Loan Status Prediction System

(Machine Learning)

A project report submitted in the partial fulfillment of
Requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY IN
COMPUTER SCIENCE AND ENGINEERING

Name(Registration Number) :
JAMMULA VARSHINI(21501A0568)
KOGANTI KAUSHIK(21501A0589)
KOLLA BHARGAVA CHOWDARY(21501A0591)
MOHAMMED FAIZUDDIN(21501A05B4)
MANGALAGIRI GAYATRI(21501A05A8)

Name of the college: **Prasad V. Potluri Siddhartha Institute of Technology**

Under the Esteemed Guidance of
Ms. Y. Surekha, M.Tech, (Ph.D.)
Assistant Professor,
Department of CSE

Department of Computer Science and Engineering
PRASAD V. POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY
(Permanently affiliated to JNTU:Kakinada, Approved by AICTE)
(An NBA&NAAC A+ accredited and ISO 9001:2015 certified Institution)
Kanuru, Vijayawada –520007

2023-2024

CONTENTS

CHAPTER-1 : OVERVIEW OF THE PROJECT

Introduction:

A Finance company wants to automate the loan eligibility process based on the customer details provided while filling application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have given a problem to identify the customers segments, those are eligible for loan amount so that they can specifically target these customers.

Brief overview of the project's objective: to predict whether a loan application will be approved or not based on various applicant features.

Importance of loan prediction in financial institutions for risk assessment and decision-making.

Data Collection and Exploration:

Description of the dataset used, including its source and structure.

Exploration of key features such as applicant income, co-applicant income, loan amount, etc.

Identification of any missing values or outliers and strategies employed to handle them.

Data Preprocessing:

Overview of preprocessing steps performed on the dataset, including:

Handling missing values: imputation or removal.

Encoding categorical variables: one-hot encoding or label encoding.

Feature scaling: standardization or normalization.

Feature engineering: creating new features or transforming existing ones.

Model Training:

Description of the machine learning algorithm used for loan prediction (e.g., logistic regression, decision trees, etc.).

Training process, including the splitting of data into training and validation sets.

Evaluation metrics used to assess model performance (e.g., accuracy, precision, recall, F1-score).

Model Deployment:

Development of a Flask web application to deploy the trained model for real-time predictions.

Overview of the web application's user interface and functionality.

Explanation of how users can interact with the application to input their data and receive loan approval predictions.

Command-Line Interface (CLI):

Description of the command-line script provided for making predictions directly from the terminal.

Instructions on how to use the CLI tool to make predictions using the trained model.

Conclusion:

Summary of the project's objectives, methodologies, and outcomes.

Discussion of any challenges faced during the project and potential areas for improvement.

Overall assessment of the project's success in achieving its goals and contributing to the field of loan prediction.

Future Directions:
Suggestions for future enhancements or extensions to the project, such as:
Incorporating additional features for improved prediction accuracy.
Implementing advanced machine learning techniques.
Scaling the application for deployment in production environments.

CHAPTER 2: DESCRIPTION OF THE CODE

Here we used Random Forest Classifier algorithm. The codebase consists of a Flask web application for real-time loan predictions, HTML templates for user interaction, and utility functions for model loading and prediction. Additionally, there's a command-line interface for making predictions directly from the terminal.

Flask Web Application (app.py):
This script initializes a Flask web application.
It loads the trained machine learning model (model.pkl) using pickle.
The / route renders the home page (index.html).
The /predict route handles form submissions for loan prediction.
It preprocesses the user input, makes predictions using the loaded model, and renders the prediction result on the prediction.html template.
The application runs locally using app.run(debug=True).

HTML Templates (index.html and prediction.html):
index.html: This template represents the home page of the web application.
It contains a navigation bar with links to different sections of the website.
The main section features a heading and a call-to-action button to navigate to the prediction page.
prediction.html: This template is used to display the loan prediction result.
It includes a form for users to input their loan application details.
After submission, it displays the prediction result ("Loan is Approved" or "Loan is Not Approved").

Machine Learning Model Loading (load_model Function):
The load_model function loads the trained machine learning model (model.pkl) using pickle.
It handles exceptions and returns None if the model loading fails.

Prediction Function (predict Function):
The predict function takes the loaded model and input data as parameters.
It makes predictions using the loaded model and returns the prediction result.
It handles exceptions and returns None if prediction fails.

