**Assessment Report**

on

# “LOAN DEFAULT PREDICTION ”

submitted as partial fulfillment for the award of

**BACHELOR OF TECHNOLOGY DEGREE**

SESSION 2024-25

In

**CSE(AI)**

By

**GROUP - 8**

Name : Raamanjal Singh Gangwar

Roll Number : 202401100300187

Section: C

## 1. INTRODUCTION

In today’s banking and financial landscape, managing credit risk is a critical task. One of the most important challenges is identifying whether a loan applicant is likely to default. This project, **Loan Default Prediction Using Machine Learning**, aims to address this challenge by developing a predictive model that classifies applicants based on the likelihood of loan repayment.

## 2. Problem Statement

**Loan Default Prediction**

**Predict whether a loan applicant will default using classification algorithms and decision trees. Visualize feature importance.**

## 3.Objectives

- **Predict whether a loan applicant will default on a loan using supervised classification algorithms.**
- **Compare the performance of multiple models such as Naive Bayes and Support Vector Machines (SVM).**
- **Handle real-world challenges including missing data, categorical encoding, and imbalanced classes.**
- **Generate predictions on unseen data to support practical decision-making.**

# **4-Methodology**

**1. Data Loading:**

- **Train and test datasets are loaded using pandas.read_csv().**

**2. Data Cleaning:**

- **Missing values are checked using .isnull().sum().**
- **Missing categorical values are filled with the mode.**
- **Missing numerical values are filled with the median.**
- **Irrelevant columns such as Loan_ID are dropped.**

**3. Feature Encoding:**

- **Categorical variables are encoded using LabelEncoder from scikit-learn.**

**4. Data Splitting:**

- **Features (X) and target (y) are separated.**
- **The dataset is split into training and test sets using train_test_split() with a test size of 20%.**

**5. Feature Scaling:**

- **Features are standardized using StandardScaler.**

**6. Model Training:**

**The following classification models are trained:**

- **Logistic Regression**

- **Decision Tree**
- **Random Forest**

**7. Evaluation:**

- **Models are evaluated using:**
    - **Accuracy Score**
    - **Confusion Matrix**
    - **Classification Report (Precision, Recall, F1-score)**

**8. Visualization**

- **Confusion matrices and decision trees may be visualized using matplotlib/seaborn and plot_tree().**

