الجامعة الإسلامية العالمية شيتاغونغ
International Islamic University Chittagong
Department of Computer Science and Engineering

Project On
Credit Card Fraud Detection System Using Machine Learning

Course Code: CSE-3636
Course Title: Artificial Intelligence Lab

Submitted to:

Fatema Mostafa Tarin

Adjunct Lecturer, Dept.of CSE,IIUC

Submitted by:

Team Member's Name	:	*Sadia Akther Joyti* *Israt Jahan Abida* *Tasfia Nusrat Khanam*
Team Member's ID No	:	*C223239* *C223250* *C223260*
Semester	:	*6th*
Section	:	*6BF*
Date of Submission	:	*08-08-2025*

Table of Content

Chapter-1:Introduction

1.1 Introduction

In today's increasingly digital world, online transactions and credit card usage have become a cornerstone of modern financial systems. As more individuals and businesses shift to cashless payments for convenience, speed, and accessibility, the threat of fraudulent activities has simultaneously grown more sophisticated and damaging. Credit card fraud now poses a serious challenge to the banking sector and consumers alike, resulting in substantial financial losses, legal complications, and reputational harm. Traditional fraud detection systems, which often depend on manually designed rules and threshold-based alerts, are no longer sufficient to combat the rapidly evolving nature of fraudulent techniques. These conventional systems tend to generate high false-positive rates and are typically unable to detect new or subtle patterns of fraud.

To address these challenges, this project explores the development of a fraud detection system powered by Machine Learning (ML), a subfield of artificial intelligence that enables systems to learn from historical data and improve over time. The key advantage of ML-based models lies in their ability to analyze large volumes of transaction data, detect hidden trends, and classify transactions as either legitimate or fraudulent with higher precision. This project involves acquiring a real-world dataset, understanding its structure, and applying data preprocessing techniques to prepare it for model training. Multiple machine learning algorithms are then applied, evaluated, and compared based on their performance in terms of accuracy, precision, recall, F1-score, and confusion matrix results.

By implementing a data-driven approach, the ultimate goal of this project is to build a smart and scalable fraud detection system that can be deployed in real-world banking environments. Such a system will not only help reduce financial losses but also improve customer trust and security. This project stands as a practical application of machine learning in cybersecurity and finance, demonstrating how intelligent algorithms can significantly enhance fraud detection capabilities and contribute to safer digital transactions.

1.2 Machine Learning

Machine Learning (ML) is a branch of Artificial Intelligence (AI) that allows computers to learn from past data and make predictions without being explicitly programmed. Instead of following hard-coded rules, machine learning models use algorithms to find patterns in historical data and then apply those patterns to predict future outcomes. The main goal of machine learning is to develop systems that can automatically learn and improve from experience.In the machine learning process, training data is first given to an algorithm. This data contains input features along with known outcomes or labels. The algorithm learns the relationship between the inputs and outputs during training. Once trained, the model can then make predictions on new or unseen data. This process is essential in many real-world applications such as fraud detection, spam filtering, image recognition, and recommendation systems.

Machine learning techniques are mainly divided into three categories: supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, the algorithm is trained using labeled data, meaning each input has a corresponding correct output. This is the most commonly used method in practical applications, especially when dealing with classification problems like detecting fraudulent transactions. In contrast, unsupervised learning deals with unlabeled data, and the algorithm must identify patterns and groupings without prior knowledge of outcomes. Reinforcement learning involves a system learning by interacting with its environment and receiving feedback in the form of rewards or penalties to improve its performance.Our project focuses on supervised learning, particularly classification, where the goal is to predict a specific category or label based on the input data. For example, the system decides whether a transaction is fraudulent or not. Classification models are trained using algorithms such as logistic regression, decision trees, support vector machines (SVM), and K-nearest neighbors (KNN). In your project, logistic regression is used to classify credit card transactions. The model is trained on a dataset containing both fraudulent and non-fraudulent examples, and it learns the differences between the two classes to make

accurate predictions on new data. This approach makes fraud detection systems more efficient and intelligent.

1.3 Aims and Objectives

The main aim of this project is to develop an intelligent system that can accurately detect fraudulent credit card transactions using machine learning techniques. As digital transactions are growing rapidly, it has become essential to create a system that can automatically identify suspicious patterns and prevent fraud in real time.

To achieve this aim, the project focuses on the following key objectives:

- To understand the nature and characteristics of fraudulent transactions by analyzing a real-world dataset.
- To clean, process, and prepare the dataset for effective training and testing of machine learning models.
- To apply and compare different machine learning algorithms (such as Logistic Regression, Decision Trees, Random Forest, etc.) to identify the most effective model for fraud detection.
- To evaluate the performance of these models using metrics like accuracy, precision, recall, and F1-score.
- To reduce false positives and improve the system's ability to correctly classify both fraudulent and non-fraudulent transactions.
- To explore the possibility of deploying the model in a real-time environment for practical use in banking or financial institutions.

Through these objectives, the project aims to provide a reliable, efficient, and scalable solution that can help detect fraud early and protect users from potential financial losses.

Chapter-2:Proposed System

2.1 Proposed System

The goal of the proposed system is to develop a classification model that can accurately detect whether a credit card transaction is fraudulent or not. To achieve this, a dataset containing past credit card transactions is used as the basis for training the model. The process begins with detailed data analysis, where each column in the dataset is thoroughly examined. During this phase, necessary steps are taken to handle missing values, remove outliers, and eliminate data points that do not significantly affect the outcome.

After the data is cleaned and preprocessed, it is split into two parts: one for training the model and the other for testing its performance. Various machine learning algorithms are then applied to the training data, allowing the model to learn the underlying patterns that distinguish fraudulent transactions from legitimate ones. Once the model has been trained, it is evaluated using the test data to check how well it performs on unseen cases.

Finally, the performance of different algorithms is compared using metrics such as accuracy, precision, recall, and AUC score. The algorithm that performs best in identifying fraudulent transactions is selected as the final model. This approach ensures a more accurate, efficient, and intelligent fraud detection system.

2.2 Advantages

- Accurate fraud detection

- Efficient handling of large data
- Automated learning from past data
- Comparison of multiple algorithms
- Improved performance through data preprocessing
- Scalable and extendable model
- Useful for financial institutions to reduce loss

