Assessment Report

on

"Predict Loan Default"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

Name of discipline

By

Name:-SHIVAM AGARWAL

Roll Number:-202401100300231

Branch-section:-CSEAI-D

Under the supervision of

"ABHISHEK SHUKLA"

KIET Group of Institutions, Ghaziabad

May, 2025

INDEX:-

Introduction:

Loan default prediction is a critical task in the financial industry that helps lending institutions assess the risk associated with lending money to individuals. A **loan default** occurs when a borrower fails to repay their loan as agreed, leading to financial losses for the lender. With the increasing availability of customer data, machine learning models can be used to predict whether a borrower is likely to default, enabling lenders to make more informed decisions.

This project focuses on building a predictive model using historical loan data to classify borrowers into two categories:

- **Default (1):** The borrower is likely to default on the loan.
- **No Default (0):** The borrower is likely to repay the loan successfully.

The dataset, sourced from Kaggle, contains a wide range of borrower attributes such as income, credit score, employment history, and loan characteristics. The objective is to preprocess the data, train a classification model, evaluate its performance using standard metrics, and visualize the results using a confusion matrix heatmap. These insights can be instrumental for financial institutions in improving credit risk management and reducing default rates.

Methodology

The methodology for predicting loan default using machine learning involves several key stages, from data preprocessing to model evaluation. Below is a breakdown of each step implemented in the code:

1. Data Acquisition

The dataset titled **“1. Predict Loan Default.csv”** was sourced from Kaggle. It contains over 255,000 records with information on individual loan applicants, including their demographic details, financial history, loan terms, and default status.

2. Data Preprocessing

- **Dropping Non-informative Columns:**
  The LoanID column, being a unique identifier, was removed as it does not contribute to the prediction task.
- **Binary Encoding:**
  Columns with binary categorical values (HasMortgage, HasDependents, HasCoSigner) were encoded into numeric format (1 for "Yes", 0 for "No").
- **Feature and Target Selection:**
  The feature matrix X contains all columns except the target variable Default, which is the binary classification label.

- **Categorical Encoding:**
  Multiclass categorical columns such as Education, EmploymentType, MaritalStatus, and LoanPurpose were encoded using **One-Hot Encoding** to convert them into numeric format.
- **Numerical Scaling:**
  Numerical features (e.g., Income, CreditScore, DTIRatio) were standardized using **StandardScaler** to improve model performance.
- **Train-Test Split:**
  The dataset was split into 80% training data and 20% testing data to evaluate model generalization on unseen data.

3. Model Building

A **Random Forest Classifier** was selected as the predictive model due to its ability to handle high-dimensional data and its robustness to overfitting. It is an ensemble learning method that combines multiple decision trees to produce a more accurate and stable prediction.

The entire pipeline (preprocessing + model) was built using **Scikit-Learn's Pipeline** and **ColumnTransformer** utilities for streamlined and reproducible execution.

4. Model Evaluation

After training, the model was evaluated using the following metrics:

- **Confusion Matrix:** Shows true positives, true negatives, false positives, and false negatives in a heatmap for visual interpretation.

- **Accuracy:** Proportion of correct predictions among total predictions.
- **Precision:** Proportion of true positives among all predicted positives.
- **Recall:** Proportion of true positives among all actual positives.
- **F1-Score:** Harmonic mean of precision and recall.

These metrics provide a comprehensive view of model performance, especially in imbalanced classification scenarios.

CODE:-

```
# Step 1: Install necessary packages (optional in Colab)

!pip install seaborn scikit-learn --quiet

# Step 2: Import libraries

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion_matrix, classification_report

# Step 3: Load the dataset

# You can upload manually in Colab using the widget

from google.colab import files

uploaded = files.upload()

# Replace with the actual filename if different

df = pd.read_csv("1. Predict Loan Default.csv")

# Step 4: Data Preprocessing

df = df.drop(columns=["LoanID"])  # Drop ID column
```

