Declaration

I, Dipansu Singh , a student of BACHELOR OF SCIENCE (COMPUTER SCIENCE) at Viva College, hereby declare that the project titled “Personal Loan Approval Predictor” submitted in partial fulfillment of the requirements for the degree of BACHELOR OF SCIENCE (COMPUTER SCIENCE) is my own work. This project has been carried out by me under the guidance of **Ms. Shweta Yande** and has not been copied from any source. I further declare that this project is original and has not been submitted earlier, in part or full, for the award of any other degree or diploma at any other institution. Any work that has been taken from other authors or sources has been appropriately cited and referenced.

Name : Dipansu Singh

Acknowledgement

I would like to take this opportunity to express my sincere gratitude to all those who contributed to the successful completion of my final year project, **“Personal Loan Approval Predictor”.**

First and foremost, I would like to thank my project supervisor, **Ms. Shweta Yande**, for their continuous guidance, support, and valuable insights throughout the duration of this project. Their expertise and encouragement were instrumental in helping me navigate challenges and refine my ideas. I express our sincere thanks to all our lecturers, for their constant encouragement and support throughout our course, especially for the useful suggestions given during the course of the project period.

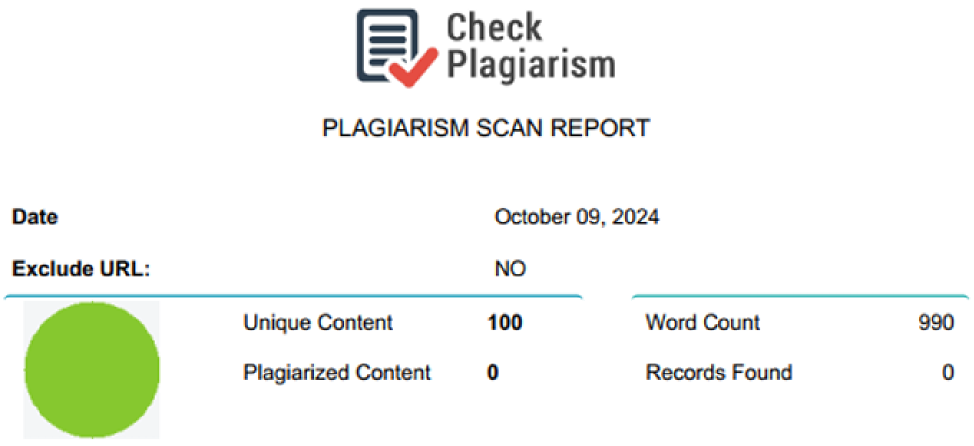
I am grateful to my internal guide **MS Shweta Yande** Lecturer, for being instrumental in the completion of our project with her complete guidance. A special thanks to my classmates and friends, who provided much-needed motivation and helped me brainstorm ideas during critical phases of the project.

Finally, I would like to extend my heartfelt appreciation to my family for their unwavering support, patience, and encouragement throughout this entire process. Their belief in me has been my greatest source of strength. Thank you all for your guidance and support in making this project a reality.

50

Plagiarism Report

To ensure the originality and integrity of the content within this document, a plagiarism analysis was conducted using the online tool **check-plagiarism**. The following report summarizes the findings and highlights any similarities detected across other sources.



Check Plagiarism

PLAGIARISM SCAN REPORT

Date	October 09, 2024
Exclude URL:	NO

Unique Content	**100**	Word Count	990
Plagiarized Content	**0**	Records Found	0

Chapter 1 : Introduction

The **Personal Loan Approval Predictor** is a machine learning-based project designed to streamline and automate the loan approval process for financial institutions. In today’s fast-paced world, the manual evaluation of loan applications is not only time-consuming but also prone to human error and bias. This project aims to address these challenges by leveraging data-driven techniques to predict whether a loan application should be approved or rejected based on various customer attributes.

The project uses a dataset containing synthetic and real-world loan application data, which includes features such as the applicant's gender, marital status, number of dependents, education level, credit card debt, employment type, and more. By analyzing these features, the system can predict the likelihood of loan approval, helping financial institutions make faster, more accurate, and consistent decisions.

The core of the project involves preprocessing the data, handling missing values, encoding categorical variables, and training multiple machine learning models to identify the best-performing algorithm. The Random Forest model emerged as the most accurate, with a prediction accuracy of 78.80%. The model is then deployed as a web application using Flask, providing a user-friendly interface for inputting customer details and receiving real-time predictions.

The **Personal Loan Approval Predictor** not only reduces the manual effort involved in loan processing but also minimizes the risk of errors and biases. It is a practical solution for financial institutions looking to modernize their operations and improve customer satisfaction. By automating the loan approval process, this project contributes to faster decision-making, increased efficiency, and better resource allocation in the financial sector.

In summary, this project demonstrates the power of machine learning in solving real-world problems and highlights the potential for data-driven solutions to transform traditional processes in the banking and finance industry

1.1 Background

The Personal Loan Approval Predictor is a machine learning-based project designed to predict whether a loan application will be approved or rejected based on various customer attributes. The project leverages a dataset containing information about applicants, such as their income, credit card debt, employment type, education, and other relevant features. The goal is to build a predictive model that can assist financial institutions in making informed decisions about loan approvals.
The project uses a combination of data preprocessing, feature engineering, and machine learning algorithms to achieve its objectives. The model is trained on a dataset containing synthetic and real-world loan application data, and it is deployed using a Flask-based web application for real-time predictions.

1.2 Objectives

- To develop a machine learning model that predicts loan approval status based on customer attributes.
- To preprocess and clean the dataset to ensure accurate predictions.
- To handle imbalanced data using oversampling techniques.
- To evaluate the performance of multiple machine learning algorithms **(Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors).**
- To deploy the model as a web application for real-time predictions.

1.3 Scope of the Project

The project focuses on predicting loan approval status for personal loans. It is designed to assist financial institutions in automating the loan approval process, reducing manual effort, and improving decision-making accuracy. The scope includes:

- Data preprocessing and feature engineering.
- Model training and evaluation.
- Deployment of the model as a web application.
- Providing a user-friendly interface for inputting customer details and receiving predictions.

Chapter 2 : Requirement Analysis

2.1 Problem Definition

The loan approval process in financial institutions is often time-consuming and prone to human error. Manual evaluation of loan applications can lead to inconsistencies and delays. This project aims to automate the loan approval process by predicting the likelihood of loan approval based on customer attributes.

2.2 Requirement Specifications

1. Minimum Software Requirements For

Development

- **Programming Languages: Python**
- **Libraries: Pandas, NumPy, Scikit-learn, Flask, Joblib, Seaborn, Matplotlib**
- **Development Environment: Jupyter Notebook, VS Code**
- **Web Framework: Flask**
- **Frontend: HTML, CSS, Bootstrap**
- **Version Control: Git**

For Deployment (If Hosted Online)

- **Server Environment**: PythonAnywhere, Heroku, or any cloud service supporting Flask
- **Web Server**: Gunicorn (for handling web requests)

2. Minimum Hardware Requirements For

Development

- **Processor**: Intel Core i3 (or AMD equivalent)
- **RAM**: 4GB (8GB recommended for smooth performance)
- **Storage**: 20GB of free disk space
- **GPU**: Not required (since basic ML models do not need GPU acceleration)

For Hosting/Deployment (If Deployed Online)

