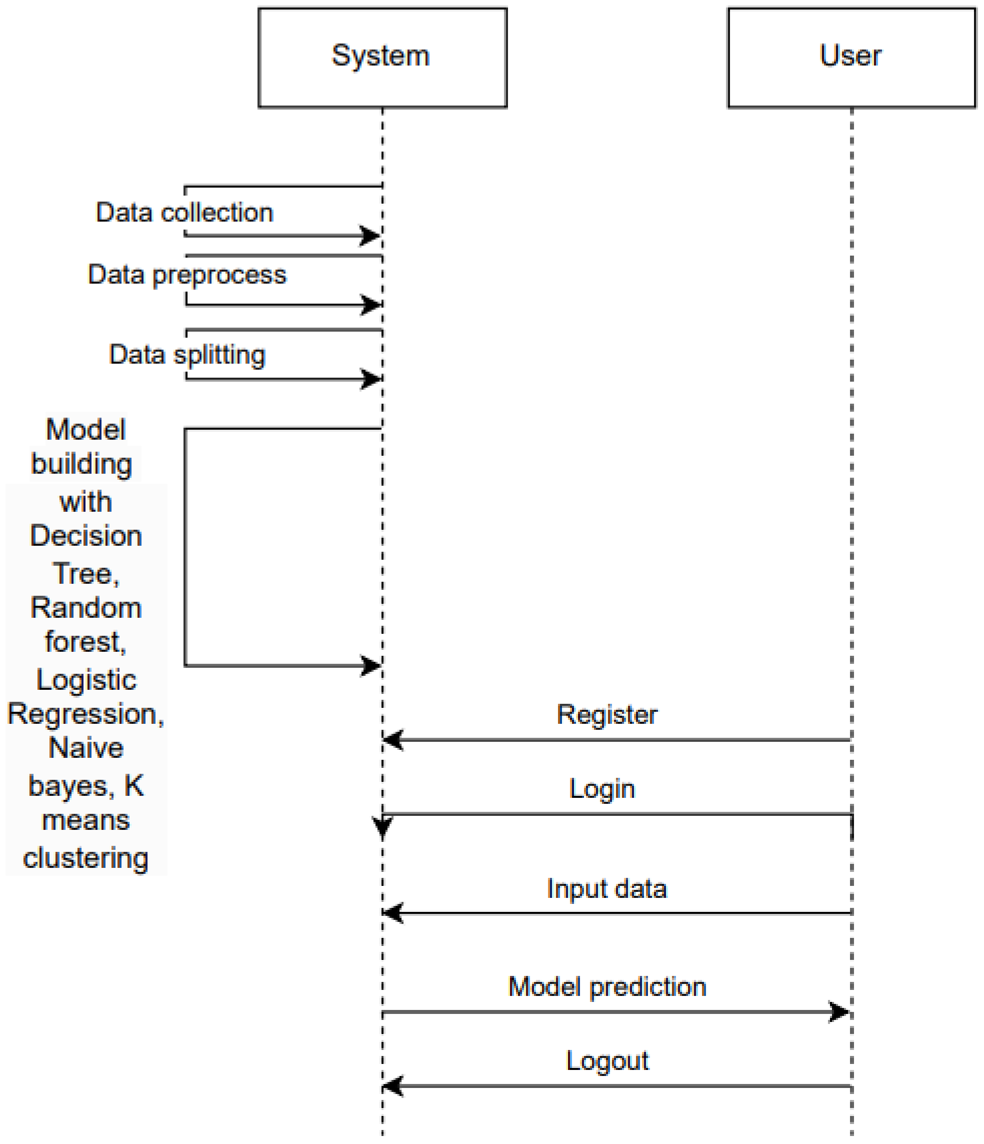

System
User
Data collection
Data preprocess
Data splitting
Model building with Decision Tree, Random forest, Logistic Regression, Naive bayes, K means clustering
Register
Login
Input data
Model prediction
Logout

Figure 6.3: Sequence Diagram

6.3.4 COLLABORATION DIAGRAM:

The method invocation sequence is reflected in collaboration diagram via a numbering mechanism as illustrated on the following diagram :The number expresses, in a cumulative way how are methods invoke each other's We employed a similar approach that we introduce order management, previously and here is to what extents the Collaboration diagram explains :Method Invocation is similar between Collaboration and the Sequence diagrams the distinction this is no explication in items order diagram where the Collaboration does tell of object' s organization end.

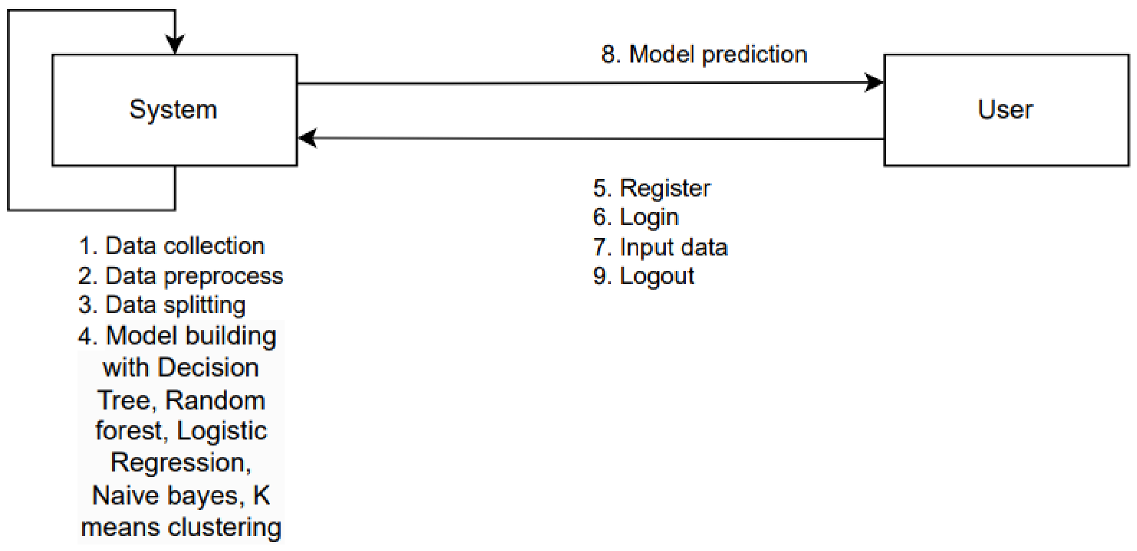


Figure 6.4: Collaboration Diagram

6.3.5 DEPLOYMENT DIAGRAM

Deployment diagrams represent the view of a system. It is related to the component diagram , the components are deployed using deployment diagrams. Deployment diagram constitutes nodes. Node is a physical hardware used for deploying an application.

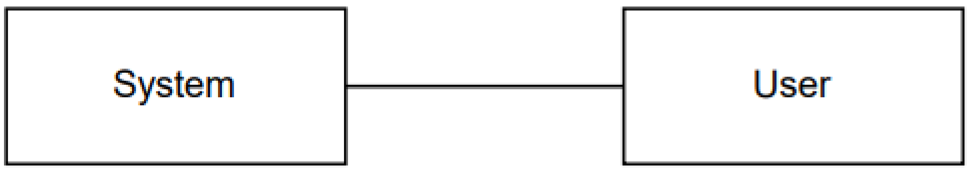


Figure 6.5: Deployment Diagram

6.3.6 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations. They describe the workflows stepwise with activities and actions that support primitive objectives such as choice, iteration, and concurrency.

Activity diagrams can be used for the illustration of business and operational step-by-step workflow.

Number of components in a system. An activity diagram depicts the control flow in general.

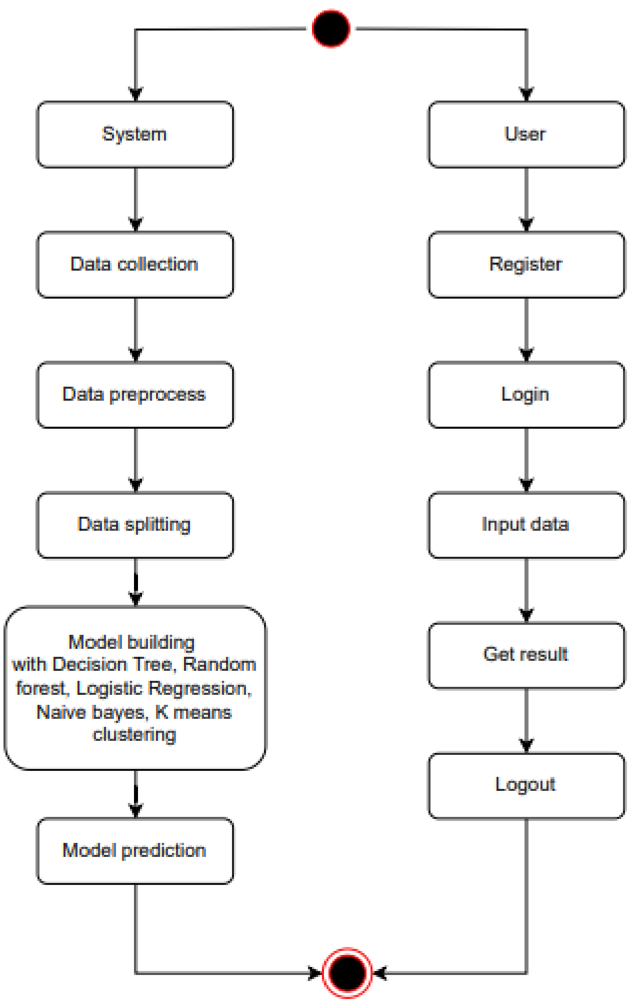


Figure 6.6: Activity Diagram

6.3.7 ER DIAGRAM:

The so-called "Entity-Relationship Model" mainly describes, in the forms of diagrams named Entity-Relationship diagrams, how "a" selected database is structured. Generally, an ER model is the structural design or blueprint of later implementable Databases. Therefore, E-R Model may essentially consist of all those elements present in an E-R Diagram, essentially Entity-set or relationship set.

An ER diagram illustrates the relationships between entity sets. An entity set is an arrangement of similar entities that may have features. In a DBMS, an entity is a table or a table's attribute. Thus, by displaying the links between tables and their attributes, an ER diagram displays the entire logical structure of a database. To better comprehend this concept, imagine a simple ER diagram.

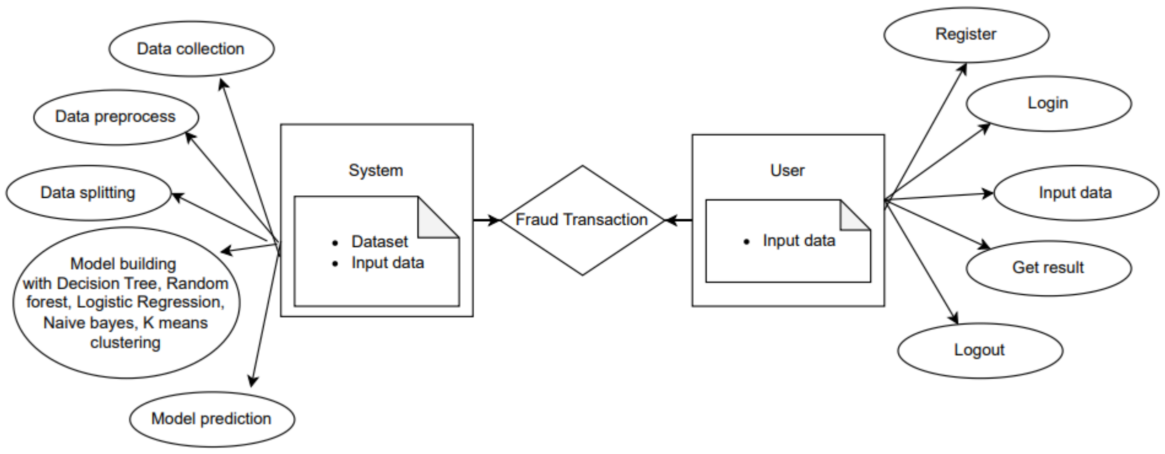


Figure 6.7: ER Diagram

6.3.8 COMPONENT DIAGRAM:

A component diagram, or simply UML component diagram, draws out how physical components in the system is arranged and wired. Component diagrams are sometimes drawn to help model Implementation details and check again that every part of the system's required function is Covered by a development that is preplanned

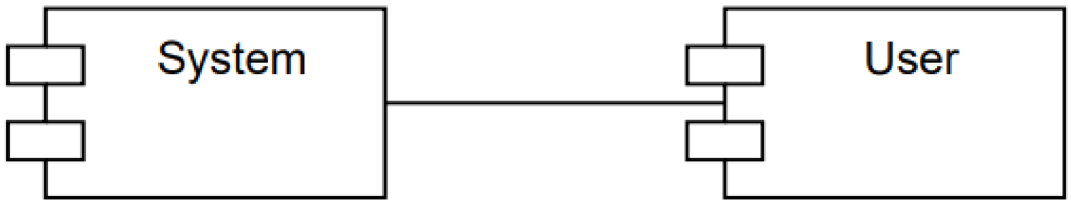


Figure 6.8: Component Diagram

6.4 DFD DIAGRAM:

Data Flow Diagrams (DFDs): These are conventional ways of displaying information flows within a system. A tidy and straightforward DFD may graphically express an important section of the system requirements. It might be manual, built-in, or a combination of the two. They show how information enters and exits the system, what changes the information, and where It is stored. A DFD should be able to show the scope and boundary of a whole system. A DFD is a major communication tool used by a system analyst with individuals who have important roles in beginning the process of redesign for a system.