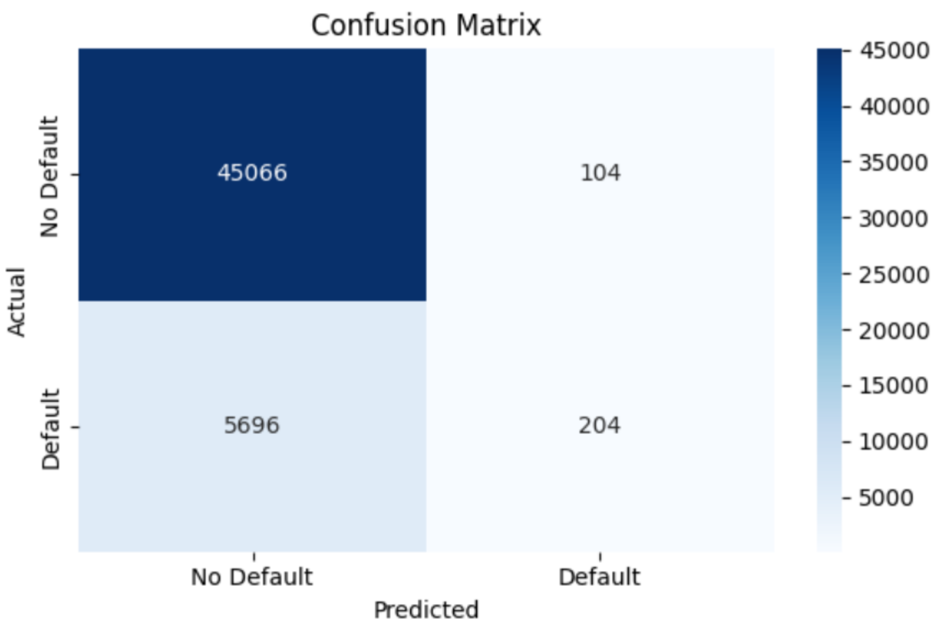
```
# Convert 'Yes'/'No' to 1/0

binary_columns = ['HasMortgage', 'HasDependents', 'HasCoSigner']

df[binary_columns] = df[binary_columns].replace({'Yes': 1, 'No': 0})

# Define features and target

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

y = df['Default']

# Identify column types

categorical_cols = ['Education', 'EmploymentType', 'MaritalStatus', 'LoanPurpose']

numeric_cols = X.select_dtypes(include=['int64',
'float64']).columns.difference(binary_columns)

# Step 5: Build pipeline

preprocessor = ColumnTransformer(

    transformers=[

        ('num', StandardScaler(), numeric_cols),

        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_cols)

    ],

    remainder='passthrough'  # Leave binary columns as-is

)

clf = Pipeline(steps=[

    ('preprocessor', preprocessor),

    ('classifier', RandomForestClassifier(random_state=42))

])

# Step 6: Train/Test Split and Fit
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

clf.fit(X_train, y_train)

# Step 7: Predictions and Evaluation

y_pred = clf.predict(X_test)

cm = confusion_matrix(y_test, y_pred)

report = classification_report(y_test, y_pred, output_dict=True)

# Step 8: Plot Confusion Matrix

plt.figure(figsize=(6, 4))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No Default', 'Default'],
yticklabels=['No Default', 'Default'])

plt.title('Confusion Matrix')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.tight_layout()

plt.show()

# Step 9: Print Evaluation Metrics

report_df = pd.DataFrame(report).transpose()

display(report_df)
```

OUTPUT/RESULT:-

- **1. Predict Loan Default.csv**(text/csv) - 24834870 bytes, last modified: 4/18/2025 - 100% done

```
Saving 1. Predict Loan Default.csv to 1. Predict Loan Default (1).csv
<ipython-input-1-40299cebf0f5>:30: FutureWarning: Downcasting behavior in `replace` is deprecated
  df[binary_columns] = df[binary_columns].replace({'Yes': 1, 'No': 0})
```



	precision	recall	f1-score	support
0	0.887790	0.997698	0.939541	45170.00000
1	0.662338	0.034576	0.065722	5900.00000
accuracy	0.886430	0.886430	0.886430	0.88643
macro avg	0.775064	0.516137	0.502631	51070.00000
weighted avg	0.861744	0.886430	0.838590	51070.00000