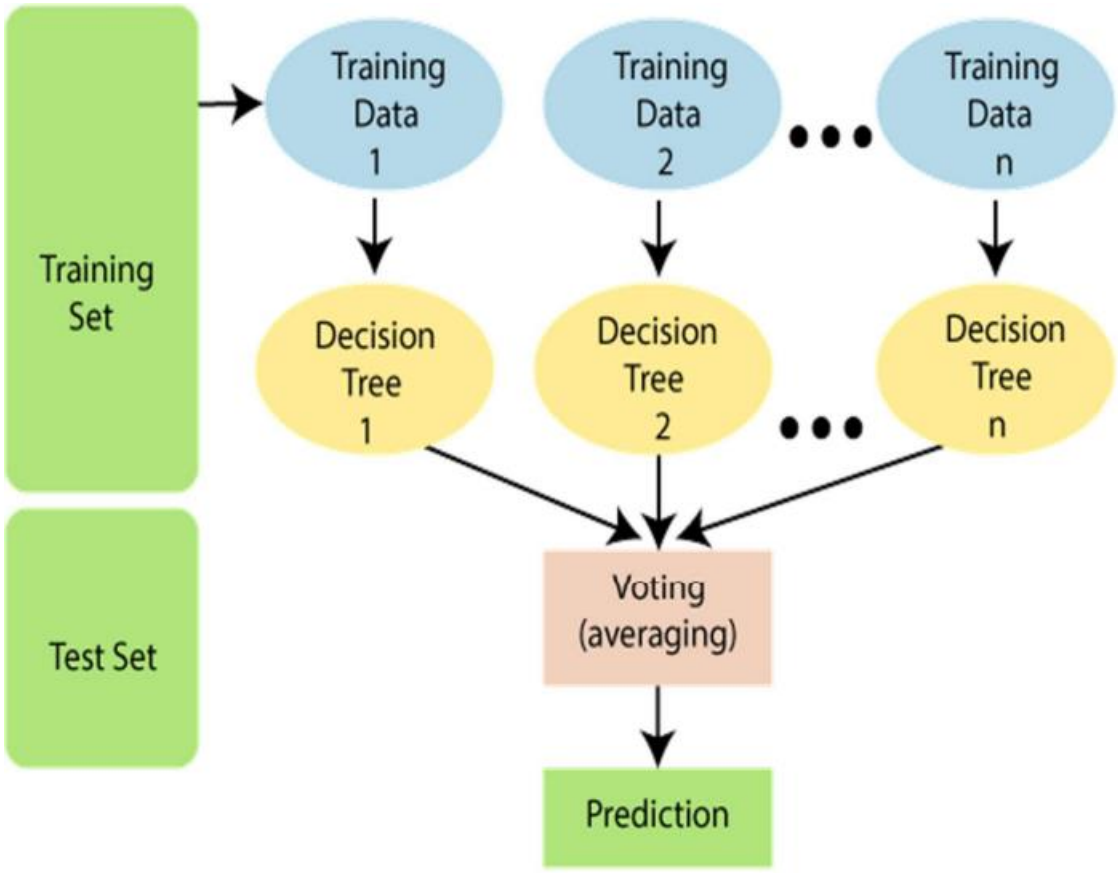
Command-Line Interface (predict_cli.py):
This script provides a command-line interface for making loan predictions.
It loads the trained model using the load_model function and preprocesses input data.
After making predictions using the predict function, it prints the result to the console.

About ML algorithm:
The Random Forest Classifier is a powerful machine learning algorithm that can be used for classification tasks with high accuracy and robustness. In the above code, it is used to predict the loan status based on various features, and the results show that it achieves a high level of accuracy.

Random Forest Classifier is a popular machine learning algorithm that belongs to the category of ensemble learning methods. It is based on the concept of constructing multiple decision trees and combining their predictions to improve the overall performance and accuracy of the model. The name "Random Forest" comes from the idea of creating a "forest" of decision trees, where each tree is trained on a random subset of the training data and features.

In the above code, the Random Forest Classifier is used to predict the loan status (whether it is approved or not) based on various features such as credit history, income, loan amount, etc. The algorithm is trained on a subset of the data (x_train and y_train) and then used to predict the loan status of the remaining data (x_test). The accuracy of the model is calculated by comparing the predicted values (y_pred) with the actual values (y_test) using the confusion matrix.

The Random Forest Classifier is chosen because of its ability to handle large datasets with high dimensionality, and its robustness to outliers and noise. It also provides a measure of feature importance, which can be useful in identifying the most relevant features for the prediction task.



CHAPTER 3: CODE OF THE PROJECT

FRONTEND:

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 Prediction</title>
  <style>
   body {
     font-family: Arial, sans-serif;
     margin: 0;
     padding: 0;
     background-image:url('https://images.unsplash.com/photo-1551434678-e076c223a692?ixlib=rb-
1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=2850&q=80') ;
     background-size: cover;
   }
   .container {
     max-width: 1200px;
     margin: 0 auto;
     padding: 0 20px;
   }
   .navbar {
     background-color: #4F46E5;
     color: #FFF;
     padding: 20px 0;
   }
   .navbar ul {
     list-style-type: none;
     margin: 0;
     padding: 0;
     text-align: right;
   }
   .navbar ul li {
     display: inline;
     margin-left: 20px;
   }
   .logo img {
     height: 40px;
     width: auto;
     vertical-align: middle;
   }
```

```
    .hero {
      padding: 100px 0;
      text-align: center;
    }
    .hero h1 {
      font-size: 3rem;
      margin-bottom: 20px;
      color: #333;
    }
    .hero p {
      font-size: 1.2rem;
      color: #666;
    }
    .btn {
      display: inline-block;
      padding: 10px 20px;
      background-color: #4F46E5;
      color: #FFF;
      text-decoration: none;
      border-radius: 5px;
      transition: background-color 0.3s ease;
    }
    .btn:hover {
      background-color: #6E63FF;
    }
  </style>
</head>
<body>
  <div class="navbar">
    <div class="container">
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="./predict">Prediction</a></li>
        <li><a href="#">About us</a></li>
        <li><a href="#">Contact</a></li>
      </ul>
    </div>
  </div>
  <div class="hero">
    <div class="container">
      <h1>Loan Prediction</h1>
      <p>Machine Learning</p>
      <a href="./predict" class="btn">Prediction</a>
    </div>
  </div>
```