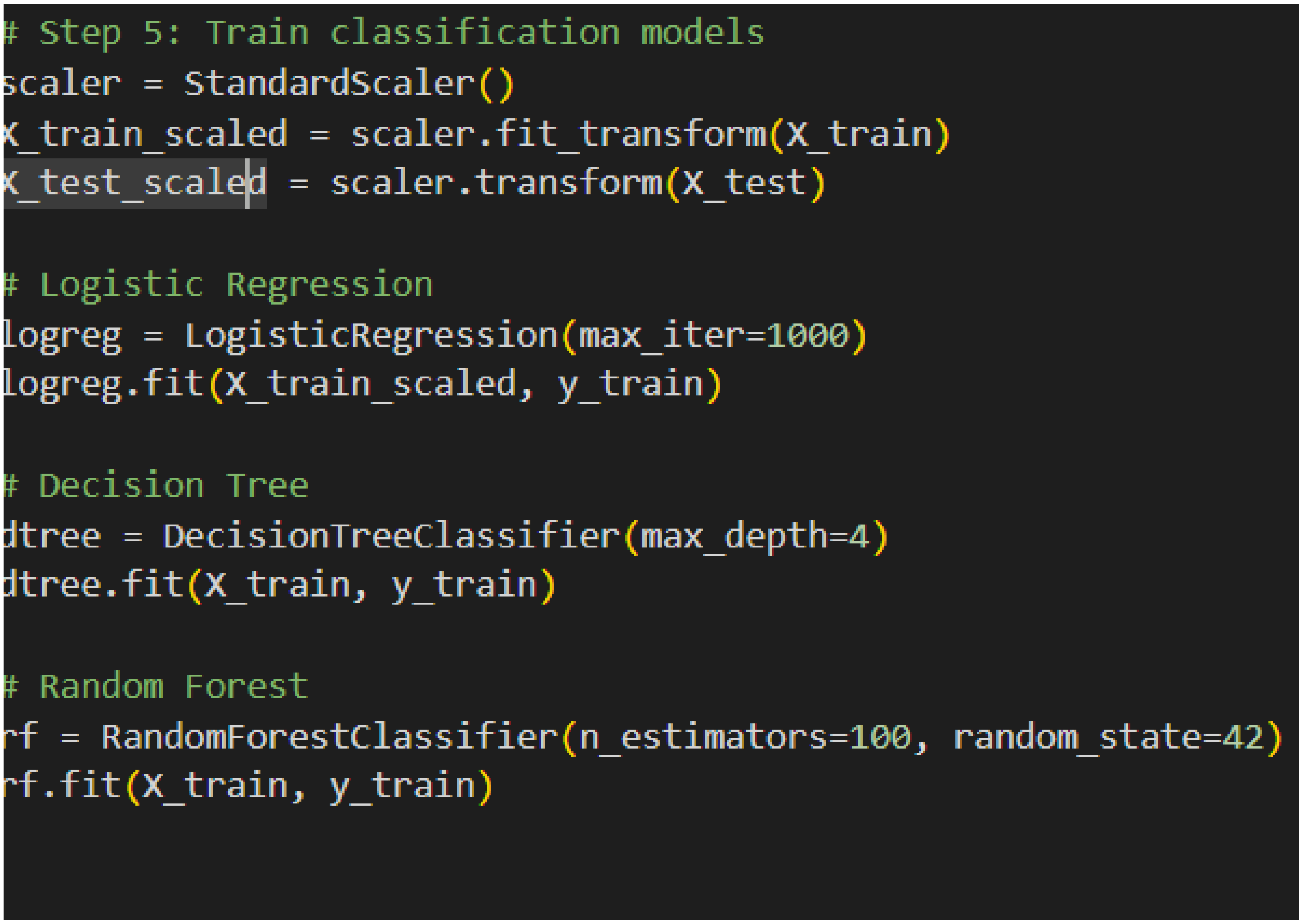
## 5.Dataset Description

**The dataset appears to be related to loan applications and contains various features that can influence loan default predictions. The dataset is split into train.csv and test.csv.**

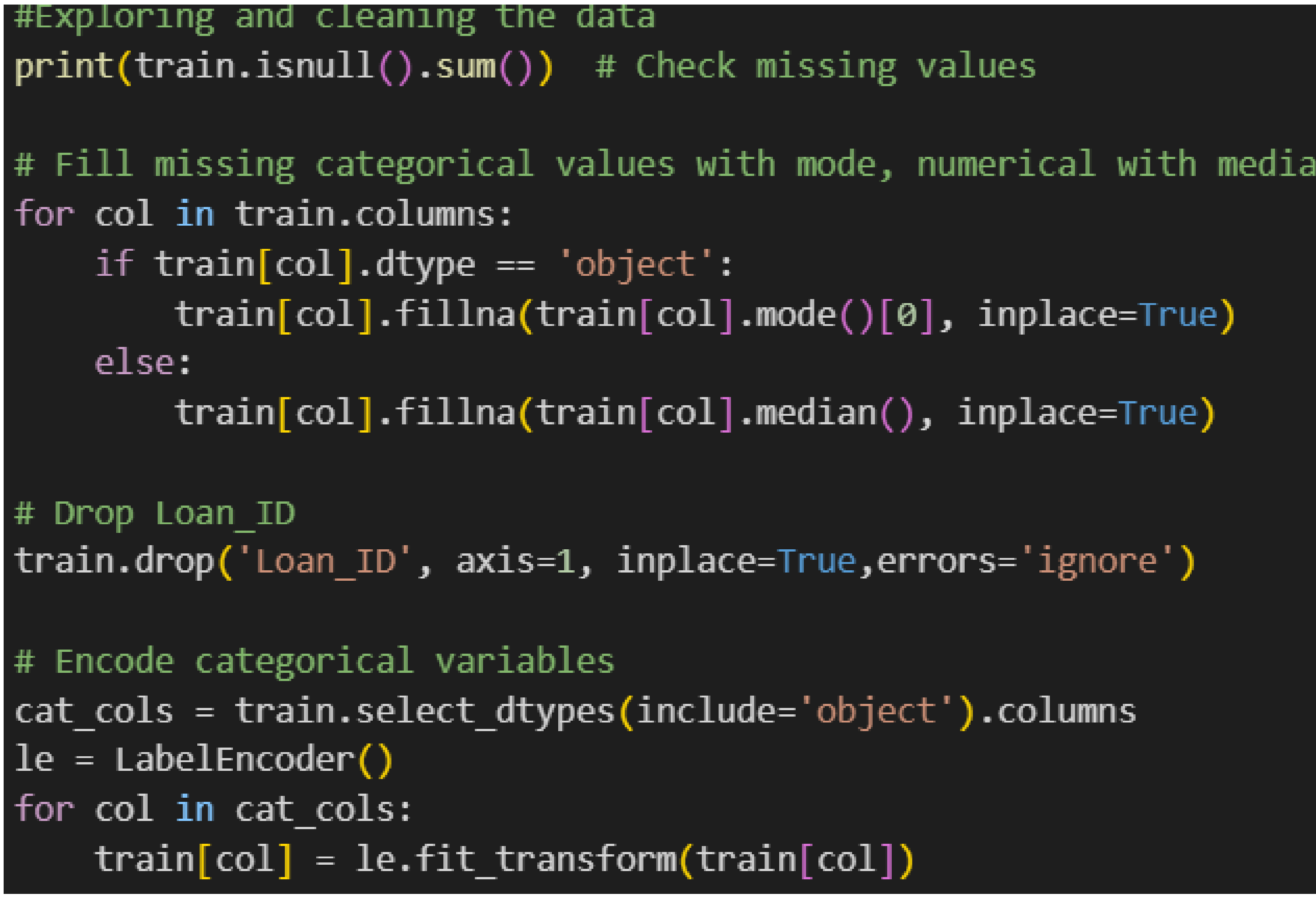
**Common Features (likely based on naming and typical loan datasets):**

