# 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.

# 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.

[Declaration 1](#_Toc193301946)

[Acknowledgement 2](#_Toc193301947)

[Chapter 1 : Introduction 4](#_Toc193301948)

[Chapter 2 : Requirement and Analysis 4](#_Toc193301949)

[Chapter 3 : System Design 5](#_Toc193301950)

[Chapter 4 : Implementation 5](#_Toc193301951)

[Chapter 5 : Results and Conclusions 5](#_Toc193301952)

[Plagiarism Report 5](#_Toc193301953)

[Chapter 1 : Introduction 6](#_Toc193301954)

[1.1 Background 7](#_Toc193301955)

[1.2 Objectives 7](#_Toc193301956)

[1.3 Scope of the Project 7](#_Toc193301957)

[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: 8](#_Toc193301958)

[ Data preprocessing and feature engineering. 8](#_Toc193301959)

[ Model training and evaluation. 8](#_Toc193301960)

[ Deployment of the model as a web application. 8](#_Toc193301961)

[ Providing a user-friendly interface for inputting customer details and receiving predictions. 8](#_Toc193301962)

[2.1 Problem Definition 9](#_Toc193301963)

[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. 9](#_Toc193301964)

[2.2 Requirement Specifications 9](#_Toc193301965)

[ Programming Languages: Python 9](#_Toc193301966)

[ Libraries: Pandas, NumPy, Scikit-learn, Flask, Joblib, Seaborn, Matplotlib 9](#_Toc193301967)

[ Development Environment: Jupyter Notebook, VS Code 9](#_Toc193301968)

[ Web Framework: Flask 9](#_Toc193301969)

[ Frontend: HTML, CSS, Bootstrap 9](#_Toc193301970)

[ Version Control: Git 9](#_Toc193301971)

[For Deployment (If Hosted Online) 9](#_Toc193301972)

[2. Minimum Hardware Requirements For Development 9](#_Toc193301973)

[Chapter 3 : System Design 12](#_Toc193301974)

[3.2 Test Cases for the project 12](#_Toc193301975)

[Dataset loading and preprocessing 12](#_Toc193301976)

[Model Training and Evaluation 13](#_Toc193301977)

[User interface testing 13](#_Toc193301978)

[4.1 Program Code 14](#_Toc193301979)

[5.3 Testing Approaches 31](#_Toc193301980)

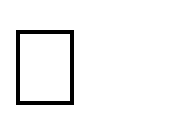
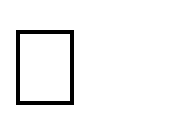
[5.1 Test Report 32](#_Toc193301981)

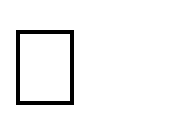
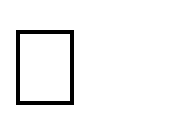
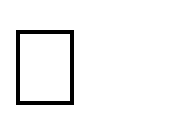
[5.2 Conclusion 33](#_Toc193301982)

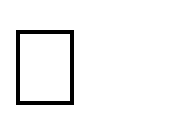
[5.3 Limitation of the System 34](#_Toc193301983)

[5.4 Future Scope 34](#_Toc193301984)

[References 35](#_Toc193301985)

[ Dataset Source – Kaggle Dataset https://www.kaggle.com  Machine Learning Framework – TensorFlow and Scikit-Learn Official 35](#_Toc193301986)

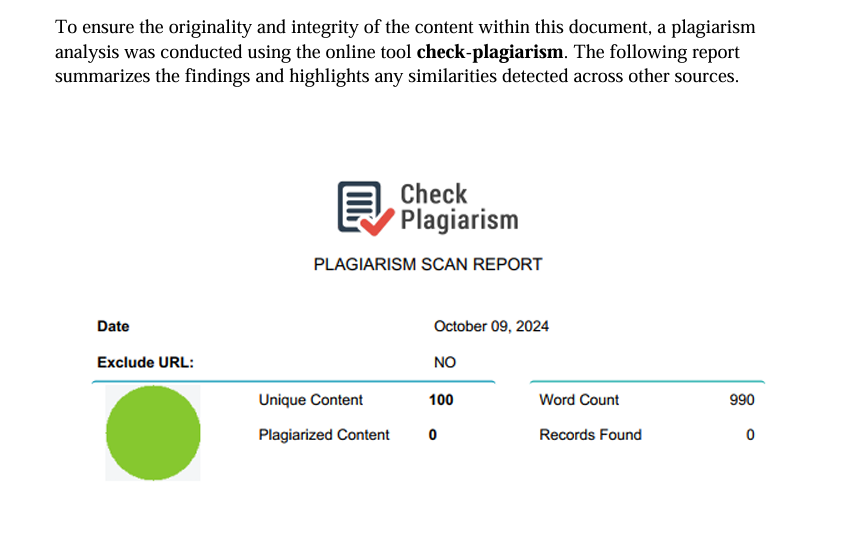
[ Flask Web Framework – Flask Official Documentation https://flask.palletsprojects.com/  Medical Research on PCOS – Rotterdam ESHRE/ASRM-Sponsored PCOS Consensus Workshop Group https://academic.oup.com/humrep/article/19/1/41/2902555  Online Machine Learning Community for Support and Discussions – 35](#_Toc193301987)

