Assessment Report

on

“Loan Default prediction”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

Introduction to AI

By

Utkarsh Singh (20240110300271, CSE-AI- D)

Under the supervision of

“MR.ABHISHEK SHUKLA”

KIET Group of Institutions, Ghaziabad

19 April,2025

Introduction

Predicting loan defaults is a critical task for financial institutions to minimize risk and make informed lending decisions. Loan default occurs when a borrower fails to repay a loan, resulting in financial losses for the lender. By leveraging machine learning, it is possible to analyze historical loan data and build predictive models that estimate the likelihood of default before a loan is granted.

In this project, we develop a classification model to predict whether a borrower will default on a loan using features such as age, income, credit score, loan amount, interest rate, and employment status. The dataset is preprocessed to handle categorical variables and imbalanced classes. A Random Forest Classifier is chosen for its robustness, interpretability, and ability to handle both numerical and categorical data.

Model performance is evaluated using accuracy, precision, and recall. While the model achieves high accuracy and reasonable precision, the recall is notably low, indicating that the model struggles to correctly identify actual defaulters. This reflects a common challenge in imbalanced classification problems, where the minority class (defaulters) is underrepresented.

To address this, future improvements could include techniques like oversampling, undersampling, or algorithmic adjustments to enhance recall and build a more balanced, risk-aware model. Overall, this project highlights the potential and limitations of machine learning in financial risk prediction.

Methodology for Loan Default Prediction

1. Problem Understanding

The goal is to predict whether a borrower will default on a loan using historical data. This helps financial institutions minimize risk and make informed lending decisions. The prediction is treated as a binary classification problem where the target variable is "Default" (yes or no).

2. Data Acquisition

The dataset is loaded from a CSV file and contains various features including age, income, loan amount, interest rate, credit score, employment details, and more. Each row represents a borrower's loan application.

3. Exploratory Data Analysis (EDA)

Initial exploration involves checking the structure and contents of the dataset. Summary statistics (mean, median, etc.) are reviewed, and potential issues such as missing values or incorrect data types are identified.

4. Data Cleaning

Irrelevant or non-informative columns like unique identifiers (e.g., Loan ID) are removed. Missing values are either dropped or imputed depending on their nature and frequency. Consistency in data types is ensured, and duplicate entries are eliminated if any are found.

5. Encoding Categorical Variables

Since machine learning models require numerical inputs, all categorical features (like employment status, education, etc.) are converted into numerical format using encoding techniques such as label encoding. This step helps the model interpret categorical information.

6. Feature and Target Selection

The target variable (whether the borrower defaulted or not) is separated from the feature set. The rest of the columns serve as independent variables that the model will learn from to predict the target.

7. Optional Sampling

If the dataset is large, a small representative subset (e.g., 20%) is taken for quicker experimentation and model prototyping, while still maintaining statistical relevance.

8. Train-Test Split

The dataset is divided into training and testing sets. Typically, 80% of the data is used to train the model, and the remaining 20% is used to evaluate how well the model generalizes to unseen data.

9. Model Selection and Training

A Random Forest Classifier is chosen for its robustness and ability to handle both categorical and numerical features. It is trained using the training data. The model builds an ensemble of decision trees and averages their predictions to improve accuracy and reduce overfitting.

10. Model Evaluation

Once trained, the model's predictions are compared to actual outcomes in the test set. Evaluation metrics include:

- **Accuracy: Proportion of total correct predictions.**
- **Precision: Proportion of predicted defaulters who were actual defaulters.**
- **Recall: Proportion of actual defaulters correctly identified.**

A confusion matrix is also used to visualize the true positives, true negatives, false positives, and false negatives.

11. Model Insights

The model may show high accuracy and precision, but low recall. This indicates it fails to identify a significant number of actual defaulters, which could be due to class imbalance (i.e., far fewer defaulters than non-defaulters in the data).

12. Dealing with Class Imbalance

To improve recall and better detect defaulters:

- **Resampling: Techniques like oversampling (SMOTE) or undersampling can be used to balance the dataset.**
- **Class weights: Adjusting the model to penalize misclassification of defaulters more heavily.**
- **Advanced models: Trying other algorithms such as XGBoost or Logistic Regression with penalty terms.**

CODE

```
# 🎁 Import necessary libraries

from google.colab import files

import pandas as pd

import io

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy_score, precision_score, recall_score,
confusion_matrix


# 📂 Upload and load the dataset

uploaded = files.upload()

filename = list(uploaded.keys())[0]

df = pd.read_csv(io.BytesIO(uploaded[filename]))


# 🔍 Display the first few rows

print("First 5 rows of the dataset:")

print(df.head())
```

```
# □ Drop 'LoanID' column if it exists
if 'LoanID' in df.columns:
    df = df.drop("LoanID", axis=1)

# 🏷️ Encode categorical variables
label_encoders = {}
categorical_cols = df.select_dtypes(include='object').columns
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# 🎯 Define features and target
X = df.drop("Default", axis=1)
y = df["Default"]

# □ Sample 20% of data for faster training (optional)
df_sampled = df.sample(frac=0.2, random_state=42)
```

```
X = df_sampled.drop("Default", axis=1)

y = df_sampled["Default"]



# 🔀 Train-test split

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



# 🌲 Train a Random Forest Classifier

model = RandomForestClassifier(n_estimators=100, max_depth=10,
random_state=42)

model.fit(X_train, y_train)



# □ Predict on test data

y_pred = model.predict(X_test)



# 📊 Calculate evaluation metrics

accuracy = accuracy_score(y_test, y_pred)

precision = precision_score(y_test, y_pred)

recall = recall_score(y_test, y_pred)
```

```
# 🖨️ Print the evaluation results
print("\nEvaluation Metrics:")
print(f"Accuracy : {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall   : {recall:.4f}")


# 🔥 Confusion Matrix Heatmap
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=["No Default", "Default"],
            yticklabels=["No Default", "Default"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix Heatmap")
plt.tight_layout()
plt.show()
```

Result

```
Evaluation Matrics:
Accuracy : 0.8878
Precision: 0.7021
Recall   : 0.0283
```