Context level Diagram:

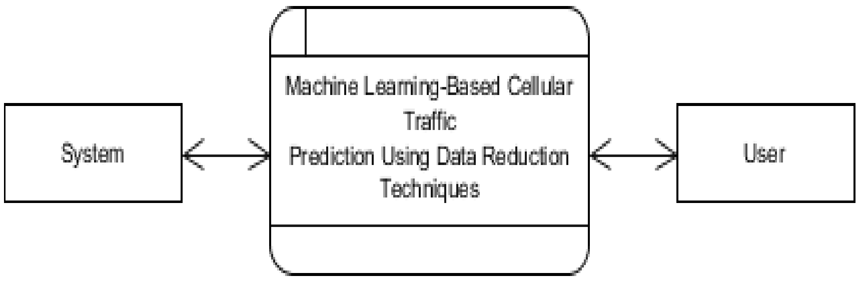

System
Machine Learning-Based Cellular Traffic Prediction Using Data Reduction Techniques
User

Figure 6.9: DFD Diagram

Level 1 Diagram:

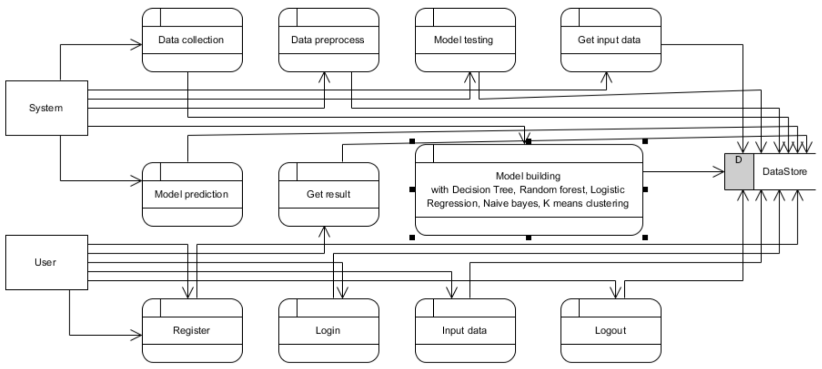


Figure 6.10: Leve 1 Diagram

Level 2 Diagram:

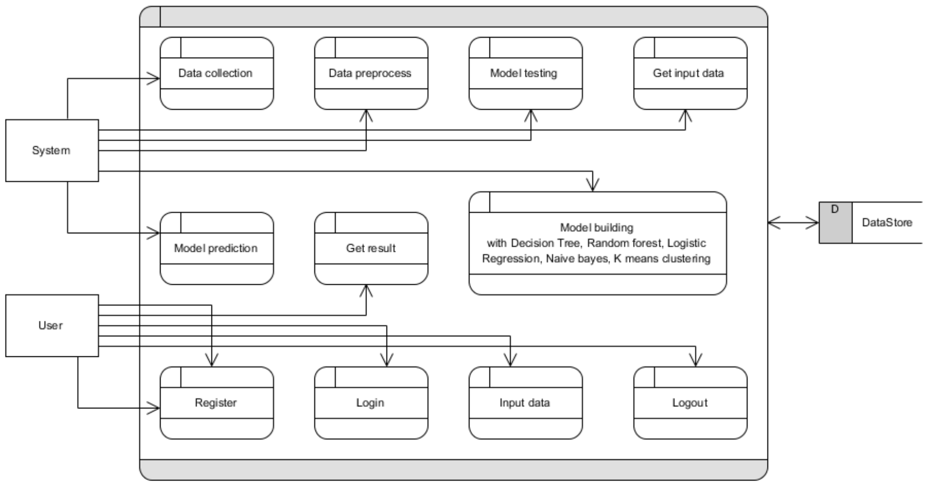


Figure 6.11: Level 2 Diagram

CHAPTER-7
TIMELINE FOR EXECUTION OF PROJECT (GANTT CHART)

7.1 Project Timeline Overview

Duration: 3 Months (12 Weeks)
Start Date: September 1, 2024
End Date: November 30, 2024

Table 7.1: Project Timeline and Milestone Overview

Phase	Duration	Deliverables
Planning	2 weeks	Project Plan
Development	8 weeks	Core System
Testing	2 weeks	Test Reports
Deployment	1 week	Live System

7.2 Gantt Chart Breakdown

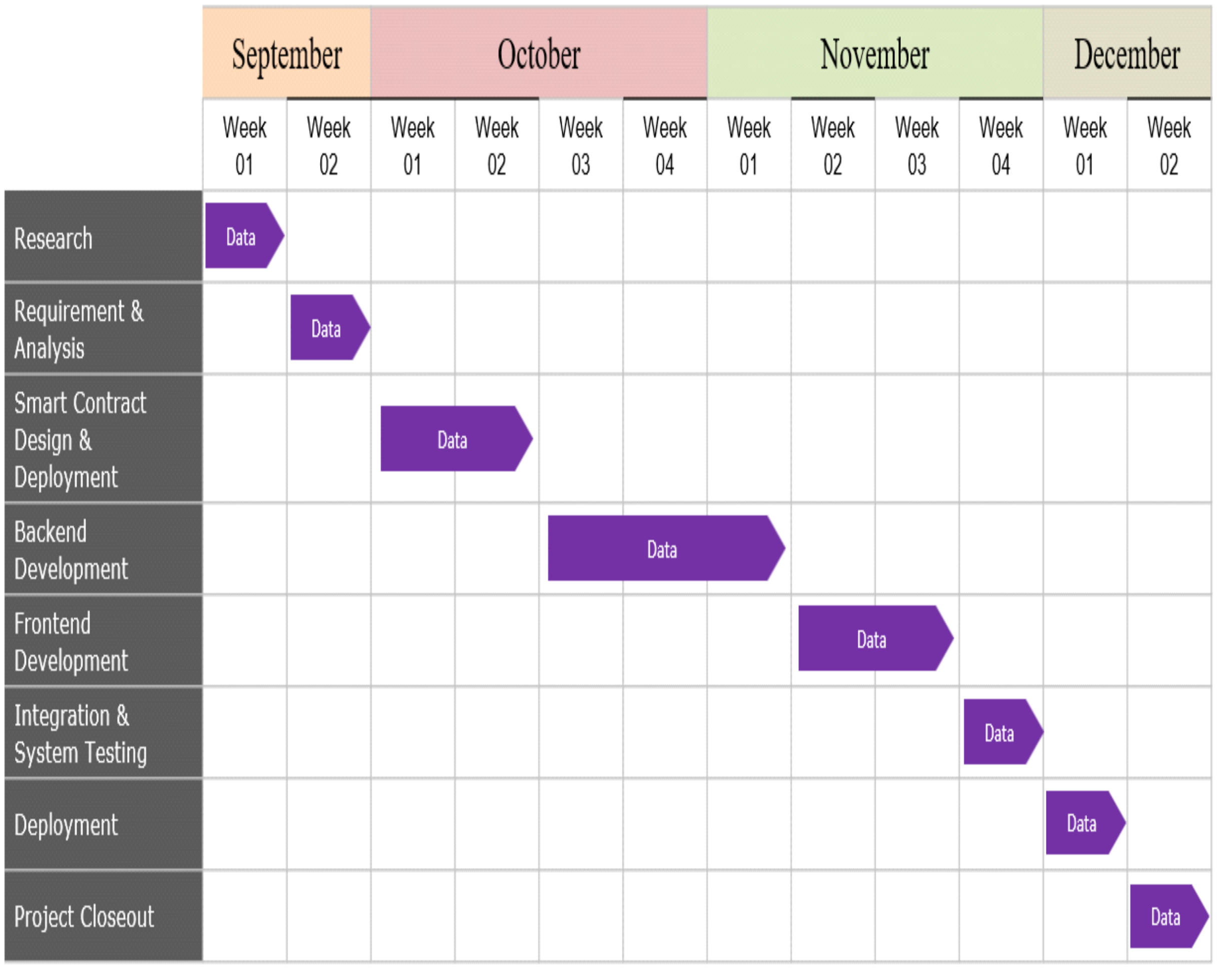


Figure 7.1: Gantt Chart Breakdown

7.3 Key Milestones

7.3.1 Phase Completion Milestones

• Week 2: Project Planning Complete

• Week 3: Development Environment Ready
•Week7: Core Features for Medical Product Tracking Implemented
• Week 9: System Integration (Blockchain, Backend, and APIs) Complete
• Week 11: Testing of Medical Product Tracking System Complete
• Week 12: Final Deployment and Handover

7.3.2 Critical Deliverables

• Week 1: Project Requirements Document (Specific to Medical Product Tracking)
• Week 2: System Architecture Design for Medical Product Supply Chain Tracking
• Week 3: Blockchain Network Setup and Smart Contract Design for Tracking Product Authenticity and Transfers
• Week 4: Database Schema for Storing Metadata of Medical Products and Transaction Logs
• Week 5: Frontend Prototype for Tracking Medical Products Across the Supply Chain
• Week 6: Backend Development and Integration with Blockchain and APIs
• Week 7: Security Implementation for Ensuring Product Data Integrity and Compliance
• Week 8: Blockchain Integration with Testing for Smart Contracts and Supply Chain Events
• Week 9: Performance and Load Testing for Real-Time Product Updates
• Week 10: User Acceptance Testing by Supply Chain Stakeholders (Manufacturers, Distributors, Pharmacies, Hospitals)
• Week 11: Test Reports and Compliance Documentation
• Week 12: Final Deployment of the Medical Product Tracking System

7.4 Resource Allocation

7.4.1 Development Team

• 2 Frontend Developers
• 2 Backend Developer
• 1 Database Administrator
• 1 Security Expert
• 1 Project Manager

7.4.2 Infrastructure Requirements

• Development
• Testing Environment
• Cloud Infrastructure
• Version Control System
• CI/CD Pipeline

CHAPTER-8
IMPLEMENTATION AND RESULTS

8.1 Modules:

8.1.1 System:

Data Collection: it gets the transaction dataset meant for the detection of fraudulent activities from PaySim1, which was obtained from Kaggle. Hence, this module collects data and prepares for processing.

• Data Preprocessing: Extensive preprocessing of the dataset collected is done by cleaning the data, handling missing values, feature engineering, and normalization. This step makes the data clean, consistent, and ready for training machine learning models.

• Data Splitting: Pre-processed data is divided into two sub-sets:

- Model Training: The training of machine learning models will be done with 80% of the data. During this phase, the model will start understanding and building up the pattern and anomaly regarding fraudulent transactions.

- Model Testing: The remaining 20% of the dataset is used for testing the performance of the models. Models will predict fraud, and accuracy, precision, recall, and F1-score are measured.

• Model Training: The training subset of the dataset is used to train different machine learning models comprising K-Means Clustering, Logistic Regression, Naive Bayes, Decision tree, and Random Forest. Iterative optimization techniques, including gradient descent, are used for fine-tuning model parameters in order to reduce prediction errors.

• Model Evaluation: Each of the trained models is evaluated on the testing subset. The key metrics that are used to evaluate the performance of each model in detecting fraudulent transactions include accuracy, precision, recall, F1-score, and AUC-ROC.

• Model Saving: The best models, after training, are saved in.pkl format so that the weights and bias learned can be maintained and may then be loaded again with ease in future use for prediction.

• Model Prediction: The saved models are used for the input of new transaction data in predicting whether the transactions are fraudulent or non-fraudulent.

8.2 User

8.2.1 Register

Users, such as financial analysts or administrators, register withtheir credentials to create an account in the system. Registration includes providing necessary details and setting up secure login credentials.

- Login: Registered users can log in with their credentials to access the system's features and functionalities.
- Input Data: Users can input new transaction data into the system. This data is sent to the trained machine learning models for fraud prediction.
- Viewing Results: After the models analyse the input data, the results are displayed to the user. The system provides detailed predictions indicating whether the transactions

are fraudulent or Non-Fraudulent, along with relevant metrics.

- Logout: Users can log out of the system to secure their session and protect their personal data.

8.3 Output Screen

8.3.1 Home Page: The Home Page serves as the landing page of your application.

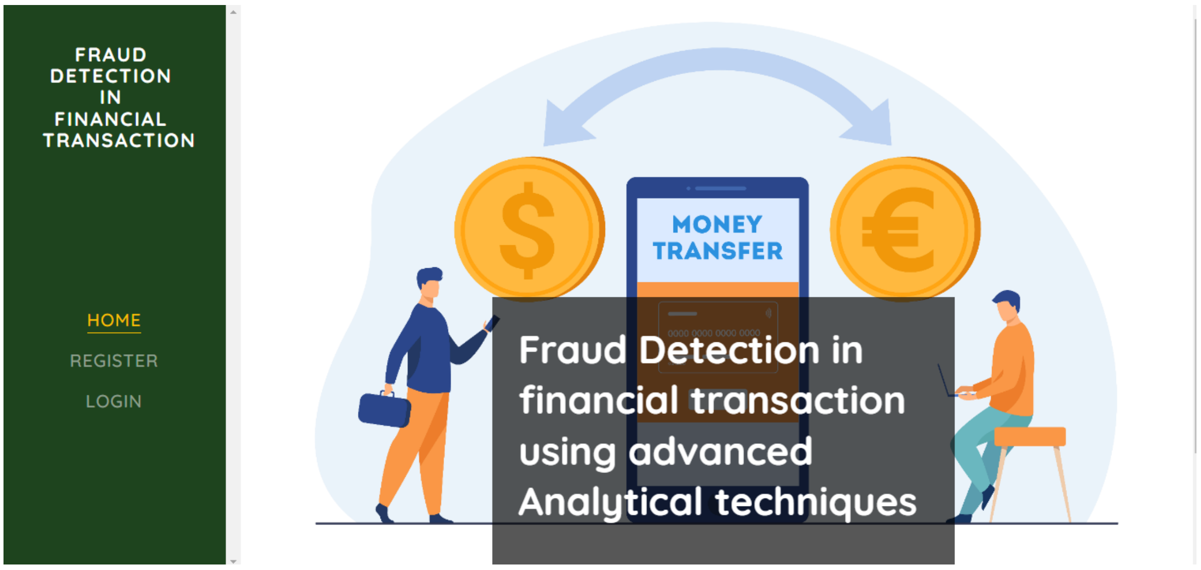

FRAUD DETECTION IN FINANCIAL TRANSACTION
HOME
REGISTER
LOGIN
MONEY TRANSFER
Fraud Detection in financial transaction using advanced Analytical techniques

Figure 8.1: Home Page

8.3.2 Registration Page: The Registration Page allows new users to create an account with the application.

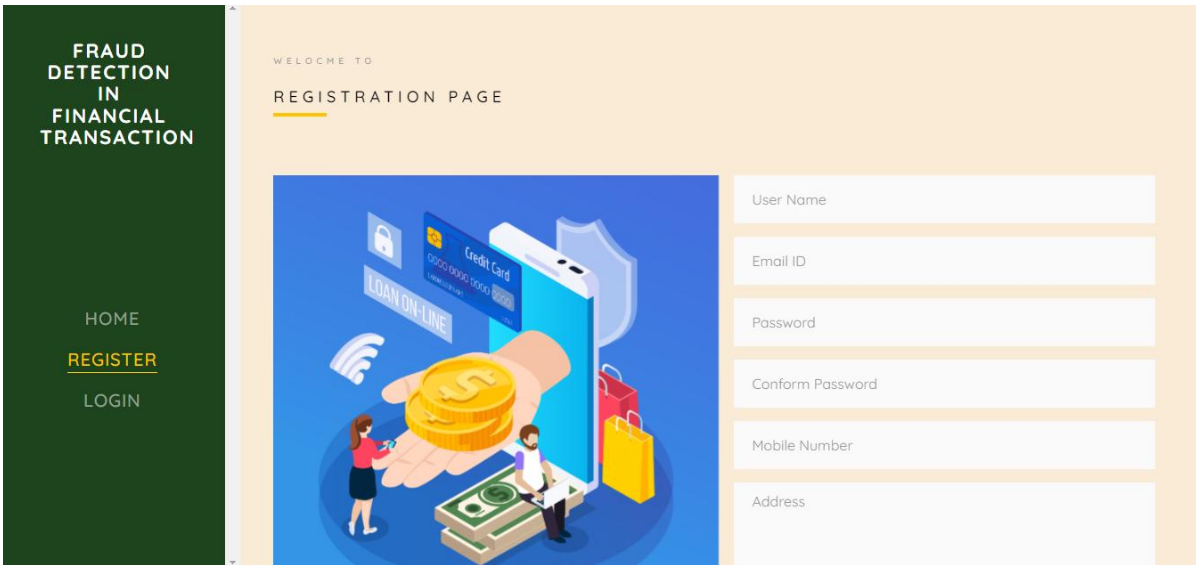


Figure 8.2: Registration Page

8.3.3 Login Page: The Login Page enables users to access their existing accounts by entering their credentials. It usually includes fields for entering a username/email and password.