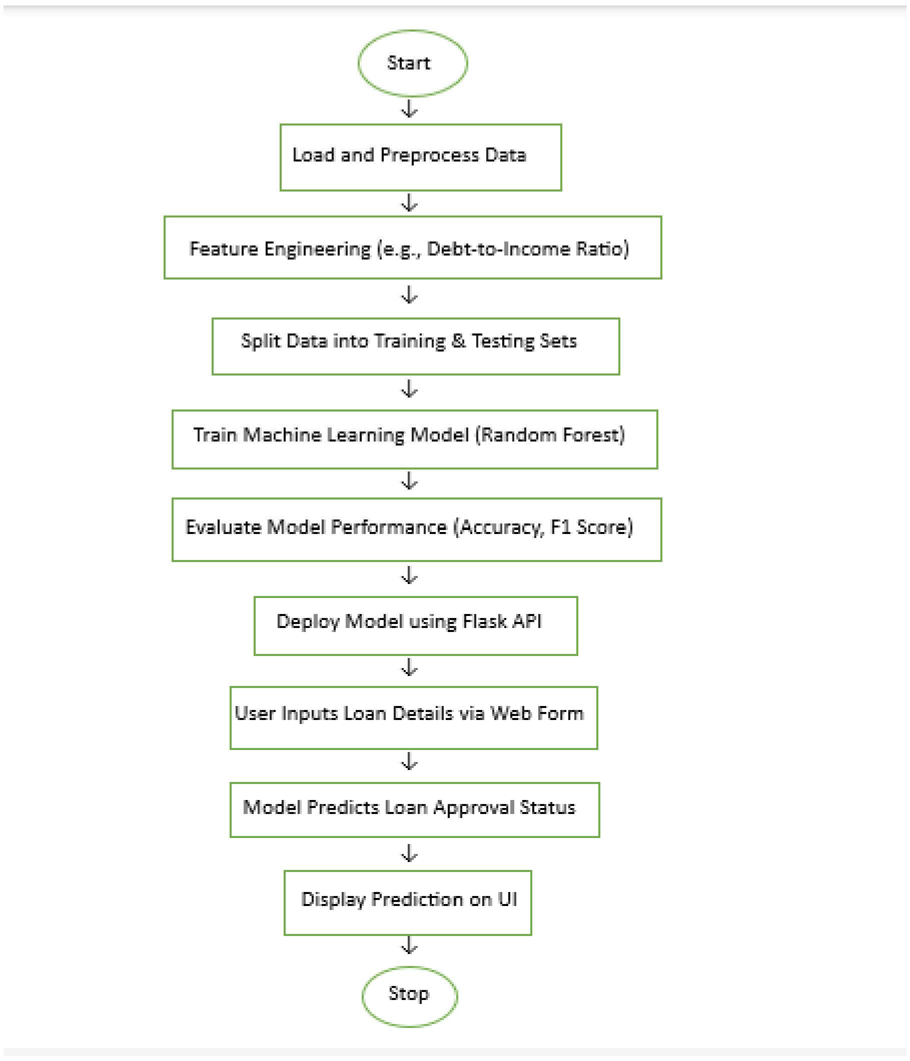
- **CPU**: 1 vCPU (Virtual CPU)

- **RAM**: 2GB (4GB recommended)
- **Storage**: 10GB SSD
- **Internet Connection**: Required for web hosting and usage

The **Loan Approval Prediction Web** is designed to run efficiently on most modern systems. A **basic laptop or PC** with at least **4GB RAM and an Intel i3 processor** can handle **model training and web app development**, while a **lightweight cloud server** can manage deployment.

Chapter 3 : System Design

✞ 3.1 System Flow Chart


Start
Load and Preprocess Data
Feature Engineering (e.g., Debt-to-Income Ratio)
Split Data into Training & Testing Sets
Train Machine Learning Model (Random Forest)
Evaluate Model Performance (Accuracy, F1 Score)
Deploy Model using Flask API
User Inputs Loan Details via Web Form
Model Predicts Loan Approval Status
Display Prediction on UI
Stop

3.2 Test Cases for the project

Dataset loading and preprocessing

Test Case ID	Test Scenario	Test Steps	Expected Result
TC_001	Check if the dataset loads correctly	Load dataset using pandas	Dataset loads without errors
TC_002	Check missing values handling	Identify and fill missing values using fillna()	Missing values are handled correctly
TC_003	Check feature scaling and normalization	Apply Min-Max Scaling or Standardization	Data is normalized properly

Name : Dipansu Singh

TC_004	Verify categorical encoding	Convert categorical features using LabelEncoder or OneHotEncoder	Categorical variables are encoded correctly
TC_005	Verify train-test split	Split dataset into training/testing (8020 ratio)	Categorical variables are encoded correctly

Model Training and Evaluation

Test Case ID	Test Scenario	Test Steps	Expected Result
TC_006	Check if model compiles without errors	Initialize model and compile	Model compiles successfully
TC_007	Train model with sample data	Train model on a small batch	Model starts training without errors
TC_008	Evaluate model	Run evaluation on test set	Model returns accuracy, precision, recall, F1-score
TC_009	Verify confusion matrix	Generate confusion matrix	Matrix correctly represents classification results

User interface testing

Test Case ID	Test Scenario	Test Steps	Expected
TC_010	Check UI responsiveness	Open the website on different devices (mobile, tablet, PC)	UI adjusts correctly for all screen sizes
TC_011	Verify button functionality	Click on all buttons (Submit, Back, Predict)	All buttons work and navigate properly
TC_012	Test loading time	Measure page load speed	Page loads within an acceptable time (<3 seconds)
TC_013	Verify input field labels	Hover over buttons and input fields	UI elements respond with visual feedback
TC_014	Check prediction display	Submit valid inputs and observe displayed result	PCOS risk prediction is displayed correctly

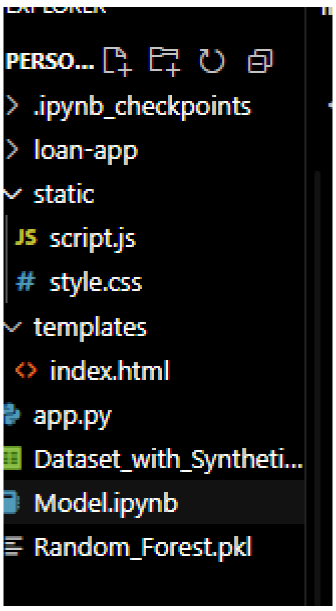
The **test cases for the Loan Approval Prediction project** are structured across different aspects of the system to ensure a **fully functional, reliable, and user-friendly application**.

1. **Dataset and Preprocessing Testing:** These test cases validate that the dataset loads correctly, preprocessing functions (such as resizing and normalization) work as expected, and the dataset is split appropriately for training and testing.
2. **Model Training & Evaluation Testing:** These tests ensure that the model compiles without errors, trains successfully on sample data, and produces expected evaluation metrics like accuracy, precision, and recall.
3. **User Input Testing:** These test cases check if the form accepts user inputs correctly, validates required fields, handles incorrect or missing data, and properly calculates derived values like BMI and cycle score.
4. **Result Display Testing:** These tests verify that the model returns predictions correctly, displays appropriate messages for different risk levels, and ensures result clarity for users.
5. **UI/UX Testing:** The interface is tested for responsiveness, button functionality, navigation, readability, and alignment to ensure a seamless user experience across different devices.

