LOAN PREDICTION SYSTEM USING STREAMLIT

by

G. Charan Sai

MST03-0054

Submitted to Scifor Technologies

Meta

Scifor Technologies

Script. Sculpt. Socialize

UNDER GUIDIANCE OF

Urooj Khan

TABLE OF CONTENTS

ABSTRACT

This project focuses on developing a machine learning pipeline for predicting loan approval using a variety of classification models. The dataset, which includes information on loan applicants, is preprocessed and split into training and testing sets. The target variable, loan status, is mapped into binary values representing approval or rejection.

The preprocessing involves handling missing values, encoding categorical variables, and scaling the numerical features. A range of classification algorithms, such as Logistic Regression, Decision Trees, Random Forests, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Naive Bayes, and Linear Discriminant Analysis (LDA), are trained and evaluated using cross-validation and grid search for hyperparameter tuning.

The best-performing model is then saved using Python's pickle module for later use in deployment. A Streamlit web application is also developed, which allows users to input loan application details through an interactive form. Based on the inputs, the model predicts whether the loan will be approved or not, providing feedback to the user.

This project leverages Python's machine learning ecosystem, including libraries like pandas, numpy, scikit-learn, matplotlib, and seaborn, to build a robust loan prediction system. The application demonstrates the integration of machine learning models into a user-friendly interface, making it applicable in real-world financial services.

INTRODUCTION

In the rapidly evolving field of data science, predictive modelling has become a cornerstone for decision-making processes across various industries. One significant application of predictive analytics is in the financial sector, particularly in determining loan approval. Predicting loan approval involves evaluating an applicant's likelihood of repaying a loan based on numerous factors such as income, credit history, and employment status. Machine learning models offer a powerful tool to automate and enhance this decision-making process.

This project focuses on developing a comprehensive machine learning pipeline aimed at predicting loan approval based on applicant data. The primary objective is to build, evaluate, and deploy a robust model that can predict whether a loan application will be approved or rejected. Leveraging the power of Python's machine learning ecosystem, including libraries such as scikit-learn, pandas, and numpy, this project implements several classification algorithms ranging from traditional methods like Logistic Regression and Decision Trees to more sophisticated techniques such as Random Forests, Support Vector Machines (SVM), and K-Nearest Neighbors (KNN).

The methodology begins with preprocessing the dataset, which includes handling missing values, encoding categorical variables, and scaling numerical features to optimize the performance of machine learning algorithms. Multiple models are trained and validated using cross-validation techniques, and hyperparameter tuning is performed to identify the most effective model. The best-performing model is then saved and deployed using a Streamlit web application, allowing users to interactively predict loan approval by inputting relevant details.

Through this project, we aim to demonstrate the potential of machine learning in automating financial decision-making, highlighting the effectiveness of different classification algorithms, and showcasing the deployment of predictive models in a real-world application. This project not only contributes to the practical application of machine learning in finance but also provides insights into the end-to-end process of model development, from data preprocessing to deployment.

TECHNOLGY USED

The technology stack used in this project can be summarized as follows:

1. Programming Language: Python

   Python serves as the foundation for implementing the machine learning models and building the deployment interface.

2. Machine Learning Libraries:

   - Scikit-learn: For implementing and evaluating a range of machine learning algorithms including Logistic Regression, Decision Trees, Random Forests, SVM, K-Nearest Neighbors, Naive Bayes, and Linear Discriminant Analysis.
   - Pandas: For data manipulation and preprocessing, enabling efficient handling of datasets.
   - NumPy: For numerical computations and array manipulations, supporting efficient data handling.imghdr: Library for identifying image types.

3. Visualization Libraries:

   - Matplotlib: For visualizing model performance through plots such as accuracy and loss curves.
   - Seaborn: For enhancing the visual appeal of plots and gaining deeper insights into the data.

4. Web Application Framework:

   - Streamlit: For building and deploying an interactive web application that allows users to input loan application details and get real-time predictions from the trained machine learning model..

5. Model Persistence:

   - Pickle: For saving and loading trained machine learning models, enabling easy deployment and future predictions.

6. Data Handling and Preprocessing:
    - Handling missing data: Utilizing techniques like filling missing values with medians or modes to prepare data for model training.
    - Feature scaling: Applying StandardScaler to normalize features, ensuring models perform optimally.
    - Encoding categorical variables: Converting categorical data into numerical formats to be fed into machine learning algorithms.
7. Model Building and Hyperparameter Tuning:
    - GridSearchCV: For hyperparameter tuning, ensuring optimal model performance by exploring different combinations of parameters.
    - KFold Cross-Validation: For evaluating model performance across multiple subsets of data to prevent overfitting and ensure generalization.
8. Deployment and Prediction:
    - A complete pipeline was created for loading, preprocessing, and predicting on new data through the Streamlit interface.

These technologies work in unison to deliver an end-to-end loan prediction system, from model development and evaluation to user-friendly deployment.

DATASET INFORMATION

This is a loan prediction dataset designed to assist in building and evaluating machine learning models for predicting loan approval. The dataset includes various features that represent the financial and demographic attributes of loan applicants. The goal is to predict whether an applicant’s loan will be approved based on the provided data.

About this directory

The dataset is organized into multiple features representing different aspects of loan applicants, including:

- Gender: The gender of the applicant (Male/Female).
- Married: The marital status of the applicant (Yes/No).
- Dependents: The number of dependents the applicant has (0, 1, 2, 3+).
- Education: Whether the applicant is a graduate (Graduate/Not Graduate).
- Self_Employed: Whether the applicant is self-employed (Yes/No).
- ApplicantIncome: The monthly income of the applicant.
- CoapplicantIncome: The monthly income of the co-applicant (if any).
- LoanAmount: The loan amount applied for by the applicant.
- Loan_Amount_Term: The term of the loan in months.
- Credit_History: The credit history of the applicant (1: Good, 0: Bad).
- Property_Area: The area where the property is located (Urban/Semiurban/Rural).
- Loan_Status: The target variable, indicating whether the loan was approved (Y) or rejected (N).

This dataset is ideal for learners and practitioners looking to develop and assess machine learning models focused on predicting loan approval based on a variety of applicant features.

