

**An**

**Assessment Report**

on

**“Predict Loan Default”**

submitted as partial fulfillment for the award of

**BACHELOR OF TECHNOLOGY**

**DEGREE**

SESSION 2024-25

in

**CSE (AI & ML)**

By

Shashi Ranjan(202401100400171)

**Under the supervision of**

“Abhishek Shukla”

**KIET Group of Institutions, Ghaziabad**

**Introduction**

Loan default prediction is a crucial task in the financial sector. By predicting whether a borrower will default on a loan, financial institutions can reduce risk and make informed lending decisions. In this task, we use a dataset containing financial and demographic information of borrowers to build a classification model that predicts the likelihood of a loan default. Visualization and evaluation of results help validate the model performance.

**Methodology**

1. **Data Loading:** Loaded the dataset 1. Predict Loan Default.csv.
2. **Preprocessing:**
   1. Dropped irrelevant columns (e.g., LoanID).
   2. Encoded categorical variables using Label Encoding.
3. **Sampling:** Sampled 20,000 rows from the dataset for faster model training.
4. **Splitting:** Divided the data into training and testing sets (80%-20%).
5. **Model Training:** Trained a Random Forest classifier with 10 estimators.
6. **Evaluation:** Calculated accuracy, precision, recall and plotted a confusion matrix heatmap.

**Code**

import pandas aspd

importmatplotlib.pyplotasplt

importseabornassns

fromsklearn.model\_selectionimporttrain\_test\_split

fromsklearn.preprocessingimportLabelEncoder

fromsklearn.ensembleimportRandomForestClassifier

fromsklearn.metricsimportconfusion\_matrix, accuracy\_score, precision\_score, recall\_score

# Load the dataset

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

# Drop the LoanID column

df = df.drop(columns=["LoanID"])

# Encode categorical variables

categorical\_cols = df.select\_dtypes(include='object').columns

label\_encoders = {}

for col incategorical\_cols:

le = LabelEncoder()

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

label\_encoders[col] = le

# Sample a subset for faster training

df\_sampled = df.sample(n=20000, random\_state=42)

# Features and target

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

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, stratify=y

)

# Train a Random Forest classifier

clf = RandomForestClassifier(n\_estimators=10, random\_state=42)

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

# Predictions

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

# Evaluation metrics

cm = confusion\_matrix(y\_test, y\_pred)

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

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

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

# Print metrics

print("Accuracy:", round(accuracy, 4))

print("Precision:", round(precision, 4))

print("Recall:", round(recall, 4))

# Confusion matrix heatmap

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

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',

xticklabels=["No Default", "Default"],

yticklabels=["No Default", "Default"])

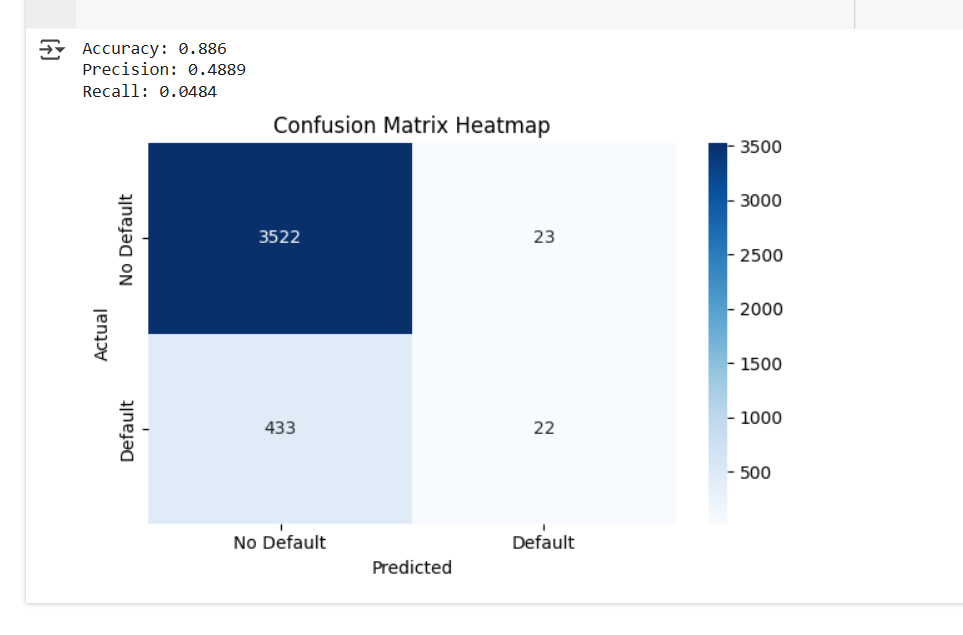
plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix Heatmap')

plt.tight\_layout()plt.show()

**OUTPUT**



**References/Credits**

* Dataset: Provided for AI MSE
* Libraries used: pandas, matplotlib, seaborn, scikit-learn
* Implementation: Done in Google Colab