Chapter 4 : Implementation

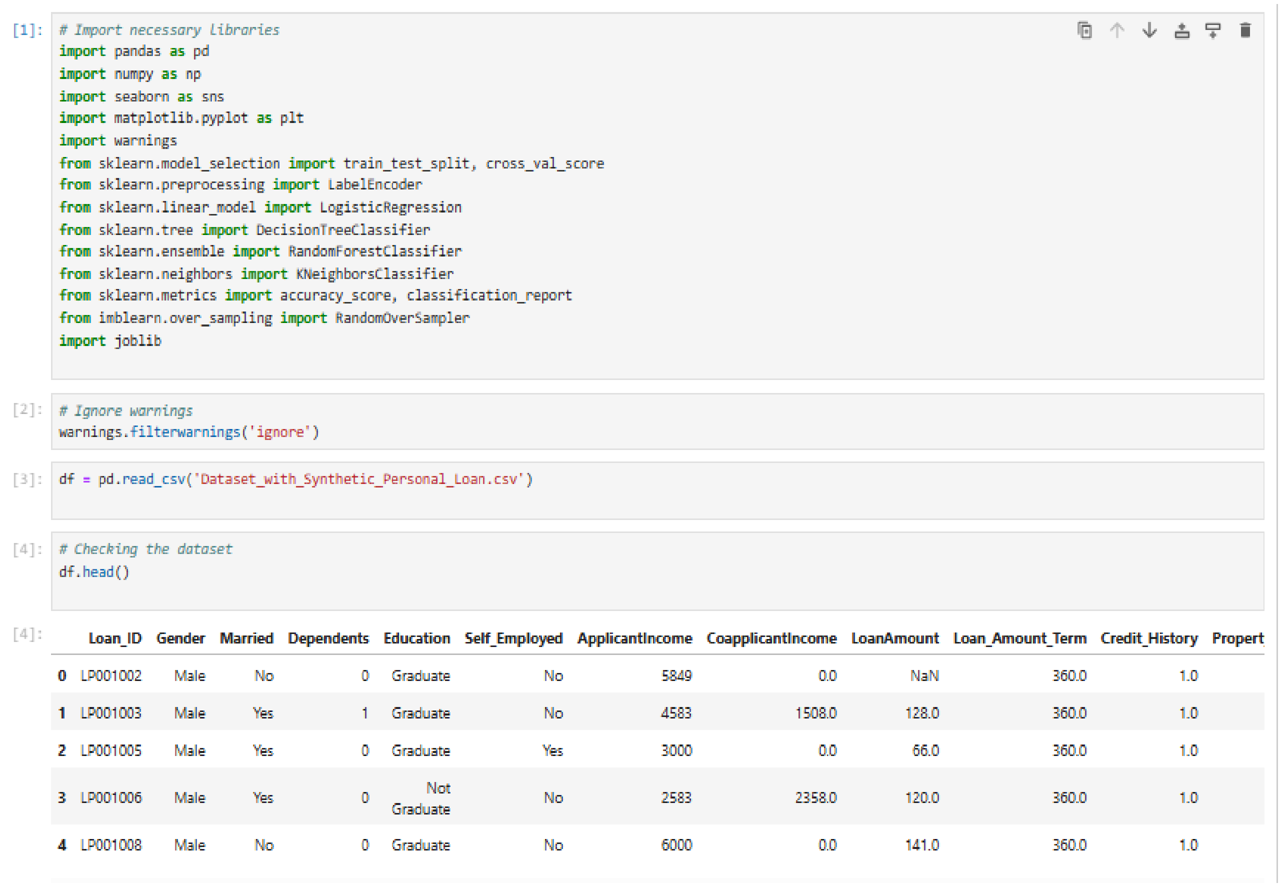
4.1 Program Code

Program Structure :



```
PERSO...
> .ipynb_checkpoints
> loan-app
v static
    script.js
    style.css
v templates
    index.html
  app.py
  Dataset_with_Syntheti...
  Model.ipynb
  Random_Forest.pkl
```

Code :



```
[1]: # Import necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
from imblearn.over_sampling import RandomOverSampler
import joblib
```

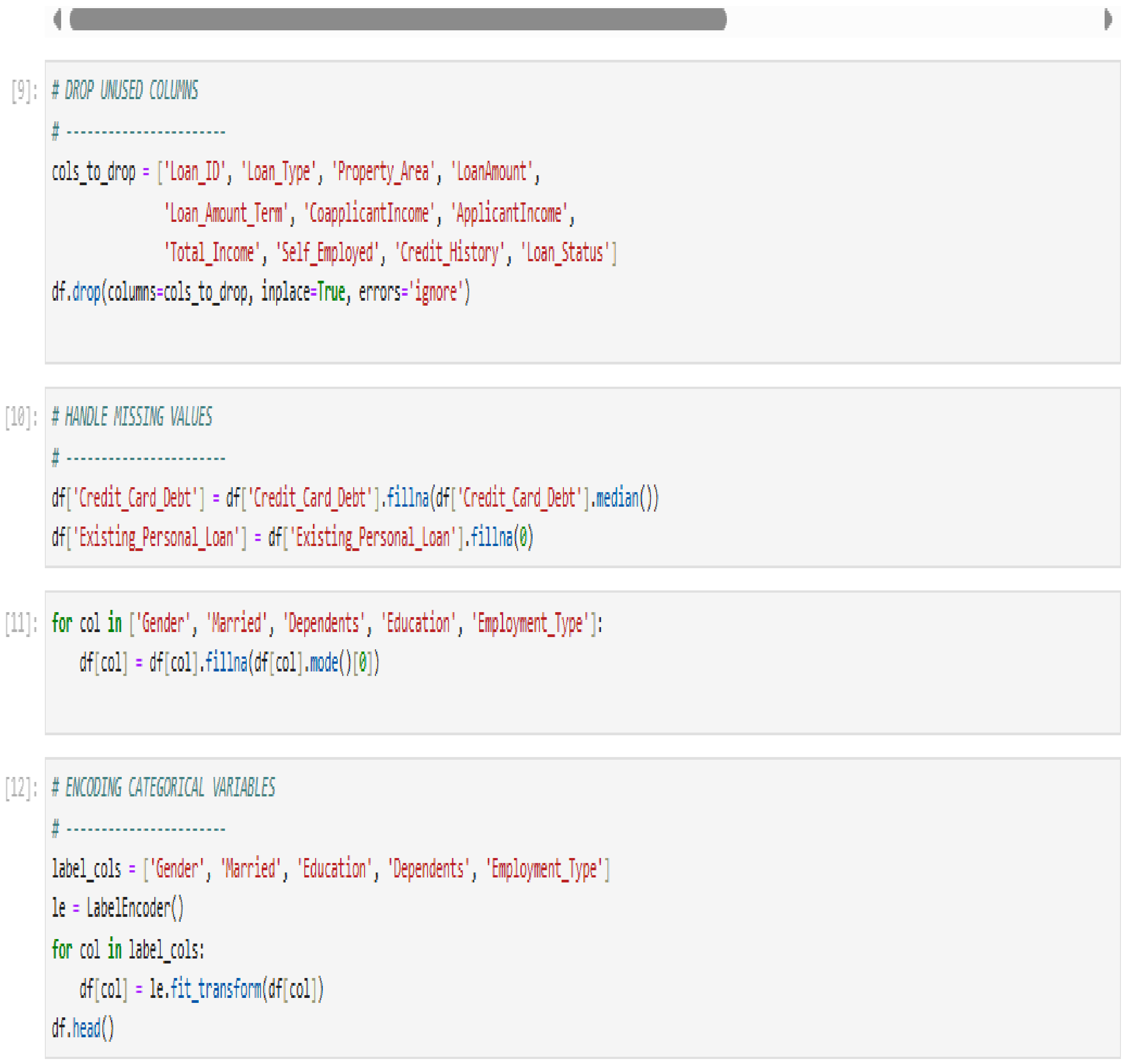
```
[2]: # Ignore warnings
warnings.filterwarnings('ignore')
```

```
[3]: df = pd.read_csv('Dataset_with_Synthetic_Personal_Loan.csv')
```

```
[4]: # Checking the dataset
df.head()
```

[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Propert
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	