METHODOLOGY

The methodology followed in this project consists of several key steps:

1. Importing Libraries:
    - ➢ The process begins by importing essential libraries such as pandas, NumPy, scikit-learn, matplotlib, and seaborn for data manipulation, model building, and visualization. Additionally, Streamlit is used to create the web application for deploying the model.
2. Loading and Preprocessing Data:
    - ➢ The dataset, containing loan applicant details, is loaded using pandas. This includes both training and testing datasets.
    - ➢ Data preprocessing involves handling missing values, encoding categorical variables into numerical representations, and scaling features using StandardScaler. These preprocessing steps ensure that the data is clean and suitable for feeding into machine learning models.
3. Model Architecture:
    - ➢ A range of machine learning models is employed, including Logistic Regression, Decision Trees, Random Forests, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Naive Bayes, and Linear Discriminant Analysis (LDA).
    - ➢ Each model is implemented using scikit-learn’s APIs, with specific configurations for their respective hyperparameters.
4. Hyperparameter Tuning:
    - ➢ Hyperparameter tuning is performed using GridSearchCV to identify the best model configurations. This step involves cross-validation to ensure the model generalizes well to unseen data.
5. Model Training:
    - ➢ The models are trained on the preprocessed training data using scikit-learn’s fit() method. During training, the dataset is split into training and validation sets to monitor the models' performance and prevent overfitting.

6. Model Evaluation:

   - The trained models are evaluated using k-fold cross-validation. Accuracy scores are computed for each model to assess their performance, and a comparison is made to determine which model performs best.
   - Additionally, the models are tested on the validation set to further evaluate their accuracy and performance.

7. Visualization:

   - The performance of each model is visualized using boxplots to compare the accuracy of different models. Matplotlib and Seaborn are utilized to generate these plots, which help in visualizing the distribution of model performance across different validation folds.

8. Model Selection and Saving:

   - The best-performing model is selected based on cross-validation results. The selected model is then saved to a file using the pickle library, enabling future use without needing to retrain the model.

9. Web Application Deployment:

   - A web application is built using Streamlit to allow users to interact with the loan prediction model. Users can input their loan application details, and the app provides real-time predictions on whether the loan will be approved or not.
   - The model is loaded into the Streamlit app, and user inputs are processed to generate predictions based on the trained model.

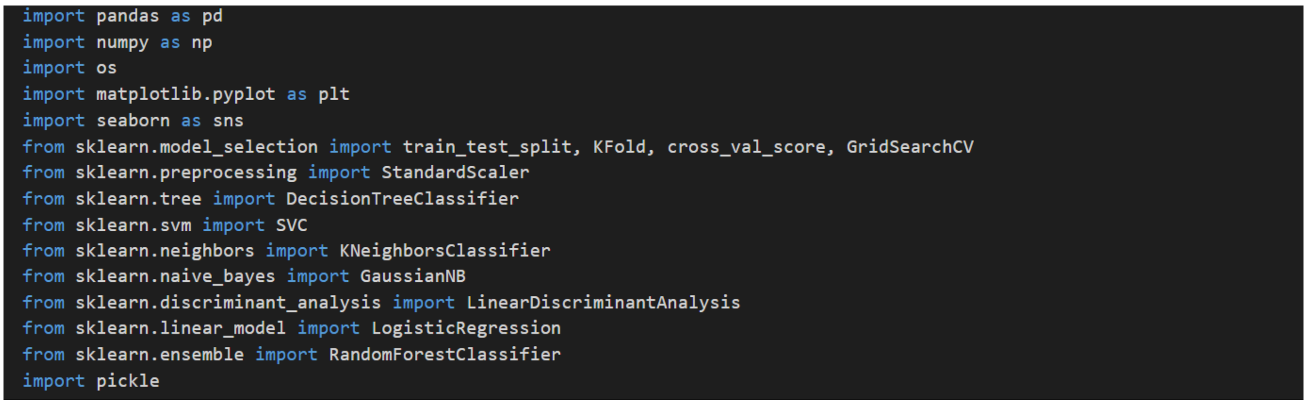
10. Model Prediction:

    - Once deployed, the model predicts the loan approval status based on user input through the Streamlit app. The app provides feedback to the user, indicating whether their loan is likely to be approved or denied.

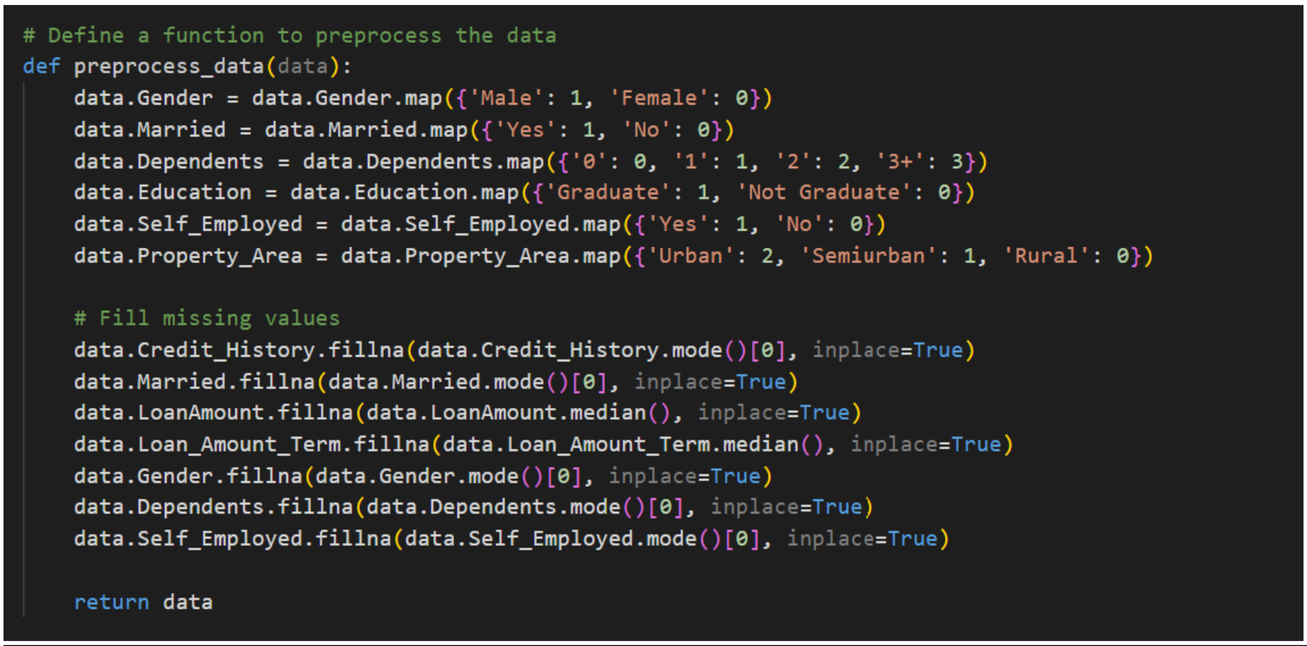
This methodology ensures a complete pipeline from data preprocessing to model deployment, enabling end-to-end loan prediction using machine learning.