[ GitHub Repository for Project Code – Community Contributions https://github.com/ 35](#_Toc193301988)

**50**

# 

# Plagiarism Report

****

## 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

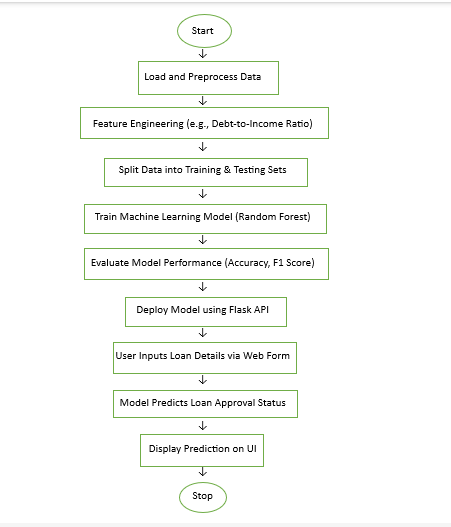
* **Processor**: Intel Core i3 (or AMD equivalent)
* **RAM**: 4GB (8GB recommended for smooth performance)
* **Storage**: 20GB of free disk space
* **GPU**: Not required (since basic ML models do not need GPU acceleration) **For Hosting/Deployment (If Deployed Online)**
* **CPU**: 1 vCPU (Virtual CPU)
* **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.

|  |  |
| --- | --- |
|  | 9 |

## Chapter 3 : System Design

❖ **3.1 System Flow Chart**



## 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 |
| **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 |

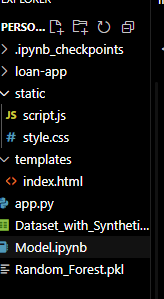
The **test cases for the Loan Approval Prediction project** are structured across different aspects of the system to ensure a **fully functional, reliable, and user-friendly application**.

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.