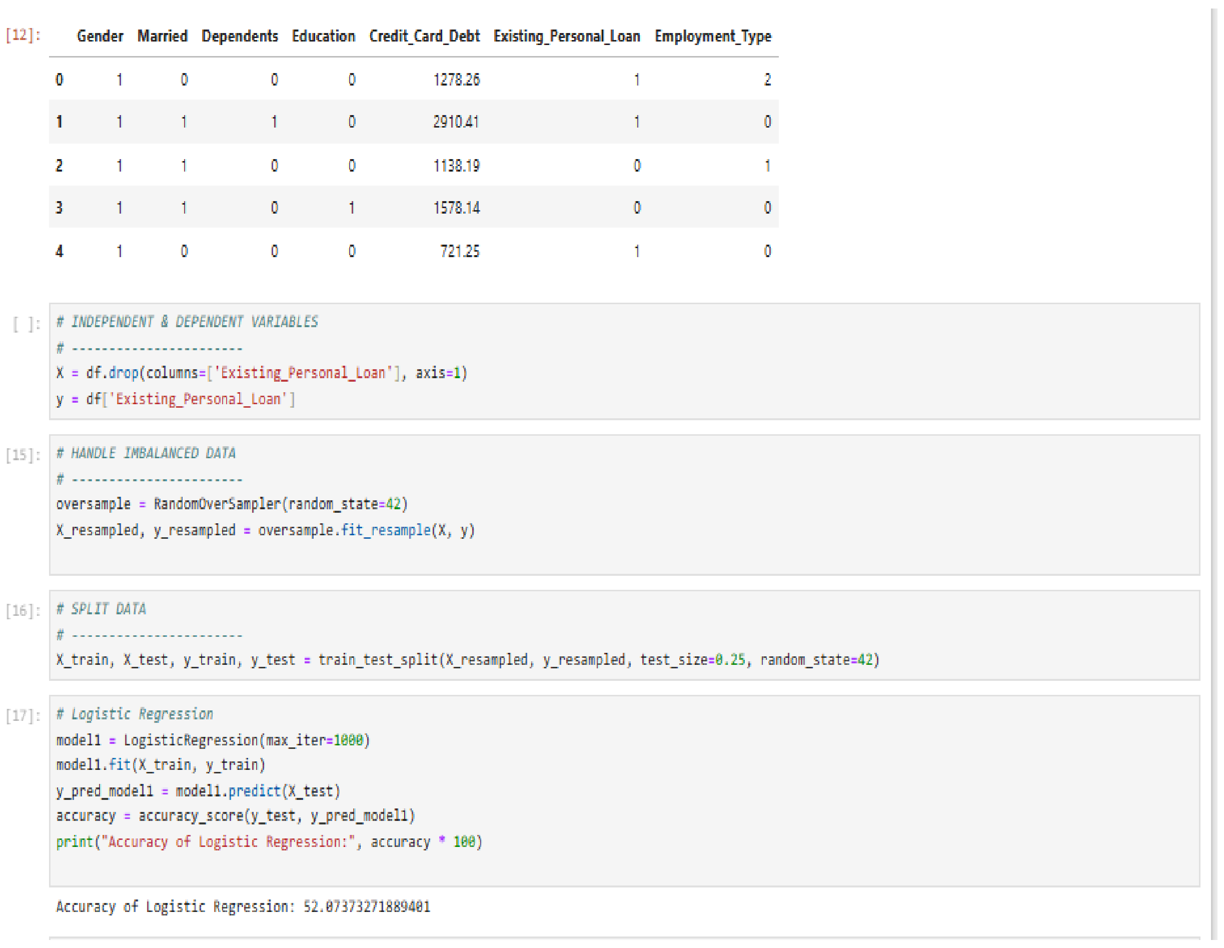


[9]:
```python
# DROP UNUSED COLUMNS
# ----------------------
cols_to_drop = ['Loan_ID', 'Loan_Type', 'Property_Area', 'LoanAmount',
                'Loan_Amount_Term', 'CoapplicantIncome', 'ApplicantIncome',
                'Total_Income', 'Self_Employed', 'Credit_History', 'Loan_Status']
df.drop(columns=cols_to_drop, inplace=True, errors='ignore')
```

[10]:
```python
# HANDLE MISSING VALUES
# ----------------------
df['Credit_Card_Debt'] = df['Credit_Card_Debt'].fillna(df['Credit_Card_Debt'].median())
df['Existing_Personal_Loan'] = df['Existing_Personal_Loan'].fillna(0)
```

[11]:
```python
for col in ['Gender', 'Married', 'Dependents', 'Education', 'Employment_Type']:
    df[col] = df[col].fillna(df[col].mode()[0])
```

[12]:
```python
# ENCODING CATEGORICAL VARIABLES
# ----------------------
label_cols = ['Gender', 'Married', 'Education', 'Dependents', 'Employment_Type']
le = LabelEncoder()
for col in label_cols:
    df[col] = le.fit_transform(df[col])
df.head()
```



[12]:

	Gender	Married	Dependents	Education	Credit_Card_Debt	Existing_Personal_Loan	Employment_Type
0	1	0	0	0	1278.26	1	2
1	1	1	1	0	2910.41	1	0
2	1	1	0	0	1138.19	0	1
3	1	1	0	1	1578.14	0	0
4	1	0	0	0	721.25	1	0

[ ]:

```
# INDEPENDENT & DEPENDENT VARIABLES
# ----------------------
X = df.drop(columns=['Existing_Personal_Loan'], axis=1)
y = df['Existing_Personal_Loan']
```

[15]:

```
# HANDLE IMBALANCED DATA
# ----------------------
oversample = RandomOverSampler(random_state=42)
X_resampled, y_resampled = oversample.fit_resample(X, y)
```

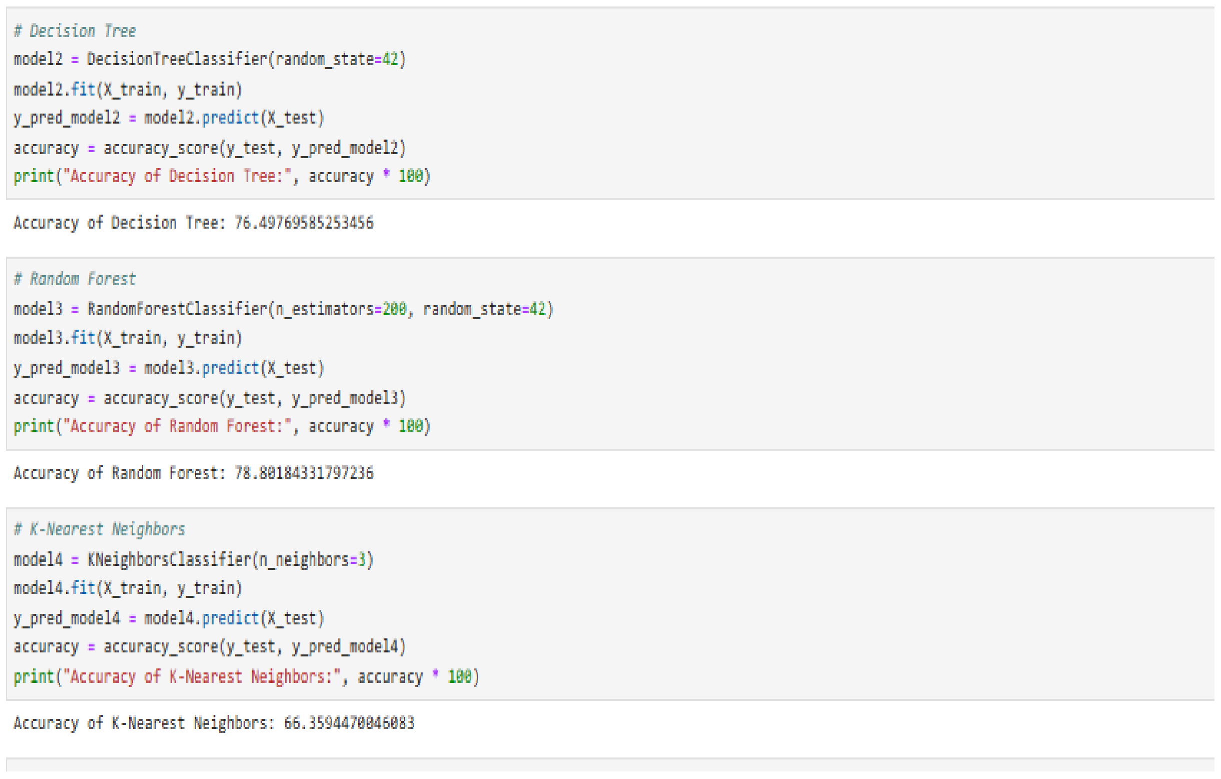
[16]:

```
# SPLIT DATA
# ----------------------
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.25, random_state=42)
```

[17]:

```
# Logistic Regression
model1 = LogisticRegression(max_iter=1000)
model1.fit(X_train, y_train)
y_pred_model1 = model1.predict(X_test)
accuracy = accuracy_score(y_test, y_pred_model1)
print("Accuracy of Logistic Regression:", accuracy * 100)
```

```
Accuracy of Logistic Regression: 52.07373271889401
```



```
# Decision Tree
model2 = DecisionTreeClassifier(random_state=42)
model2.fit(X_train, y_train)
y_pred_model2 = model2.predict(X_test)
accuracy = accuracy_score(y_test, y_pred_model2)
print("Accuracy of Decision Tree:", accuracy * 100)
```

