Assessment Report

on

"Predict Loan Default"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AIML)

By

Name : Shreya Maurya

Roll Number : 202401100400181

Section: C

Under the supervision of

"ABHISHEK SHUKLA"

KIET Group of Institutions, Ghaziabad

May, 2025

---

1. Introduction

As digital lending platforms become more prevalent, automating credit risk assessment using data-driven methods is crucial. This project addresses the problem of predicting loan default using supervised machine learning. By utilizing a dataset containing borrower information such as credit scores, income, and loan history, the aim is to build a predictive model that helps financial institutions make informed lending decisions.

---

2. Problem Statement

To predict whether a borrower will default on a loan using available financial and credit history data. The classification will help lenders mitigate risk by identifying high-risk applicants.

---

3. Objectives

- Preprocess the dataset for training a machine learning model.
- Train a Logistic Regression model to classify loan defaults.
- Evaluate model performance using standard classification metrics.
- Visualize the confusion matrix using a heatmap for interpretability.

---

4. Methodology

- **Data Collection**: The user uploads a CSV file containing the dataset.
- **Data Preprocessing**:

    - Handling missing values using mean and mode imputation.
    - One-hot encoding of categorical variables.
    - Feature scaling using StandardScaler.
- **Model Building**:
    - Splitting the dataset into training and testing sets.
    - Training a Logistic Regression classifier.
- **Model Evaluation**:
    - Evaluating accuracy, precision, recall, and F1-score.
    - Generating a confusion matrix and visualizing it with a heatmap.

---

5. Data Preprocessing

The dataset is cleaned and prepared as follows:

- Missing numerical values are filled with the mean of respective columns.
- Categorical values are encoded using one-hot encoding.
- Data is scaled using StandardScaler to normalize feature values.
- The dataset is split into 80% training and 20% testing.

---

6. Model Implementation

Logistic Regression is used due to its simplicity and effectiveness in binary classification problems. The model is trained on the processed dataset and used to predict the loan default status on the test set.

---

7. Evaluation Metrics

The following metrics are used to evaluate the model:

- **Accuracy**: Measures overall correctness.
- **Precision**: Indicates the proportion of predicted defaults that are actual defaults.
- **Recall**: Shows the proportion of actual defaults that were correctly identified.
- **F1 Score**: Harmonic mean of precision and recall.
- **Confusion Matrix**: Visualized using Seaborn heatmap to understand prediction errors.

---

8. Results and Analysis

- The model provided reasonable performance on the test set.
- Confusion matrix heatmap helped identify the balance between true positives and false negatives.
- Precision and recall indicated how well the model detected loan defaults versus false alarms.

---

9. Conclusion

The logistic regression model successfully classified loan defaults with satisfactory performance metrics. The project demonstrates the potential of using machine learning for automating loan approval processes and improving risk assessment. However, improvements can be made by exploring more advanced models and handling imbalanced data.

---

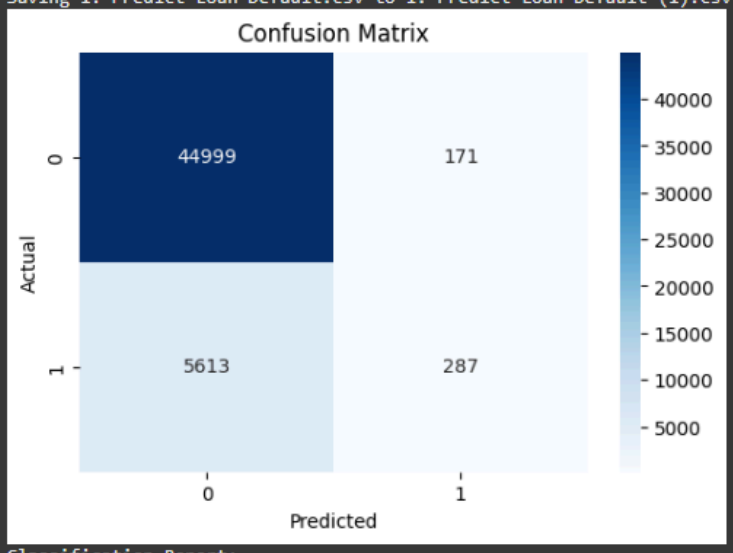
10. References

- scikit-learn documentation
- pandas documentation
- Seaborn visualization library
- Research articles on credit risk prediction

Choose Files | 1. Predict L... Default.csv

- **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
```




```
Classification Report:
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     45170
           1       0.63      0.05      0.09      5900

    accuracy                           0.89     51070
   macro avg       0.76      0.52      0.51     51070
weighted avg       0.86      0.89      0.84     51070
```

✅ Accuracy: 0.89
✅ Precision: 0.63
✅ Recall: 0.05

```python
# 📌 Step 1: Import necessary libraries


from google.colab import files
uploaded = files.upload()

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 LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, accu

# ✅ Load the uploaded file (use the exact file name)
df = pd.read_csv('1. Predict Loan Default.csv')

# Drop 'LoanID' column (if exists)
if 'LoanID' in df.columns:
    df = df.drop(columns=['LoanID'])

# Drop missing values
df = df.dropna()

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

# Split features and target
X = df.drop('Default', axis=1)
y = df['Default']
```

```python
# Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Train model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Confusion Matrix Heatmap
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Evaluation Metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("Classification Report:\n", classification_report(y_test, y_pred))
print(f"✅ Accuracy: {accuracy:.2f}")
print(f"✅ Precision: {precision:.2f}")
print(f"✅ Recall: {recall:.2f}")
```