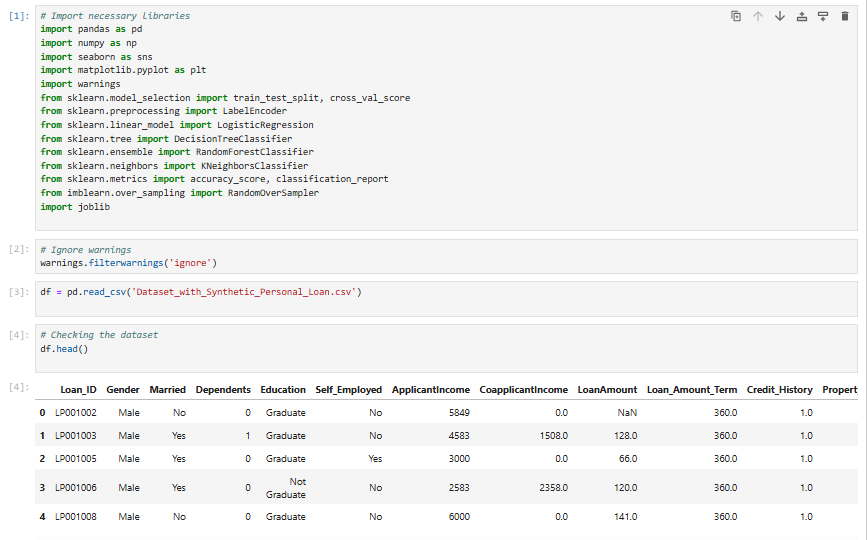
**Chapter 4 : Implementation**

## 4.1 Program Code

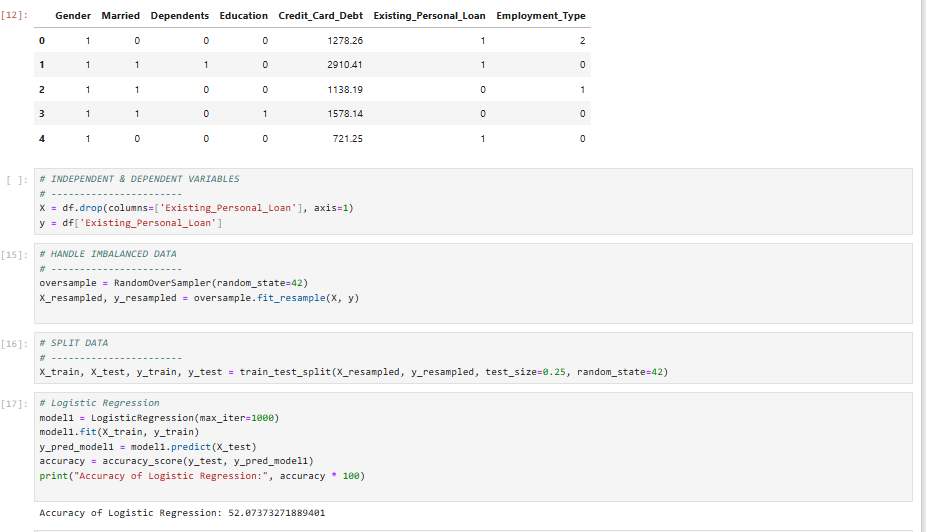
**Program Structure :**

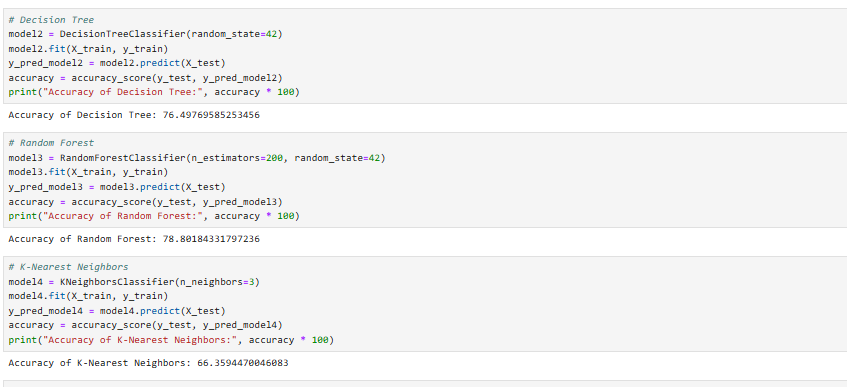


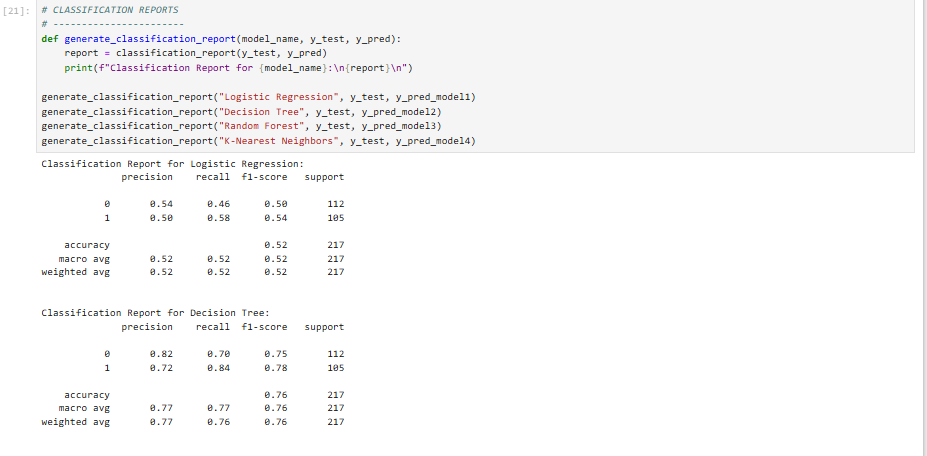
**Code :**

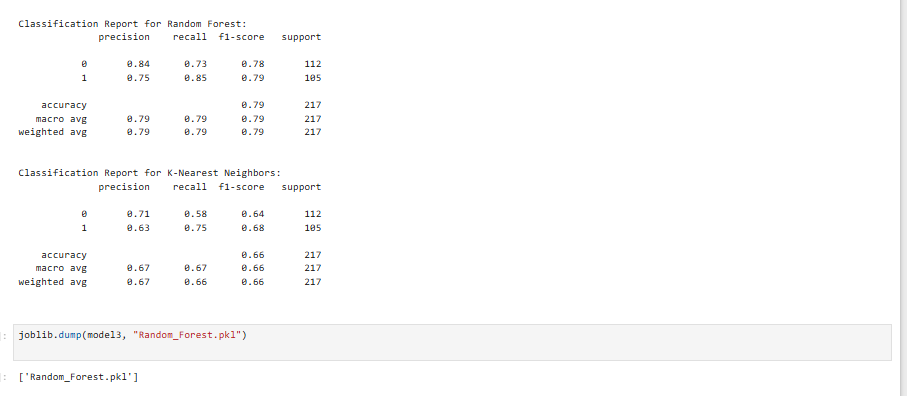
