****

**Assessment Report**

on

**“Problem Statement”**

submitted as partial fulfillment for the award of

**BACHELOR OF TECHNOLOGY**

**DEGREE**

SESSION 2024-25

in

**CSE(AIML)**

By

Name - Kanishak Tyagi

Roll Number – 202401100400104

Branch- CSE(aiml)

Section - B

**Under the supervision of**

“Abhishek Shukla Sir”

**KIET Group of Institutions, Ghaziabad**

**18,May, 2025**

**Introduction**

**Problem Overview**:

Loan default prediction is a critical problem in the finance sector. Financial institutions need to predict whether a borrower will default on a loan based on various factors like credit score, income, and loan amount. This prediction helps banks reduce financial risk by making informed lending decisions.

**Goal**:

The objective is to build a classification model that predicts whether a borrower will default on a loan (yes/no) based on their financial data.

**Key Features:**

* **Credit Score**: This score is an indication of a borrower’s creditworthiness and is based on their credit history. A low credit score often indicates a higher risk of default.
* **Income**: A borrower’s income plays a significant role in their ability to repay a loan. A higher income generally implies a higher capacity to repay.
* **Loan Amount**: The amount of money being borrowed. A larger loan might increase the likelihood of default, particularly if the borrower’s income is not sufficient to meet repayment obligations.
* **Other Financial History**: This could include factors such as existing debt, repayment history, and other loans held by the borrower.

**Methodology**

**1. Data Preprocessing:**

* **Data Cleaning**: The dataset is inspected for missing or inconsistent values and cleaned accordingly.
* **Feature Selection**: Relevant features such as **credit score**, **income**, and **loan amount** are selected to predict loan default.

**2. Model Selection:**

**Random Forest Classifier** is chosen due to its ability to handle complex relationships between features and target variable, and its robustness in classification tasks.

**3. Train-Test Split:**

The dataset is split into training (80%) and testing (20%) sets. This allows the model to learn from the training set and be evaluated on unseen data using the test set.

**4. Model Training:**

The **Random Forest Classifier** is trained on the training data. It constructs decision trees and aggregates their results to make final predictions.

**5. Model Evaluation:**

The model is evaluated using:

* **Accuracy**: Proportion of correct predictions.
* **Precision**: Accuracy of predicted defaults.
* **Recall**: Ability to identify actual defaults.
* **Confusion Matrix**: A heatmap is generated to visualize the performance.

**CODE**

**# Import libraries**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn.metrics import confusion\_matrix, classification\_report**

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

**from sklearn.preprocessing import LabelEncoder, StandardScaler**

**# Step 1: Load dataset**

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

**# Step 2: Drop LoanID column if present**

**if 'LoanID' in df.columns:**

**df.drop(columns=['LoanID'], inplace=True)**

**# Step 3: Encode categorical columns**

**for col in df.select\_dtypes(include='object').columns:**

**df[col] = LabelEncoder().fit\_transform(df[col])**

**# Step 4: Split data into features and target**

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

**y = df['Default']**

**# Step 5: Train-test split**

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

**# Step 6: Scale features**

**scaler = StandardScaler()**

**X\_train\_scaled = scaler.fit\_transform(X\_train)**

**X\_test\_scaled = scaler.transform(X\_test)**

**# Step 7: Train Logistic Regression model**

**model = LogisticRegression(max\_iter=1000)**

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

**# Step 8: Predict**

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

**# Step 9: Confusion Matrix**

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

**# Step 10: Plot Confusion Matrix Heatmap**

**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.show()**

**# Step 11: Evaluation Metrics**

**print("Classification Report:")**

**print(classification\_report(y\_test, y\_pred, target\_names=["No Default", "Default"]))**

**OUTPUT**

A graph with numbers and a number

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

**References:**

1. **Dataset**: (1.Predict Loan Default Provide by SIR.)
2. **Images**:
   * Confusion matrix heatmap generated using Seaborn library.
3. **Libraries**:
   * Python libraries used include pandas, numpy, scikit-learn, matplotlib, and seaborn.
4. **External Content**:
   * Model evaluation techniques based on scikit-learn documentation: <https://scikit-learn.org>