2.3 Architecture of Proposed Model

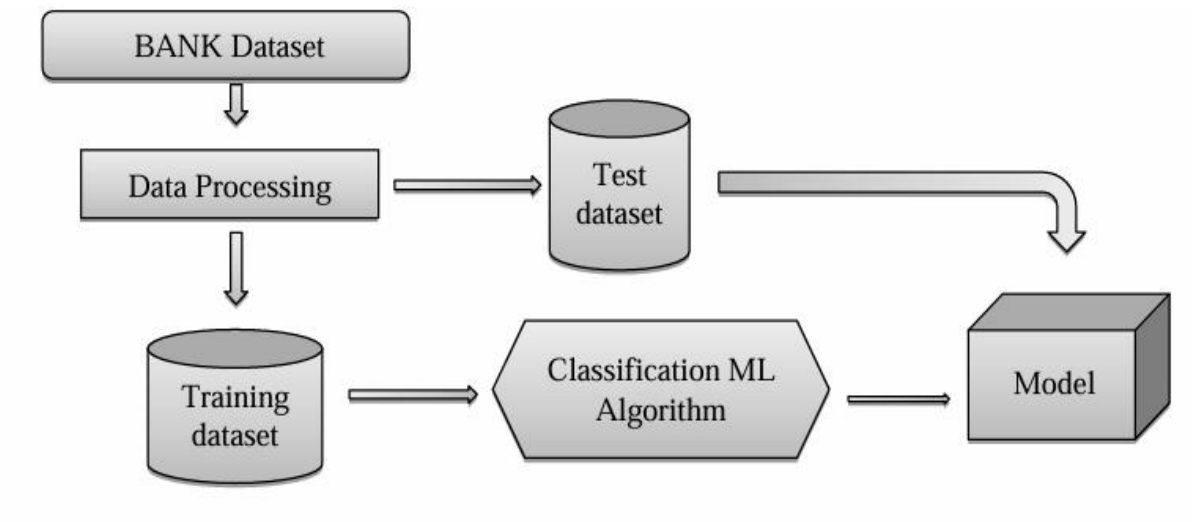


Figure -1

2.4 System Architecture

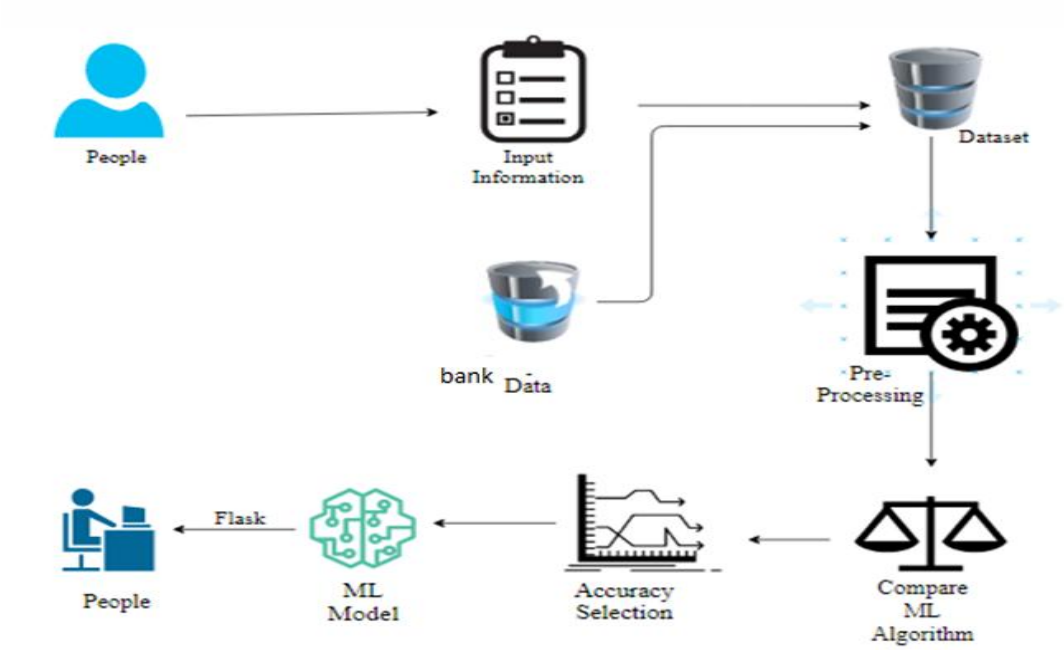


Figure-2

Chapter-3:Literature Review

Many approaches have been explored to address the challenge of credit card fraud detection. Earlier systems were largely rule-based, using fixed logic and predefined thresholds to identify suspicious activities. While these methods were simple to implement, they lacked adaptability and often produced a high number of false alerts or failed to detect evolving fraud techniques.

Machine learning techniques have brought significant improvements to fraud detection by enabling systems to learn from patterns in transactional data. Algorithms such as logistic regression, support vector machines (SVM), decision trees, and k-nearest neighbors (KNN) have been widely applied in this domain. Among these, ensemble methods like Random Forest have gained popularity for their accuracy and ability to handle complex datasets by combining the output of multiple decision trees.

In this project, we have applied Decision Tree and Random Forest classifiers to detect fraudulent transactions. Following best practices seen in existing research, we focused on preprocessing steps like handling imbalanced datasets, which is a common issue in fraud detection, as genuine transactions heavily outnumber fraudulent ones. We used accuracy score and confusion matrix to evaluate model performance, ensuring that the system can reliably differentiate between legitimate and fraudulent activity.

Research in this area suggests that effective fraud detection requires a combination of thoughtful preprocessing, balanced data representation, and the right choice of algorithms. Our work aligns with these insights, aiming to provide a reliable, machine learning-based solution for credit card fraud detection.

Chapter-4:Project Requirements

4.1 General Requirements

Requirements are the basic constraints that are essential for developing a system. These are identified during the system design phase. The following are the key types of requirements to be considered:

1. Functional Requirements
2. Non-Functional Requirements

4.2 Functional Requirements

The software requirements specification defines the necessary technical aspects of the software product. It is the initial step of requirement analysis and specifies what the software should be able to do.
For this fraud detection system, the following libraries and modules are required:

- sklearn-for machine learning algorithms
- pandas- for data manipulation
- numpy- for numerical operations
- matplotlib- for data visualization
- seaborn- for plotting graphs and correlation
- pickle- for saving and loadind(seriealizing)

4.3 Non Functional Requirements

Steps involved in the functional process of the project include:

1. Problem Definition
2. Data Preprocessing
3. Model Training and Evaluation

4. Performance Comparison
5. Prediction of Fraudulent Transactions

Chapter-5:Methodology

5.1 Methodology Overview

The methodology followed in this project focuses on building an effective machine learning-based system for detecting fraudulent credit card transactions. The overall workflow consists of several key phases: data understanding, data preprocessing, feature analysis, model training, and performance evaluation.

The process begins with importing and exploring the dataset to understand the structure, distribution, and characteristics of both fraudulent and legitimate transactions. Since fraud cases are significantly fewer than genuine ones, special attention is given to handling class imbalance using techniques such as under-sampling. Next, the data is preprocessed by selecting relevant features and separating input variables (X) from the target variable (y).

After preprocessing, two classification models Decision Tree and Random Forest are trained to detect fraudulent behavior. These models are chosen for their interpretability and strong performance on classification tasks involving structured data. The trained models are then evaluated using accuracy score and confusion matrix to assess their effectiveness in identifying fraudulent transactions while minimizing false positives.

The project uses Python programming in a Jupyter Notebook environment, leveraging popular libraries such as Pandas, NumPy, Matplotlib, Seaborn, and Scikit-learn for data handling, visualization, and machine learning implementation. This structured approach ensures that the system is built on solid data science practices while remaining practical for real-world use in fraud detection scenarios.