- **Categorical Features: Gender, Married, Education, Self_Employed, Property_Area, etc.**
- **Numerical Features: ApplicantIncome, CoapplicantIncome, LoanAmount, Loan_Amount_Term, Credit_History, etc.**
- **Target Variable: Loan_Status (indicating loan approval or default)**

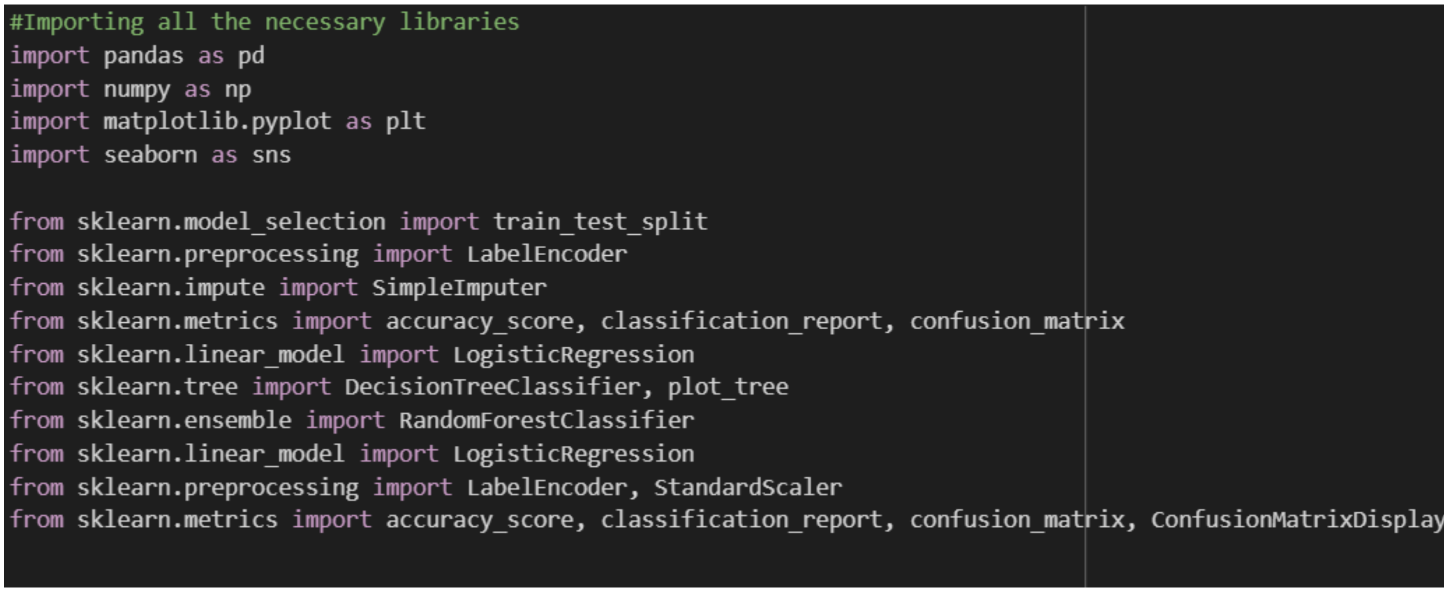
## 6.CODE



```
# Step 5: Train classification models
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Logistic Regression
logreg = LogisticRegression(max_iter=1000)
logreg.fit(X_train_scaled, y_train)

# Decision Tree
dtree = DecisionTreeClassifier(max_depth=4)
dtree.fit(X_train, y_train)

# Random Forest
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
```