****













**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:

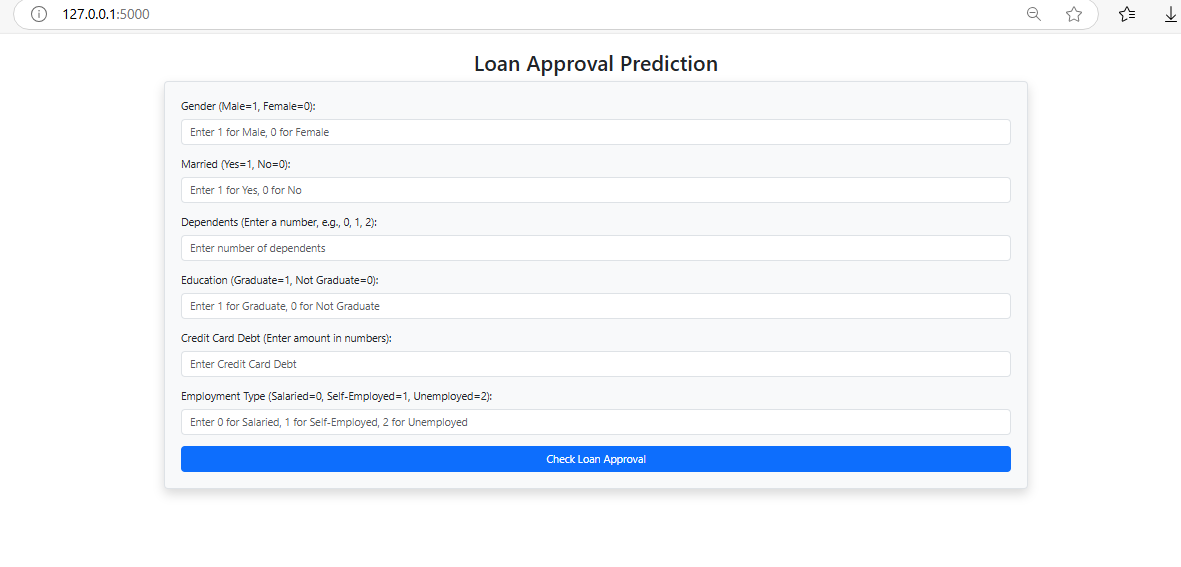
**return** jsonify({"error": str(e)}), 500

**if** \_\_name\_\_ **==** '\_\_main\_\_':

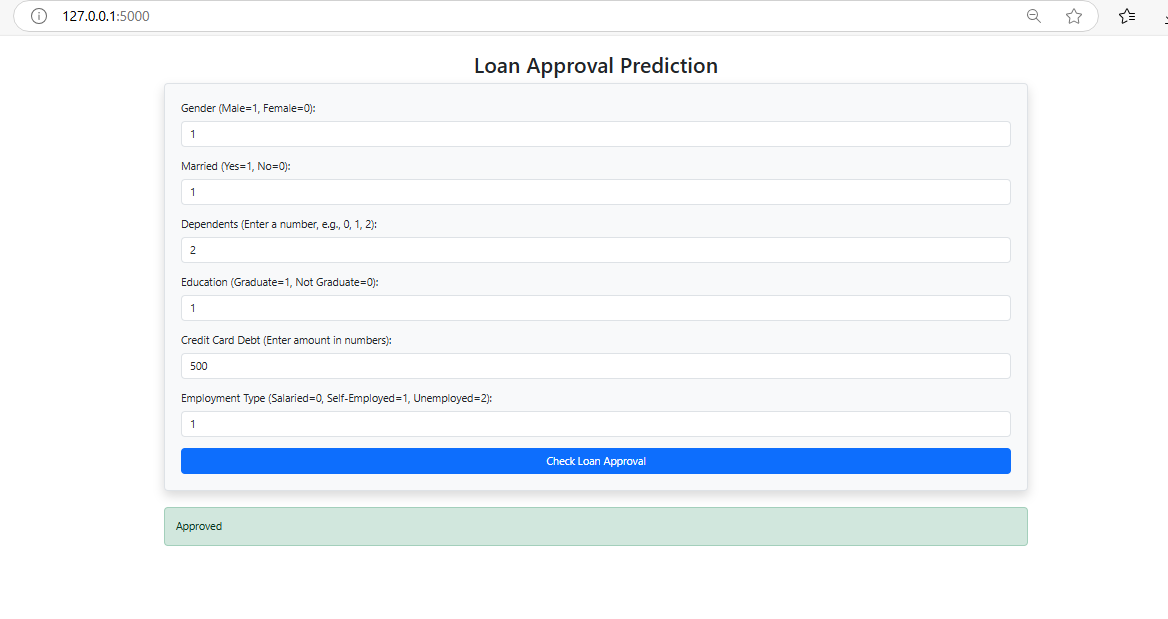
    app.run(**debug=**True)