5.2 Methodology Diagram

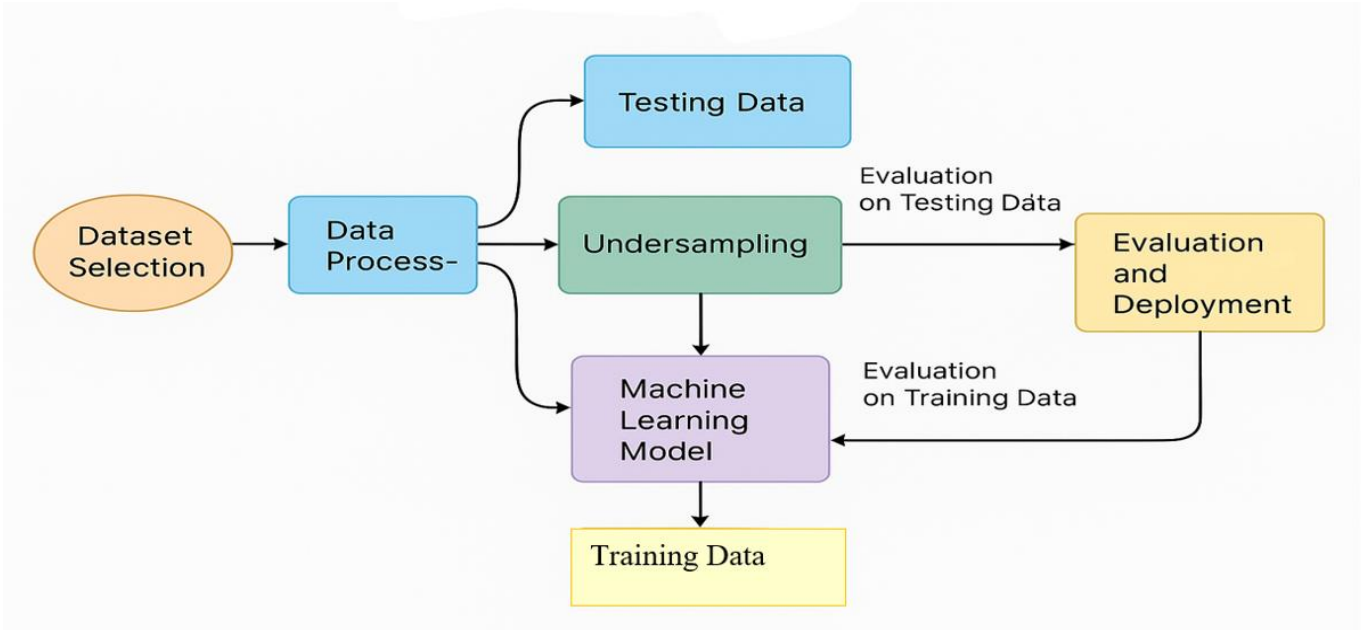


Figure-3

5.3 DFD

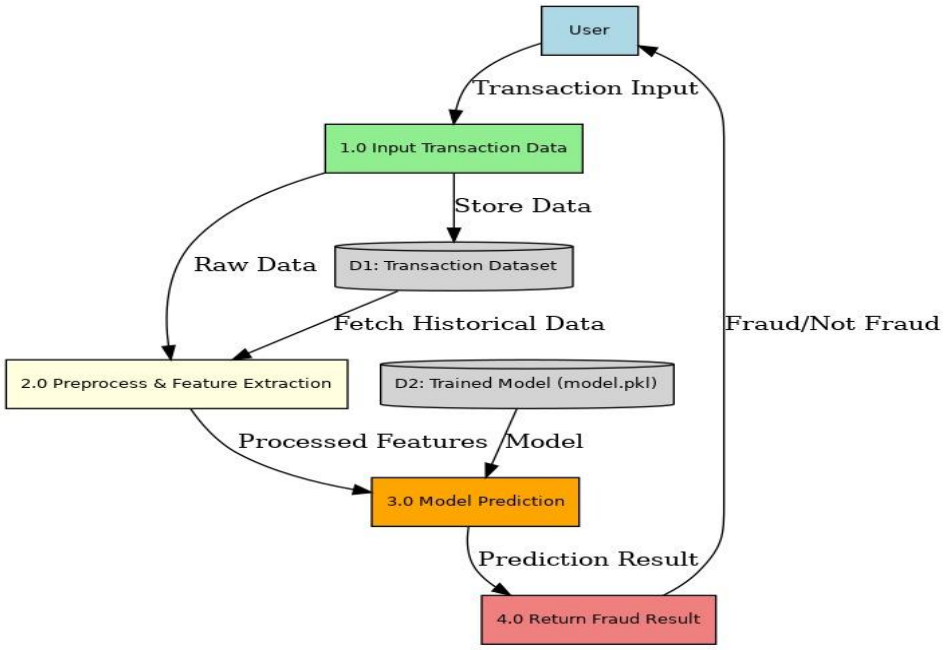


Figure-4

5.4 Use Case Diagram

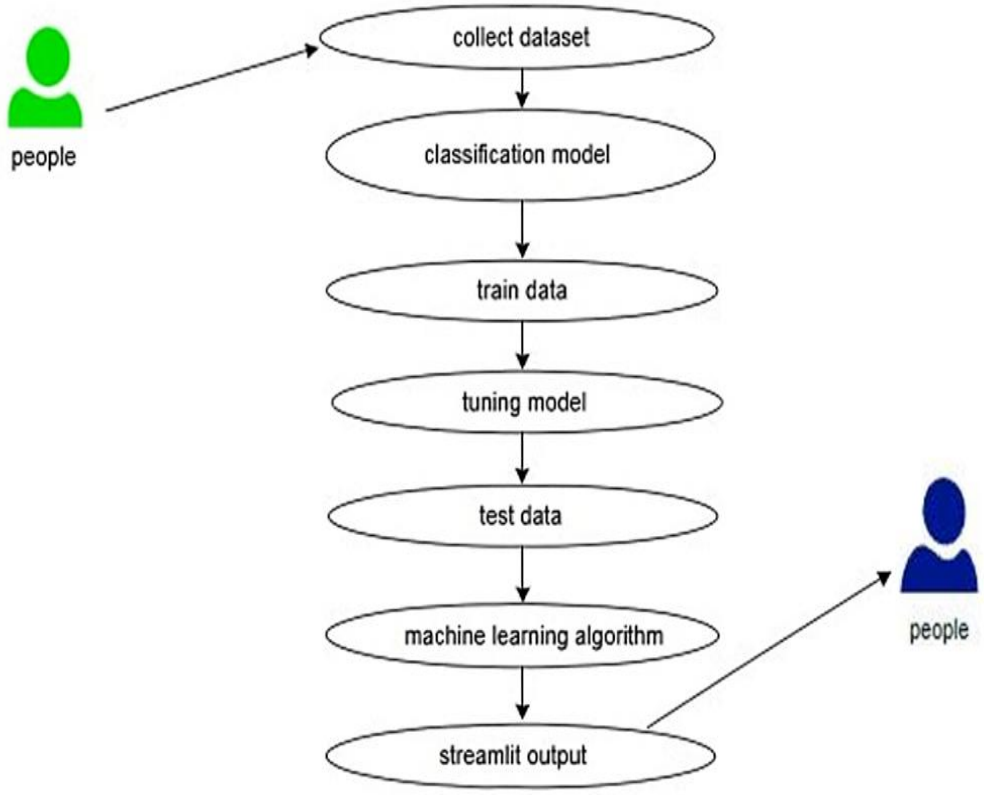


Figure-5

5.5 Data Sources

The dataset was retrieved from an open-source website, Kaggle.com. It contains data of transactions that were made in 2013 by credit card users in Europe, covering a period of only two days. The dataset consists of 31 attributes and 284,808 rows. Out of these, 28 attributes are numeric variables that have been transformed using PCA (Principal Component Analysis) to ensure customer confidentiality and privacy. The remaining three attributes are "Time", which represents the elapsed seconds between the first transaction and the rest, "Amount", which indicates the amount of each transaction, and "Class", which is a binary variable where "1" indicates a fraudulent transaction and "0" indicates a non-fraudulent (legitimate) transaction.

Dataset Link: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Chapter-6:Data Analysis and Algorithms

6.1 Data Preparation

code:
```
# dataset informations
credit_card_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
...
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

Figure 2-Dataset Structure

The figure below shows the structure of the dataset, displaying all the attributes along with their data types and a brief preview of the values within each column. The target column, "Class", is initially represented as an integer type, where 0 stands for legitimate (non-fraudulent) transactions and 1 stands for fraudulent transactions. For better clarity during analysis and visualization, the labels were interpreted accordingly as "Not Fraud" and "Fraud", helping to make the results easier to understand when building the model and generating plots.

6.2 Dataset Overview and Imbalance Class Distribution

The following code blocks are used to check for any missing values in the dataset and to analyze the distribution of the target variable “Class”, which helps understand the proportion of fraudulent and legitimate transactions before performing further analysis.

```
code: # checking the number of missing values in each column
credit_card_data.isnull().sum()
```

```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
...
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

Figure 3-Missing Values

```
code: # distribution of legit transactions & fraudulent transactions
credit_card_data['Class'].value_counts()
```

```
Class
0    284315
1       492
Name: count, dtype: int64

This Dataset is highly unblanced

0 --> Normal Transaction

1 --> fraudulent transaction
```

Figure 4-Imbalance Distribution of Legit & Fraud

6.3 Data Visualization

The following visualizations help to better understand the distribution of the dataset. The first plot displays the imbalance between fraudulent and legitimate transactions, clearly showing that the dataset is highly skewed towards non-fraudulent cases. The second histogram compares the transaction amounts for both classes, revealing that fraudulent transactions generally involve smaller amounts than legitimate ones.

```
code: #Data Visualization

plt.figure(figsize=(6,4))
sns.countplot(x='Class', data=credit_card_data)
plt.title("Class Distribution (Unbalanced)")
plt.show()

plt.figure(figsize=(8, 4))
sns.histplot(credit_card_data[credit_card_data['Class'] == 0]['Amount'], bins=50,
color='green', label='Legit', kde=True)
sns.histplot(credit_card_data[credit_card_data['Class'] == 1]['Amount'], bins=50,
color='red', label='Fraud', kde=True)
plt.legend()
plt.title("Transaction Amount Distribution")
plt.show()
```

