Predict Loan Default: Classify whether a borrower will default on a loan using financial history and credit scores.

NAME ->HARSH VARDHAN SINGH

ROLL NUMBER->20

Introduction

In this project, the goal is to build a model that predicts whether a loan applicant will default or not based on their financial history. The dataset includes attributes like income, credit score, employment status, and more. This classification problem helps financial institutions assess lending risk

Code

# Step 1: Upload the CSV file

from google.colab import files

uploaded = files.upload()

# Step 2: Import libraries

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import classification\_report, accuracy\_score

# Step 3: Load the CSV file (update the name if different)

df = pd.read\_csv("1. Predict Loan Default (1).csv")

# Step 4: Drop non-informative columns

if "LoanID" in df.columns:

    df = df.drop(columns=["LoanID"])

# Step 5: Encode categorical variables

categorical\_cols = df.select\_dtypes(include=["object"]).columns

for col in categorical\_cols:

    le = LabelEncoder()

    df[col] = le.fit\_transform(df[col])

# Step 6: Split the data

X = df.drop(columns=["Default"])

y = df["Default"]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Step 7: Train the Random Forest model with class\_weight to handle imbalance

model = RandomForestClassifier(class\_weight='balanced', random\_state=42)

model.fit(X\_train, y\_train)

# Step 8: Predict and evaluate

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

report = classification\_report(y\_test, y\_pred)

# Step 9: Print results

print("Accuracy:", accuracy)

print("Classification Report:\n", report)

# Step 10: Plot Feature Importance

importances = model.feature\_importances\_

features = X.columns

feature\_df = pd.DataFrame({'Feature': features, 'Importance': importances})

feature\_df = feature\_df.sort\_values(by='Importance', ascending=False)

# Plot

plt.figure(figsize=(12, 6))

sns.barplot(x='Importance', y='Feature', data=feature\_df, palette='viridis')

plt.title("Feature Importance for Loan Default Prediction")

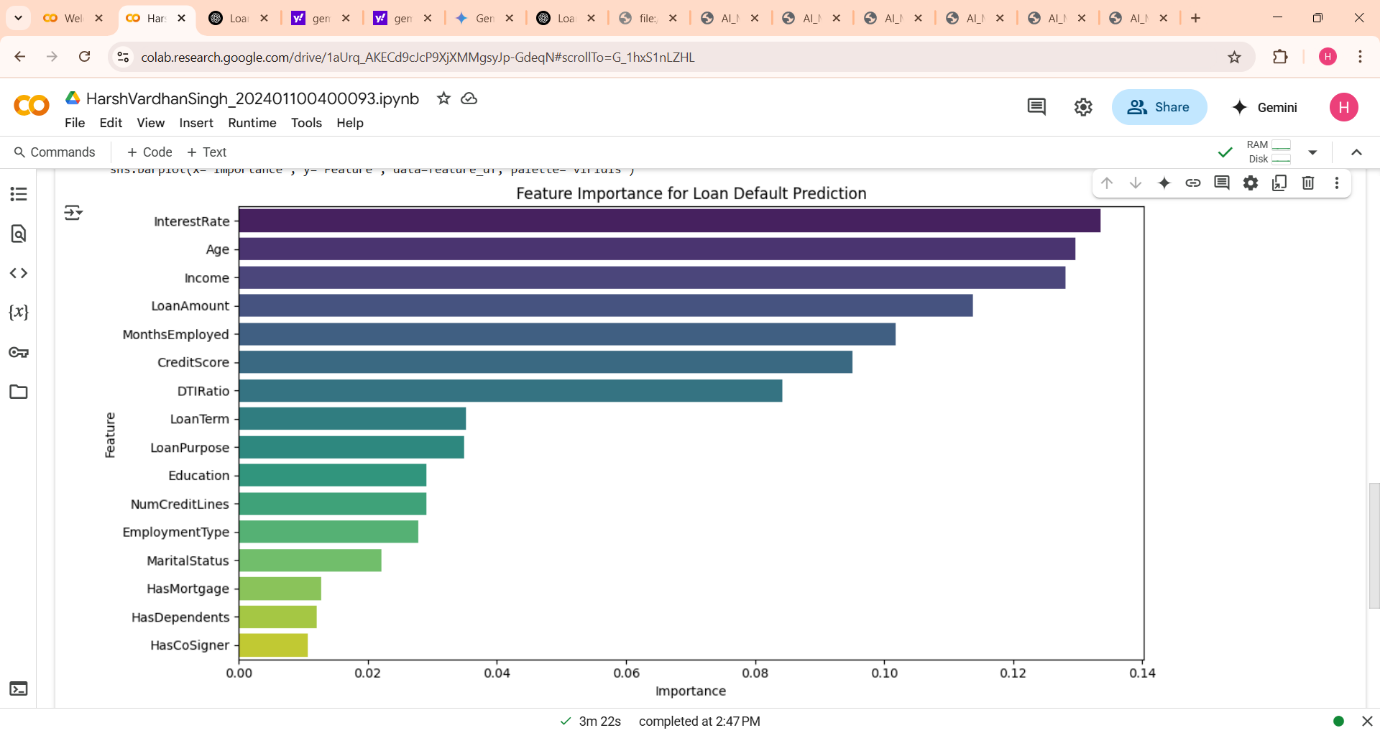
plt.xlabel("Importance")

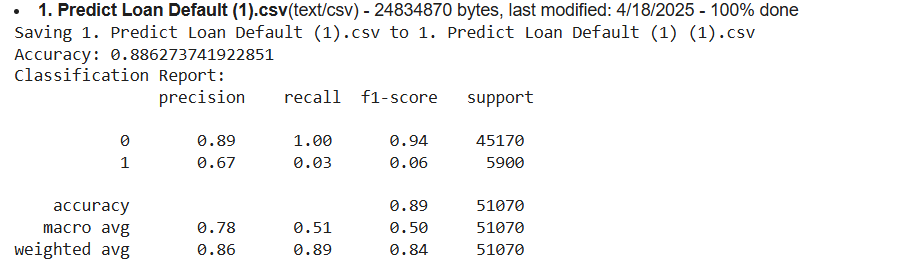
plt.ylabel("Feature")

plt.tight\_layout()

plt.show()

Output/Result





Methodology

1. Load the dataset and remove irrelevant columns. 2. Encode categorical variables using Label Encoding. 3. Split the data into training and testing sets. 4. Train a Random Forest Classifier on the training set. 5. Evaluate model performance using accuracy, confusion matrix, and classification report.

References/Credits - Dataset: Provided by the AI MSE exam committee.