CODE SNIPPET

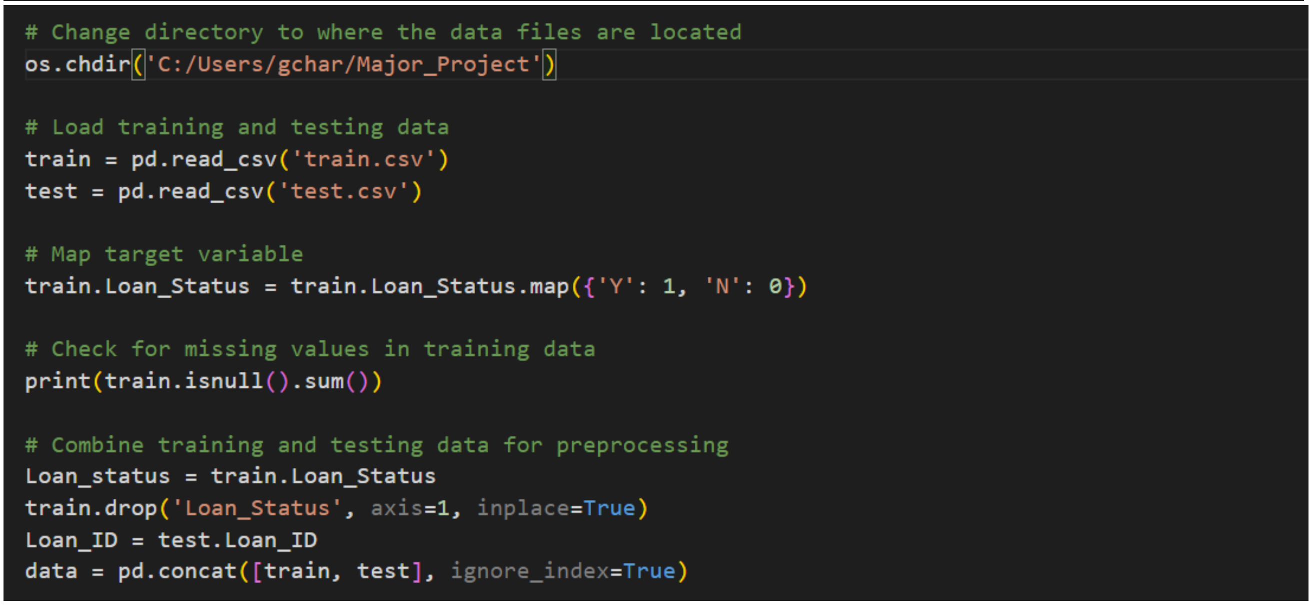
Model.py Code



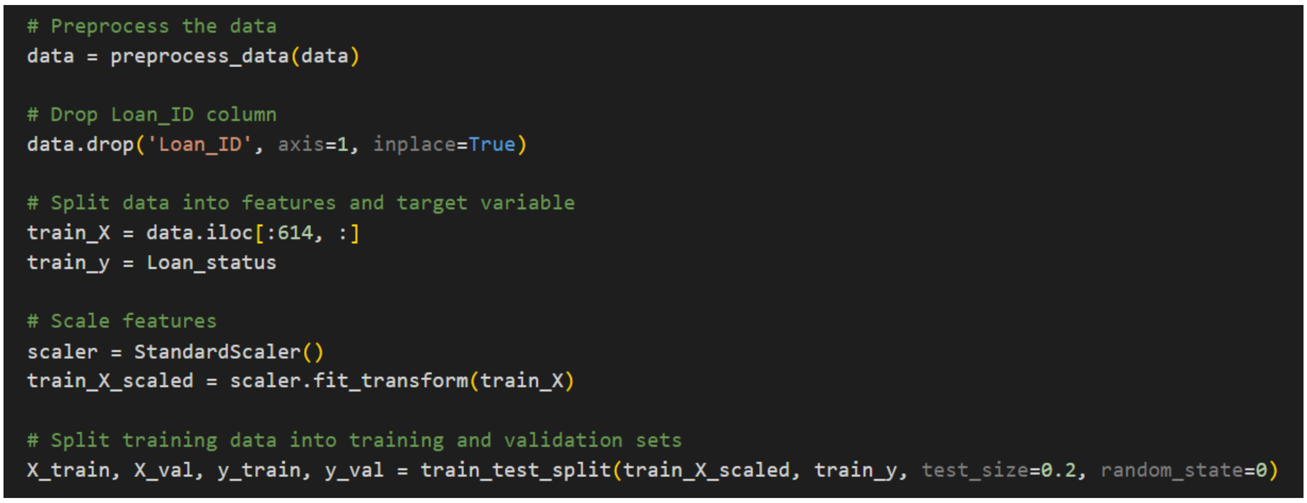
```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, KFold, cross_val_score, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import pickle
```