```
</body>
</html>
```

Prediction.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Loan Prediction</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      background-color: #f3f4f6;
      /*background-size: cover;
      background-image: url('https://protium.co.in/wp-content/uploads/2022/11/7-Simple-Tips-to-Get-Your-SME-Loan-Approved.png');*/
    }
    .container {
      max-width: 800px;
      margin: 0 auto;
      padding: 20px;
    }
    h1 {
      font-size: 24px;
      font-weight: bold;
      text-align: center;
      margin-bottom: 20px;
    }
    p {
      font-size: 16px;
      text-align: center;
      margin-bottom: 30px;
    }
    form {
      background-color: #ffffff;
      padding: 20px;
      border-radius: 10px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

    }
    label {
      font-weight: bold;
```

```
      display: block;
      margin-bottom: 5px;
    }
    input, select {
      width: calc(100% - 22px);
      padding: 10px;
      margin-bottom: 15px;
      border: 1px solid #ccc;
      border-radius: 5px;
    }
    button {
      background-color: #4F46E5;
      color: #ffffff;
      padding: 10px 20px;
      border: none;
      border-radius: 5px;
      cursor: pointer;
      font-size: 16px;
      width: calc(100% - 22px);
    }
    button:hover {
      background-color: #6E63FF;
    }
    res{
      align-items:center ;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Loan Prediction Project</h1>
    <p>Fill the form for prediction</p>
    <form action="/predict" method="POST">
      <div class="res">
      <h2><b>{{prediction_text}}</b></h2>
      </div>
      <label for="gender">Gender</label>
      <select id="gender" name="gender"aria-label="Default select example">
        <option selected>-- select gender --</option>
        <option value="Male">Male</option>
        <option value="Female">Female</option>
      </select>

      <label for="married">Married Status</label>
      <select id="married" name="married">
```

```
  <option selected>-- select marriage status--</option>
  <option value="Yes">Yes</option>
  <option value="No">No</option>
</select>

<label for="dependents">Dependents</label>
<select id="dependents" name="dependents">
  <option selected>-- select dependents--</option>
  <option value="0">0</option>
  <option value="1">1</option>
  <option value="2">2</option>
  <option value="3+">3+</option>
</select>

<label for="education">Education</label>
<select id="education" name="education">
  <option selected>-- select education --</option>
  <option value="Graduate">Graduate</option>
  <option value="Not Graduate">Not Graduate</option>
</select>

<label for="employed">Self Employed</label>
<select id="employed" name="employed">
  <option selected>-- self employeed? --</option>
  <option value="Yes">Yes</option>
  <option value="No">No</option>
</select>

<label for="credit">Credit History</label>
<select id="credit" name="credit"placeholder="Credit History">
  <option selected>-- select CreditHistory --</option>
  <option value="1.000000">1.000000</option>
  <option value="0.000000">0.000000</option>
  <option value="0.842199">0.842199</option>
</select>

<label for="area">Property Area</label>
<select id="area" name="area">
  <option selected>-- select Area type --</option>
  <option value="Semiurban">Semiurban</option>
  <option value="Urban">Urban</option>
  <option value="Rural">Rural</option>
</select>

<label for="ApplicantIncome">Applicant Income</label>
```

```
    <input type="text" id="ApplicantIncome" name="ApplicantIncome" placeholder="Enter
Applicant Income">

    <label for="CoapplicantIncome">Coapplicant Income</label>
    <input type="text" id="CoapplicantIncome" name="CoapplicantIncome" placeholder="Enter
Coapplicant Income">

    <label for="LoanAmount">Loan Amount</label>
    <input type="text" id="LoanAmount" name="LoanAmount" placeholder="Enter Loan Amount">

    <label for="Loan_Amount_Term">Loan Amount Term</label>
    <input type="text" id="Loan_Amount_Term" name="Loan_Amount_Term" placeholder="Enter
Loan Amount Term">

    <button type="submit">Predict</button>
  </form>
  <a href="./">Back</a>
 </div>
</body>
</html>
```

BACKEND:

App.py

```
from flask import Flask,  request, render_template
import pickle
import numpy as np
app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))
@app.route('/')
def home():
   return render_template("index.html")
@app.route('/predict', methods=['GET', 'POST'])
def predict():
   if request.method ==  'POST':
      gender = request.form['gender']
      married = request.form['married']
      dependents = request.form['dependents']
      education = request.form['education']
      employed = request.form['employed']
      credit = float(request.form['credit'])
      area = request.form['area']
      ApplicantIncome = float(request.form['ApplicantIncome'])
      CoapplicantIncome = float(request.form['CoapplicantIncome'])
      LoanAmount = float(request.form['LoanAmount'])
      Loan_Amount_Term = float(request.form['Loan_Amount_Term'])
```

```
if (gender == "Male"):
    male=1
else:
    male=0
if(married=="Yes"):
    married_yes = 1
else:
    married_yes=0
if(dependents=='1'):
    dependents_1 = 1
    dependents_2 = 0
    dependents_3 = 0
elif(dependents == '2'):
    dependents_1 = 0
    dependents_2 = 1
    dependents_3 = 0
elif(dependents=="3+"):
    dependents_1 = 0
    dependents_2 = 0
    dependents_3 = 1
else:
    dependents_1 = 0
    dependents_2 = 0
    dependents_3 = 0
if (education=="Not Graduate"):
    not_graduate=1
else:
    not_graduate=0
if (employed == "Yes"):
    employed_yes=1
else:
    employed_yes=0
if(area=="Semiurban"):
    semiurban=1
    urban=0
elif(area=="Urban"):
    semiurban=0
    urban=1
else:
    semiurban=0
    urban=0
ApplicantIncomelog = np.log(ApplicantIncome)
totalincomelog = np.log(ApplicantIncome+CoapplicantIncome)
LoanAmountlog = np.log(LoanAmount)
Loan_Amount_Termlog = np.log(Loan_Amount_Term)
```

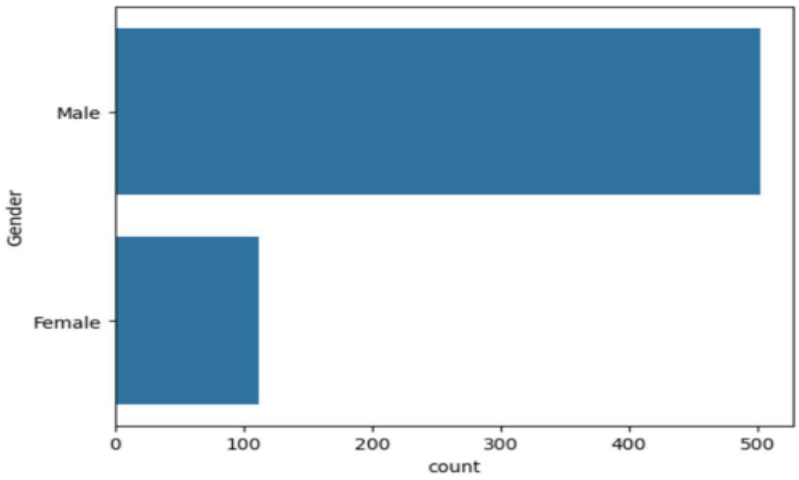
```
        prediction = model.predict([[credit, ApplicantIncomelog,LoanAmountlog,
Loan_Amount_Termlog, totalincomelog, male, married_yes, dependents_1, dependents_2,
dependents_3, not_graduate, employed_yes,semiurban, urban ]])
        if(prediction=="N"):
            prediction="Not Approved"
        else:
            prediction="Approved"
        return render_template("prediction.html", prediction_text="Loan is {}".format(prediction))
    else:
        return render_template("prediction.html")
if __name__ == "__main__":
    app.run(debug=True)
```

Loan Prediction.ipynb

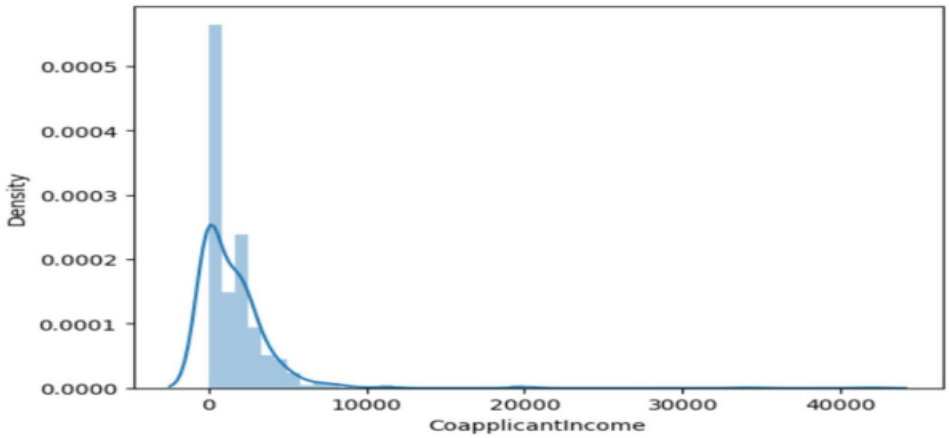
```
import pandas as pd
import numpy as np
df=pd.read_csv("train.csv")
df.head()
```