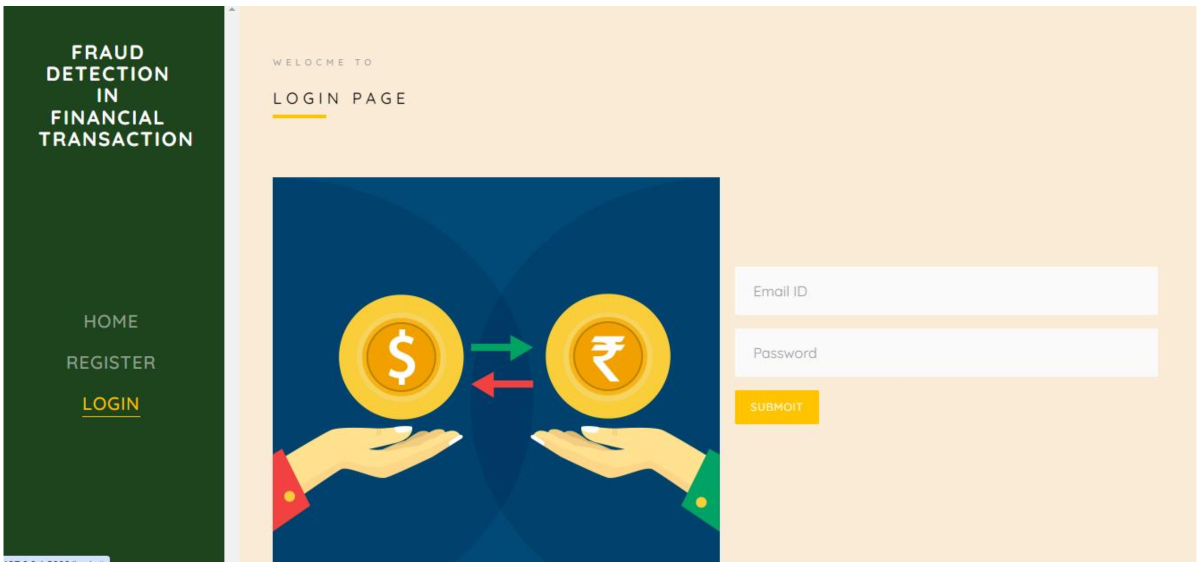


Figure 8.3: Login Page

8.3.4 User home Page: After user successfully login this page will be appear.

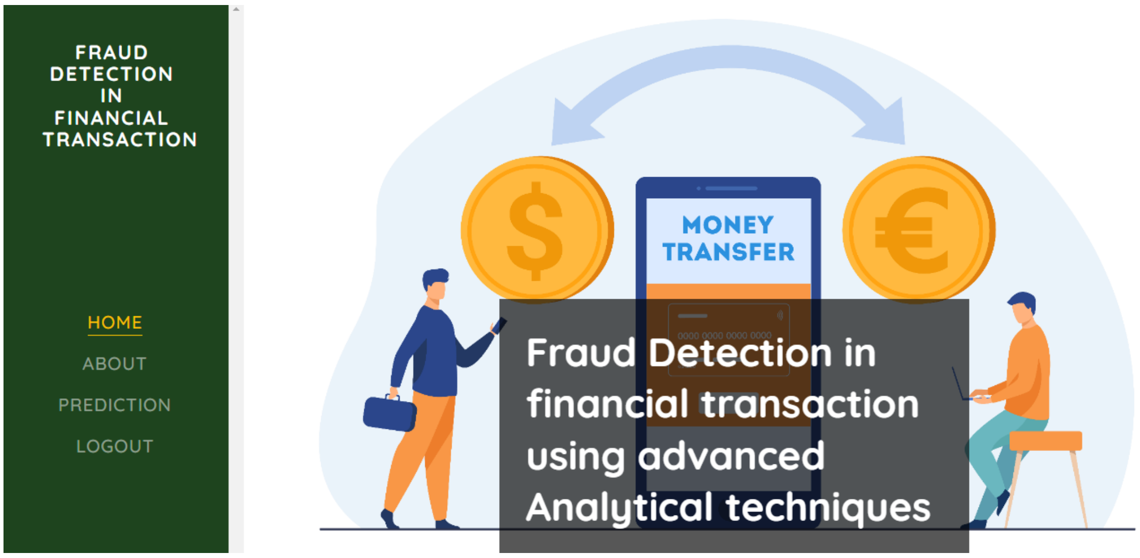
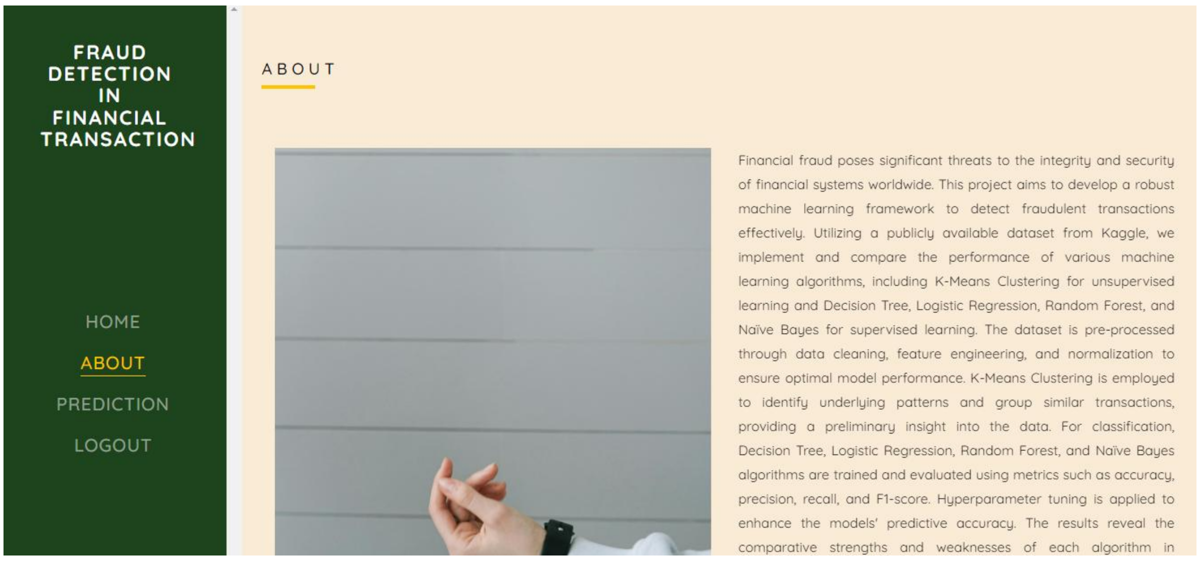

FRAUD DETECTION IN FINANCIAL TRANSACTION
HOME
ABOUT
PREDICTION
LOGOUT
MONEY TRANSFER
Fraud Detection in financial transaction using advanced Analytical techniques

Figure 8.4 : User home Page

8.3.5 About Page: The About Page offers detailed information about the project, including its purpose, goals, and the technology used.



Financial fraud poses significant threats to the integrity and security of financial systems worldwide. This project aims to develop a robust machine learning framework to detect fraudulent transactions effectively. Utilizing a publicly available dataset from Kaggle, we implement and compare the performance of various machine learning algorithms, including K-Means Clustering for unsupervised learning and Decision Tree, Logistic Regression, Random Forest, and Naïve Bayes for supervised learning. The dataset is pre-processed through data cleaning, feature engineering, and normalization to ensure optimal model performance. K-Means Clustering is employed to identify underlying patterns and group similar transactions, providing a preliminary insight into the data. For classification, Decision Tree, Logistic Regression, Random Forest, and Naïve Bayes algorithms are trained and evaluated using metrics such as accuracy, precision, recall, and F1-score. Hyperparameter tuning is applied to enhance the models' predictive accuracy. The results reveal the comparative strengths and weaknesses of each algorithm in

Figure 8.5: About Page

8.3.6 Prediction Page: The Prediction Page allows users to input data and receive predictions based on the trained machine learning models. This page typically includes a form or interface for uploading or entering data.

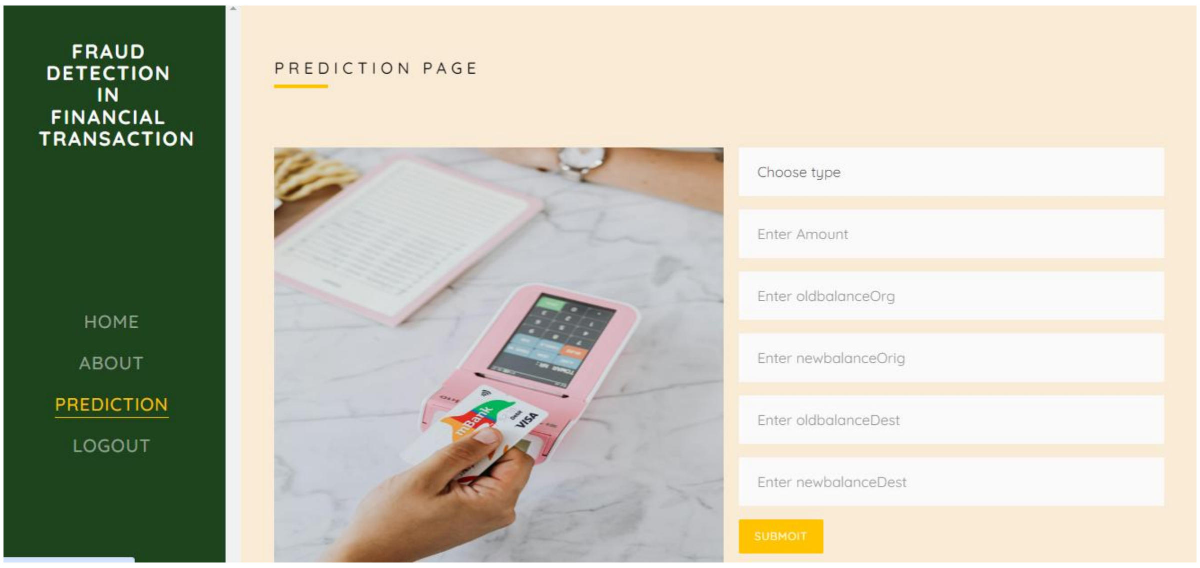
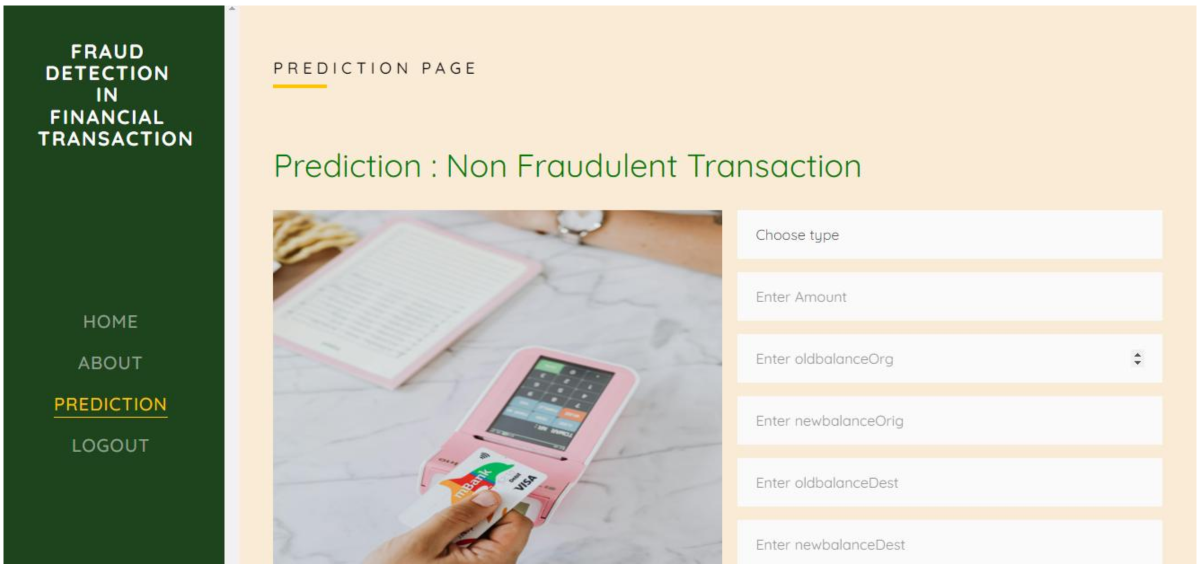


Figure 8.6: Prediction Page

8.3.7 Result Page: In here result will be display which is predicted by our trained model.


FRAUD DETECTION IN FINANCIAL TRANSACTION

HOME

ABOUT

PREDICTION

LOGOUT

PREDICTION PAGE

Prediction : Non Fraudulent Transaction

Choose type

Enter Amount

Enter oldbalanceOrg

Enter newbalanceOrig

Enter oldbalanceDest

Enter newbalanceDest

Figure 8.7: Result Page

CHAPTER-9
SYSTEM STUDY AND STUDY

9.1 Feasibility Study

• Feasibility Study:

The feasibility study assesses whether it is possible to integrate sophisticated machine learning algorithms in real-time fraud detection during financial transactions. This is determined by the different algorithms' capability of classifying transactions as either fraudulent or not (Decision Tree, Logistic Regression, Random Forest, Naïve Bayes, and K-Means Clustering).

• Prototype Development:

Developing a prototype system that incorporates these machine learning algorithms into fraud detection. This will include algorithm design to preprocess and analyze the transactional data, train models, and iteratively improve their performance based on feature selection and model tuning.

•Testing and Evaluation:

Conduct thorough testing to verify the performance of the prototype. This includes:

-Accuracy Evaluation: Comparing the precision of different machine learning models in detecting fraudulent transactions. This is followed by feedback and further iterations for training.