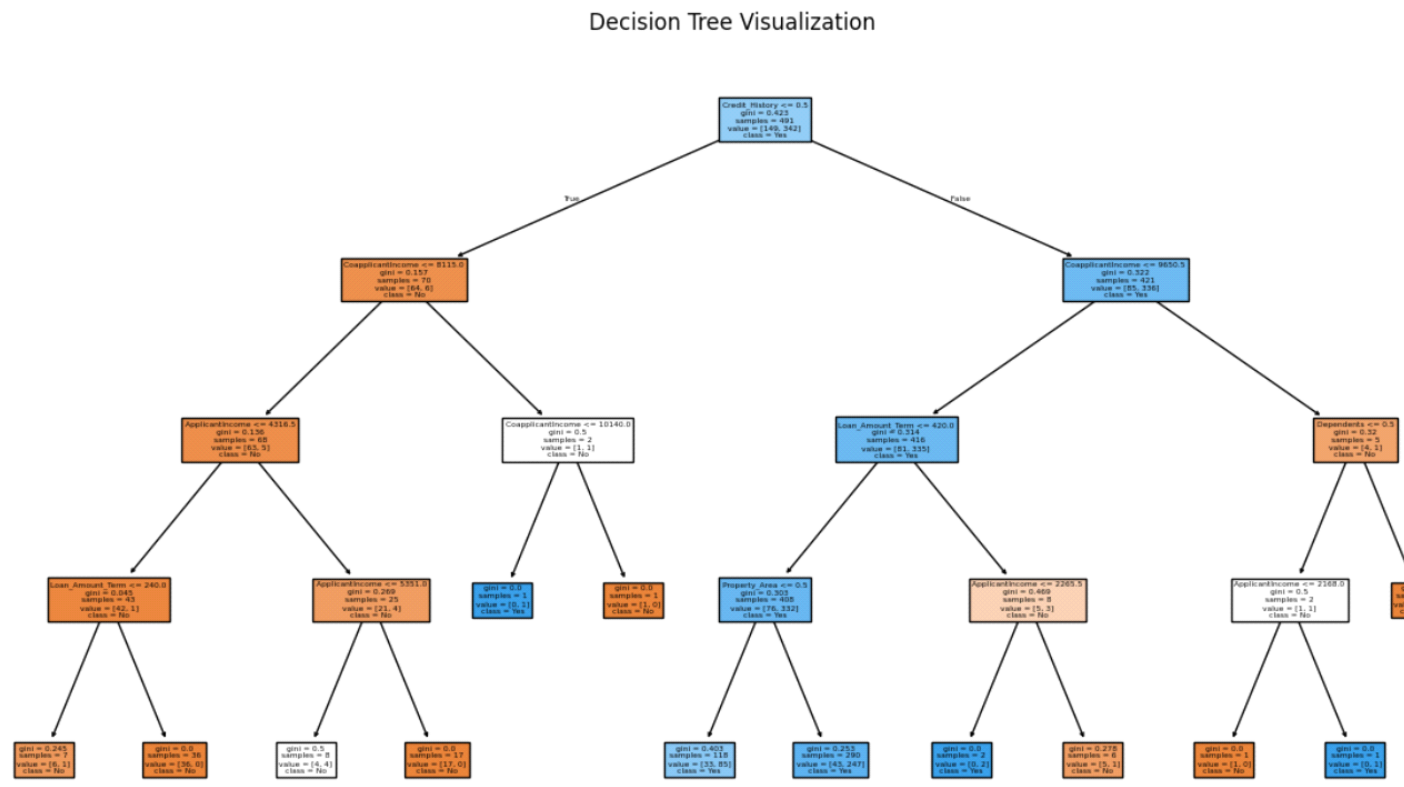


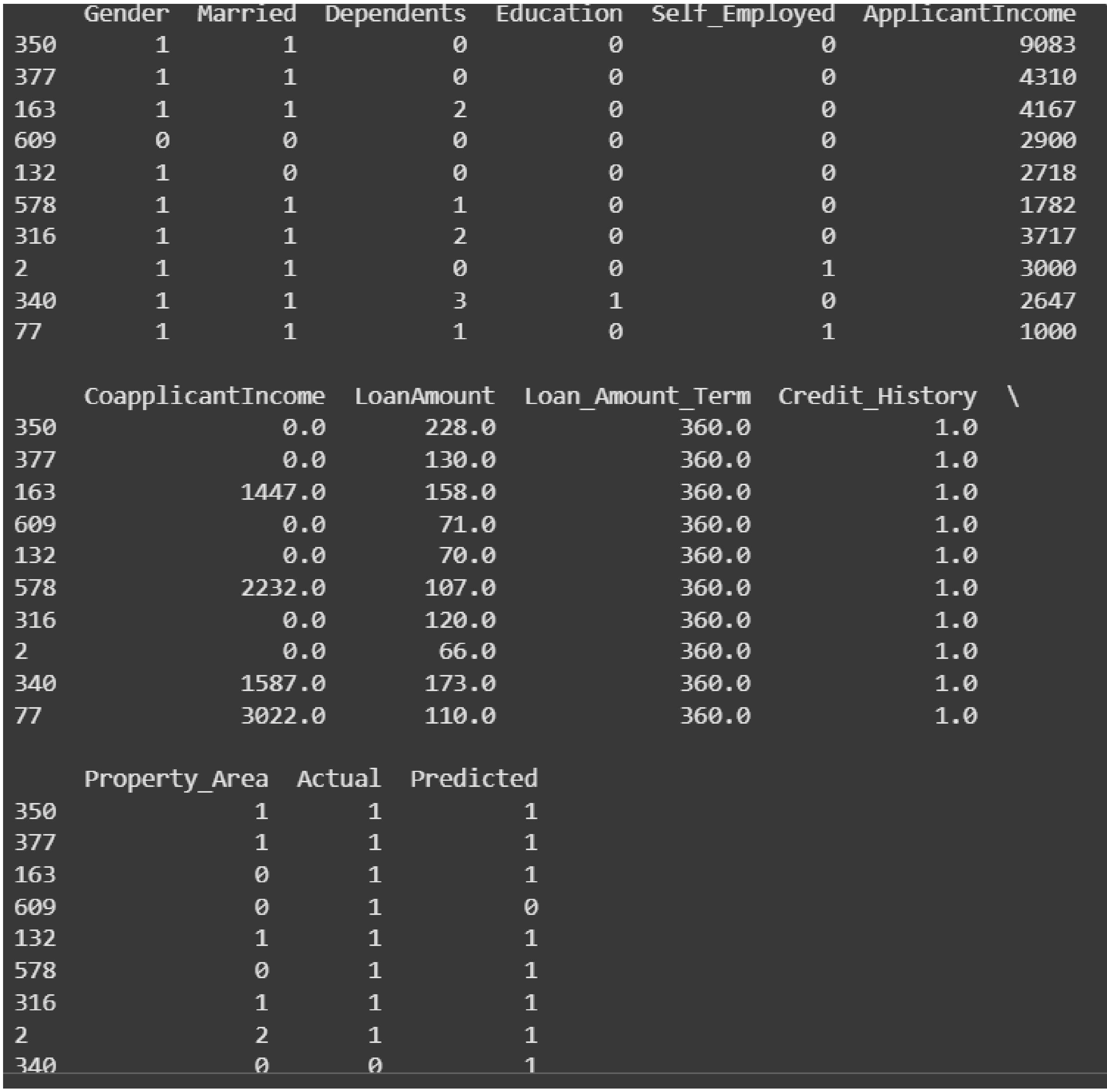
```
#Exploring and cleaning the data
print(train.isnull().sum())  # Check missing values

# Fill missing categorical values with mode, numerical with media
for col in train.columns:
    if train[col].dtype == 'object':
        train[col].fillna(train[col].mode()[0], inplace=True)
    else:
        train[col].fillna(train[col].median(), inplace=True)

# Drop Loan_ID
train.drop('Loan_ID', axis=1, inplace=True,errors='ignore')

# Encode categorical variables
cat_cols = train.select_dtypes(include='object').columns
le = LabelEncoder()
for col in cat_cols:
    train[col] = le.fit_transform(train[col])
```