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

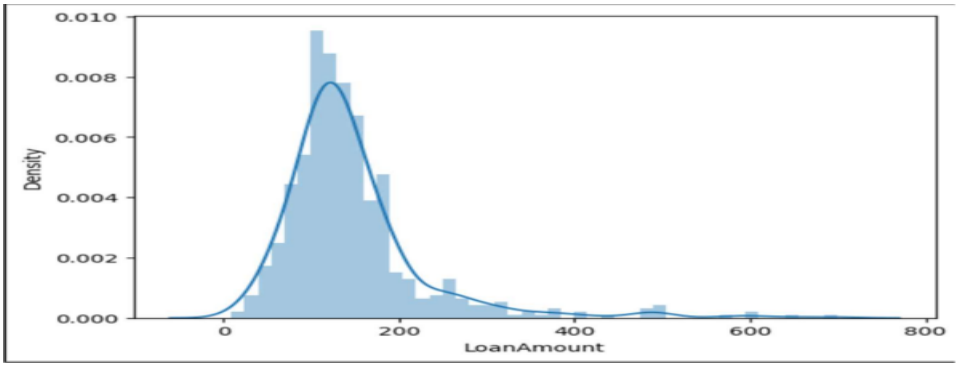
```
df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].mean())
df['Loan_Amount_Term'] = df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mean())
df['Credit_History'] = df['Credit_History'].fillna(df['Credit_History'].mean())
df['Gender'].mode()[0]
df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])
df['Married'] = df['Married'].fillna(df['Married'].mode()[0])
df['Dependents'] = df['Dependents'].fillna(df['Dependents'].mode()[0])
df['Self_Employed'] = df['Self_Employed'].fillna(df['Self_Employed'].mode()[0])
df['Total_income'] = df['ApplicantIncome']+df['CoapplicantIncome']
df['ApplicantIncomeLog'] = np.log(df['ApplicantIncome'])
df['CoapplicantIncomeLog'] = np.log(df['CoapplicantIncome'])
df['Loan_Amount_Term_Log'] = np.log(df['Loan_Amount_Term'])
df['Total_Income_Log'] = np.log(df['Total_income'])
import seaborn as sns
sns.countplot(df['Gender'])
```



sns.distplot(df.CoapplicantIncome)



sns.distplot(df.LoanAmount)




```
cols = ['ApplicantIncome', 'CoapplicantIncome', "LoanAmount", "Loan_Amount_Term",
"Total_income", 'Loan_ID', 'CoapplicantIncomeLog']
df = df.drop(columns=cols, axis=1)
d1 = pd.get_dummies(df['Gender'], drop_first= True)
d2 = pd.get_dummies(df['Married'], drop_first= True)
d3 = pd.get_dummies(df['Dependents'], drop_first= True)
d4 = pd.get_dummies(df['Education'], drop_first= True)
```

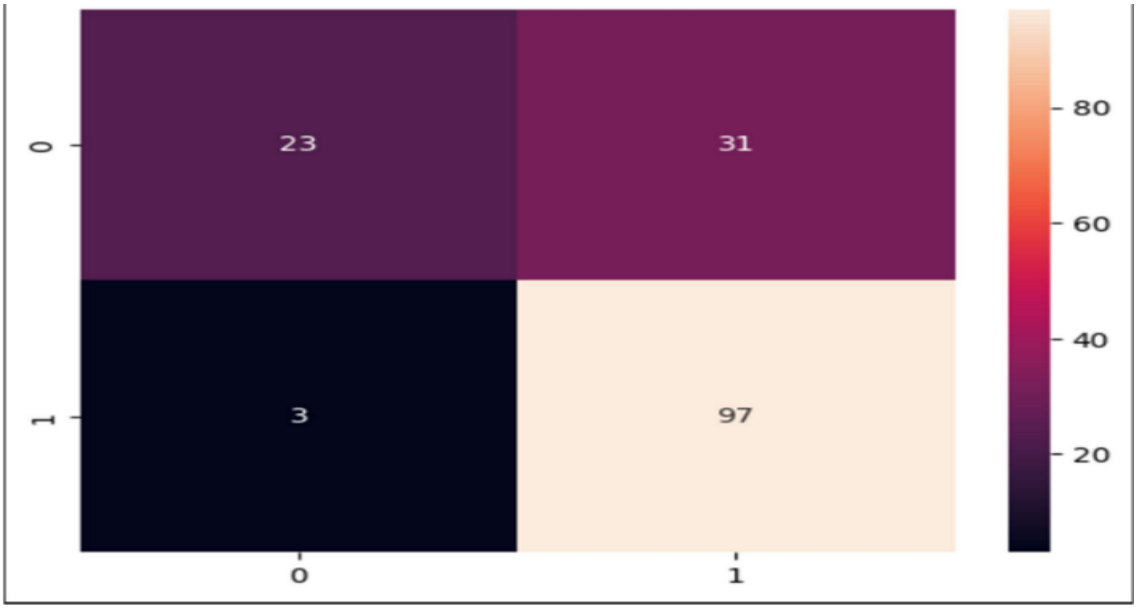
```
d5 = pd.get_dummies(df['Self_Employed'], drop_first= True)
d6 = pd.get_dummies(df['Property_Area'], drop_first= True)
df1 = pd.concat([df, d1, d2, d3, d4, d5, d6], axis = 1)
df=df1
cols = ['Gender', 'Married', "Dependents", "Education", "Self_Employed", 'Property_Area']
df = df.drop(columns=cols, axis=1)
test = pd.read_csv("test.csv")
test['LoanAmount']=test['LoanAmount'].fillna(test['LoanAmount'].mean())
test['Loan_Amount_Term']=test['Loan_Amount_Term'].fillna(test['Loan_Amount_Term'].mean())
test['Credit_History']=test['Credit_History'].fillna(test['Credit_History'].mean())
test['Gender']=test['Gender'].fillna(test['Gender'].mode()[0])
test['Married']=test['Married'].fillna(test['Married'].mode()[0])
test['Dependents']=test['Dependents'].fillna(test['Dependents'].mode()[0])
test['Self_Employed']=test['Self_Employed'].fillna(test['Self_Employed'].mode()[0])
test['Total_income'] = test['ApplicantIncome']+test['CoapplicantIncome']
np.seterr(divide = 'ignore')
test['ApplicantIncomeLog'] = np.where(test['ApplicantIncome']>0,np.log(test['ApplicantIncome']),0)
test['CoapplicantIncomeLog'] = np.log(test['CoapplicantIncome'])
test['LoanAmountLog'] = np.log(test['LoanAmount'])
test['Loan_Amount_Term_Log'] = np.log(test['Loan_Amount_Term'])
test['Total_Income_Log'] = np.log(test['Total_income'])
cols = ['ApplicantIncome', 'CoapplicantIncome', "LoanAmount", "Loan_Amount_Term",
"Total_income", 'Loan_ID', 'CoapplicantIncomeLog']
test = test.drop(columns=cols, axis=1)
t1 = pd.get_dummies(test['Gender'], drop_first= True)
t2 = pd.get_dummies(test['Married'], drop_first= True)
t3 = pd.get_dummies(test['Dependents'], drop_first= True)
t4 = pd.get_dummies(test['Education'], drop_first= True)
t5 = pd.get_dummies(test['Self_Employed'], drop_first= True)
t6 = pd.get_dummies(test['Property_Area'], drop_first= True)
df1 = pd.concat([test, t1, t2, t3, t4, t5, t6], axis = 1)
test=df1
cols = ['Gender', 'Married', "Dependents", "Education", "Self_Employed", 'Property_Area']
test = test.drop(columns=cols, axis=1)
x = df.drop(columns=['Loan_Status'], axis=1)
y = df['Loan_Status']
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=42)
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(x_train, y_train)
print("Accuracy is", model.score(x_test, y_test)*100)
```