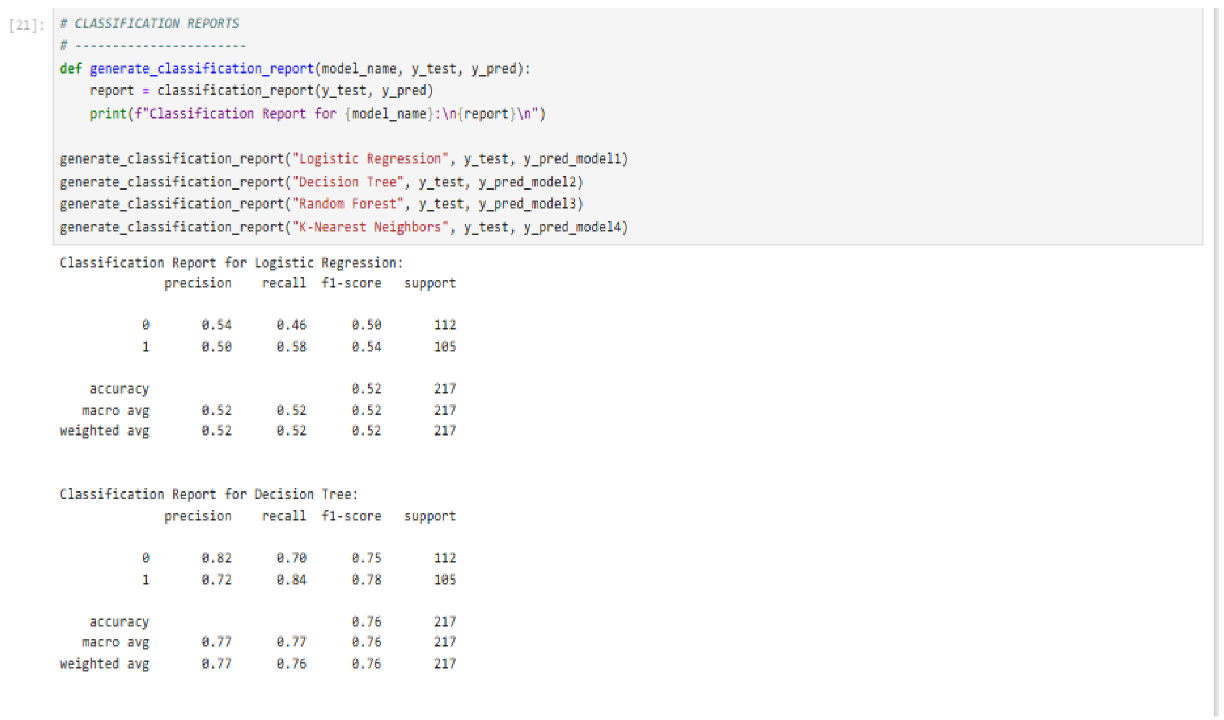
```
Accuracy of Decision Tree: 76.49769585253456
```

```
# Random Forest
model3 = RandomForestClassifier(n_estimators=200, random_state=42)
model3.fit(X_train, y_train)
y_pred_model3 = model3.predict(X_test)
accuracy = accuracy_score(y_test, y_pred_model3)
print("Accuracy of Random Forest:", accuracy * 100)
```

```
Accuracy of Random Forest: 78.80184331797236
```

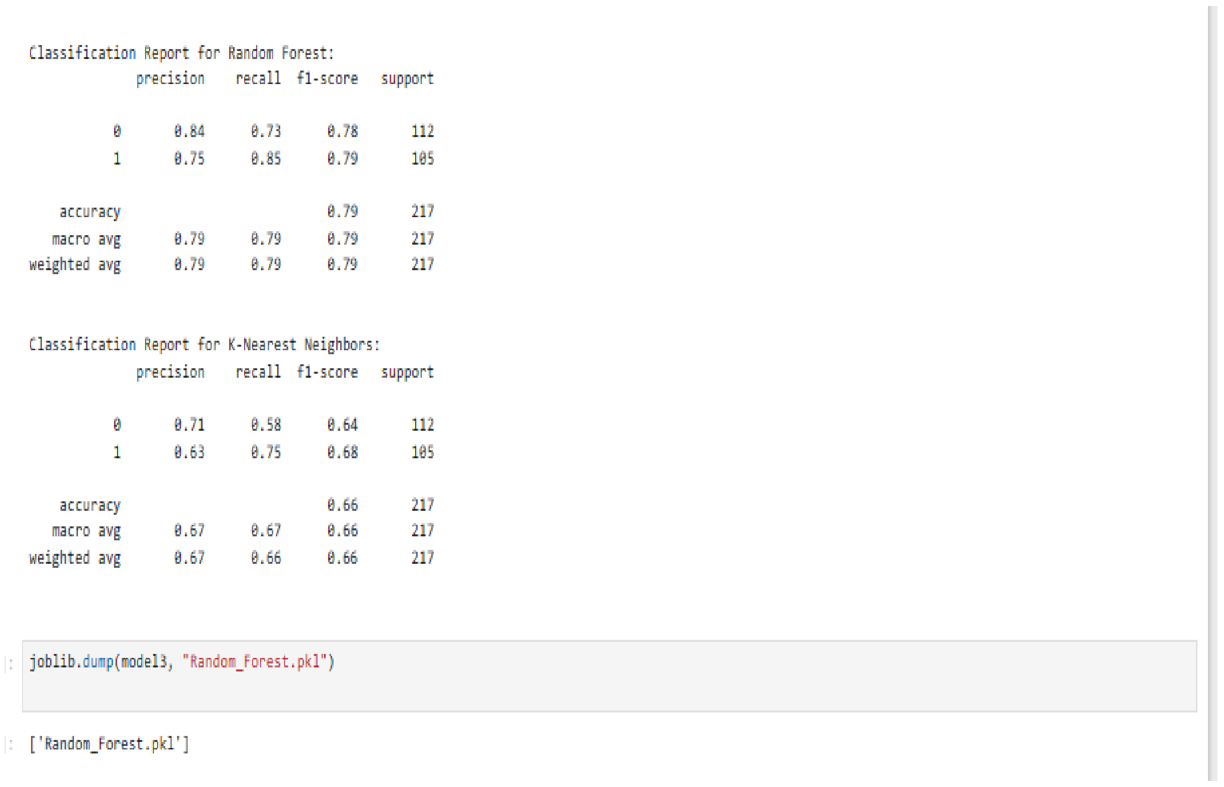
```
# K-Nearest Neighbors
model4 = KNeighborsClassifier(n_neighbors=3)
model4.fit(X_train, y_train)
y_pred_model4 = model4.predict(X_test)
accuracy = accuracy_score(y_test, y_pred_model4)
print("Accuracy of K-Nearest Neighbors:", accuracy * 100)
```

```
Accuracy of K-Nearest Neighbors: 66.3594470046083
```



```
[21]: # CLASSIFICATION REPORTS
      # ----------------------
      def generate_classification_report(model_name, y_test, y_pred):
          report = classification_report(y_test, y_pred)
          print(f"Classification Report for {model_name}:\n{report}\n")

      generate_classification_report("Logistic Regression", y_test, y_pred_model1)
      generate_classification_report("Decision Tree", y_test, y_pred_model2)
      generate_classification_report("Random Forest", y_test, y_pred_model3)
      generate_classification_report("K-Nearest Neighbors", y_test, y_pred_model4)
```

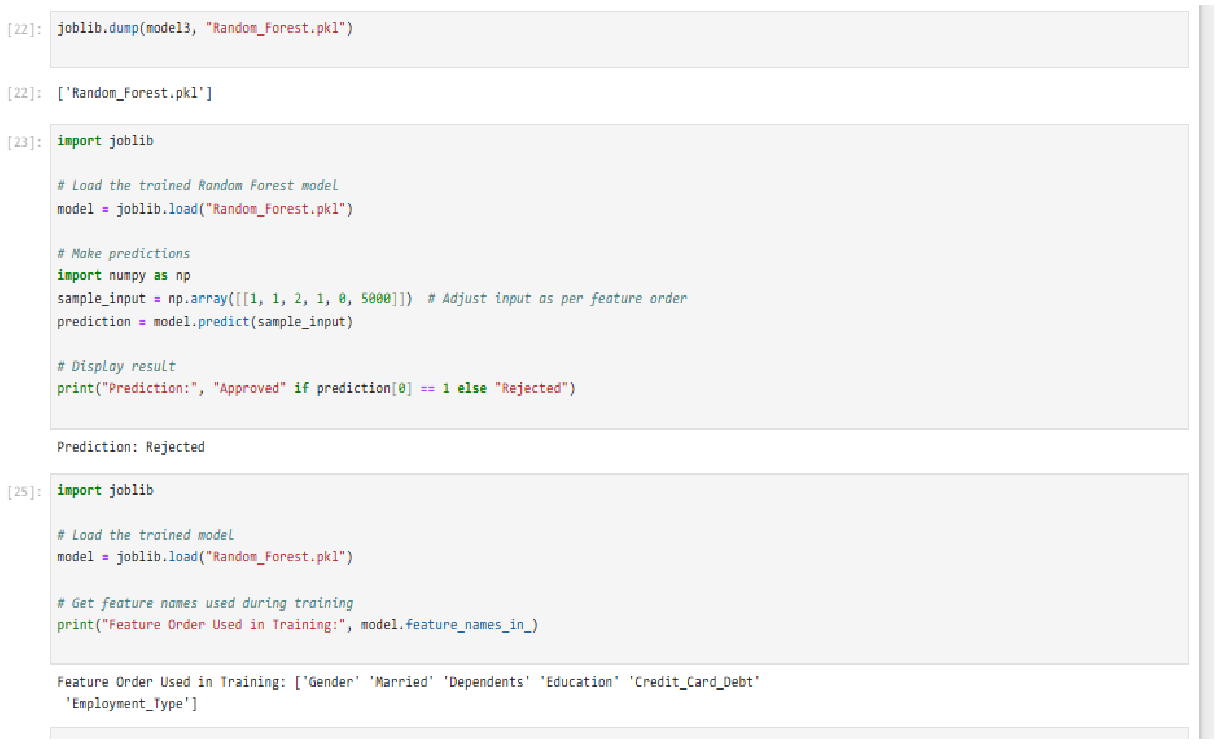
```
Classification Report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.54      0.46      0.50       112
           1       0.50      0.58      0.54       105

    accuracy                           0.52       217
   macro avg       0.52      0.52      0.52       217
weighted avg       0.52      0.52      0.52       217


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

           0       0.82      0.70      0.75       112
           1       0.72      0.84      0.78       105

    accuracy                           0.76       217
   macro avg       0.77      0.77      0.76       217
weighted avg       0.77      0.76      0.76       217
```



```
Classification Report for Random Forest:
              precision    recall  f1-score   support

           0       0.84      0.73      0.78       112
           1       0.75      0.85      0.79       105

    accuracy                           0.79       217
   macro avg       0.79      0.79      0.79       217
weighted avg       0.79      0.79      0.79       217


Classification Report for K-Nearest Neighbors:
              precision    recall  f1-score   support

           0       0.71      0.58      0.64       112
           1       0.63      0.75      0.68       105

    accuracy                           0.66       217
   macro avg       0.67      0.67      0.66       217
weighted avg       0.67      0.66      0.66       217
```

```
: joblib.dump(model3, "Random_Forest.pkl")

: ['Random_Forest.pkl']
```