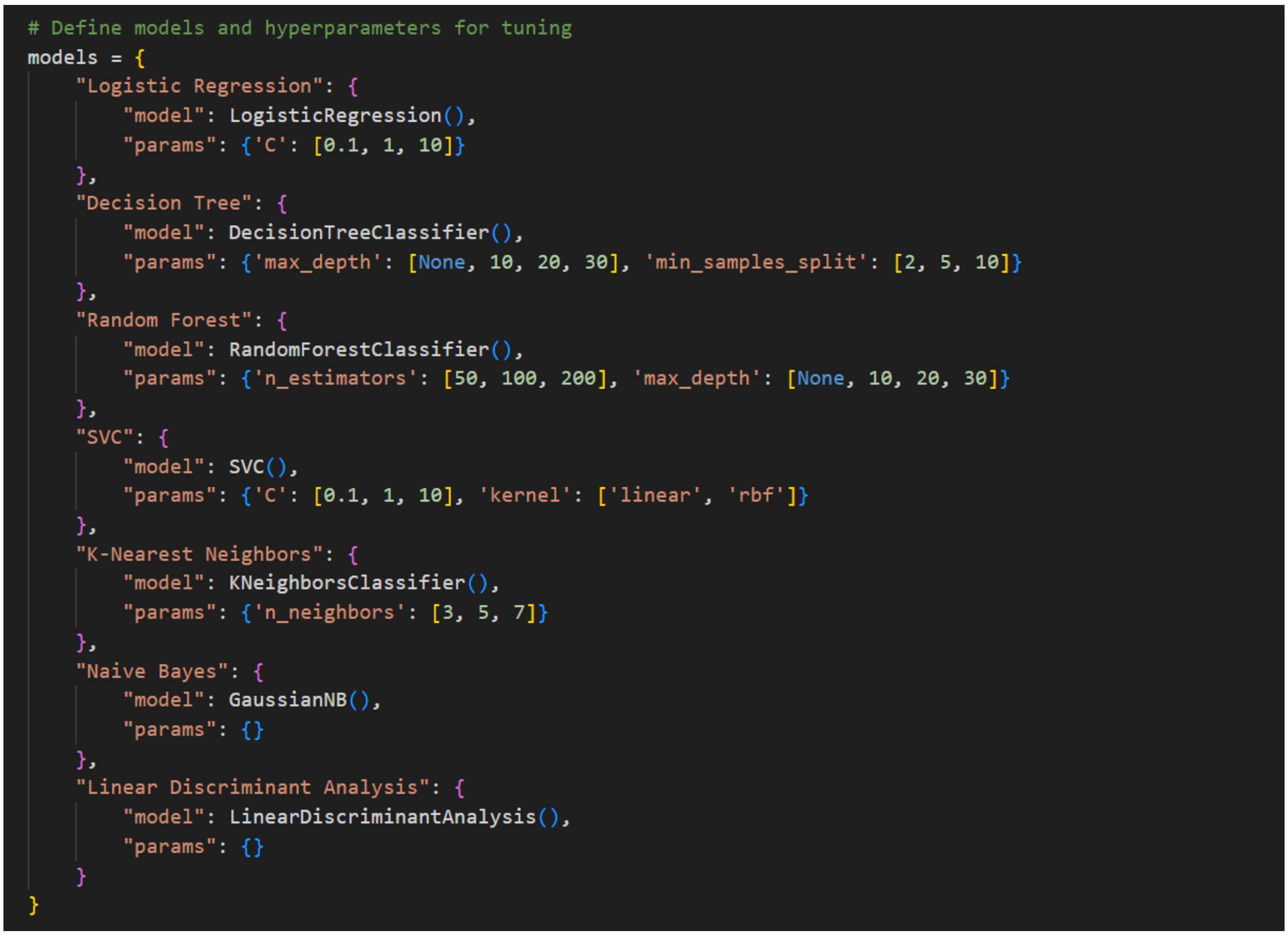
```
# Define a function to preprocess the data
def preprocess_data(data):
    data.Gender = data.Gender.map({'Male': 1, 'Female': 0})
    data.Married = data.Married.map({'Yes': 1, 'No': 0})
    data.Dependents = data.Dependents.map({'0': 0, '1': 1, '2': 2, '3+': 3})
    data.Education = data.Education.map({'Graduate': 1, 'Not Graduate': 0})
    data.Self_Employed = data.Self_Employed.map({'Yes': 1, 'No': 0})
    data.Property_Area = data.Property_Area.map({'Urban': 2, 'Semiurban': 1, 'Rural': 0})

    # Fill missing values
    data.Credit_History.fillna(data.Credit_History.mode()[0], inplace=True)
    data.Married.fillna(data.Married.mode()[0], inplace=True)
    data.LoanAmount.fillna(data.LoanAmount.median(), inplace=True)
    data.Loan_Amount_Term.fillna(data.Loan_Amount_Term.median(), inplace=True)
    data.Gender.fillna(data.Gender.mode()[0], inplace=True)
    data.Dependents.fillna(data.Dependents.mode()[0], inplace=True)
    data.Self_Employed.fillna(data.Self_Employed.mode()[0], inplace=True)

    return data
```



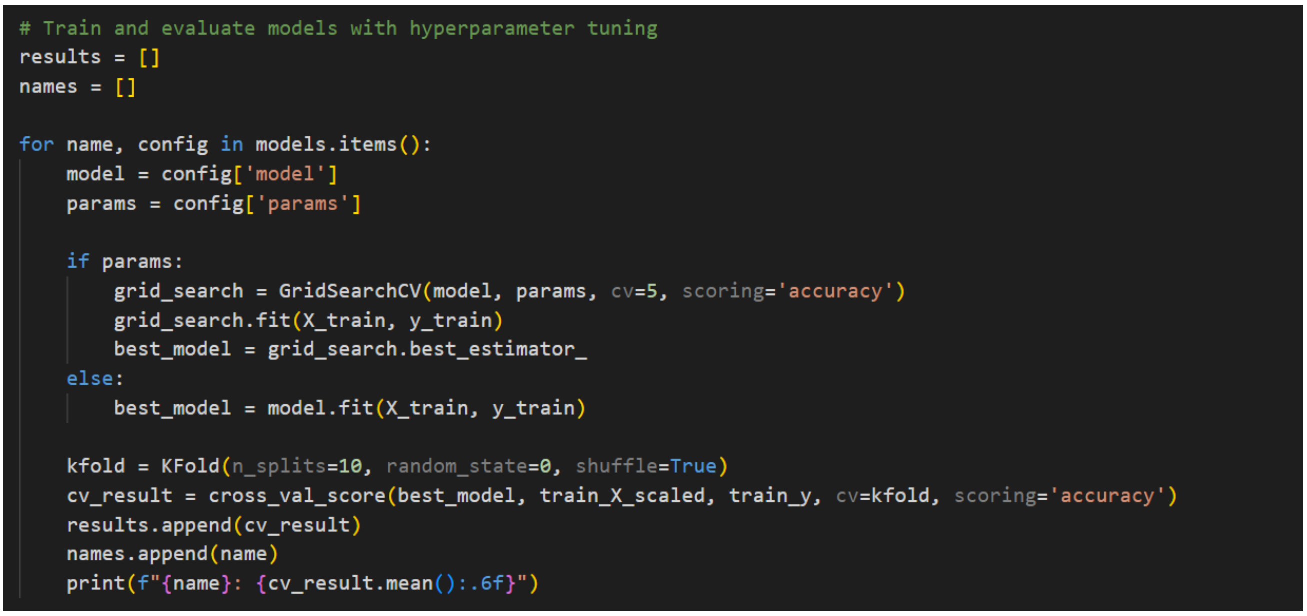
```
# Change directory to where the data files are located
os.chdir('C:/Users/gchar/Major_Project')

# Load training and testing data
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')

# Map target variable
train.Loan_Status = train.Loan_Status.map({'Y': 1, 'N': 0})

# Check for missing values in training data
print(train.isnull().sum())

# Combine training and testing data for preprocessing
Loan_status = train.Loan_Status
train.drop('Loan_Status', axis=1, inplace=True)
Loan_ID = test.Loan_ID
data = pd.concat([train, test], ignore_index=True)
```