- Robustness Testing: Testing whether fraud can be successfully detected using this system for diverse conditions such as different transaction volume and different kinds of frauds.

- Real-time Monitoring: Testing if fraud detection will actually take place in a real-time context that reflects normal financial operations

• User Feedback and Iterative Improvement:

Get users' opinions or feedback on users' perceptions. Improve the model iteratively. Improve it as per users' input as well as according to performance measures.

9.2 SYSTEM TESTING

• Feasibility Study:

This phase studies the project's feasibility and produces a business proposal with a general project plan and cost estimates. A feasibility evaluation for the proposed system occurs throughout system analysis to guarantee that the system will not be a burden to the firm. Major system requirements must be understood. The feasibility analysis contains three major considerations:

- Economic Feasibility: This is the checking of the impact on the economy the system will have within the organization. The system designed must remain within the budget since most of the technologies applied are free of charge; thus only customized products will be charged
- Technical Feasibility: Ascertainment of whether the system is technically infeasible with the demands placed on available technical resources: Any system developed from this must have modest requirements.
- Social Feasibility: The degree to which the system will be accepted by the users. This includes training the users to use the system effectively and making them accept it as a given. User confidence should be increased so that they may provide constructive

criticism for improvement.

• System Testing:

The purpose of testing is to find faults. Testing verifies the functionality of components, sub-assemblies, and the final product. It verifies that the software system satisfies its requirements and user needs without failing in an unacceptable manner.

9.2.1 Types of Tests

- **Unit Testing:**

  Unit testing creates test cases that check the internal program logic, ensuring that program inputs produce correct outputs. It tests individual software components in the program once they are fully developed but before they are integrated. This structural testing is intrusive and depends on the architecture of the structure.

  Unit tests perform simple tests at the component level, ensuring that every different path of a business process meets defined requirements

- **Integration Testing:**

  The integration software testing is the incremental evaluation of multiple software multiple software components that are interdependent on a single platform that generate failures caused by dependence on one another. Integration tests are event-driven and are concerned with the basic outcome of screens or fields.

- **Acceptance Testing:**

  Acceptance Testing is one of the very important phases that require great participation by the end user. It guarantees that the system satisfies functional requirements. Test results show all test cases passed successfully without any defects detected.

- **White Box Testing:**

  White Box Testing: This tests a software having an understanding of all its inner details, its structure, and even the language. Testing areas that may not be reached from a black box level, where internal logic of the software is tested to its fullest.

- **Black Box Testing:**

  Testing the software with no knowledge of the inner details, structure, or language of the software. Tests are written from a definitive source document, such as a specification or requirements document

Table 9.1 TEST CASES:

Input	**Output**	**Result**
Input	Various model are tested as per instructions given by user on the different model.	**Success**
Model	Testing different input given by the user on numerous models are creating using the different algorithms and data.	**Success**
Prediction	Predicting the performance using the different models build from the algorithms.	**Success**

Table 9.2 Test cases Model building:

S.NO	**Test cases**	**I/O**	**Expected O/T**	**Actual O/T**	**P/F**
1	Read datasets.	Dataset path.	Dataset need to be read successfully.	Datasets fetched successfully.	produces P. Otherwise we get F.
2	Registration	Valid username email, password.	Verify that the registration form accepts valid user inputs and successfully creates a new account.	User is successfully registered, and an account is created	produced P. If this is not, it will undergo F.
3	Login	Valid username and password	Verify that users can log in with valid credentials	User is successfully logged in and redirected to the dashboard	It produced P. If this is not, it will undergo F.
4	Find Fraudulent	Input Financial data	Output as predicted Fraudulent or Non Fraudulent	Output as predicted Fraudulent or Non Fraudulent	It produced P. If this is not, it will undergo F

CHAPTER-10
CONCLUSION

This project successfully developed an advanced fraud detection framework using machine learning algorithms to classify financial transactions into "Fraudulent" and "Non-fraudulent" categories. By leveraging Decision Tree, Logistic Regression, Random Forest, Naïve Bayes, and K-Means Clustering, the system demonstrated enhanced detection accuracy, adaptability, and scalability. The integration of comprehensive data preprocessing techniques, real-time processing capabilities, and robust evaluation metrics ensured the system's reliability and effectiveness. This project addressed the limitations of traditional rule-based systems with a proactive, automated approach to fraud detection that reduces false positives and greatly reduces operational impedance. Results provide a case for several advanced analytical techniques in combination to further advance the security and trust of financial systems and provide significant benefits in combating financial fraud. In this context, the work will be continued by optimizing models further, adding other machine learning techniques, and practical implementation of the system in real financial environments.

CHAPTER-11
FUTURE ENHANCEMENT

Some future enhancements to the fraud detection system are through the incorporation of deep learning techniques such as CNN and RNN to grasp complex patterns within transaction data. Real-time anomaly detection will be enabled with streaming data technologies that will further fine-tune the detection and response time of the system against fraudulent activities. The work can also be extended by the integration of advanced feature selection methods with hyperparameter optimization techniques to further improve the accuracy and the performance of the models. Additionally, a user-friendly interface with interactive visualizations will be developed to support analysts in the interpretation and investigation of fraud cases. Further extension of the system to support multi-class classification will provide further granularity into the different types of fraud. This work will also collaborate with financial institutions on more diverse and extensive datasets to refine and validate the models for robustness and scalability across a variety of financial environments**.**

CHAPTER 12
REFERENCES

1. Chen, Y., Wang, Y., & Jiang, Y. (2020). A survey on fraud detection approaches in financial domain. IEEE Access, 8, 37373-37390.
2. Liu, C., & Fan, J. (2020). Financial fraud detection model: Based on random forest. Journal of Computational and Applied Mathematics, 371, 112668.
3. Phua, C., Lee, V., Smith, K., & Gayler, R. (2010). A comprehensive survey of data mining-based fraud detection research. Artificial Intelligence Review, 34(1), 1-14.
4. Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. Decision Support Systems, 50(3), 602-613.
5. Ngai, E. W. T., Hu, Y., Wong, Y. H., Chen, Y., & Sun, X. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. Decision Support Systems, 50(3), 559-569.
6. Kou, Y., Lu, C. T., Sirwongwattana, S., & Huang, Y. P. (2004). Survey of fraud detection techniques. IEEE International Conference on Networking, Sensing and Control, 2004, 749-754.
7. Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2018). Credit card fraud detection: A realistic modeling and a novel learning strategy. IEEE Transactions on Neural Networks and Learning Systems, 29(8), 3784-3797.
8. Carcillo, F., Borgne, Y. A., Caelen, O., Kessaci, Y., Oblé, F., Bontempi, G., & Termier, A. (2019). Combining unsupervised and supervised learning in detection. Information Sciences, 557, 317-331.
9. Bolton, R. J., & Hand, D. J. (2002). Statistical fraud detection: A review. Statistical Science, 17(3), 235-249.
10. Van Vlasselaer, V., Eliassi-Rad, T., Akoglu, L., Snoeck, M., & Baesens, B. (2017). AFRAID: Fraud detection via active feature space augmentation. International Conference on Information and Knowledge Management, 2017, 1961-1964.
11. Whitrow, C., Hand, D. J., Juszczak, P., Weston, D., & Adams, N. M. (2009). Transaction aggregation as a strategy for credit card fraud detection. Data Mining and Knowledge Discovery, 18(1), 30-55.
12. Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19-31.
13. Wei, W., Wang, L., Zhu, M., & Yang, S. (2013). A comprehensive survey on hybrid data mining techniques in credit card fraud detection. International Conference on Machine Learning and Cybernetics, 2013, 1127-1131.
14. West, J., & Bhattacharya, M. (2016). Intelligent financial fraud detection: A comprehensive review. Computers & Security, 57, 47-66.
15. Pourhabibi, T., Ong, S. H., Ismail, R., & Lai, K. K. (2020). Fraud detection: A systematic literature review of graph-based anomaly detection approaches. Decision Support Systems, 133, 113303.
16. Chen, C. H., & Hu, Y. H. (2013). The detection and prevention of financial statement fraud: A study of banks. Journal of Forecasting, 32(4), 368-382.

17. Bauder, R. A., & Khoshgoftaar, T. M. (2018). The detection of credit card fraud using transaction aggregation strategy and machine learning methods. 2018 IEEE International Conference on Information Reuse and Integration (IRI), 191-198.
18. Iyer, R. M., & Parthiban, L. (2020). Credit card fraud detection using machine learning algorithms. International Journal of Engineering and Advanced Technology (IJEAT), 9(4), 1065-1068.
19. Patil, R. B., & Joshi, M. A. (2014). Survey on fraud detection techniques in healthcare. 2014 International Conference on Computational Intelligence and Computing Research, 1-5.
20. Jurgovsky, J., Granitzer, G., Ziegler, K., Calabretto, S., Portier, P. E., He-Guelton, L., & Caelen, O. (2018). Sequence classification for credit-card fraud detection. Expert Systems with Applications, 100, 234-245.

APPENDIX-A
PSUEDOCODE

BackEnd Code:

```
from sklearn.utils import resample
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix,
silhouette_score
from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.cluster import KMeans

import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
# Load the dataset
file_path = r'DATASET\DATASET.csv'
data = pd.read_csv(file_path)
# data's row, column counts
data.shape
data.info()
# retrive first 5 rows
data.head()

# retrive last 5 rows
data.tail()
# Check for missing values
missing_values = data.isnull().sum()
print("Missing values in each column:\n", missing_values)
# Drop rows with missing values (if any)
data = data.dropna()
# Plot distributions of numerical features
numerical_features = ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
'newbalanceDest']

plt.figure(figsize=(15, 10))
for i, feature in enumerate(numerical_features):
    plt.subplot(2, 3, i + 1)
    sns.histplot(data[feature], bins=50, kde=True)
    plt.title(f'Distribution of {feature}')

plt.tight_layout()
plt.show()
import matplotlib.pyplot as plt
import seaborn as sns

# Plot distribution of categorical features
plt.figure(figsize=(15, 5))



# Transaction type
plt.subplot(1, 2, 1)
sns.countplot(x='type', data=data)
plt.title('Transaction Type')
# Fraudulent vs Non-Fraudulent Transactions
plt.subplot(1, 2, 2)
sns.countplot(x='isFraud', data=data)
plt.title('Fraudulent vs Non-Fraudulent Transactions')

plt.tight_layout()
plt.show()
# Correlation matrix
plt.figure(figsize=(6, 4))
correlation_matrix = data[numerical_features].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix of Numerical Features')
plt.show()
# Boxplots for numerical features with respect to isFraud
plt.figure(figsize=(15, 10))
for i, feature in enumerate(numerical_features):
    plt.subplot(2, 3, i + 1)
    sns.boxplot(x='isFraud', y=feature, data=data)
    plt.title(f'{feature} by Fraudulent vs Non-Fraudulent')

plt.tight_layout()
plt.show()

# Separate the majority and minority classes
data_majority = data[data.isFraud == 0]
data_minority = data[data.isFraud == 1]
# Undersample the majority class
data_majority_undersampled = resample(data_majority,
                              replace=False,     # sample without replacement
n_samples=len(data_minority),
# to match minority class
                              random_state=42)   # reproducible results

# Combine the undersampled majority class with the minority class
data_undersampled = pd.concat([data_majority_undersampled, data_minority])
# Value counts for target column
data_undersampled["isFraud"].value_counts()
# Fraudulent vs Non-Fraudulent Transactions
plt.subplot(1, 2, 2)
sns.countplot(x='isFraud', data=data_undersampled)
plt.title('Fraudulent vs Non-Fraudulent Transactions')
plt.tight_layout()
plt.show()
# row, colun counts for undersampled data
data_undersampled.shape
# read the data_undersampled
data_undersampled
# drop unneccessary columns from data_undersampled
data_undersampled.drop(columns=["step", "nameOrig", "nameDest", "isFlaggedFraud"],
inplace=True)
# retrive first 5 rows
data_undersampled.head()
# Correlation matrix
plt.figure(figsize=(6, 4))
correlation_matrix = data_undersampled[numerical_features].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix of Numerical Features')
plt.show()
# Boxplots for numerical features with respect to isFraud
numerical_features = ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
'newbalanceDest']
plt.figure(figsize=(9, 6))
for i, feature in enumerate(numerical_features):
    plt.subplot(2, 3, i + 1)
    sns.boxplot(x='isFraud', y=feature, data=data_undersampled)
    plt.title(f'{feature} by Fraudulent vs Non-Fraudulent')

plt.tight_layout()
plt.show()
# information about data_undersampled
data_undersampled.info()
# value counts for type column (object column)
```

```
data_undersampled["type"].value_counts()
from sklearn.model_selection import GridSearchCV

# Hyperparameter tuning for K-Means using GridSearchCV
param_grid_kmeans = {
    'n_clusters': [2, 3, 4, 5, 6, 7, 8, 9, 10],  # range of clusters
    'init': ['k-means++', 'random'],
    'n_init': [10, 20, 30],  # Number of time the k-means algorithm will be run with different
centroid seeds
    'max_iter': [300, 600, 900]  # Maximum number of iterations of the k-means algorithm for
a single run
}

# Custom scoring function for silhouette score
def silhouette_scorer(estimator, X):
    clusters = estimator.fit_predict(X)
    score = silhouette_score(X, clusters)
    return score

# GridSearchCV for K-Means
grid_kmeans = GridSearchCV(KMeans(random_state=42), param_grid_kmeans,
scoring=silhouette_scorer, cv=3)
grid_kmeans.fit(X_train_under)

# Best parameters
print("Best parameters for K-Means Clustering:", grid_kmeans.best_params_)

# Predicting the clusters using the best model
best_kmeans = grid_kmeans.best_estimator_
clusters = best_kmeans.predict(X_test_under)

# Evaluating the clustering
sil_score = silhouette_score(X_test_under, clusters)
print(f'Silhouette Score for K-Means Clustering: {sil_score}')
# Plotting the clusters
plt.figure(figsize=(5, 3))
plt.scatter(X_test_under[:, 0], X_test_under[:, 1], c=clusters, cmap='viridis')
plt.title('K-Means Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
from sklearn.tree import DecisionTreeClassifier

# Initialize and train the decision tree classifier
```

```
dt_clf = DecisionTreeClassifier(random_state=42)
dt_clf.fit(X_train_under, y_train_under)

# Predict on the test data
y_pred_dt = dt_clf.predict(X_test_under)

# Evaluate the model
accuracy = accuracy_score(y_test_under, y_pred_dt)
conf_matrix = confusion_matrix(y_test_under, y_pred_dt)
class_report = classification_report(y_test_under, y_pred_dt)

print(f'Classification Report for Decision Tree:\n{class_report}')
print(f'Accuracy for Decision Tree model: {accuracy}')

# Plot confusion matrix
cm = confusion_matrix(y_test_under, y_pred_dt)
sns.heatmap(cm, annot=True, cmap="coolwarm", fmt='d')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
# Logistic Regression with Hyperparameter Tuning

from sklearn.linear_model import LogisticRegression

# Initialize the Logistic Regression
log_clf = LogisticRegression()

# Define the parameter grid
param_grid_log = {
    'C': [0.01, 0.1, 1, 10, 100],
    'penalty': ['l1', 'l2'],
    'solver': ['liblinear']  # 'liblinear' solver supports both 'l1' and 'l2' penalties
}

# Perform grid search with cross-validation
grid_log_clf = GridSearchCV(log_clf, param_grid_log, cv=3, scoring='accuracy')
grid_log_clf.fit(X_train_under, y_train_under)

# Make predictions on the test set
y_pred_log = grid_log_clf.predict(X_test_under)

# Print best parameters
print("Best parameters for Logistic Regression:", grid_log_clf.best_params_)
```

```
# Print performance metrics
print("Logistic Regression Performance with Hyperparameter Tuning:")
print(classification_report(y_test_under, y_pred_log))
print("Logistic Regression Accuracy:", accuracy_score(y_test_under, y_pred_log))

# Plot confusion matrix
cm = confusion_matrix(y_test_under, y_pred_log)
sns.heatmap(cm, annot=True, cmap="coolwarm", fmt='d')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
# Random Forest with Hyperparameter Tuning

rf_clf = RandomForestClassifier()
param_grid_rf = {
    'n_estimators': [50, 100],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

grid_rf_clf = GridSearchCV(rf_clf, param_grid_rf, cv=3, scoring='accuracy')
grid_rf_clf.fit(X_train_under, y_train_under)
y_pred_rf = grid_rf_clf.predict(X_test_under)

print("Best parameters for Random Forest:", grid_rf_clf.best_params_)
print("Random Forest Performance with Hyperparameter Tuning:")
print(classification_report(y_test_under, y_pred_rf))
print("Random Forest Accuracy:", accuracy_score(y_test_under, y_pred_rf))

# Plot confusion matrix
cm = confusion_matrix(y_test_under, y_pred_log)
sns.heatmap(cm, annot=True, cmap="coolwarm", fmt='d')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
# GaussianNB with Hyperparameter Tuning

from sklearn.naive_bayes import GaussianNB

# Define the Naive Bayes model
```

```
nb_clf = GaussianNB()

# Define the parameter grid
param_grid = {
    'var_smoothing': [1e-9, 1e-8, 1e-7, 1e-6, 1e-5]
}

# Initialize GridSearchCV
grid_nb_clf = GridSearchCV(estimator=nb_clf, param_grid=param_grid, cv=3,
scoring='accuracy')

# Fit the model
grid_nb_clf.fit(X_train_under, y_train_under)

# Get the best parameters and model
best_params_nb = grid_nb_clf.best_params_
best_nb_model = grid_nb_clf.best_estimator_

# Make predictions
y_pred_nb = best_nb_model.predict(X_test_under)

# Evaluate the model
print("Best parameters for Naive Bayes:", best_params_nb)
print("Gaussian Naive Bayes Performance with Hyperparameter Tuning:")
print(classification_report(y_test_under, y_pred_nb))
print("Gaussian Naive Bayes Accuracy:", accuracy_score(y_test_under, y_pred_nb))

# Plot confusion matrix
cm = confusion_matrix(y_test_under, y_pred_nb)
sns.heatmap(cm, annot=True, cmap="coolwarm", fmt='d')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import roc_curve, auc

# Data for comparison plots
models = ['Decision Tree', 'Logistic Regression', 'Random Forest', 'Naive Bayes']
accuracies = [0.99, 0.90, 0.99, 0.66]
precision = [0.99, 0.91, 0.99, 0.75]
recall = [0.99, 0.90, 0.99, 0.66]
f1_scores = [0.99, 0.90, 0.99, 0.63]
```