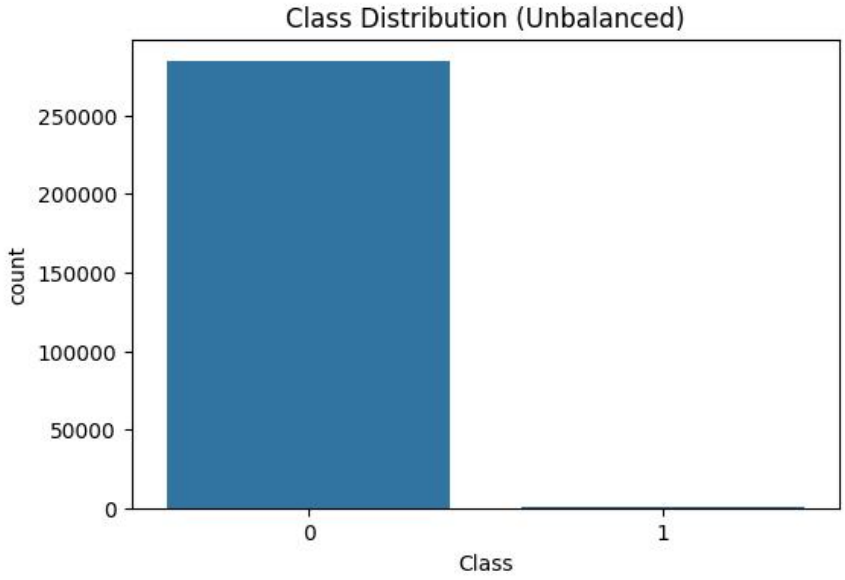


Figure 5-Class Distribution

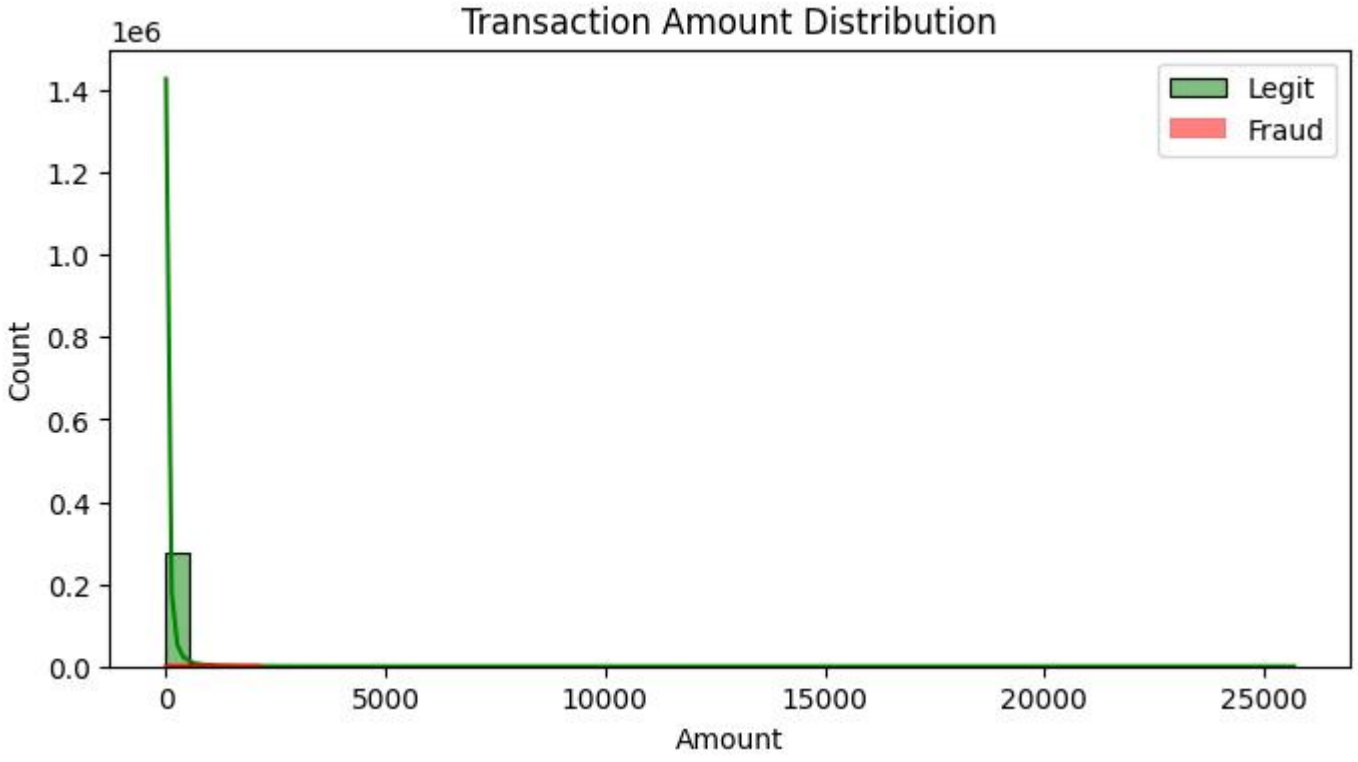


Figure 6-Sample Transaction Distribution

6.4 Balanced Dataset Class Distribution

```
code: print("\nBalanced Dataset Class Distribution:")
new_dataset['Class'].value_counts()
```

```
Balanced Dataset Class Distribution:

Class
0    492
1    492
Name: count, dtype: int64
```

Figure 7-Balanced Dataset

6.5 Algorithm Explanation

In machine learning and statistics, classification is a supervised learning technique where the algorithm learns from labeled input data and then uses that learning to classify new, unseen observations. The dataset can either be binary (such as identifying whether an email is spam or not, or whether a person is male or female) or multi-class. Examples of classification problems include speech recognition, handwriting recognition, biometric identification, and document classification.

In supervised learning, the algorithm is trained using labeled data. Once it understands the pattern of the data, it can assign labels to new, unlabeled data by recognizing similar patterns.Used Python Packages:

Sklearn
Scikit-learn (sklearn) is a powerful machine learning library in Python that provides various tools for building machine learning models. It includes modules such as train test split, DecisionTreeClassifier, LogisticRegression, and accuracy score, which are used for model training, evaluation, and accuracy measurement.

Numpy
NumPy is a numerical computation library in Python. It provides high-performance mathematical functions and array operations. It is commonly used to handle data in numerical format and is useful for mathematical and matrix operations.

Pandas
Pandas is a data manipulation and analysis library that allows easy handling of structured data through data frames. It is used for reading and writing different file formats (like CSV, Excel), and for cleaning and preprocessing data.

Matplotlib
Matplotlib is a popular data visualization library used to create static, animated, and interactive plots in Python. It helps in identifying patterns and trends in datasets through graphs and charts.

Pickle
Pickle is a Python module used for saving and loading machine learning models. It allows the serialization of trained models, which can then be reused without retraining, making deployment and reuse more efficient.

6.6 Accuracy Calculation

Accuracy is one of the most intuitive and commonly used performance metrics in machine learning classification problems. It measures the ratio of correctly predicted observations—including both true positives (TP) and true negatives (TN)—to the total number of predictions made. The formula for accuracy is:
Accuracy = (TP + TN) / (TP + TN + FP + FN)
This metric is reliable when the dataset is balanced, meaning the number of actual positive and negative cases are nearly equal. However, in situations with class imbalance (e.g., when fraud cases are much fewer than non-fraud cases), accuracy alone can be misleading as it may give a false impression of model performance.

Precision is another important metric that answers the question: Of all the transactions predicted as fraud, how many were truly fraud? It is calculated using the formula:
Precision = TP / (TP + FP)
High precision indicates that the model makes fewer false positive predictions, which is especially crucial in fraud detection where wrongly labeling a legitimate transaction as fraud could impact customer trust.Recall, also known as sensitivity or true positive rate, focuses on the model's ability to correctly detect actual fraud cases. It answers: Out of all the real fraudulent transactions, how many did the model catch? The formula for recall is:
Recall = TP / (TP + FN)
A high recall means the model successfully detects most of the frauds, minimizing false negatives, which is essential in applications where missing a fraudulent case can be costly.The F1 Score is the harmonic mean of precision and recall and provides a balanced evaluation by considering both false positives and false negatives. It is especially useful in datasets with uneven class distribution. The standard formula for F1 Score is:
F1 Score = 2 × (Precision × Recall) / (Precision + Recall)
Alternatively, in terms of TP, FP, and FN, it can also be expressed as:
F1 Score = 2TP / (2TP + FP + FN)
The F1 Score becomes more informative than accuracy when the consequences of false positives and false negatives differ significantly, making it a preferred metric for evaluating fraud detection systems.

code:
```
# accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
print("\nTraining Accuracy:", training_data_accuracy)
```

Training Accuracy: 0.9479034307496823

code:
```
# accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy score on Test Data : ', test_data_accuracy)
```

Accuracy score on Test Data : 0.9441624365482234

6.7 Logistic Regression

Logistic Regression is a statistical method used to analyze datasets in which one or more independent variables influence a binary outcome. This outcome is represented by a dichotomous variable—meaning there are only two possible results (e.g., yes or no, success or failure). The main objective of logistic regression is to find the most appropriate model that describes the relationship between a binary dependent

variable (also known as the response or outcome variable) and a set of independent (predictor or explanatory) variables.In machine learning, logistic regression functions as a classification algorithm that predicts the probability of a categorical dependent variable. Typically, this dependent variable is binary and is coded as 1 for success (yes, true) and 0 for failure (no, false). Essentially, the model calculates the probability that Y equals 1 (P(Y = 1)) based on the given input variables X.

Logistic regression relies on several assumptions. First, the dependent variable must be binary. Second, the value labeled as "1" in the dataset should represent the desired or positive outcome. Third, only variables that are meaningful and relevant should be included in the model. Additionally, the independent variables should not be highly correlated with each other—they must be statistically independent. Another important assumption is that the independent variables should have a linear relationship with the log odds of the outcome. Finally, logistic regression generally requires a relatively large sample size to yield reliable results.

```
code: # training the Logistic Regression Model with Training Data
model = LogisticRegression(max_iter=1000)
model.fit(X_train, Y_train)
```

In this code,A Logistic Regression model is trained using the training dataset, where x-train contains the input features and y_train holds the corresponding labels. The max_iter=1000 parameter ensures that the model is allowed sufficient iterations to converge during the optimization process.