Output:

Accuracy is 77.92207792207793

```
from sklearn.tree import DecisionTreeClassifier
```

```
model2 = DecisionTreeClassifier()
model2.fit(x_train, y_train)
print("Accuracy is", model2.score(x_test, y_test)*100)
```
Output:-
Accuracy is 72.07792207792207
```
from sklearn.linear_model import LogisticRegression
model3 = LogisticRegression()
model3.fit(x_train, y_train)
print("Accuracy is", model3.score(x_test, y_test)*100)
```
Output:-
Accuracy is 77.27272727272727
```
from sklearn.metrics import confusion_matrix
y_pred = model.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
cm
```
Output:-
```
array([[23, 31],
 [ 3, 97]], dtype=int64)
import seaborn as sns
import matplotlib as plt
from sklearn.metrics import confusion_matrix
y_pred = model.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm,annot=True)
plt.savefig('confusion.png')
cm
```




```
import pickle
file=open("model.pkl", 'wb')
```

pickle.dump(model, file)

Steps to execute the project:

1. Install Python: Ensure you have Python installed on your machine.
2. Install Required Libraries: You need several Python libraries including pandas, numpy, seaborn, scikit-learn, and flask. You can install these using pip.

pip install pandas numpy seaborn scikit-learn flask

3. Prepare Your Data
   - Read the Dataset
   - Handle Missing Values
   - Feature Engineering
   - Data Visualization
   - One-Hot Encoding
4. Prepare the Test Data - Read and Process Test Data
5. Model Training and Evaluation
   - Split the Data
   - Train Models
   - Evaluate Model with Confusion Matrix
6. Save the Model - Serialize the Model
7. Create the Flask Application
   - Flask App (`app.py`)
   - HTML Template (`index.html`)
8. Run the Flask App
   - Start the Flask Server