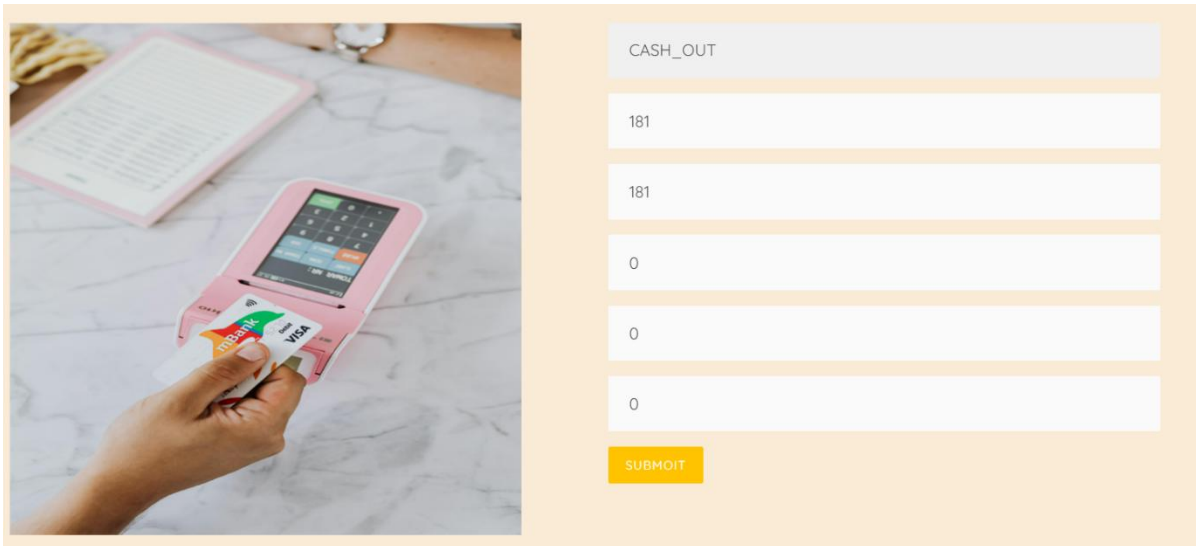
```
def complarison_plot(name, color, values):
    plt.figure(figsize=(50, 5))
    plt.subplot(1, 3, 1)
    plt.barh(models, values, color=color)
    plt.xlabel(name)
    plt.title(f'{name} Comparison')

complarison_plot(name="Accuracy", color="Skyblue", values=accuracies)
complarison_plot(name="Precision", color="lightgreen", values=precision)
complarison_plot(name="Recall", color="salmon", values=recall)
complarison_plot(name="F1 Scores", color="orange", values=f1_scores)
# Function to plot ROC curve
def plot_roc_curve(fpr, tpr, roc_auc, model_name):
    plt.plot(fpr, tpr, lw=2, label=f'{model_name} (area = {roc_auc:.2f})')

# Calculate ROC curves and AUC for each model
dt_clf_fpr, dt_clf_tpr, _ = roc_curve(y_test_under, dt_best_model.predict_proba(X_test_under)[:,1])
dt_clf_auc = auc(dt_clf_fpr, dt_clf_tpr)

log_clf_fpr, log_clf_tpr, _ = roc_curve(y_test_under, log_best_model.predict_proba(X_test_under)[:,1])
log_clf_auc = auc(log_clf_fpr, log_clf_tpr)

rf_fpr, rf_tpr, _ = roc_curve(y_test_under, rf_best_model.predict_proba(X_test_under)[:,1])
rf_auc = auc(rf_fpr, rf_tpr)

nb_fpr, nb_tpr, _ = roc_curve(y_test_under, nb_best_model.predict_proba(X_test_under)[:,1])
nb_auc = auc(nb_fpr, nb_tpr)

# Plot all ROC curves
plt.figure(figsize=(5, 4))
plot_roc_curve(dt_clf_fpr, dt_clf_tpr, dt_clf_auc, 'Decision Tree')
plot_roc_curve(log_clf_fpr, log_clf_tpr, log_clf_auc, 'Logistic Regression')
plot_roc_curve(rf_fpr, rf_tpr, rf_auc, 'Random Forest')
plot_roc_curve(nb_fpr, nb_tpr, nb_auc, 'Naive Bayes')
plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curves')
plt.legend(loc="lower right")
plt.show()
# Save the best models

import joblib

joblib.dump(log_best_model, 'MODELS/logistic_regression_model.joblib')
joblib.dump(rf_best_model, 'MODELS/random_forest_model.joblib')
joblib.dump(dt_best_model, 'MODELS/decision_tree_model.joblib')
joblib.dump(nb_best_model, 'MODELS/naive_bytes_model.joblib')
import joblib

# Load the scaler
scaler = joblib.load(r"MODELS\scaler.pkl")

# Load the Model
model = joblib.load(r"MODELS\logistic_regression_model.pkl")

classes = {0 : "Non Fraudulent Transaction", 1 : "Fraudulent Transaction"}
def prediction_function(inputs):
    scaled_features = scaler.transform(inputs)
    prediction = model.predict(scaled_features)
    result = classes[prediction[0]]
    print("Predicted result: ", result)
inputs = [[3, 7817.71, 53860, 46042.29, 0, 0]]
prediction_function(inputs)
inputs = [[54664, 4, 496, 494, 6949, 94]]
prediction_function(inputs)
```

FRONTEND CODE:

```
from flask import Flask, url_for, redirect, render_template, request
import mysql.connector, os, re
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
import joblib

app = Flask(__name__)
app.secret_key = 'admin'

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    port="3306",
    database='transaction_db'
)

mycursor = mydb.cursor()

def executionquery(query,values):
    mycursor.execute(query,values)
    mydb.commit()
    return

def retrivequery1(query,values):
    mycursor.execute(query,values)
    data = mycursor.fetchall()
    return data

def retrivequery2(query):
    mycursor.execute(query)
    data = mycursor.fetchall()
    return data

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/register', methods=["GET", "POST"])
def register():
    if request.method == "POST":
        email = request.form['email']
        password = request.form['password']
        c_password = request.form['c_password']
        if password == c_password:
            query = "SELECT UPPER(email) FROM users"
            email_data = retrivequery2(query)
            email_data_list = []
            for i in email_data:
                email_data_list.append(i[0])
            if email.upper() not in email_data_list:
                query = "INSERT INTO users (email, password) VALUES (%s, %s)"
                values = (email, password)
                executionquery(query, values)
                return render_template('login.html', message="Successfully Registered!")
            return render_template('register.html', message="This email ID is already exists!")
        return render_template('register.html', message="Conform password is not match!")
    return render_template('register.html')

@app.route('/login', methods=["GET", "POST"])
def login():
    if request.method == "POST":
        email = request.form['email']
        password = request.form['password']

        query = "SELECT UPPER(email) FROM users"
        email_data = retrivequery2(query)
        email_data_list = []
        for i in email_data:
            email_data_list.append(i[0])

        if email.upper() in email_data_list:
            query = "SELECT UPPER(password) FROM users WHERE email = %s"
            values = (email,)
            password__data = retrivequery1(query, values)
            if password.upper() == password__data[0][0]:
                global user_email
                user_email = email

                return render_template('home.html')
            return render_template('login.html', message= "Invalid Password!!")
        return render_template('login.html', message= "This email ID does not exist!")
    return render_template('login.html')

@app.route('/home')
def home():
    return render_template('home.html')

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/prediction', methods=["GET", "POST"])
def prediction():
    result = None
    if request.method == "POST":

        type = int(request.form['type'])
        amount = int(request.form['amount'])
        oldbalanceOrg = int(request.form['oldbalanceOrg'])
        newbalanceOrig = float(request.form['newbalanceOrig'])
        oldbalanceDest = int(request.form['oldbalanceDest'])
        newbalanceDest = int(request.form['newbalanceDest'])

        # Load the saved models
        scaler = joblib.load(r'Model\scaler.joblib')
        model = joblib.load(r'Model\random_forest_model.joblib')

        def prediction_function(inputs):
            classes = {0 : "Non Fraudulent Transaction", 1 : "Fraudulent Transaction"
            scaled_features = scaler.transform(inputs)
            prediction = model.predict(scaled_features)
            result = classes[prediction[0]]
            return result

            inputs =[[type, amount, oldbalanceOrg, newbalanceOrig, oldbalanceDest,
newbalanceDest]]

        result = prediction_function(inputs)

    return render_template('prediction.html', prediction = result)

@app.route('/result')
def result():
    return render_template('result.html')

if __name__ == '__main__':
    app.run(debug = True)
```

APPENDIX-B
SCREENSHOTS

CASE-1:


CASH_OUT

181

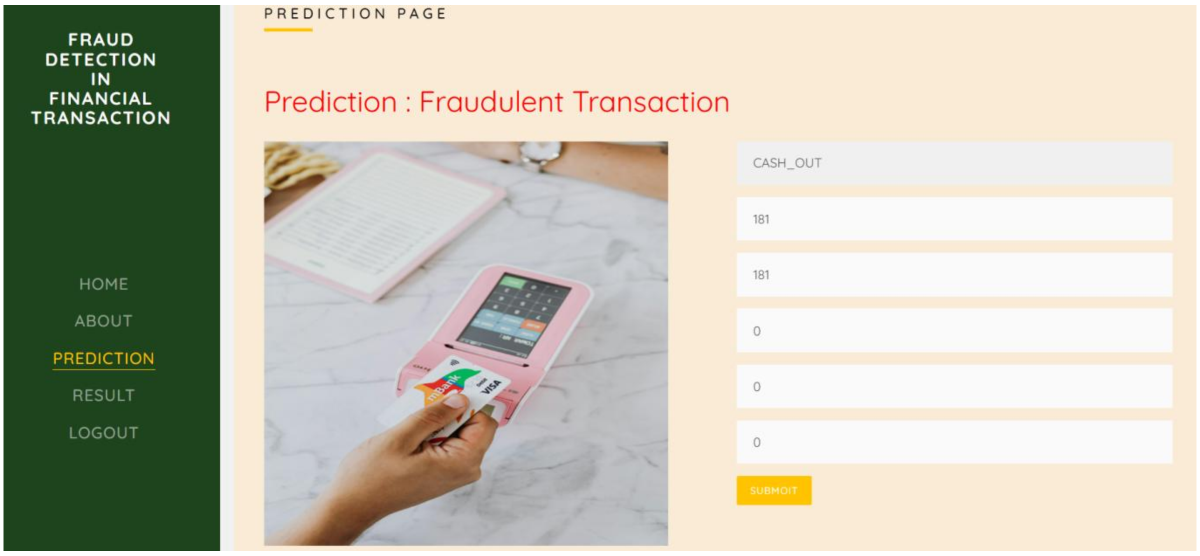
181

0

0

0

ScreenShot 1



Prediction : Fraudulent Transaction

CASH_OUT

181

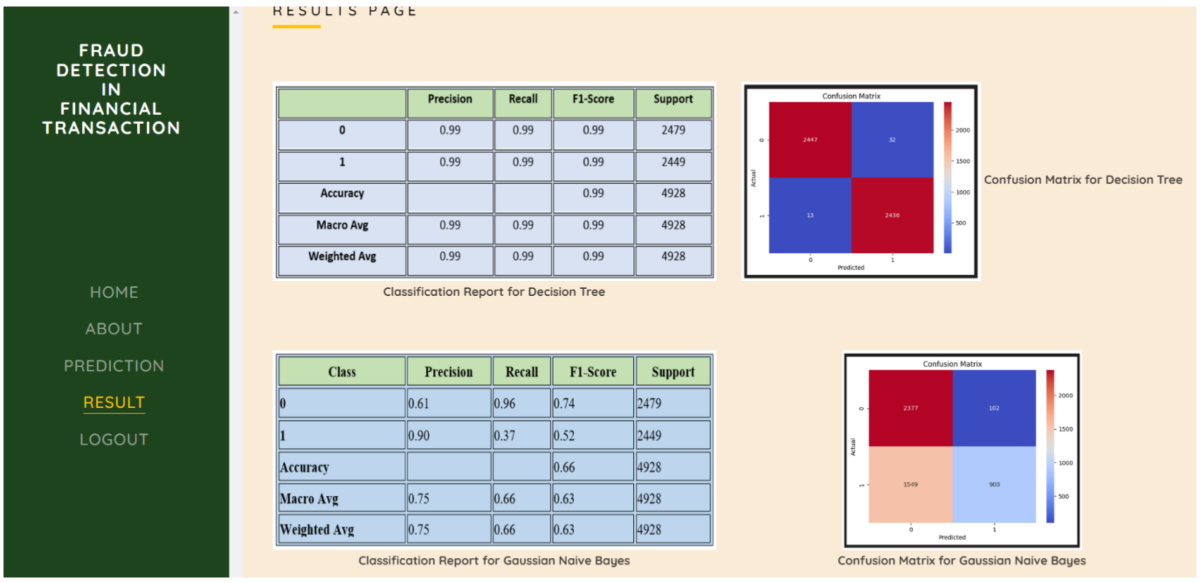
181

0

0

0

ScreenShot 2



	Precision	Recall	F1-Score	Support
0	0.99	0.99	0.99	2479
1	0.99	0.99	0.99	2449
Accuracy			0.99	4928
Macro Avg	0.99	0.99	0.99	4928
Weighted Avg	0.99	0.99	0.99	4928

Classification Report for Decision Tree

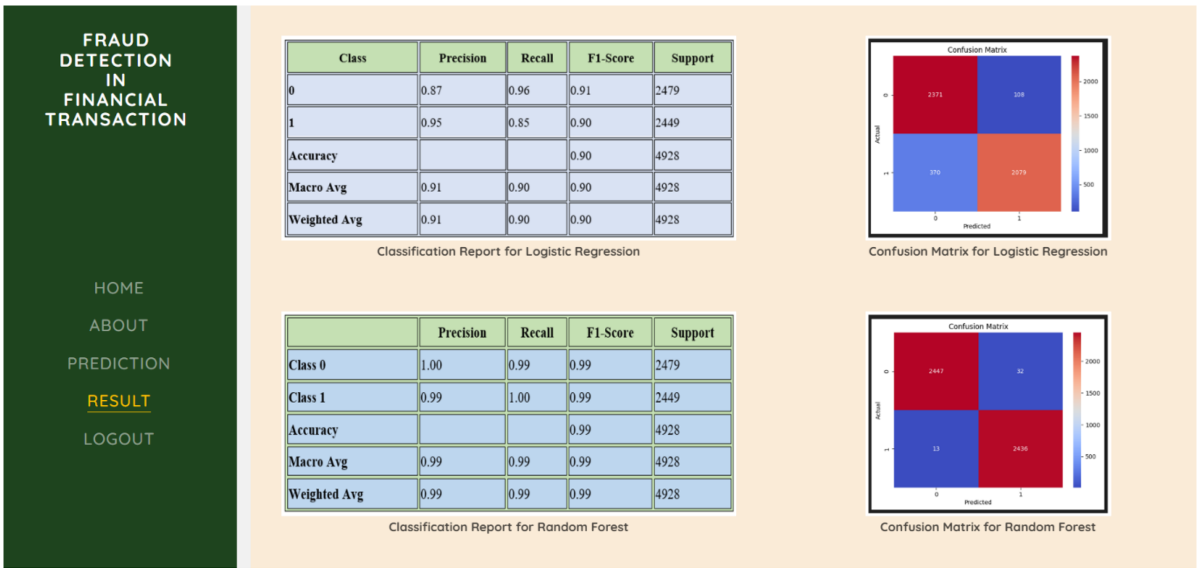
Confusion Matrix for Decision Tree

Class	Precision	Recall	F1-Score	Support
0	0.61	0.96	0.74	2479
1	0.90	0.37	0.52	2449
Accuracy			0.66	4928
Macro Avg	0.75	0.66	0.63	4928
Weighted Avg	0.75	0.66	0.63	4928

Classification Report for Gaussian Naive Bayes

Confusion Matrix for Gaussian Naive Bayes

ScreenShot 3



Class	Precision	Recall	F1-Score	Support
0	0.87	0.96	0.91	2479
1	0.95	0.85	0.90	2449
Accuracy			0.90	4928
Macro Avg	0.91	0.90	0.90	4928
Weighted Avg	0.91	0.90	0.90	4928

Classification Report for Logistic Regression

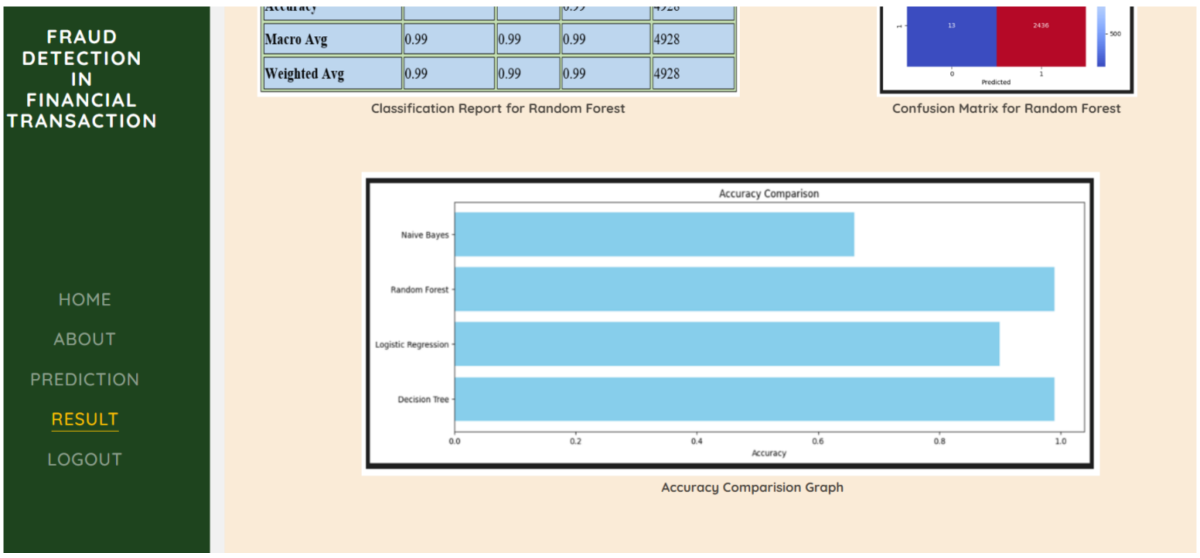
Confusion Matrix for Logistic Regression

	Precision	Recall	F1-Score	Support
Class 0	1.00	0.99	0.99	2479
Class 1	0.99	1.00	0.99	2449
Accuracy			0.99	4928
Macro Avg	0.99	0.99	0.99	4928
Weighted Avg	0.99	0.99	0.99	4928

Classification Report for Random Forest

Confusion Matrix for Random Forest

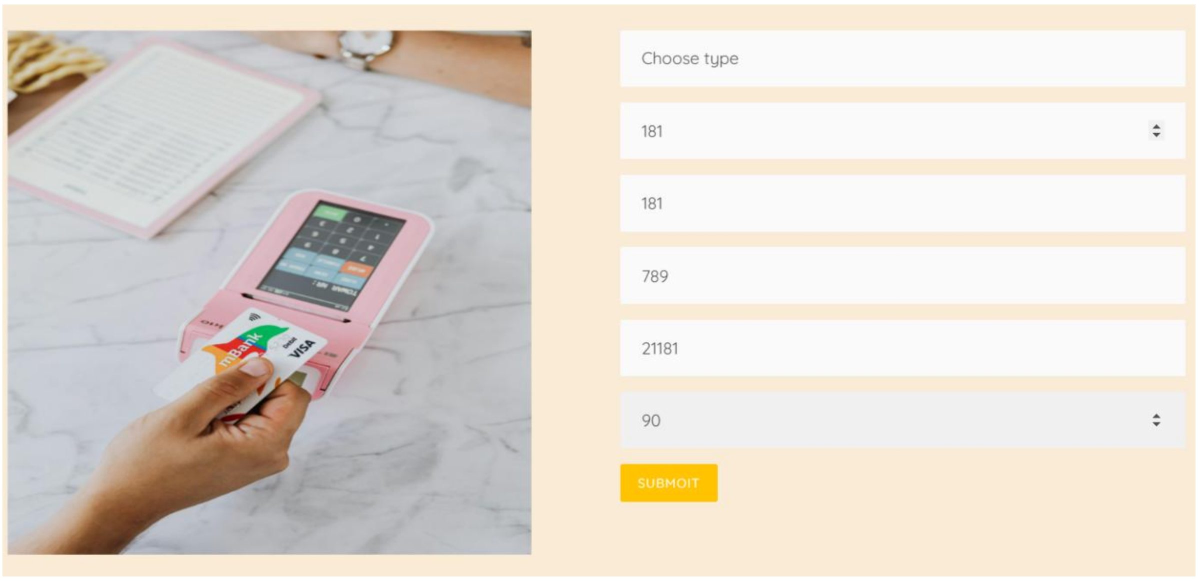
ScreenShot 4



Accuracy			0.99	4928
Macro Avg	0.99	0.99	0.99	4928
Weighted Avg	0.99	0.99	0.99	4928

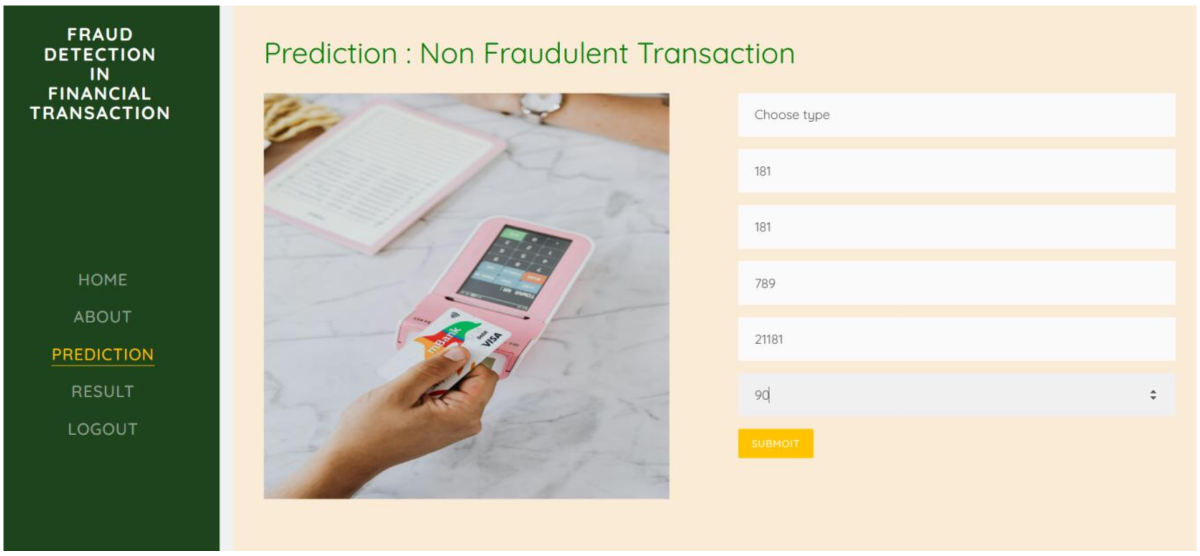
ScreenShot 5

CASE-2:



Choose type
181
181
789
21181
90

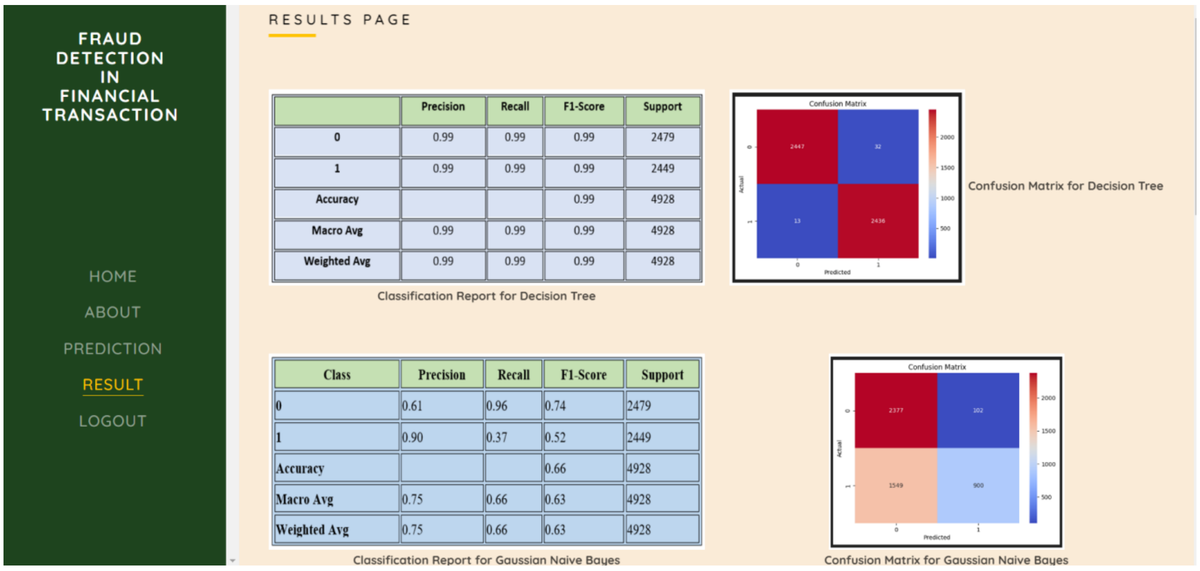
ScreenShot 6



Prediction : Non Fraudulent Transaction

Choose type
181
181
789
21181
90

ScreenShot 7



RESULTS PAGE

	Precision	Recall	F1-Score	Support
0	0.99	0.99	0.99	2479
1	0.99	0.99	0.99	2449
Accuracy			0.99	4928
Macro Avg	0.99	0.99	0.99	4928
Weighted Avg	0.99	0.99	0.99	4928