6.8 Evaluation and Deployment

The performance of the model was checked using the confusion matrix and ROC curve. The confusion matrix shows how well the model predicted fraudulent and non-fraudulent transactions by comparing actual results with predicted results. It helps to understand if the model is making mistakes, like marking a safe transaction as fraud or missing a real fraud.The ROC curve helps to measure how well the model can separate fraud from non-fraud cases. A higher area under this curve means better performance.Although the project was run and tested in a notebook environment, the trained model can be saved and used later in real-time applications. This makes it possible to apply the model to live data and detect fraud as transactions happen.

```
code: # Confusion Matrix
cm = confusion_matrix(Y_test, X_test_prediction)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

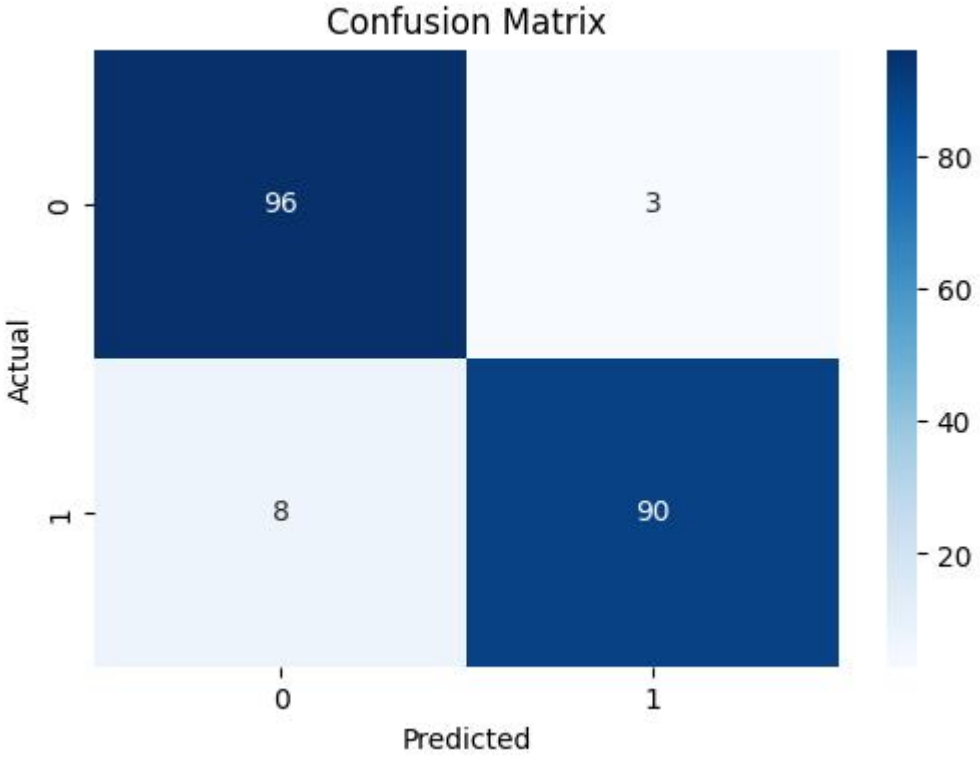


Figure 8-Predicted Value

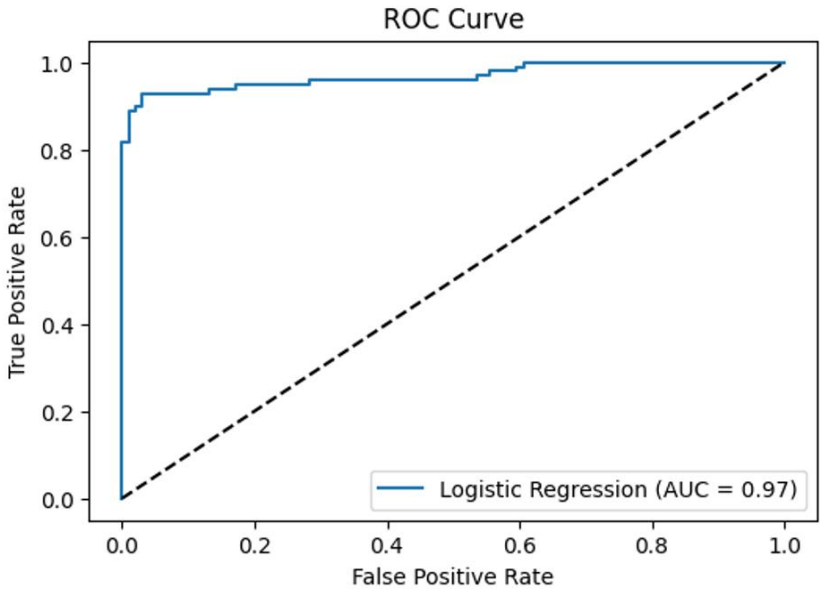


Figure 9-ROC Curve

Chapter-7:Framework

7.1 Framework Code

```
import pickle
import streamlit as st
import base64
import pandas as pd
```

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    classification_report,
    roc_curve,
    precision_recall_curve,
    auc
)

# Background Image via CSS
def add_bg_from_local(image_file):
    with open(image_file, "rb") as image:
        encoded = base64.b64encode(image.read()).decode()
    st.markdown(
        f"""
        <style>
        .stApp {{
            background-image: url("{encoded}");
            background-size: cover;
            background-position: center;
            background-repeat: no-repeat;
            background-attachment: fixed;
        }}
        </style>
        """,
        unsafe_allow_html=True
    )

# Call the function
add_bg_from_local("img1.jpg")

# Page config
st.set_page_config(page_title="Credit Card Fraud Detection", page_icon="💳", layout="wide")

st.title("💳 Smart Credit Card Fraud Detection System")
st.markdown("Upload your dataset & predict fraudulent transactions!")

uploaded_file = st.file_uploader("📂 Upload your creditcard.csv file", type="csv")

if uploaded_file is not None:
    # Load dataset
    data = pd.read_csv(uploaded_file)
    st.success(" Dataset Loaded Successfully!")

    with st.expander("Dataset Overview"):
        st.write("*Shape:*", data.shape)
        st.write("*Preview:*")
        st.write(data.head(20))

    # Split legit and fraud
    legit = data[data.Class == 0]
```

```
    fraud = data[data.Class == 1]

    # Under-sampling
    legit_sample = legit.sample(n=492, random_state=2)
    balanced_data = pd.concat([legit_sample, fraud], axis=0)

    # Features & Target
    X = balanced_data.drop(columns='Class', axis=1)
    Y = balanced_data['Class']

    # Train-test split
    X_train, X_test, Y_train, Y_test = train_test_split(
        X, Y, test_size=0.2, stratify=Y, random_state=2)

    # Load model (pickle)
    with open('model.pkl', 'rb') as f:
        model = pickle.load(f)

    # Predictions and accuracy
    y_train_pred = model.predict(X_train)
    training_data_accuracy = accuracy_score(Y_train, y_train_pred)
    y_test_pred = model.predict(X_test)
    test_data_accuracy = accuracy_score(Y_test, y_test_pred)

    # Classification Report
    report_dict = classification_report(Y_test, y_test_pred, target_names=['Legit', 'Fraud'], output_dict=True)

    # Markdown report
    report_md = """
  Classification Report

| Class  | Precision | Recall | F1-Score | Support |
|--------|-----------|--------|----------|---------|
"""
    for cls in ['Legit', 'Fraud']:
        precision = report_dict[cls]['precision']
        recall = report_dict[cls]['recall']
        f1_score = report_dict[cls]['f1-score']
        support = int(report_dict[cls]['support'])
        report_md += f"| {cls} | {precision:.2f} | {recall:.2f} | {f1_score:.2f} | {support} |\n"

    # Accuracy percentages
    training_accuracy_percent = round(training_data_accuracy * 100, 2)
    test_accuracy_percent = round(test_data_accuracy * 100, 2)

    # Show metrics
    st.markdown("## Model Evaluation")
    col1, col2 = st.columns(2)
    col1.metric("Training Accuracy", f"{training_accuracy_percent}%")
    col2.metric("Test Accuracy", f"{test_accuracy_percent}%")

    # ROC Curve probabilities
    y_prob = model.predict_proba(X_test)[:, 1]
```

```
    # Plots
    col1, col2, col3 = st.columns(3)

    with col1:
        st.write("### Confusion Matrix")
        cm = confusion_matrix(Y_test, y_test_pred)
        fig, ax = plt.subplots(figsize=(6, 4))
        sns.heatmap(
            cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=["Legit", "Fraud"], yticklabels=["Legit", "Fraud"],
            ax=ax,
            annot_kws={"size": 14}
        )
        ax.set_xlabel("Predicted", fontsize=12)
        ax.set_ylabel("Actual", fontsize=12)
        ax.tick_params(axis='both', labelsize=12)
        st.pyplot(fig)

    with col2:
        st.write("### ROC Curve")
        fpr, tpr, _ = roc_curve(Y_test, y_prob)
        roc_auc = auc(fpr, tpr)
        fig2, ax2 = plt.subplots(figsize=(4, 3))
        ax2.plot(fpr, tpr, color='darkorange', label=f"AUC = {roc_auc:.2f}")
        ax2.plot([0, 1], [0, 1], color='navy', linestyle='--')
        ax2.set_xlabel('False Positive Rate', fontsize=11)
        ax2.set_ylabel('True Positive Rate', fontsize=11)
        ax2.legend(loc="lower right", fontsize=10)
        ax2.tick_params(axis='both', labelsize=11)
        st.pyplot(fig2)

    with col3:
        st.write("### Precision-Recall Curve")
        precision, recall, _ = precision_recall_curve(Y_test, y_prob)
        fig3, ax3 = plt.subplots(figsize=(4, 3))
        ax3.plot(recall, precision, marker='.')
        ax3.set_xlabel('Recall', fontsize=11)
        ax3.set_ylabel('Precision', fontsize=11)
        ax3.tick_params(axis='both', labelsize=11)
        st.pyplot(fig3)

    # Classification Report
    with st.expander("Classification Report"):
        st.markdown(report_md)

    with st.expander(" Detection Summary"):
        st.code(f"""
 Credit Card Fraud Detection Summary
- Model Used: Logistic Regression
- Accuracy Achieved: {test_accuracy_percent:.2f}%

- Techniques Applied:
  • Under-sampling for dataset balancing
  • Confusion Matrix for error visualization
```

```
• ROC Curve & Precision-Recall Curve for classifier evaluation
    """)

  st.markdown("---")
  st.header(" Predict Fraud for a New Transaction")

  example_legit = legit[(legit['Time'] != 0) & (legit['Amount'] != 0)].iloc[0].drop('Class').to_dict()
  example_fraud = fraud[(fraud['Time'] != 0) & (fraud['Amount'] != 0)].iloc[0].drop('Class').to_dict()

  col1, col2 = st.columns(2)
  if col1.button(" Auto-fill Legit Transaction"):
      st.session_state.inputs = example_legit
  if col2.button("Auto-fill Fraud Transaction"):
      st.session_state.inputs = example_fraud

  input_data = {}
  cols = st.columns(4)
  for i, col in enumerate(X.columns):
      default = 0.0
      if "inputs" in st.session_state:
          default = float(st.session_state.inputs.get(col, 0.0))
      input_val = cols[i % 4].number_input(col, value=default, format="%.6f")
      input_data[col] = input_val

  if st.button(" Predict Transaction Type"):
      input_df = pd.DataFrame([input_data])
      prediction = model.predict(input_df)[0]
      prob = model.predict_proba(input_df)[0][1]

      if prediction == 1:
          st.error(f" Fraud Detected with Probability: {prob*100:.2f}%")
      else:
          st.success(f"Legitimate Transaction with Probability: {(1-prob)*100:.2f}%")

else:
  st.info("Please upload the creditcard.csv file to begin.")
```