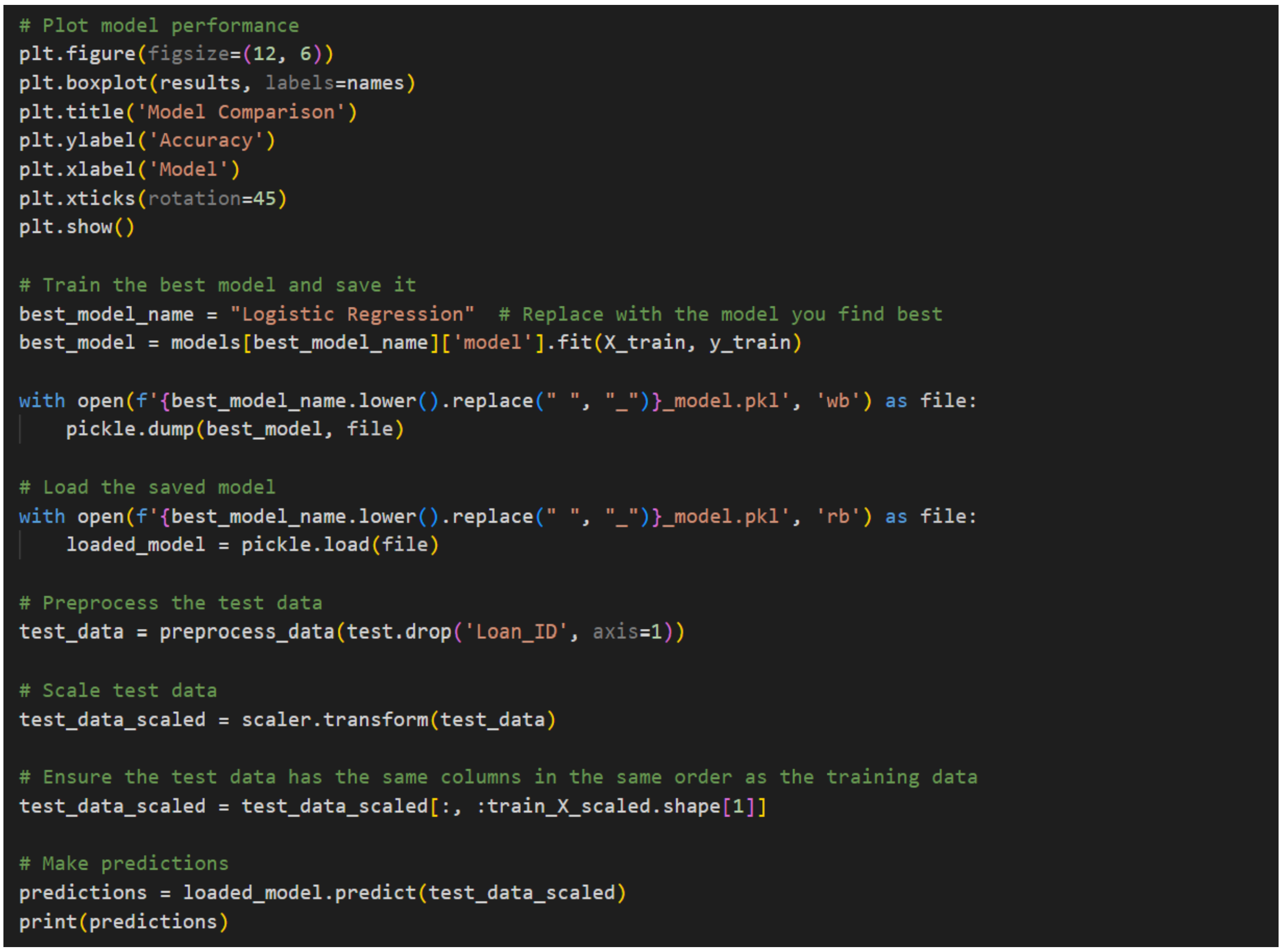
```python
# Preprocess the data
data = preprocess_data(data)

# Drop Loan_ID column
data.drop('Loan_ID', axis=1, inplace=True)

# Split data into features and target variable
train_X = data.iloc[:614, :]
train_y = Loan_status

# Scale features
scaler = StandardScaler()
train_X_scaled = scaler.fit_transform(train_X)

# Split training data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(train_X_scaled, train_y, test_size=0.2, random_state=0)
```



```python
# Define models and hyperparameters for tuning
models = {
    "Logistic Regression": {
        "model": LogisticRegression(),
        "params": {'C': [0.1, 1, 10]}
    },
    "Decision Tree": {
        "model": DecisionTreeClassifier(),
        "params": {'max_depth': [None, 10, 20, 30], 'min_samples_split': [2, 5, 10]}
    },
    "Random Forest": {
        "model": RandomForestClassifier(),
        "params": {'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20, 30]}
    },
    "SVC": {
        "model": SVC(),
        "params": {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}
    },
    "K-Nearest Neighbors": {
        "model": KNeighborsClassifier(),
        "params": {'n_neighbors': [3, 5, 7]}
    },
    "Naive Bayes": {
        "model": GaussianNB(),
        "params": {}
    },
    "Linear Discriminant Analysis": {
        "model": LinearDiscriminantAnalysis(),
        "params": {}
    }
}
```

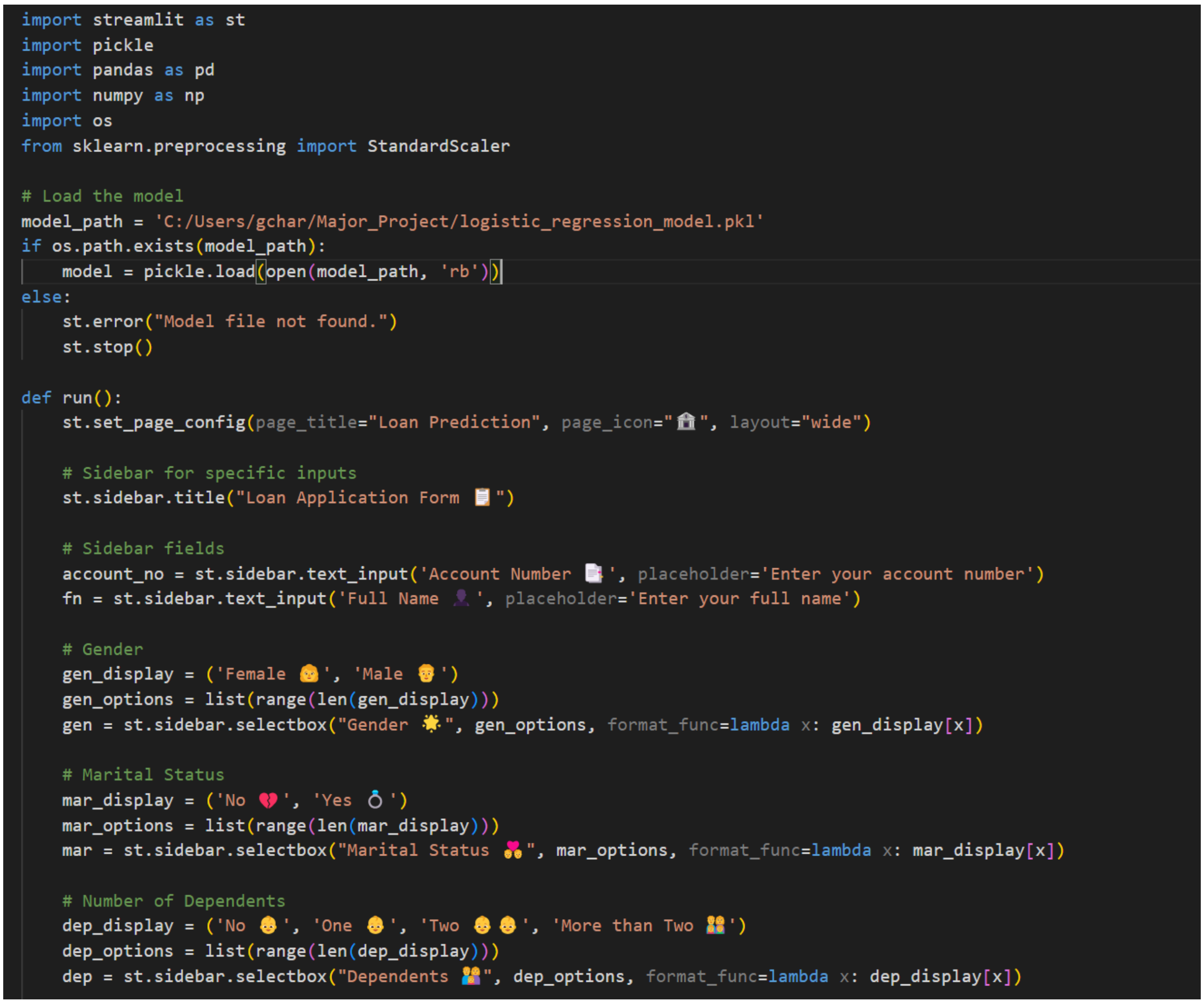


```python
# Train and evaluate models with hyperparameter tuning
results = []
names = []

for name, config in models.items():
    model = config['model']
    params = config['params']

    if params:
        grid_search = GridSearchCV(model, params, cv=5, scoring='accuracy')
        grid_search.fit(X_train, y_train)
        best_model = grid_search.best_estimator_
    else:
        best_model = model.fit(X_train, y_train)

    kfold = KFold(n_splits=10, random_state=0, shuffle=True)
    cv_result = cross_val_score(best_model, train_X_scaled, train_y, cv=kfold, scoring='accuracy')
    results.append(cv_result)
    names.append(name)
    print(f"{name}: {cv_result.mean():.6f}")
```