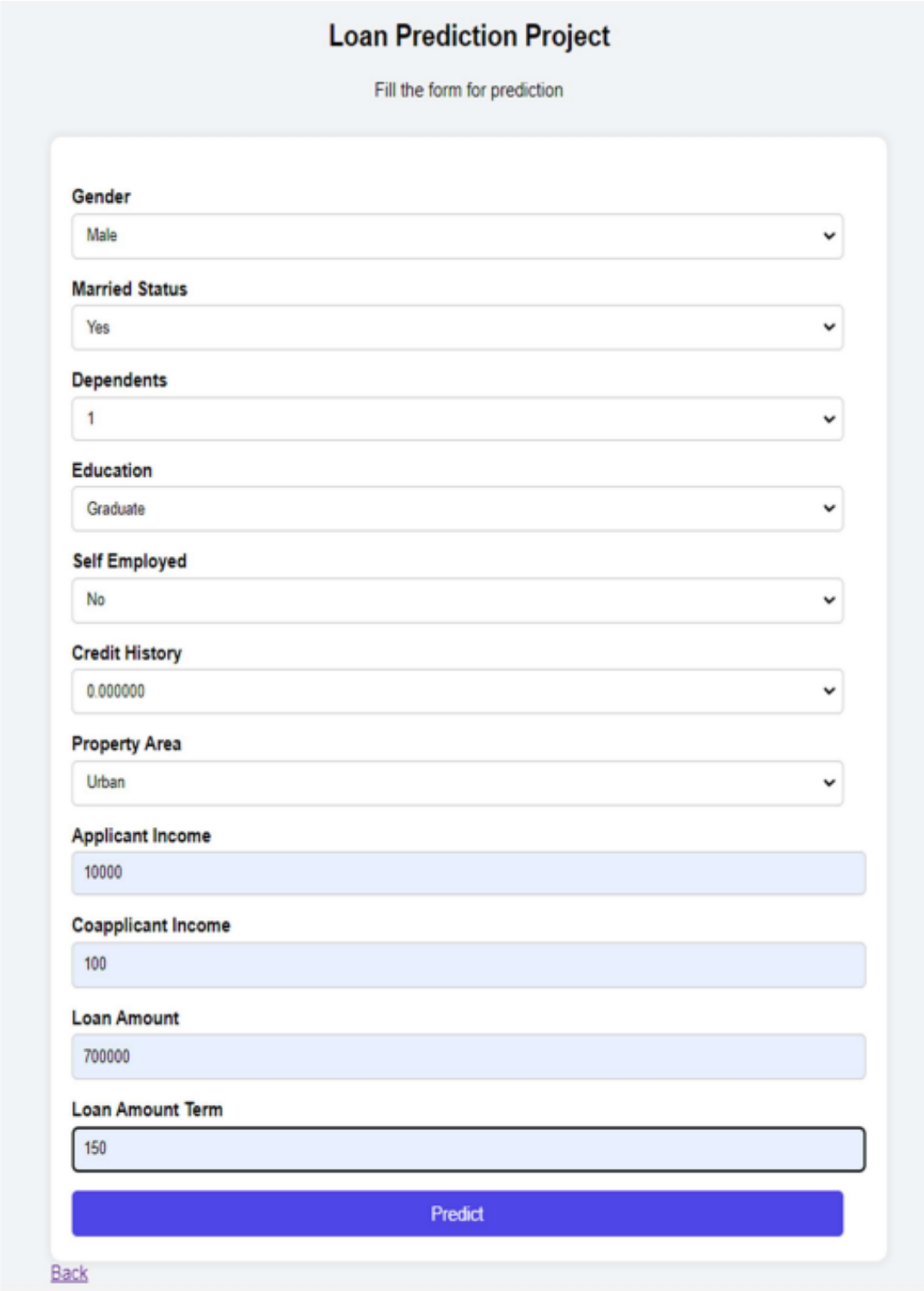
python app.py

   - Access the Application

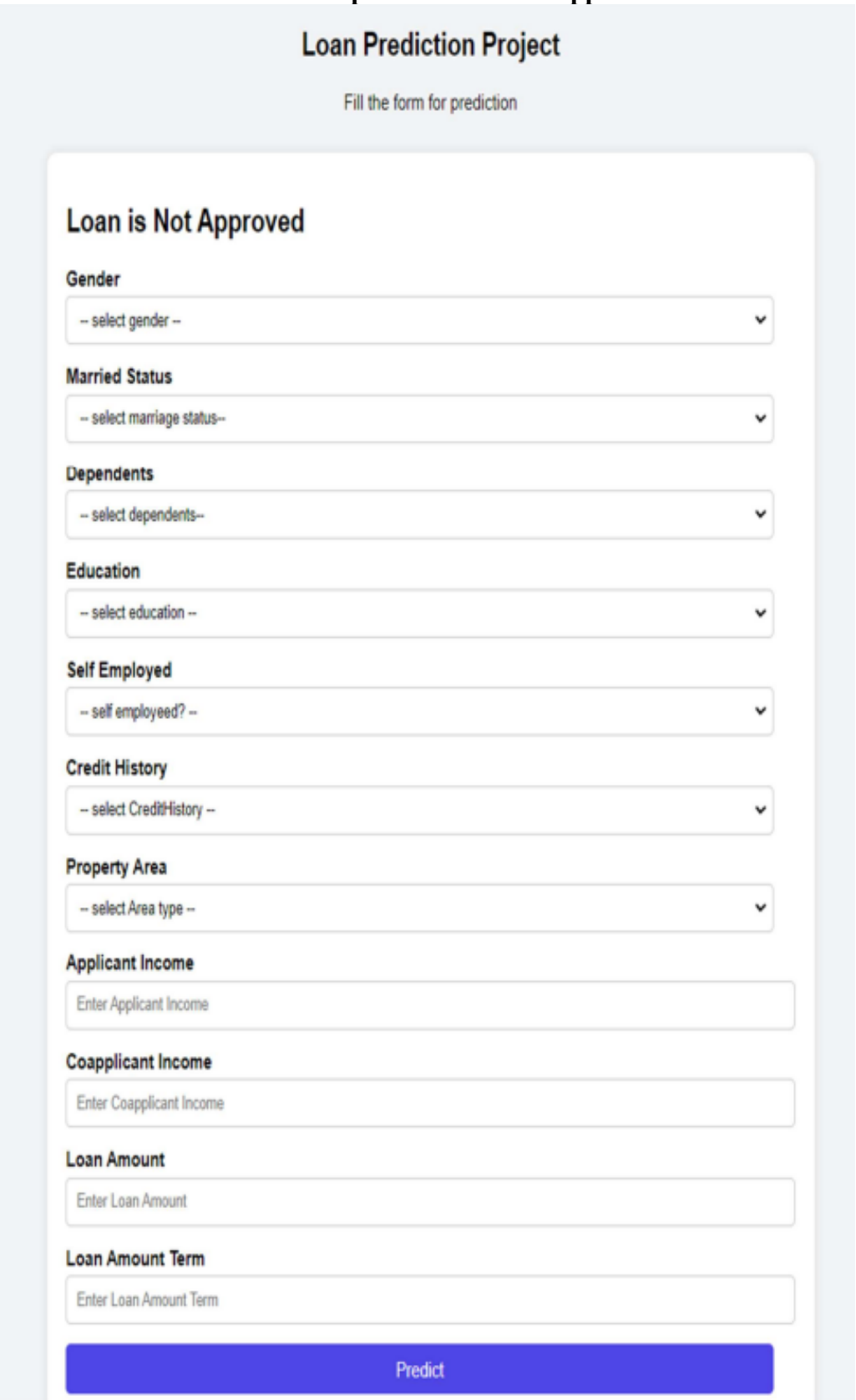
Open a web browser and go to http://127.0.0.1:5000/. You will see the loan prediction form