8.2 Screenshots of Framework

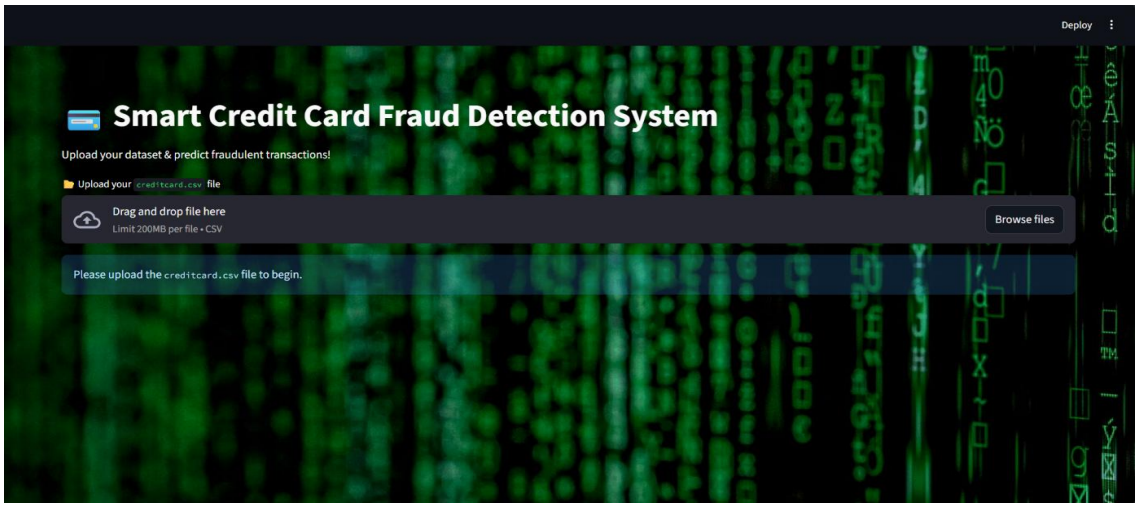


Figure-10

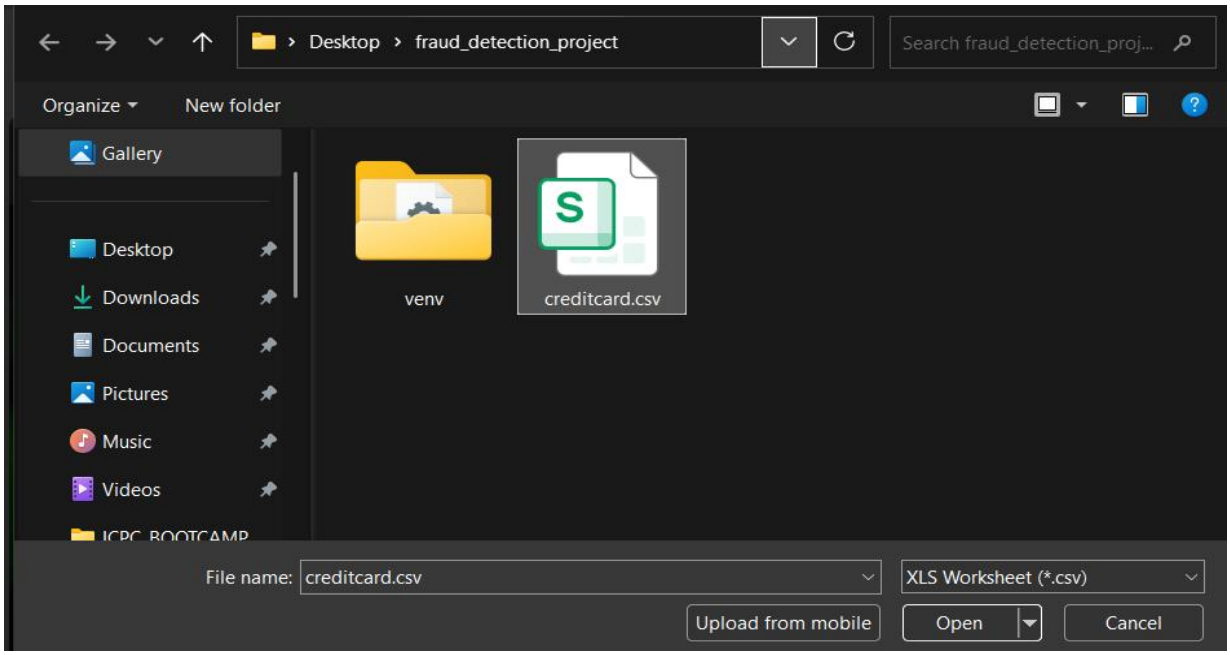
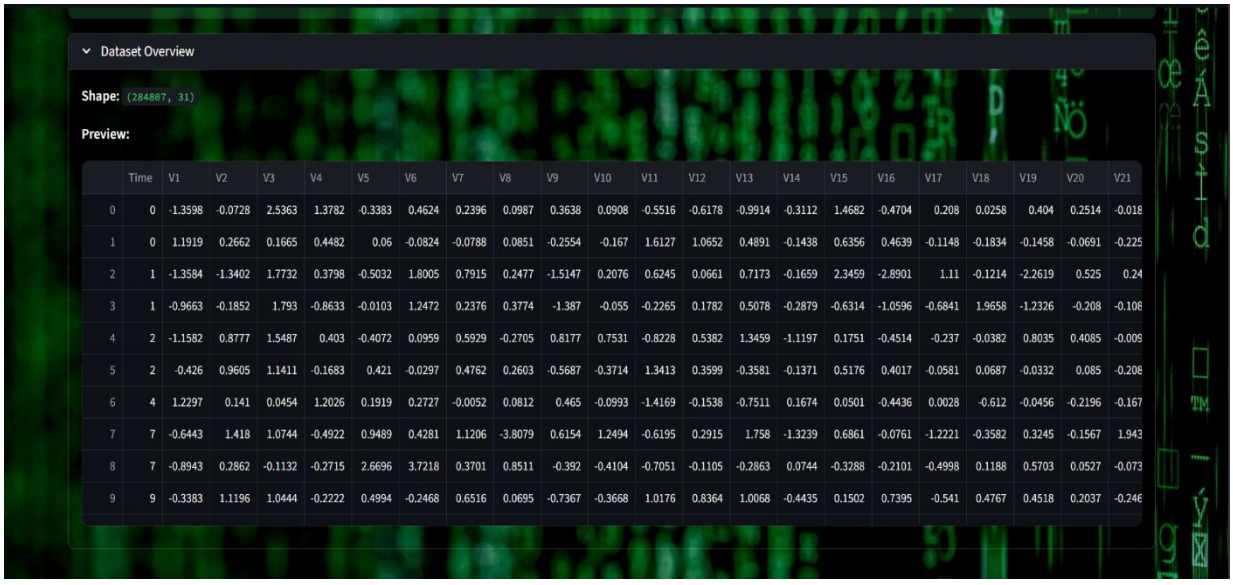