Classification Report for Decision Tree

Confusion Matrix

Actual \ Predicted	0	1
0	2447	32
1	13	2436

Confusion Matrix for Decision Tree

Class	Precision	Recall	F1-Score	Support
0	0.61	0.96	0.74	2479
1	0.90	0.37	0.52	2449
Accuracy			0.66	4928
Macro Avg	0.75	0.66	0.63	4928
Weighted Avg	0.75	0.66	0.63	4928

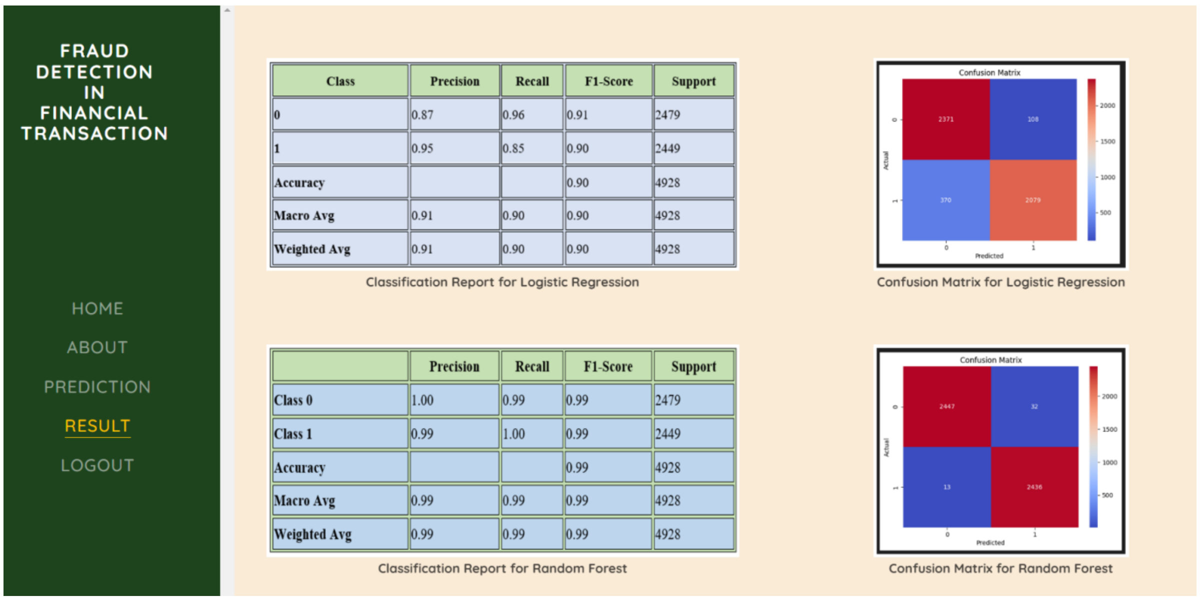
Classification Report for Gaussian Naive Bayes

Confusion Matrix

Actual \ Predicted	0	1
0	2377	102
1	1549	900

Confusion Matrix for Gaussian Naive Bayes

ScreenShot 8



Class	Precision	Recall	F1-Score	Support
0	0.87	0.96	0.91	2479
1	0.95	0.85	0.90	2449
Accuracy			0.90	4928
Macro Avg	0.91	0.90	0.90	4928
Weighted Avg	0.91	0.90	0.90	4928

Classification Report for Logistic Regression

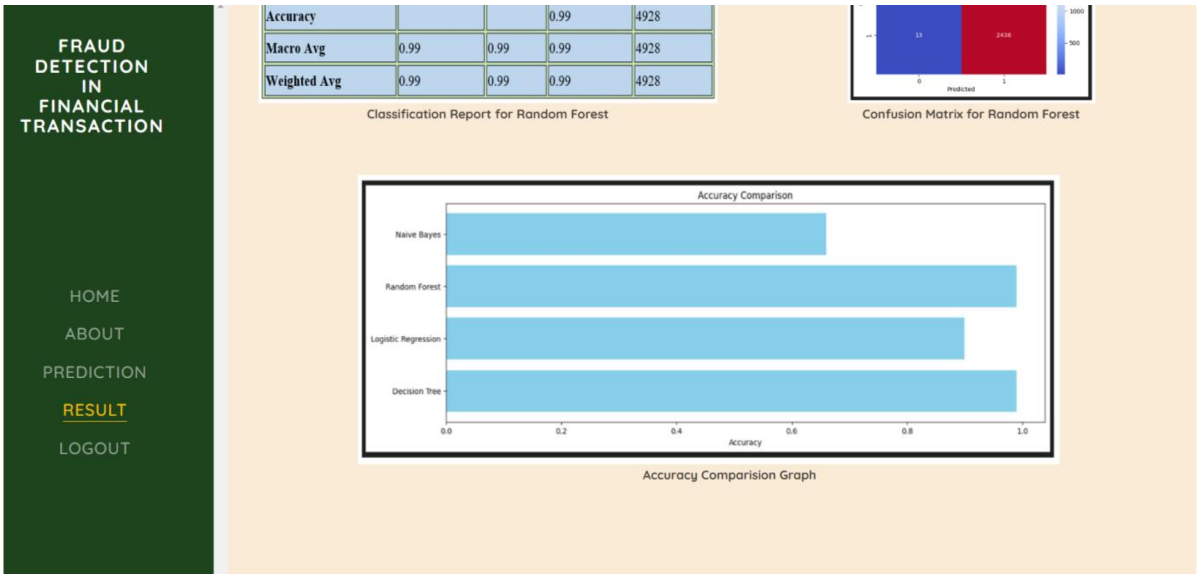
Confusion Matrix for Logistic Regression

	Precision	Recall	F1-Score	Support
Class 0	1.00	0.99	0.99	2479
Class 1	0.99	1.00	0.99	2449
Accuracy			0.99	4928
Macro Avg	0.99	0.99	0.99	4928
Weighted Avg	0.99	0.99	0.99	4928

Classification Report for Random Forest

Confusion Matrix for Random Forest

ScreenShot 9



Accuracy			0.99	4928
Macro Avg	0.99	0.99	0.99	4928
Weighted Avg	0.99	0.99	0.99	4928