Name : Dipansu Singh



```
[22]: joblib.dump(model3, "Random_Forest.pkl")

[22]: ['Random_Forest.pkl']

[23]: import joblib

      # Load the trained Random Forest model
      model = joblib.load("Random_Forest.pkl")

      # Make predictions
      import numpy as np
      sample_input = np.array([[1, 1, 2, 1, 0, 5000]])  # Adjust input as per feature order
      prediction = model.predict(sample_input)

      # Display result
      print("Prediction:", "Approved" if prediction[0] == 1 else "Rejected")

      Prediction: Rejected

[25]: import joblib

      # Load the trained model
      model = joblib.load("Random_Forest.pkl")

      # Get feature names used during training
      print("Feature Order Used in Training:", model.feature_names_in_)

      Feature Order Used in Training: ['Gender' 'Married' 'Dependents' 'Education' 'Credit_Card_Debt'
       'Employment_Type']
```

templates/index.html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Loan Approval Prediction</title>


  <!-- Bootstrap CSS -->

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">

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

</head>

<body>

  <div class="container">

    <h2 class="text-center mt-4">Loan Approval Prediction</h2>


    <form id="loanForm" class="p-4 border rounded shadow bg-light">

      <div class="mb-3">

        <label class="form-label">Gender (Male=1, Female=0):</label>

        <input type="number" name="Gender" class="form-control" placeholder="Enter 1
for Male, 0 for Female" required>

      </div>


      <div class="mb-3">

        <label class="form-label">Married (Yes=1, No=0):</label>

        <input type="number" name="Married" class="form-control" placeholder="Enter
1 for Yes, 0 for No" required>

      </div>
```

```
            <div class="mb-3">
                <label class="form-label">Dependents (Enter a number, e.g., 0, 1, 2):</label>
                <input type="number" name="Dependents" class="form-control"
placeholder="Enter number of dependents" required>
            </div>

            <div class="mb-3">
                <label class="form-label">Education (Graduate=1, Not Graduate=0):</label>
                <input type="number" name="Education" class="form-control"
placeholder="Enter 1 for Graduate, 0 for Not Graduate" required>
            </div>

            <div class="mb-3">
                <label class="form-label">Credit Card Debt (Enter amount in numbers):</label>
                <input type="number" name="Credit_Card_Debt" class="form-control"
placeholder="Enter Credit Card Debt" required>
            </div>

            <div class="mb-3">
                <label class="form-label">Employment Type (Salaried=0, Self-Employed=1,
Unemployed=2):</label>
                <input type="number" name="Employment_Type" class="form-control"
placeholder="Enter 0 for Salaried, 1 for Self-Employed, 2 for Unemployed" required>
            </div>

            <button type="submit" class="btn btn-primary w-100">Check Loan
Approval</button>
        </form>

        <div id="result" class="alert mt-4 d-none"></div>
```

```
    </div>


    <!-- Bootstrap JS -->

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

    <script src="{{ url_for('static', filename='script.js') }}"></script>

</body>

</html>
```

Static/script.js

```
document.getElementById("loanForm").addEventListener("submit", async function (e) {
  e.preventDefault();

  let formData = new FormData(this);
  let jsonData = {};
  formData.forEach((value, key) => {
    jsonData[key] = Number(value);  // Convert inputs to numbers
  });

  document.getElementById("result").classList.add("d-none");
  document.getElementById("result").textContent = "Processing...";

  try {
    let response = await fetch("/predict", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(jsonData),
    });

    let data = await response.json();
    let resultDiv = document.getElementById("result");

    resultDiv.textContent = data.prediction;
    resultDiv.classList.remove("d-none");
```

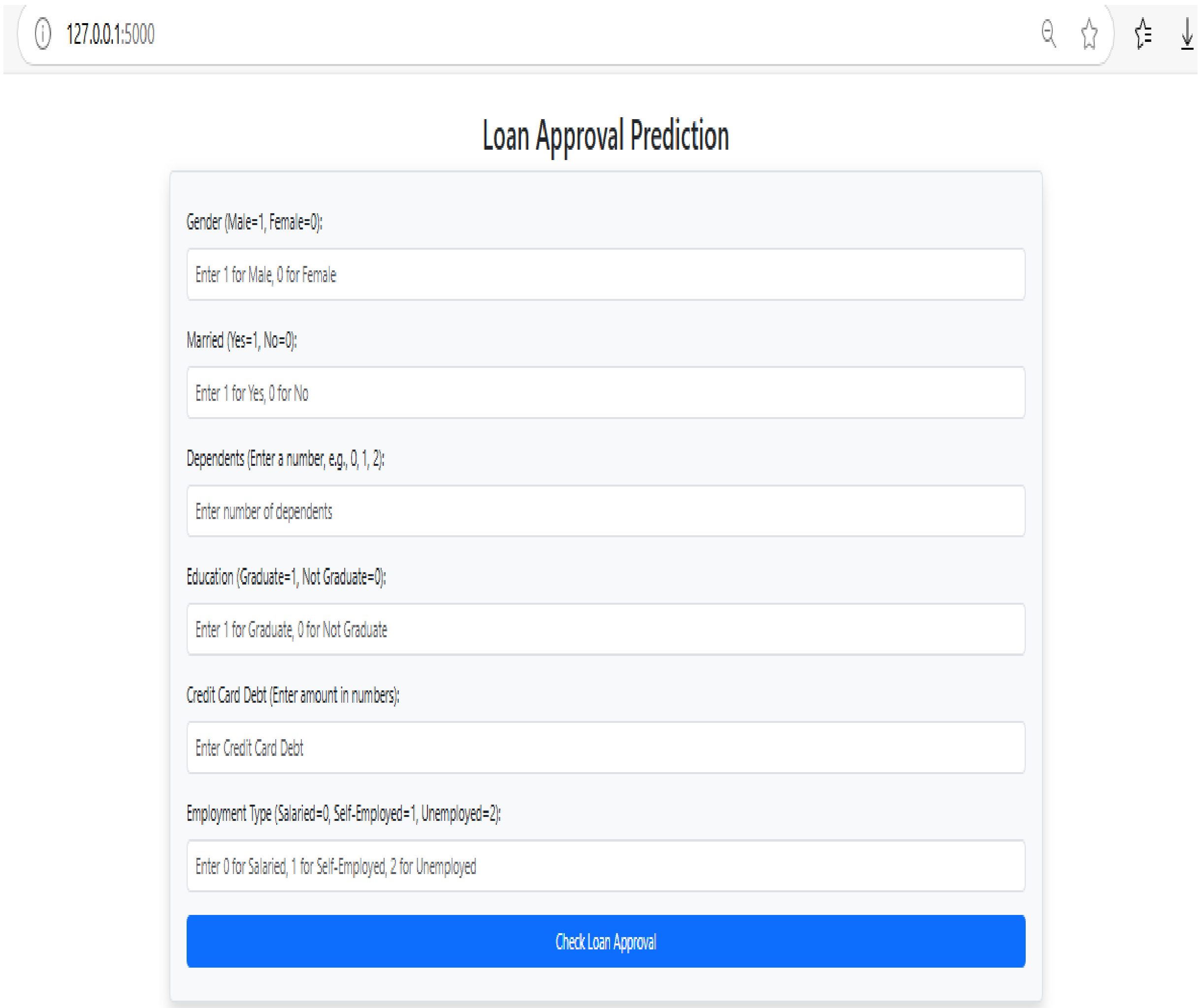
```
 if (data.prediction === "Approved") {
            resultDiv.classList.add("alert", "alert-success");
        } else {
            resultDiv.classList.add("alert", "alert-danger");
        }
    } catch (error) {
        alert("Error connecting to the server. Please try again.");
    }
});
```

App.py