Figure-11

Figure-12



	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21
0	0	-1.3598	-0.0728	2.5363	1.3782	-0.3383	0.4624	0.2396	0.0987	0.3638	0.0908	-0.5516	-0.6178	-0.9914	-0.3112	1.4682	-0.4704	0.208	0.0258	0.404	0.2514	-0.018
1	0	1.1919	0.2662	0.1665	0.4482	0.06	-0.0824	-0.0788	0.0851	-0.2554	-0.167	1.6127	1.0652	0.4891	-0.1438	0.6356	0.4639	-0.1148	-0.1834	-0.1458	-0.0691	-0.225
2	1	-1.3584	-1.3402	1.7732	0.3798	-0.5032	1.8005	0.7915	0.2477	-1.5147	0.2076	0.6245	0.0661	0.7173	-0.1659	2.3459	-2.8901	1.11	-0.1214	-2.2619	0.525	0.24
3	1	-0.9663	-0.1852	1.793	-0.8633	-0.0103	1.2472	0.2376	0.3774	-1.387	-0.055	-0.2265	0.1782	0.5078	-0.2879	-0.6314	-1.0596	-0.6841	1.9658	-1.2326	-0.208	-0.108
4	2	-1.1582	0.8777	1.5487	0.403	-0.4072	0.0959	0.5929	-0.2705	0.8177	0.7531	-0.8228	0.5382	1.3459	-1.1197	0.1751	-0.4514	-0.237	-0.0382	0.8035	0.4085	-0.009
5	2	-0.426	0.9605	1.1411	-0.1683	0.421	-0.0297	0.4762	0.2603	-0.5687	-0.3714	1.3413	0.3599	-0.3581	-0.1371	0.5176	0.4017	-0.0581	0.0687	-0.0332	0.085	-0.208
6	4	1.2297	0.141	0.0454	1.2026	0.1919	0.2727	-0.0052	0.0812	0.465	-0.0993	-1.4169	-0.1538	-0.7511	0.1674	0.0501	-0.4436	0.0028	-0.612	-0.0456	-0.2196	-0.167
7	7	-0.6443	1.418	1.0744	-0.4922	0.9489	0.4281	1.1206	-3.8079	0.6154	1.2494	-0.6195	0.2915	1.758	-1.3239	0.6861	-0.0761	-1.2221	-0.3582	0.3245	-0.1567	1.943
8	7	-0.8943	0.2862	-0.1132	-0.2715	2.6696	3.7218	0.3701	0.8511	-0.392	-0.4104	-0.7051	-0.1105	-0.2863	0.0744	-0.3288	-0.2101	-0.4998	0.1188	0.5703	0.0527	-0.073
9	9	-0.3383	1.1196	1.0444	-0.2222	0.4994	-0.2468	0.6516	0.0695	-0.7367	-0.3668	1.0176	0.8364	1.0068	-0.4435	0.1502	0.7395	-0.541	0.4767	0.4518	0.2037	-0.246