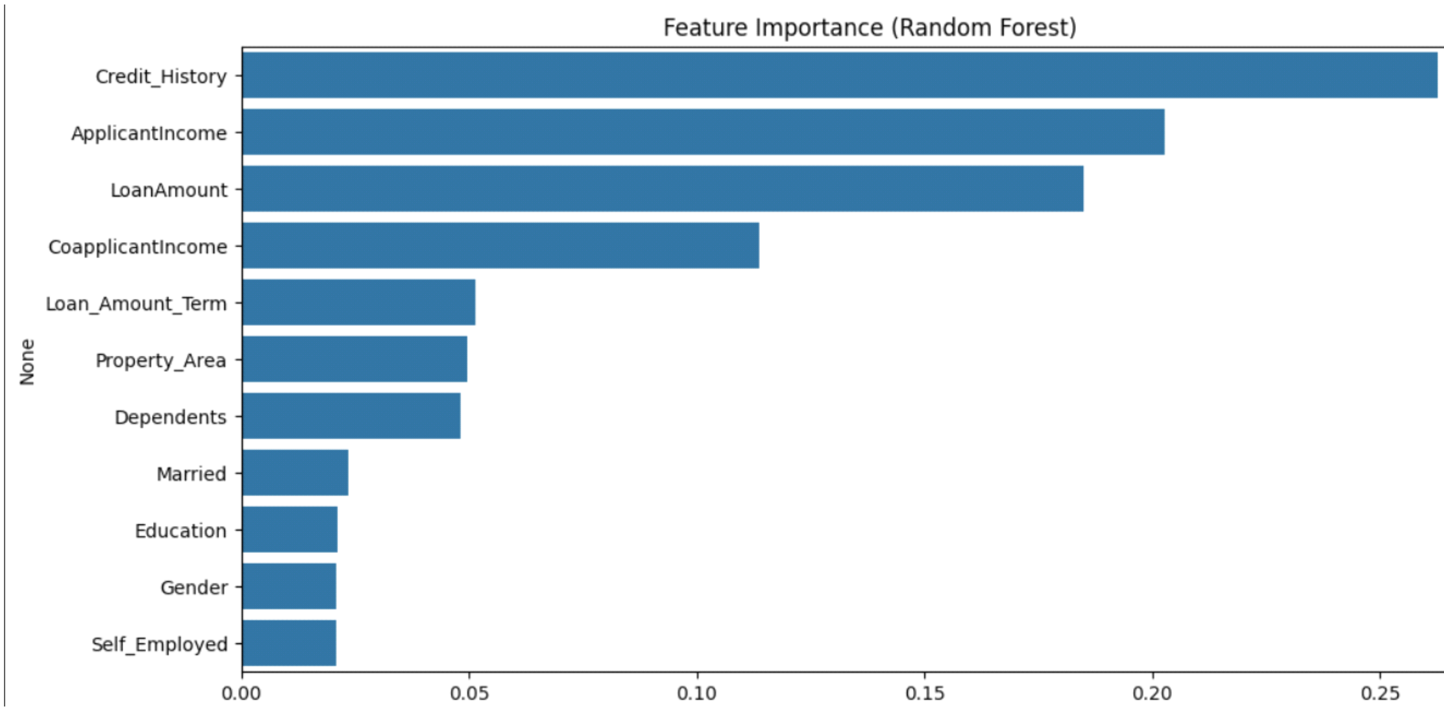
```
#Importing all the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, ConfusionMatrixDisplay
```