```
from flask import Flask, render_template, request, jsonify

import joblib

import pandas as pd


app = Flask(__name__)


# Load the trained model

model = joblib.load("Random_Forest.pkl")


# Define the required feature order

REQUIRED_FEATURES = ["Gender", "Married", "Dependents", "Education",
"Credit_Card_Debt","Employment_Type"]


# Define categorical mappings

label_mapping = {

    "Gender": {"Male": 1, "Female": 0},

    "Married": {"Yes": 1, "No": 0},

    "Education": {"Graduate": 1, "Not Graduate": 0},

    "Employment_Type": {"Salaried": 0, "Self-Employed": 1, "Unemployed": 2}

}


@app.route('/')

def home():

    return render_template("index.html")


@app.route('/predict', methods=['POST'])

def predict():

    try:
        data = request.get_json()


        if not all(feature in data for feature in REQUIRED_FEATURES):
            return jsonify({"error": "Missing required fields"}), 400


        input_df = pd.DataFrame([data])


        for col, mapping in label_mapping.items():
            if col in input_df.columns:
                input_df[col] = input_df[col].map(mapping)


        input_df = input_df[REQUIRED_FEATURES]


        prediction = model.predict(input_df)[0]
        result = "Approved" if prediction == 1 else "Rejected"


        return jsonify({"prediction": result})


    except Exception as e:
        return jsonify({"error": str(e)}), 500


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

Output/Screenshots:

Landing page:



127.0.0.1:5000

Loan Approval Prediction

Gender (Male=1, Female=0):

Enter 1 for Male, 0 for Female

Married (Yes=1, No=0):

Enter 1 for Yes, 0 for No

Dependents (Enter a number, e.g., 0, 1, 2):

Enter number of dependents

Education (Graduate=1, Not Graduate=0):

Enter 1 for Graduate, 0 for Not Graduate

Credit Card Debt (Enter amount in numbers):

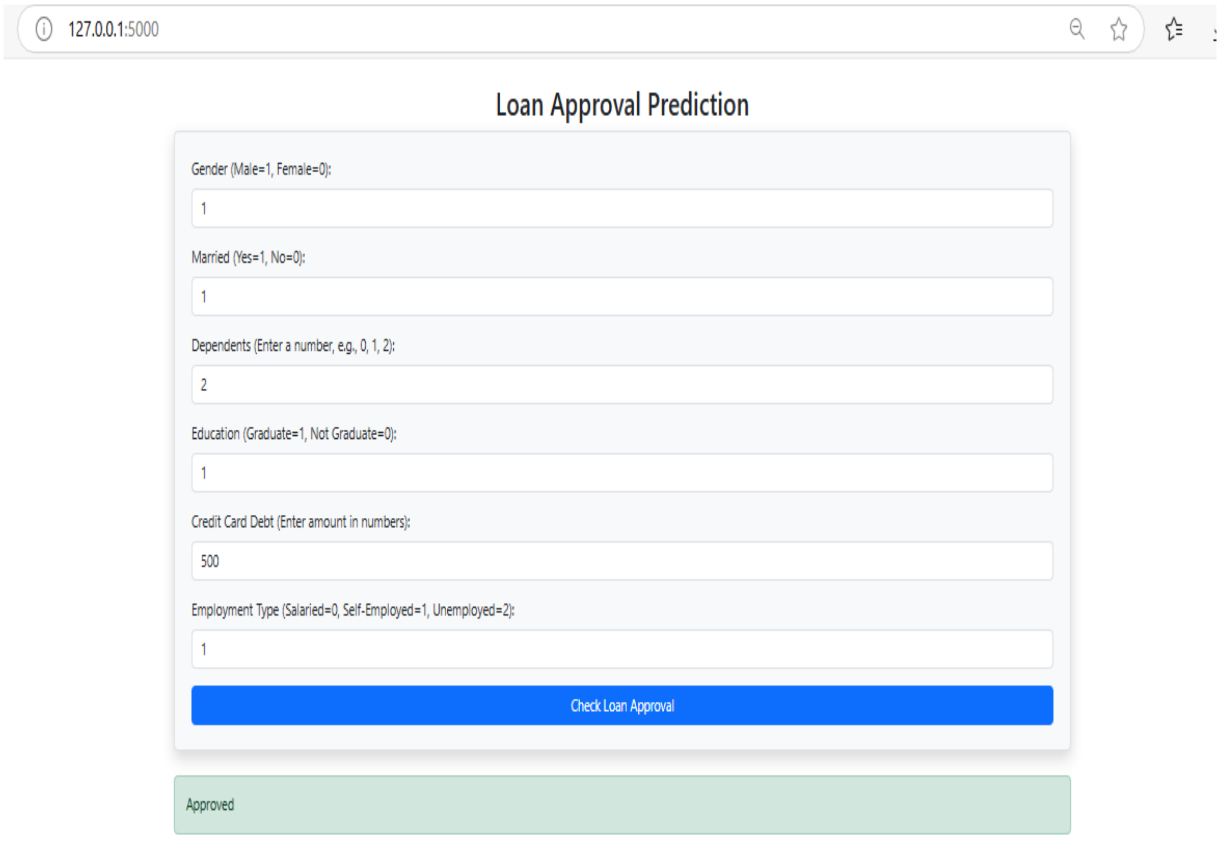
Enter Credit Card Debt

Employment Type (Salaried=0, Self-Employed=1, Unemployed=2):

Enter 0 for Salaried, 1 for Self-Employed, 2 for Unemployed

Check Loan Approval

Approved Loan:



127.0.0.1:5000

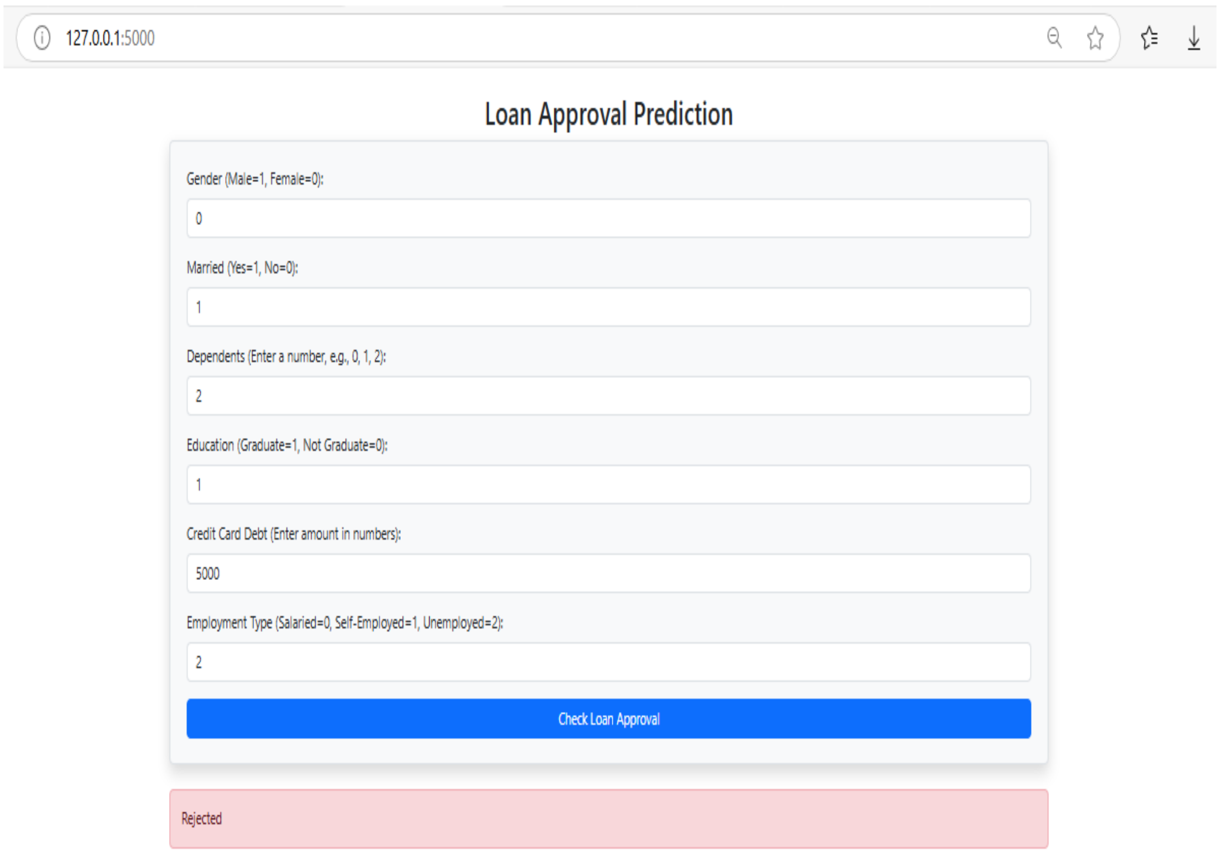
Loan Approval Prediction

Gender (Male=1, Female=0):	1
Married (Yes=1, No=0):	1
Dependents (Enter a number, e.g., 0, 1, 2):	2
Education (Graduate=1, Not Graduate=0):	1
Credit Card Debt (Enter amount in numbers):	500
Employment Type (Salaried=0, Self-Employed=1, Unemployed=2):	1

Approved

Name : Dipansu Singh

Rejected Loan:


127.0.0.1:5000

Loan Approval Prediction

Gender (Male=1, Female=0):
0

Married (Yes=1, No=0):
1

Dependents (Enter a number, e.g., 0, 1, 2):
2

Education (Graduate=1, Not Graduate=0):
1

Credit Card Debt (Enter amount in numbers):
5000

Employment Type (Salaried=0, Self-Employed=1, Unemployed=2):
2

Rejected

5.3 Testing Approaches

Test Case ID	**Test Case Description**	**Testing Approach**	**Objectives**
TC – 001	Verify dataset is loaded correctly.	Unit Testing	Ensure that all user inputs are correctly loaded into the system for prediction
TC – 002	Verify input validation and preprocessing.	Unit Testing	Ensure that user inputs (e.g., Gender, Married, Education) are correctly validated and preprocessed
TC – 003	Verify train-test split works correctly.	Unit Testing	Ensure that the dataset is properly divided into training and testing sets for model evaluation.
TC – 004	Verify machine learning model loading	Unit Testing	Ensure that the trained model is correctly loaded and ready for making predictions
TC – 005	Verify model training process.	Integration Testing	Confirm that the model trains without errors
TC – 006	Verify loss and accuracy calculations during training	Integration Testing	Ensure that the model correctly computes loss and accuracy values during training
TC – 007	Verify model evaluation on test data.	Integration Testing	Check that the trained model is tested on unseen data without errors.
TC – 008	Verify web form submission and response	System Testing	Ensure that user inputs are successfully processed, and the system returns a prediction result.
TC – 009	Verify UI responsiveness and navigation.	System Testing	Confirm that all buttons (Submit, Back) work correctly, and the UI is accessible on different screen sizes
TC – 010	Verify if the result is predicted successfully	System Testing	The expected result is displayed to the user without any issues.

Chapter 5 : Results and Conclusion

5.1 Test Report

The **Test Case Results** table provides a detailed validation of the **Loan Approval Prediction Web**, ensuring that its core functionalities operate correctly. Each test case is uniquely identified with a **Test Case ID**, making it easy to track and reference. The **Test Case Description** outlines the purpose of each test, such as verifying dataset loading, checking data preprocessing, confirming model compilation, and evaluating predictions.

The **Input** column specifies the data or actions used to perform each test, such as loading datasets, applying preprocessing techniques, or running model evaluations.

The **Expected Result** defines the anticipated outcome, ensuring that each functionality meets the desired behavior. Finally, the **Result** column records whether the test case passed or failed, with all tests successfully passing, confirming the system's robustness.

Test Case ID	**Test Case Description**	**Input**	**Objectives**	**Result**
TC – 001	Verify dataset is loaded correctly.	Load dataset using Pandas.	Dataset loads without errors.	Pass
TC – 002	Verify input validation and preprocessing.	Enter valid and invalid user inputs.	Inputs are validated, and invalid entries are handled.	Pass
TC – 003	Verify train-test split works correctly.	Split dataset into training and testing (80.20%).	Dataset is split correctly..	Pass
TC – 004	Verify machine learning model loading	Initialize and load trained ML model.	Ensure that the trained model is correctly loaded and ready for making predictions	Pass
TC – 005	Verify model training process.	Train model with a small batch.	Confirm that the model trains without errors and updates weights correctly	Pass
TC – 006	Verify loss and accuracy calculations during training	Run evaluation on test set.	Ensure that the model correctly computes loss and accuracy values during training	Pass
TC – 007	Verify model evaluation on test data.	Submit user input through the web form	Check that the trained model is tested on unseen data without errors.	Pass

TC – 008	Verify web form submission and response	Access website on different devices.	Ensure that user inputs are successfully processed, and the system returns a prediction result.	Pass
TC – 009	Verify UI responsiveness and navigation.	Input malicious scripts	Confirm that all buttons (Submit, Back) work correctly, and the UI is accessible on different screen sizes	Pass
TC – 010	Verify security and data protection	Compare prediction with known data.	Results align with expected outcomes.	Pass

5.2 Conclusion

The **Personal Loan Approval Predictor** project successfully demonstrates the application of machine learning to automate and streamline the loan approval process. By leveraging a dataset of customer attributes, the project developed a predictive model using the Random Forest algorithm, which achieved an accuracy of 78.80%. This model was deployed as a user-friendly web application using Flask, enabling real-time predictions for loan approval status.

The project highlights the potential of data-driven solutions to reduce manual effort, minimize errors, and improve decision-making in financial institutions. While the model performs well, there are limitations, such as dependency on the quality of the dataset and the need for periodic retraining.

In conclusion, the **Personal Loan Approval Predictor** is a practical and scalable solution that can significantly enhance the efficiency of loan approval processes. With further improvements, such as incorporating additional features and advanced techniques, this project has the potential to revolutionize the way financial institutions handle loan applications in the future.

5.3 Limitation of the System

1. **Dataset Quality**: The model's accuracy depends on the quality and size of the dataset. Limited or biased data can affect predictions.
2. **External Factors**: The model does not account for external factors like economic conditions or market trends.
3. **Feature Dependency**: Predictions rely heavily on the provided features, and missing or incorrect data can lead to inaccurate results.
4. **Retraining Requirement**: The model may need frequent retraining to adapt to new data and changing patterns.

5.4 Future Scope

1. **Enhanced Features**: Incorporate additional features like credit score, loan history, and financial behavior for better predictions.
2. **Advanced Techniques**: Explore deep learning or ensemble methods to improve accuracy.
3. **Larger Dataset**: Expand the dataset to include more diverse loan applications for robust training.
4. **Mobile Application**: Develop a mobile app for easier access and usability.
5. **Real-Time Integration**: Integrate the model with banking systems for real-time loan processing.
6. **Explainability**: Add explainable AI features to provide insights into why a loan was approved or rejected.

.

References

□ **Dataset Source – Kaggle Dataset https://www.kaggle.com □ Machine Learning Framework – TensorFlow and Scikit-Learn Official**
Documentation https://www.tensorflow.org/
https://scikit-learn.org/

□ **Python Programming Language – Python Official Documentation https://docs.python.org/ □ Jupyter Notebook for Model Development – Project Jupyter Documentation**
https://jupyter.org/

□ **Flask Web Framework – Flask Official Documentation**
https://flask.palletsprojects.com/ □ Medical Research on PCOS – Rotterdam ESHRE/ASRM-Sponsored PCOS Consensus Workshop Group
https://academic.oup.com/humrep/article/19/1/41/2902555 □ Online Machine Learning Community for Support and Discussions –
Kaggle and Stack Overflow
https://www.kaggle.com/
https://stackoverflow.com/

□ **GitHub Repository for Project Code – Community Contributions https://github.com/**