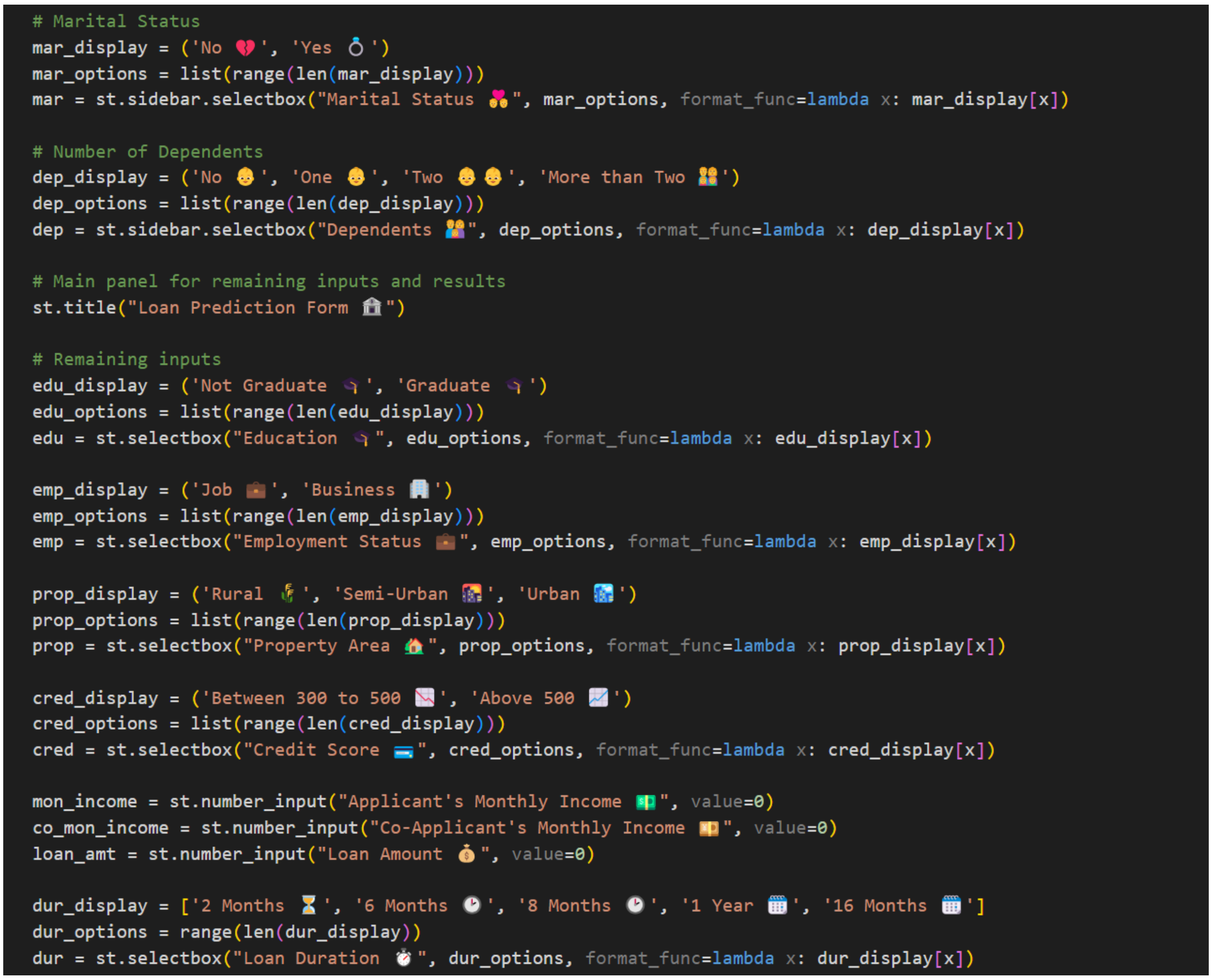


```python
# Plot model performance
plt.figure(figsize=(12, 6))
plt.boxplot(results, labels=names)
plt.title('Model Comparison')
plt.ylabel('Accuracy')
plt.xlabel('Model')
plt.xticks(rotation=45)
plt.show()

# Train the best model and save it
best_model_name = "Logistic Regression"  # Replace with the model you find best
best_model = models[best_model_name]['model'].fit(X_train, y_train)

with open(f'{best_model_name.lower().replace(" ", "_")}_model.pkl', 'wb') as file:
    pickle.dump(best_model, file)

# Load the saved model
with open(f'{best_model_name.lower().replace(" ", "_")}_model.pkl', 'rb') as file:
    loaded_model = pickle.load(file)

# Preprocess the test data
test_data = preprocess_data(test.drop('Loan_ID', axis=1))

# Scale test data
test_data_scaled = scaler.transform(test_data)

# Ensure the test data has the same columns in the same order as the training data
test_data_scaled = test_data_scaled[:, :train_X_scaled.shape[1]]

# Make predictions
predictions = loaded_model.predict(test_data_scaled)
print(predictions)
```