CHAPTER 4: OUTPUT OF THE PROJECT

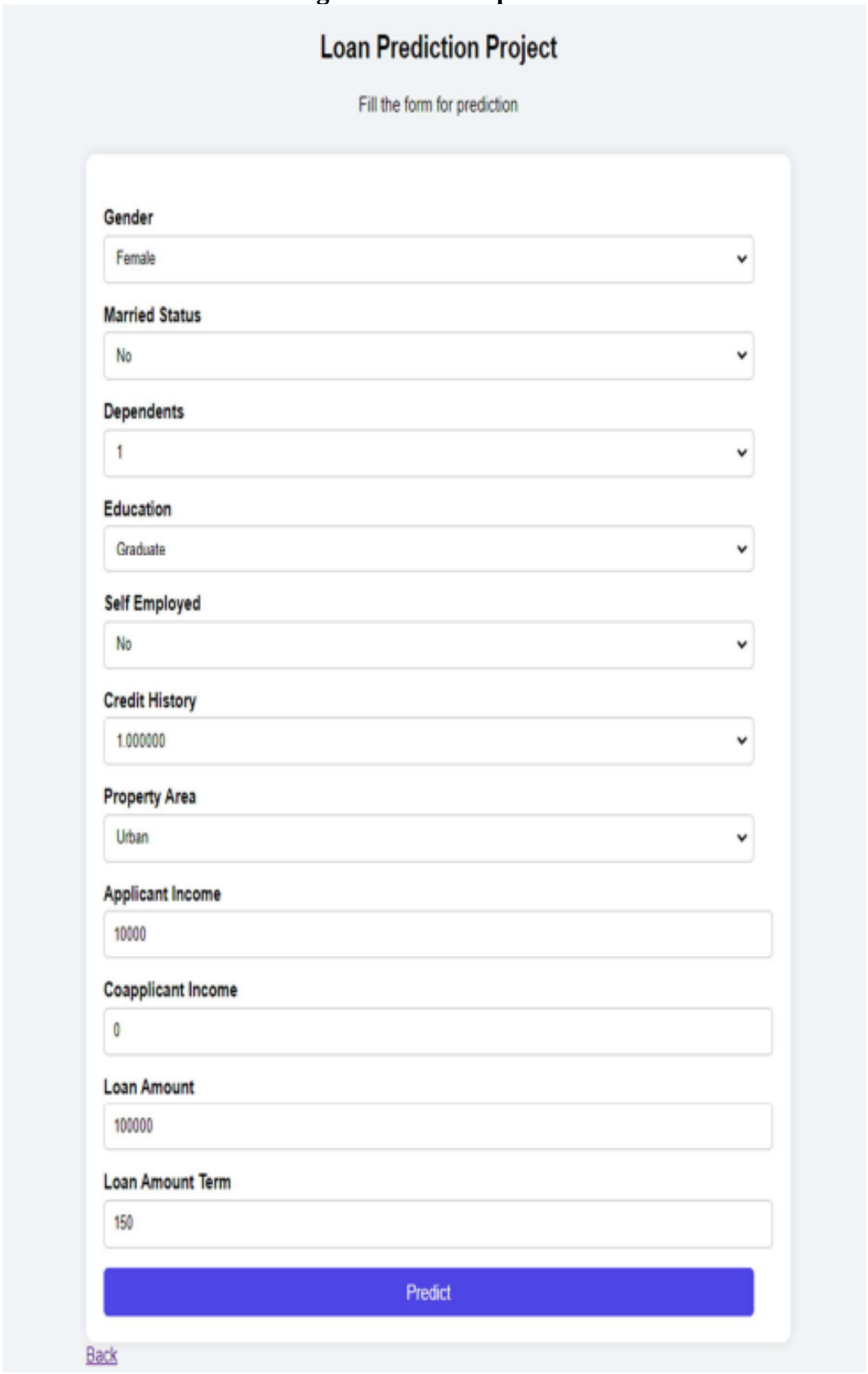
Filling the details for prediction



Predicted Output – Loan is not Approved



Filling the details for prediction



Predicted Output- Loan is Approved.