Classification Report for Random Forest

Confusion Matrix for Random Forest

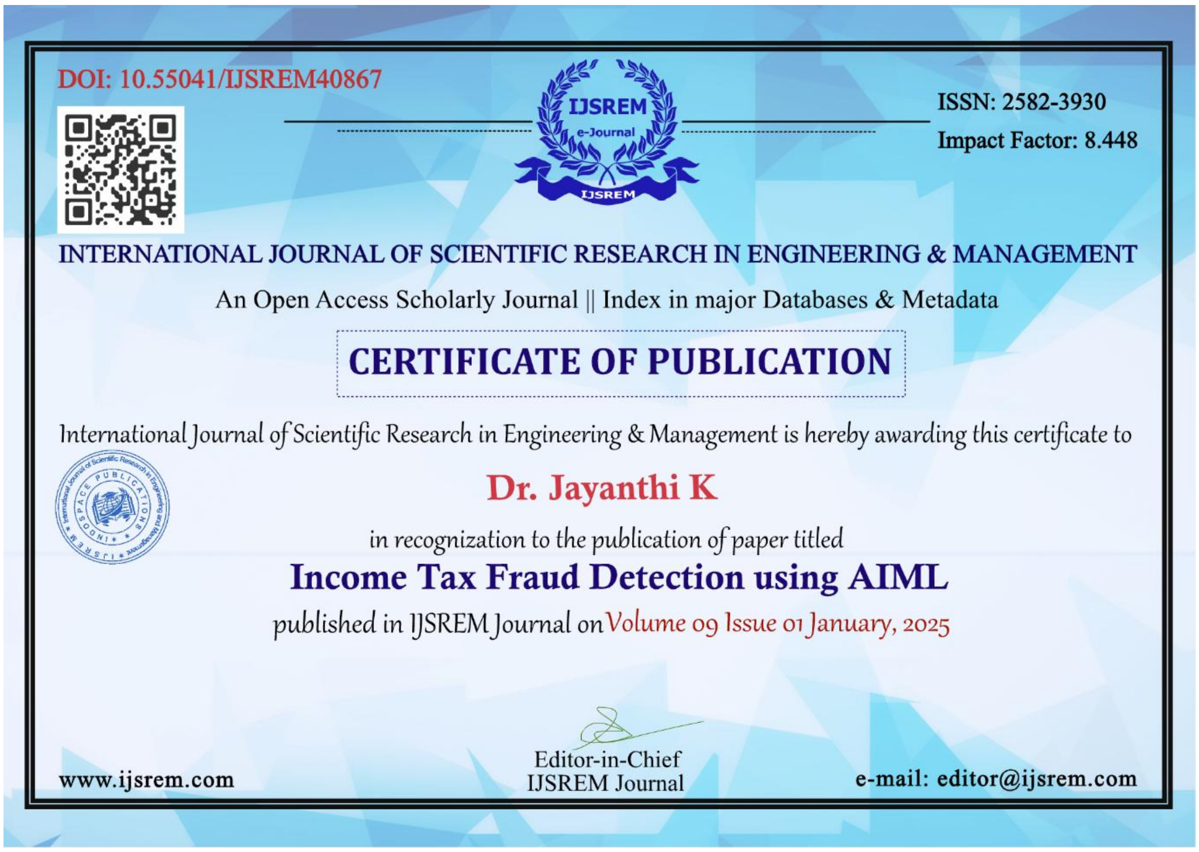
Accuracy Comparision Graph

ScreenShot 10

APPENDIX-C

ENCLOSURES

1.Journal publication/Conference Paper Presented Certificates of all students



DOI: 10.55041/IJSREM40867

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Dr. Jayanthi K

in recognization to the publication of paper titled

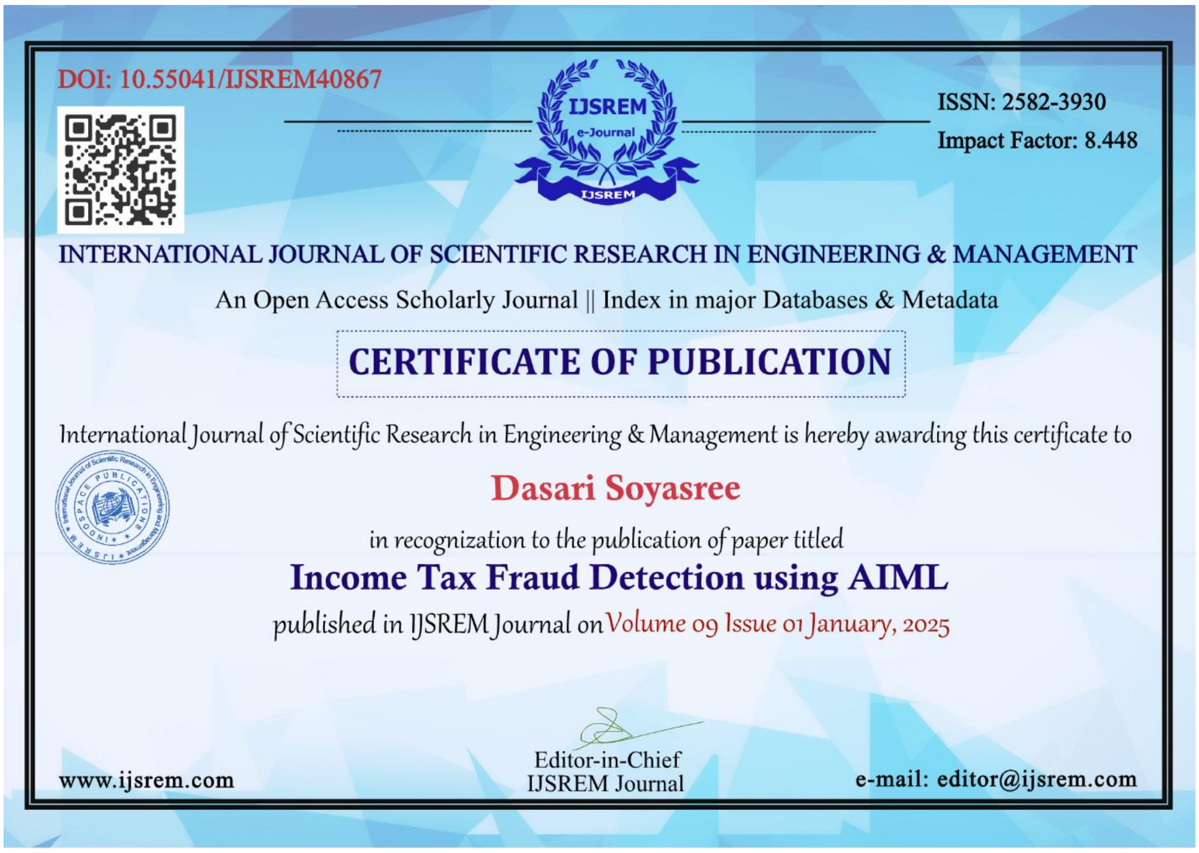
Income Tax Fraud Detection using AIML

published in IJSREM Journal on Volume 09 Issue 01 January, 2025

Editor-in-Chief
IJSREM Journal

www.ijsrem.com

e-mail: editor@ijsrem.com



DOI: 10.55041/IJSREM40867

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Dasari Soyasree

in recognization to the publication of paper titled

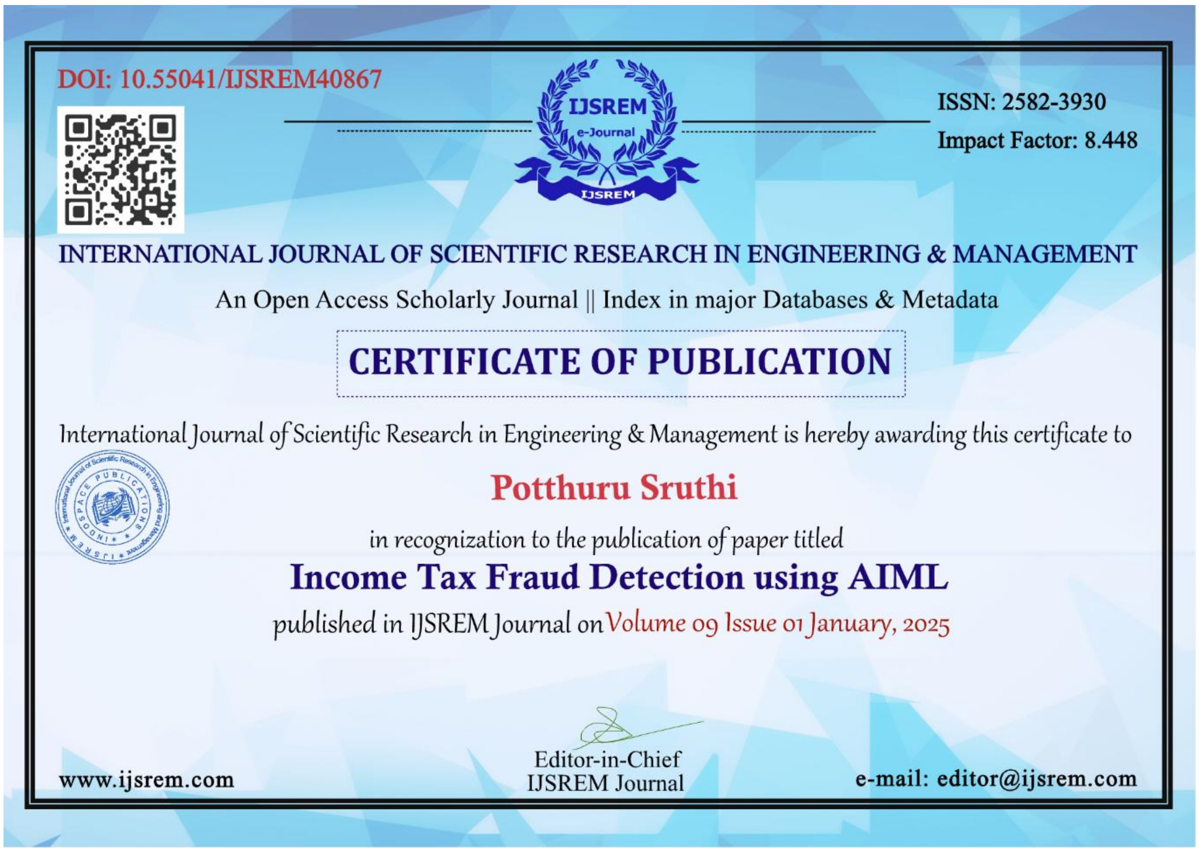
Income Tax Fraud Detection using AIML

published in IJSREM Journal on Volume 09 Issue 01 January, 2025

Editor-in-Chief
IJSREM Journal

www.ijsrem.com

e-mail: editor@ijsrem.com



DOI: 10.55041/IJSREM40867

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Potthuru Sruthi

in recognization to the publication of paper titled

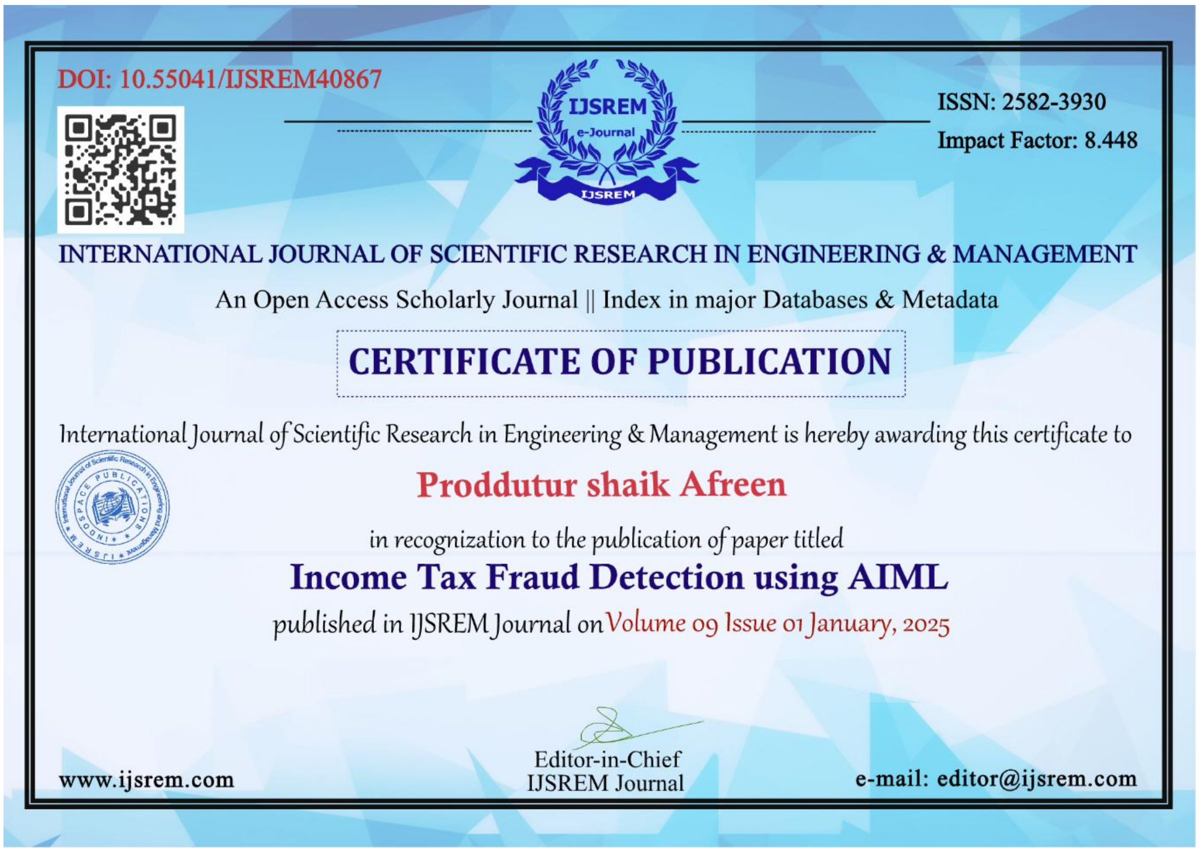
Income Tax Fraud Detection using AIML

published in IJSREM Journal on Volume 09 Issue 01 January, 2025

www.ijsrem.com

Editor-in-Chief
IJSREM Journal

e-mail: editor@ijsrem.com



DOI: 10.55041/IJSREM40867

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Proddutur shaik Afreen

in recognization to the publication of paper titled

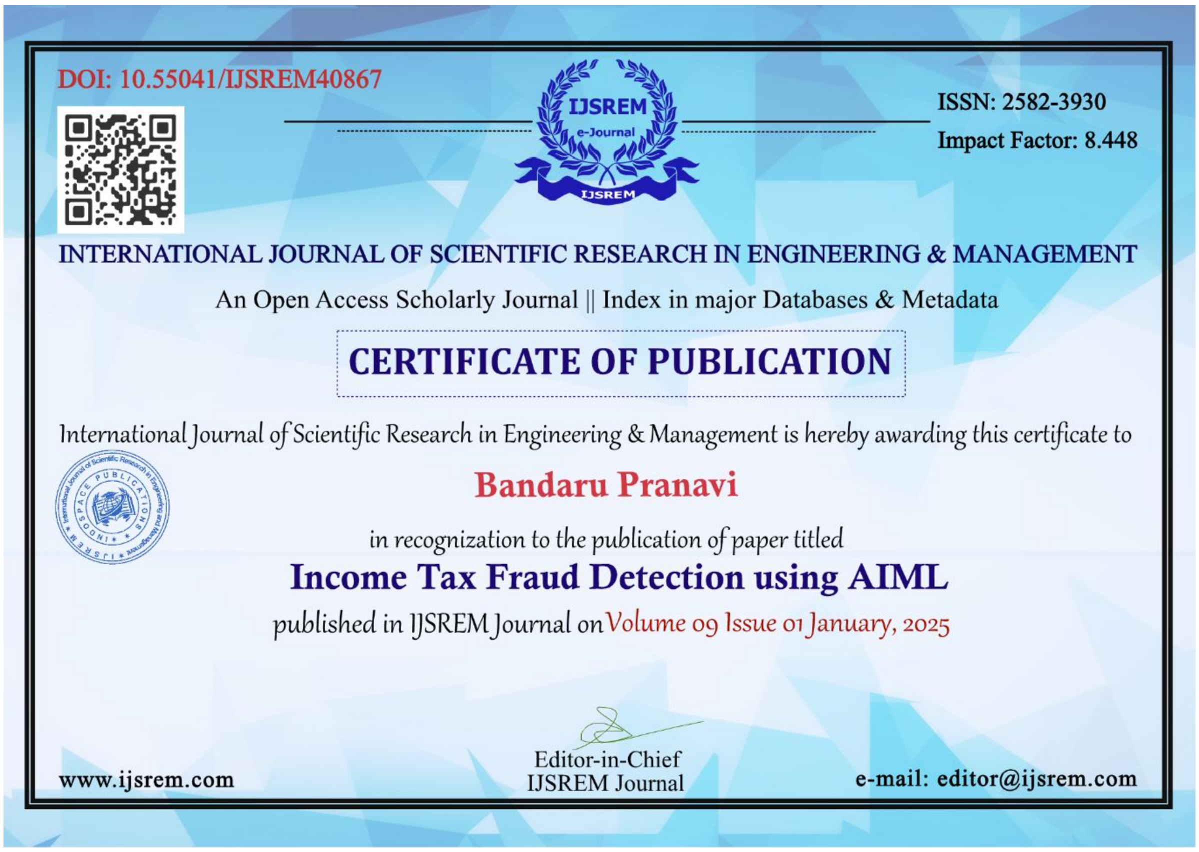
Income Tax Fraud Detection using AIML

published in IJSREM Journal on Volume 09 Issue 01 January, 2025

www.ijsrem.com

Editor-in-Chief
IJSREM Journal

e-mail: editor@ijsrem.com



DOI: 10.55041/IJSREM40867

IJSREM e-Journal IJSREM

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Bandaru Pranavi

in recognization to the publication of paper titled

Income Tax Fraud Detection using AIML

published in IJSREM Journal on Volume 09 Issue 01 January, 2025

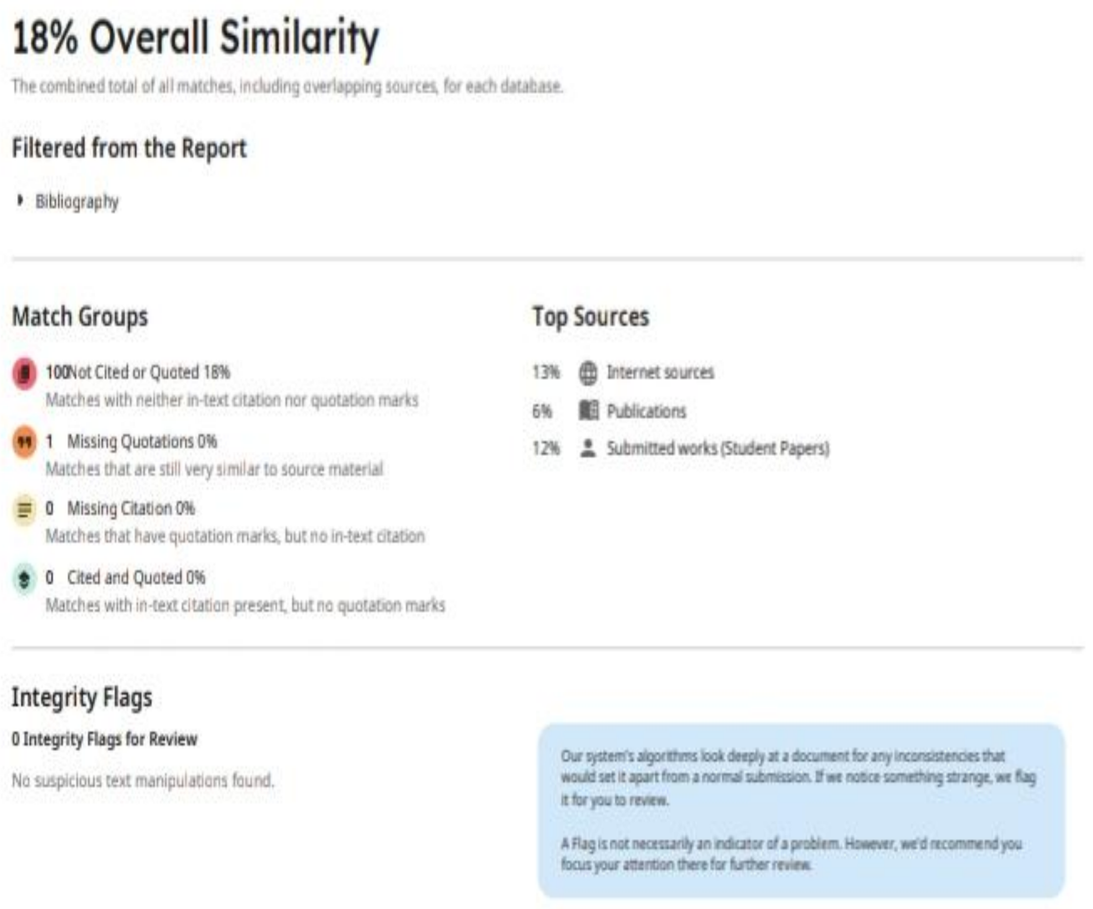
International Journal of Scientific Research in Engineering and Management INDSPACE PUBLICATIONS IJSREM

www.ijsrem.com

Editor-in-Chief
IJSREM Journal

e-mail: editor@ijsrem.com

2.Similarity Index / Plagiarism Check report clearly showing the Percentage (%). No need for a page-wise explanation.



18% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography

Match Groups

- 100 Not Cited or Quoted 18%
  Matches with neither in-text citation nor quotation marks
- 1 Missing Quotations 0%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

13%	Internet sources
6%	Publications
12%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Sustainable Development Goals (SDGs):

SUSTAINABLE DEVELOPMENT GOALS
17 GOALS TO TRANSFORM OUR WORLD
1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE, JUSTICE AND STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS
SUSTAINABLE DEVELOPMENT GOALS

This project follows SDG goal number 8 it helps us with decent work and provides economic growth .Tax fraud detection determines economic growth of a country by ensuring public with fair taxation and identifying fraudulent activities and improve trust in between organizations .It also follows SDG goal number 10 and reduce inequalities among people regarding their payments and finding out individual fraudulent transactions and prevent them .Our project helps finding out fraudulent transactions and alerting a system which perfectly helps us in improving institutions integrity and analyze financial information effectively .