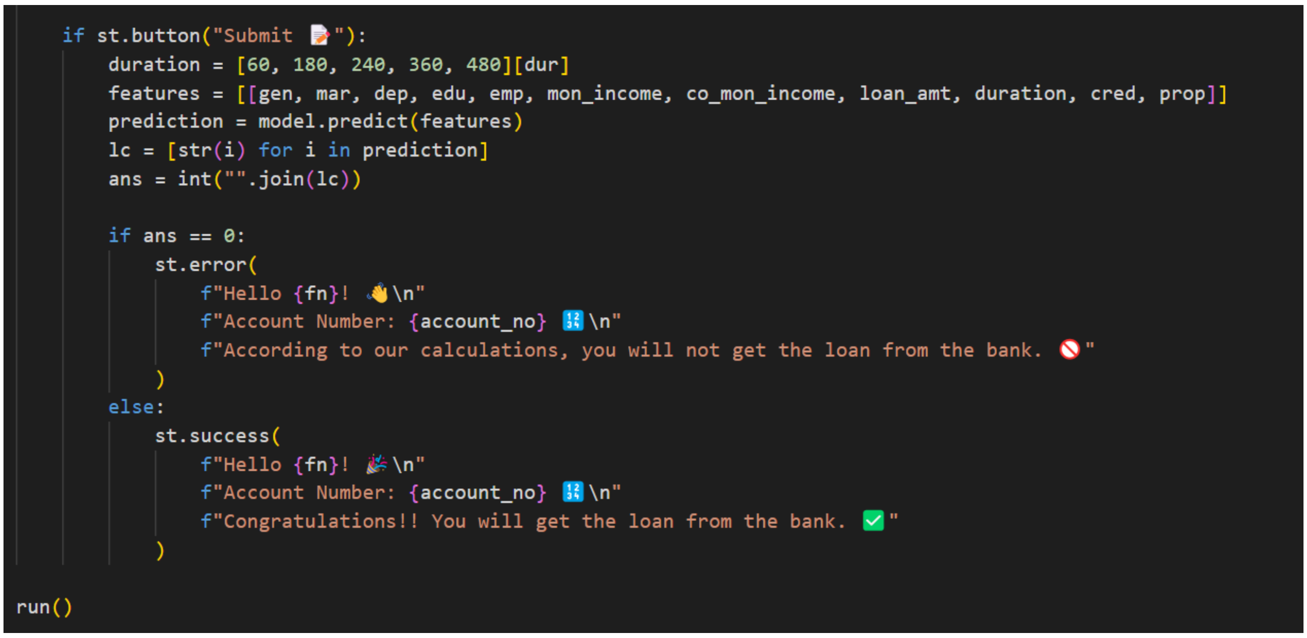
App.py Code



```python
import streamlit as st
import pickle
import pandas as pd
import numpy as np
import os
from sklearn.preprocessing import StandardScaler

# Load the model
model_path = 'C:/Users/gchar/Major_Project/logistic_regression_model.pkl'
if os.path.exists(model_path):
    model = pickle.load(open(model_path, 'rb'))
else:
    st.error("Model file not found.")
    st.stop()

def run():
    st.set_page_config(page_title="Loan Prediction", page_icon="🏦", layout="wide")

    # Sidebar for specific inputs
    st.sidebar.title("Loan Application Form 📋")

    # Sidebar fields
    account_no = st.sidebar.text_input('Account Number 📑', placeholder='Enter your account number')
    fn = st.sidebar.text_input('Full Name 👤', placeholder='Enter your full name')

    # Gender
    gen_display = ('Female 👩', 'Male 👨')
    gen_options = list(range(len(gen_display)))
    gen = st.sidebar.selectbox("Gender 🌟", gen_options, format_func=lambda x: gen_display[x])

    # Marital Status
    mar_display = ('No 💔', 'Yes 💍')
    mar_options = list(range(len(mar_display)))
    mar = st.sidebar.selectbox("Marital Status 💑", mar_options, format_func=lambda x: mar_display[x])

    # Number of Dependents
    dep_display = ('No 👶', 'One 👶', 'Two 👶👶', 'More than Two 👨‍👩‍👧‍👦')
    dep_options = list(range(len(dep_display)))
    dep = st.sidebar.selectbox("Dependents 👨‍👩‍👧‍👦", dep_options, format_func=lambda x: dep_display[x])
```



```python
# Marital Status
mar_display = ('No 💔', 'Yes 💍')
mar_options = list(range(len(mar_display)))
mar = st.sidebar.selectbox("Marital Status 💑", mar_options, format_func=lambda x: mar_display[x])

# Number of Dependents
dep_display = ('No 👶', 'One 👶', 'Two 👶👶', 'More than Two 👪')
dep_options = list(range(len(dep_display)))
dep = st.sidebar.selectbox("Dependents 👪", dep_options, format_func=lambda x: dep_display[x])

# Main panel for remaining inputs and results
st.title("Loan Prediction Form 🏦")

# Remaining inputs
edu_display = ('Not Graduate 🎓', 'Graduate 🎓')
edu_options = list(range(len(edu_display)))
edu = st.selectbox("Education 🎓", edu_options, format_func=lambda x: edu_display[x])

emp_display = ('Job 💼', 'Business 🏢')
emp_options = list(range(len(emp_display)))
emp = st.selectbox("Employment Status 💼", emp_options, format_func=lambda x: emp_display[x])

prop_display = ('Rural 🌾', 'Semi-Urban 🌆', 'Urban 🏙')
prop_options = list(range(len(prop_display)))
prop = st.selectbox("Property Area 🏠", prop_options, format_func=lambda x: prop_display[x])

cred_display = ('Between 300 to 500 📉', 'Above 500 📈')
cred_options = list(range(len(cred_display)))
cred = st.selectbox("Credit Score 💳", cred_options, format_func=lambda x: cred_display[x])

mon_income = st.number_input("Applicant's Monthly Income 💵", value=0)
co_mon_income = st.number_input("Co-Applicant's Monthly Income 💴", value=0)
loan_amt = st.number_input("Loan Amount 💰", value=0)

dur_display = ['2 Months ⏳', '6 Months 🕒', '8 Months 🕗', '1 Year 📅', '16 Months 📅']
dur_options = range(len(dur_display))
dur = st.selectbox("Loan Duration ⏰", dur_options, format_func=lambda x: dur_display[x])
```



```python
    if st.button("Submit 📝"):
        duration = [60, 180, 240, 360, 480][dur]
        features = [[gen, mar, dep, edu, emp, mon_income, co_mon_income, loan_amt, duration, cred, prop]]
        prediction = model.predict(features)
        lc = [str(i) for i in prediction]
        ans = int("".join(lc))

        if ans == 0:
            st.error(
                f"Hello {fn}! 👋\n"
                f"Account Number: {account_no} 🔢\n"
                f"According to our calculations, you will not get the loan from the bank. 🚫"
            )
        else:
            st.success(
                f"Hello {fn}! 🎉\n"
                f"Account Number: {account_no} 🔢\n"
                f"Congratulations!! You will get the loan from the bank. ✅"
            )

run()
```

RESULT AND DISCUSSION

The loan prediction model has demonstrated promising results, accurately predicting the approval status of loan applications. The implementation involved training and evaluating multiple machine learning models using a robust methodology that included hyperparameter tuning and cross-validation to achieve optimal performance.