## 7.OUTPUT:


Decision Tree Visualization
Credit_History <= 0.5 gini = 0.423 samples = 491 value = [149, 342] class = Yes
True
False
CoapplicantIncome <= 8115.0 gini = 0.157 samples = 70 value = [64, 6] class = No
CoapplicantIncome <= 9650.5 gini = 0.322 samples = 421 value = [85, 336] class = Yes
ApplicantIncome <= 4316.5 gini = 0.136 samples = 68 value = [63, 5] class = No
CoapplicantIncome <= 10140.0 gini = 0.5 samples = 2 value = [1, 1] class = No
Loan_Amount_Term <= 420.0 gini = 0.314 samples = 416 value = [81, 335] class = Yes
Dependents <= 0.5 gini = 0.32 samples = 5 value = [4, 1] class = No
Loan_Amount_Term <= 240.0 gini = 0.045 samples = 43 value = [42, 1] class = No
ApplicantIncome <= 5351.0 gini = 0.269 samples = 25 value = [21, 4] class = No
gini = 0.0 samples = 1 value = [0, 1] class = Yes
gini = 0.0 samples = 1 value = [1, 0] class = No
Property_Area <= 0.5 gini = 0.303 samples = 408 value = [76, 332] class = Yes
ApplicantIncome <= 2265.5 gini = 0.469 samples = 8 value = [5, 3] class = No
ApplicantIncome <= 2168.0 gini = 0.5 samples = 2 value = [1, 1] class = No
gini = 0.245 samples = 7 value = [6, 1] class = No
gini = 0.0 samples = 36 value = [36, 0] class = No
gini = 0.5 samples = 8 value = [4, 4] class = No
gini = 0.0 samples = 17 value = [17, 0] class = No
gini = 0.403 samples = 118 value = [33, 85] class = Yes
gini = 0.253 samples = 290 value = [43, 247] class = Yes
gini = 0.0 samples = 2 value = [0, 2] class = Yes
gini = 0.278 samples = 6 value = [5, 1] class = No
gini = 0.0 samples = 1 value = [1, 0] class = No
gini = 0.0 samples = 1 value = [0, 1] class = Yes



```
     Gender  Married  Dependents  Education  Self_Employed  ApplicantIncome
350       1        1           0          0              0             9083
377       1        1           0          0              0             4310
163       1        1           2          0              0             4167
609       0        0           0          0              0             2900
132       1        0           0          0              0             2718
578       1        1           1          0              0             1782
316       1        1           2          0              0             3717
2         1        1           0          0              1             3000
340       1        1           3          1              0             2647
77        1        1           1          0              1             1000

     CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
350                0.0       228.0             360.0             1.0
377                0.0       130.0             360.0             1.0
163             1447.0       158.0             360.0             1.0
609                0.0        71.0             360.0             1.0
132                0.0        70.0             360.0             1.0
578             2232.0       107.0             360.0             1.0
316                0.0       120.0             360.0             1.0
2                  0.0        66.0             360.0             1.0
340             1587.0       173.0             360.0             1.0
77              3022.0       110.0             360.0             1.0

     Property_Area  Actual  Predicted
350              1       1          1
377              1       1          1
163              0       1          1
609              0       1          0
132              1       1          1
578              0       1          1
316              1       1          1
2                2       1          1
340              0       0          1
```



## 8.REFRENCES:

- **Dataset**

  [**Loan default prediction**]

- **Pandas Documentation**
  ***McKinney, W. (2010).***
  **Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, 51–56.**
  **Available at: https://pandas.pydata.org**

- **Scikit-learn Documentation**
  ***Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011).***
  **Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12,* 2825–2830**

  **Available at: https://scikit-learn.org**