**Output/Screenshots:**

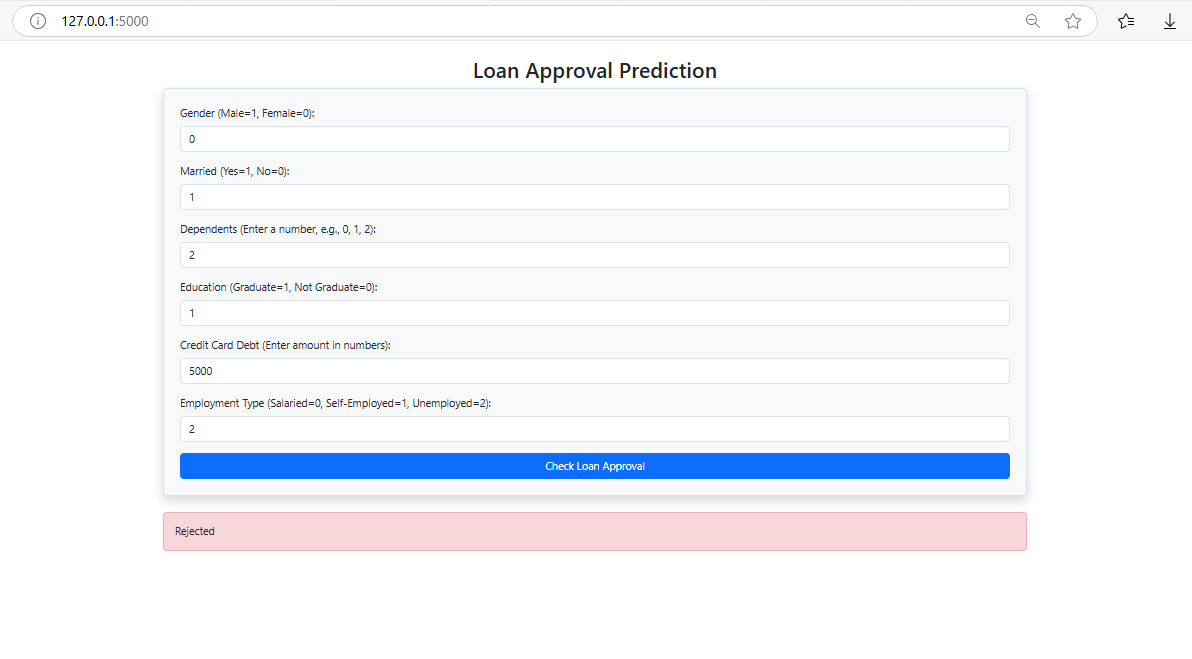
**Landing page:**

****

**Approved Loan:**



**Rejected Loan:**



## 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

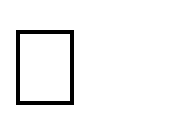
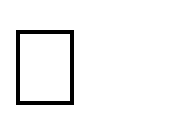
1. **Enhanced Features**: Incorporate additional features like credit score, loan history, and financial behavior for better predictions.
2. **Advanced Techniques**: Explore deep learning or ensemble methods to improve accuracy.
3. **Larger Dataset**: Expand the dataset to include more diverse loan applications 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](https://www.kaggle.com/) Machine Learning Framework – TensorFlow and Scikit-Learn Official

**Documentation** [**https://www.tensorflow.org/**](https://www.tensorflow.org/) [**https://scikit-learn.org/**](https://scikit-learn.org/)

 **Python Programming Language – Python Official Documentation** [**https://docs.python.org/**](https://docs.python.org/)  **Jupyter Notebook for Model Development – Project Jupyter Documentation**

[**https://jupyter.org/**](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://www.kaggle.com/) [**https://stackoverflow.com/**](https://stackoverflow.com/)

## GitHub Repository for Project Code – Community Contributions <https://github.com/>