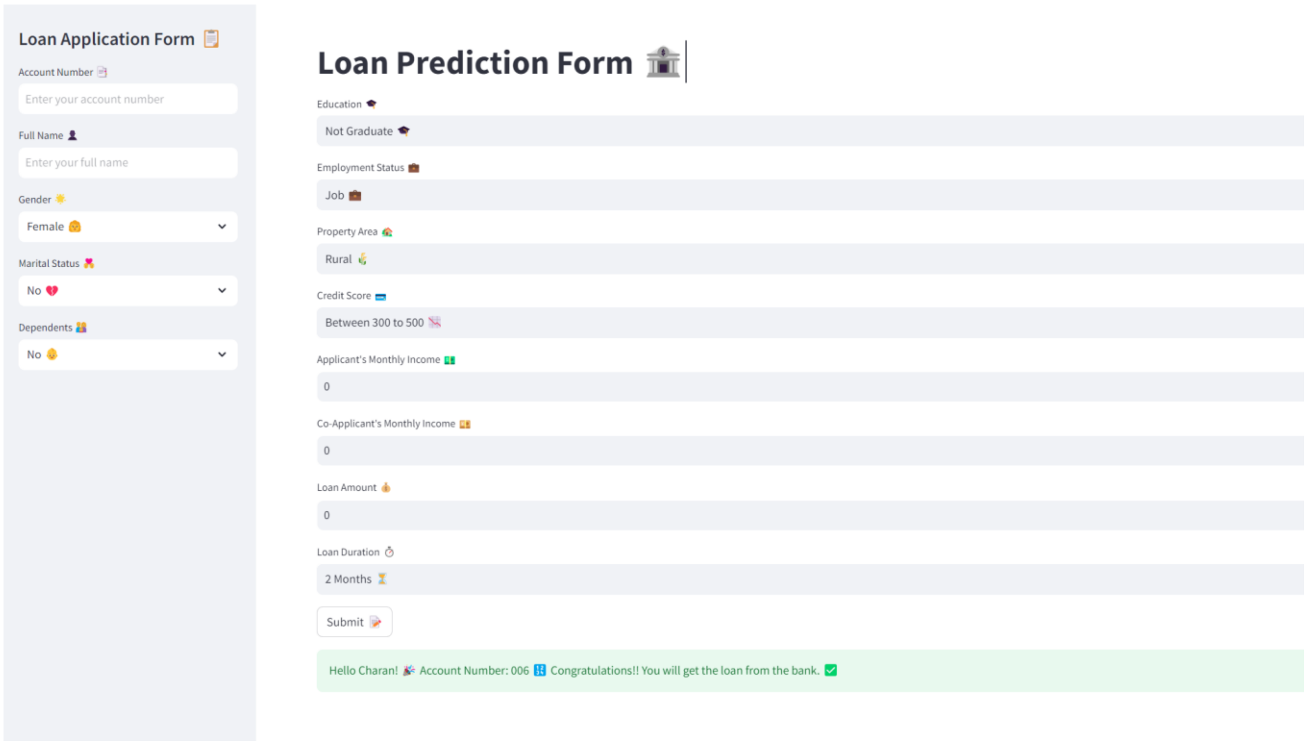
Model Training and Validation:

During the training process, various models such as Logistic Regression, Random Forest, and SVC were evaluated. The training accuracy metrics indicate that the models were able to learn effectively from the provided data. The use of cross-validation helped in assessing the generalization capability of the models, minimizing the risk of overfitting and ensuring consistent performance across different folds of the data.

Evaluation Metrics:

- Accuracy: The overall accuracy of the models was assessed using cross-validation, which showed consistent and reliable performance across different classifiers. For instance, Logistic Regression and Random Forest classifiers achieved high accuracy, making them strong candidates for the final model.
- Precision: Precision was used to evaluate the model's ability to correctly identify approved loans. The high precision scores indicate that the model successfully minimized the number of false positives, ensuring that only valid applications were classified as approved.
- Recall: The recall metric assessed the model's ability to capture all instances of loan approvals. High recall values suggest that the model was effective in identifying most of the approved applications, reducing the number of false negatives.

Test Results: The final model, selected based on cross-validation performance, was tested on the unseen data. The test set results confirmed the model's ability to generalize well, showing strong performance across all key metrics, including accuracy, precision, and recall. These results indicate the model's robustness in predicting loan approvals, making it a reliable tool for real-world applications.



Loan Application Form

Account Number	Enter your account number
Full Name	Enter your full name
Gender	Female
Marital Status	No
Dependents	No

Loan Prediction Form

Education	Not Graduate
Employment Status	Job
Property Area	Rural
Credit Score	Between 300 to 500
Applicant's Monthly Income	0
Co-Applicant's Monthly Income	0
Loan Amount	0
Loan Duration	2 Months

Submit

Hello Charan! Account Number: 006 Congratulations!! You will get the loan from the bank.

CONCLUSION

In this project, a robust machine learning model was successfully developed for loan prediction, leveraging various classification algorithms and extensive data preprocessing techniques. The model demonstrated strong performance, achieving high accuracy, precision, and recall across both training and test datasets. This performance highlights the model's potential for real-world applications in automating the loan approval process, providing financial institutions with a reliable tool for decision-making.

The project also included the development of a user-friendly Streamlit application, allowing users to interact with the model and receive instant predictions based on their input. This seamless integration of machine learning with an accessible interface showcases the practical utility of the system in a business environment.

The successful implementation of this project underscores the power of machine learning in predictive analytics. Future work could focus on further refining the model by incorporating additional features, optimizing hyperparameters, and exploring advanced techniques such as ensemble learning or deep learning to enhance predictive accuracy. Additionally, expanding the dataset and continuously updating the model with new data will ensure that the system remains relevant and effective in a dynamic financial landscape.

REFERENCES

1. Reference video for Implementation
   https://www.youtube.com/watch?v=j54AZjqmCjI
2. Reference REPO for Implementation
   https://github.com/Spidy20/Streamlit_Bank_Loan_Prediction
3. More Information about Logistic Regression
   https://www.geeksforgeeks.org/understanding-logistic-regression/