Figure-13

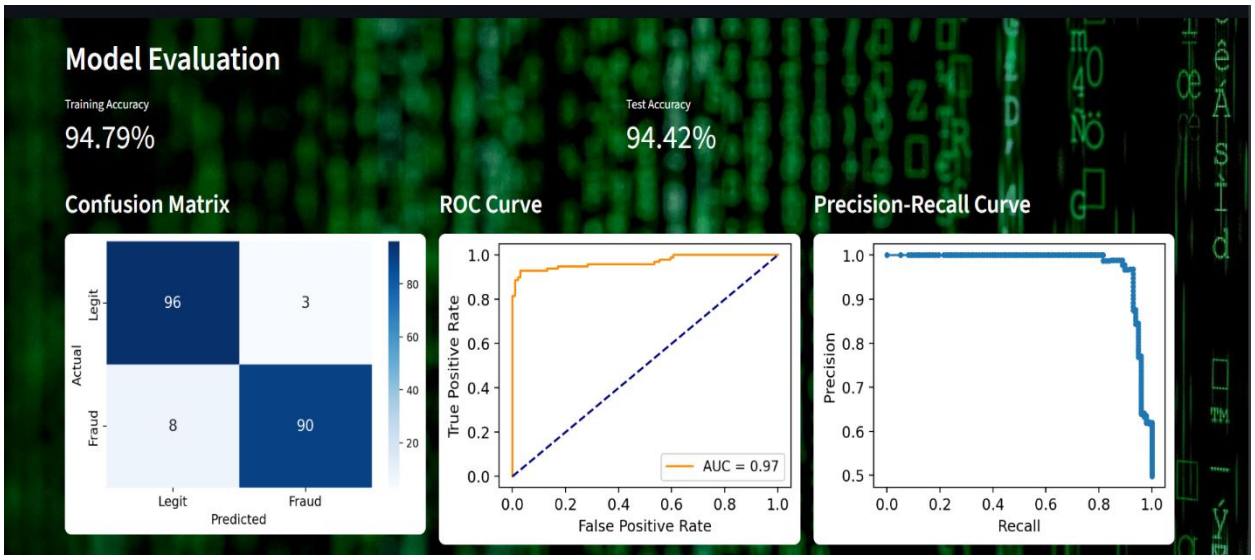


Figure-14

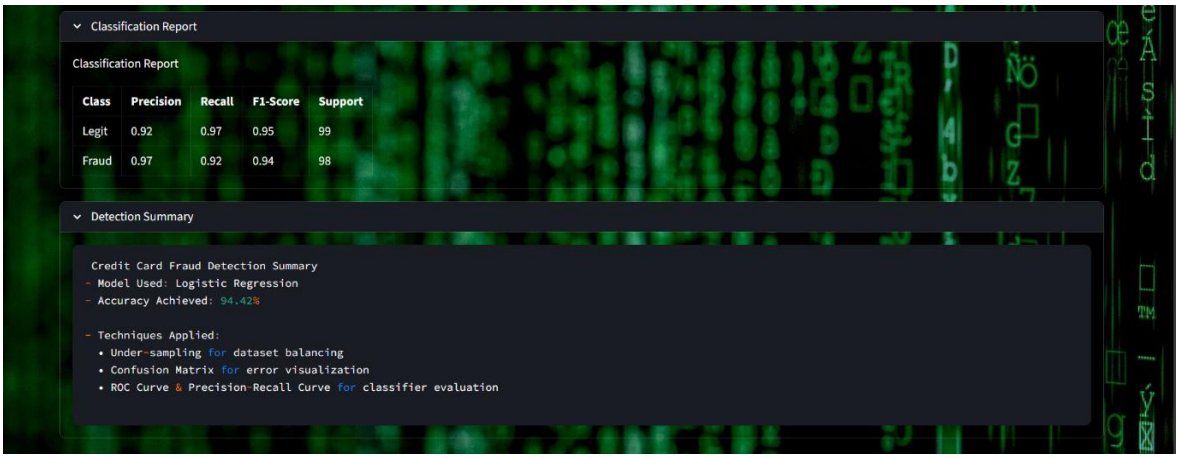


Figure-15

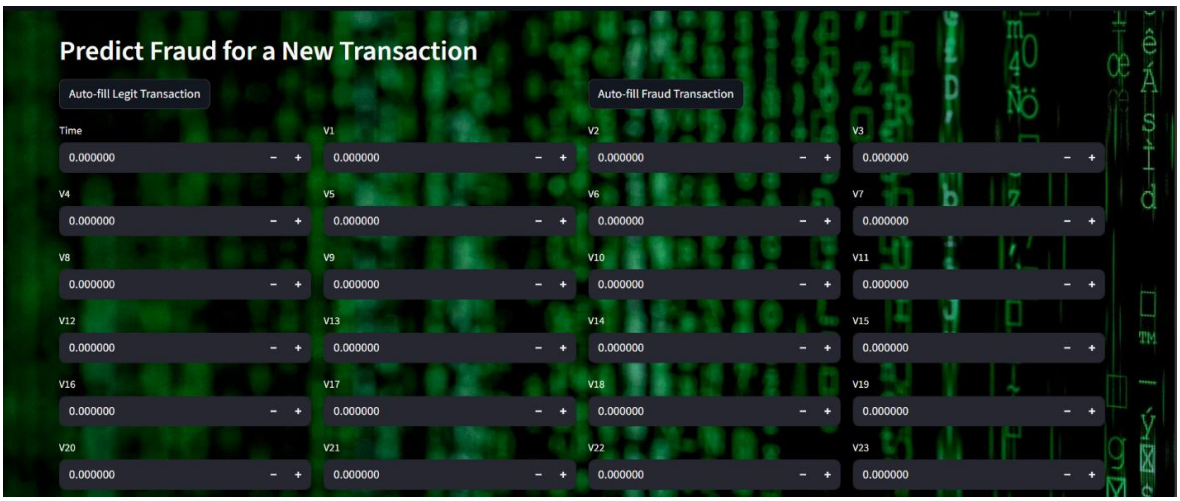


Figure-16

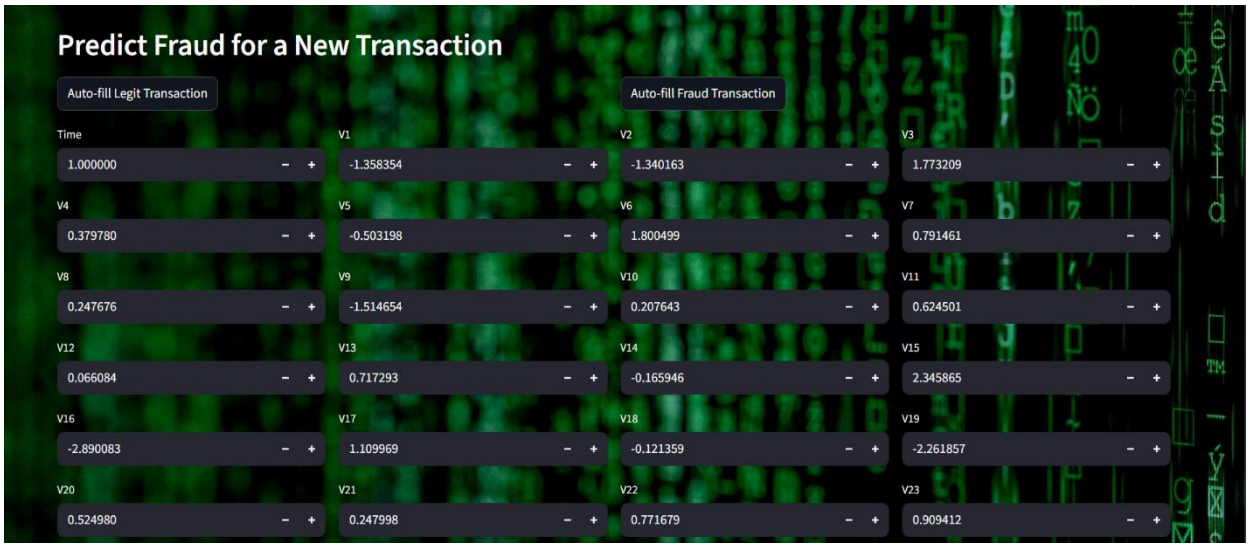


Figure-17

Figure-18

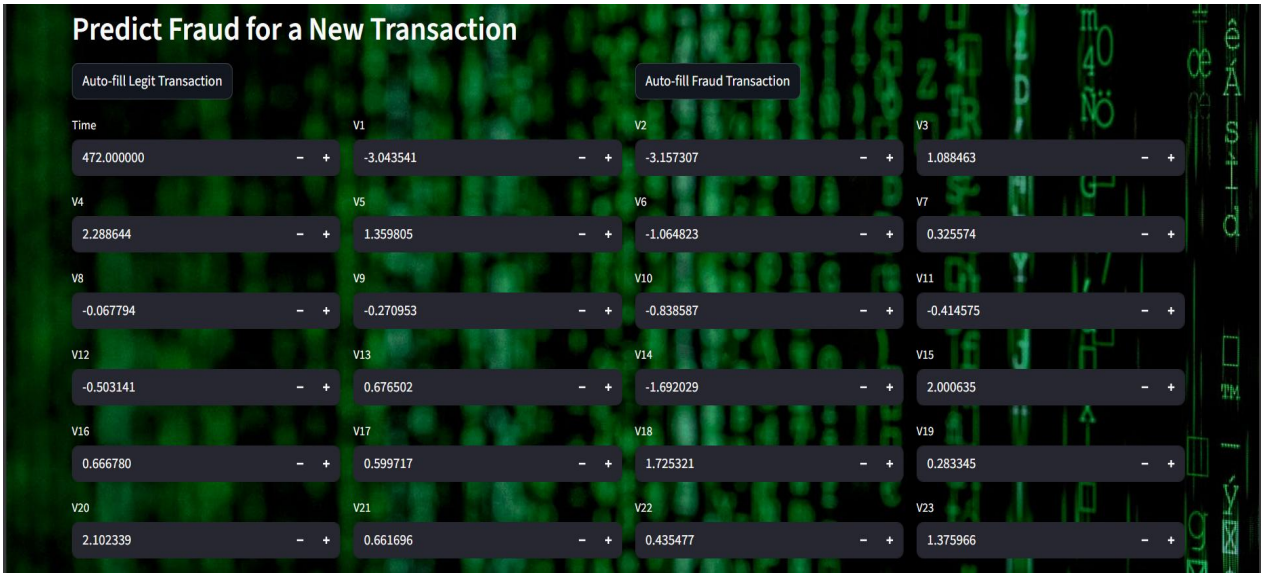


Figure-19

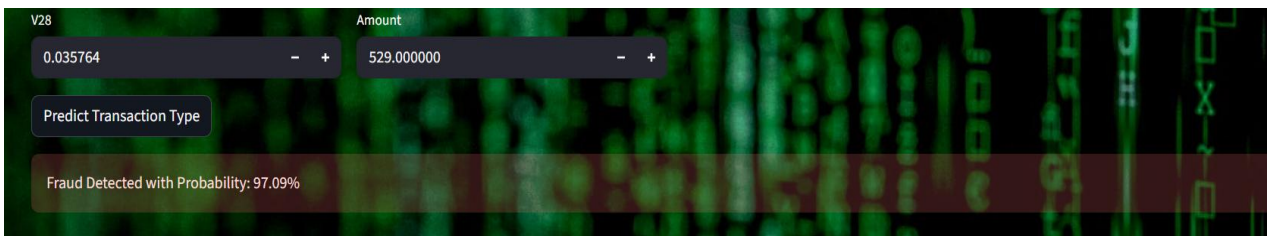


Figure-20

Chapter-8:Conclusion and Future Work

8.1 Conclusion

In this project, we explored the process of building an effective classification model using machine learning techniques. Starting from data collection and preprocessing, we moved through model training, tuning, testing, and finally deploying the model using Streamlit for user interaction. The goal was to predict accurate outcomes based on input data using algorithms like Logistic Regression and Decision Trees.

Throughout the development, we utilized essential Python libraries such as NumPy, Pandas, Scikit-learn, Matplotlib, and Pickle to handle data manipulation, modeling, visualization, and model serialization. The performance of the model was evaluated using metrics like accuracy, precision, recall, and F1-score, ensuring the reliability of the predictions.

This project demonstrates how machine learning can be effectively applied to real-world classification problems. It also highlights the importance of proper data handling, algorithm selection, and result interpretation. The final application provides a user-friendly interface where predictions can be made easily, making the model both practical and impactful.

In future work, the model can be further improved by incorporating more complex algorithms, handling larger and more diverse datasets, and optimizing performance with hyperparameter tuning and feature engineering.

8.2 Future Work

While the current project successfully demonstrates the implementation of a machine learning classification model with a user-friendly interface, there are several ways it can be improved and extended in the future:

1. **Incorporate More Algorithms:**
   Future versions can include additional algorithms such as Random Forest, XGBoost, or Support Vector Machines to compare performances and choose the best model for the problem.
2. **Hyperparameter Tuning:**
   Advanced techniques like Grid Search or Randomized Search can be used to fine-tune model parameters for better accuracy and performance.
3. **Larger and Diverse Datasets:**
   Applying the model to larger and more diverse datasets can help improve generalization and make the model more robust in real-world scenarios.
4. **Handling Imbalanced Data:**
   Techniques such as SMOTE (Synthetic Minority Over-sampling Technique) or class weighting can be used to deal with imbalanced datasets more effectively.
5. **Model Explainability:**
   Integrating tools like SHAP or LIME can help explain the predictions made by the model, making it easier for end users and stakeholders to trust and understand the decisions.
6. **Real-Time Data Integration:**
   The system can be extended to process and predict on real-time data streams, making it suitable for applications like fraud detection or spam filtering.
7. **Cloud Deployment:**
   Hosting the application on cloud platforms like AWS, Azure, or Heroku can improve accessibility, scalability, and allow continuous model updates.
8. **Security Enhancements:**
   Future improvements can also focus on enhancing the security of the deployed model, especially when handling sensitive or personal data.

By implementing these future enhancements, the project can evolve into a more powerful, flexible, and production-ready machine learning solution.

References:

1. https://youtu.be/4Od5_z28iIE?si=cp9lwae-BMDNgFtl
2. https://youtu.be/frM_7UMD_-A?si=4hrzBNrqqyhdRQHi
3. https://www.geeksforgeeks.org/machine-learning/understanding-logistic-regression/
4. https://www.geeksforgeeks.org/python/a-beginners-guide-to-streamlit/