Assessment Report

on

" Loan Default Prediction "

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AIML)

By

GROUP-8

Aditya Tyagi	202401100400018
Aman Sarkar	202401100400027
Anjali Gurjar	202401100400036
Arnav Gupta	202401100400048
Ashok Gangwar	202401100400057

Under the supervision of

'BIKKI KUMAR'

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

- Load and clean a real-world loan dataset.
- Preprocess the data, including handling categorical features.
- Train a **Decision Tree classifier** for prediction.
- Evaluate the model using accuracy, confusion matrix, and classification report.
- Visualize feature importance and data relationships.

---

4. Methodology

- **Data Collection**: Data loaded from CSV files (train_u6lujuX_CVtuZ9i.csv, test_Y3wMUE5_7gLdaTN.csv).

- **Data Preprocessing**:

- Removed extra spaces from column names.
- Encoded categorical features using LabelEncoder.
- Split the data into training and validation sets (80:20).
- No imputation or scaling was applied in the final implementation...

- **Model Evaluation**:

- Accuracy, Confusion Matrix, and Classification Report used as metrics.
- Heatmaps and bar plots were used to visualize evaluation results and feature importance

---

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 Standard Scaler to normalize feature values.
- The dataset is split into 80% training and 20% testing.

#CODE

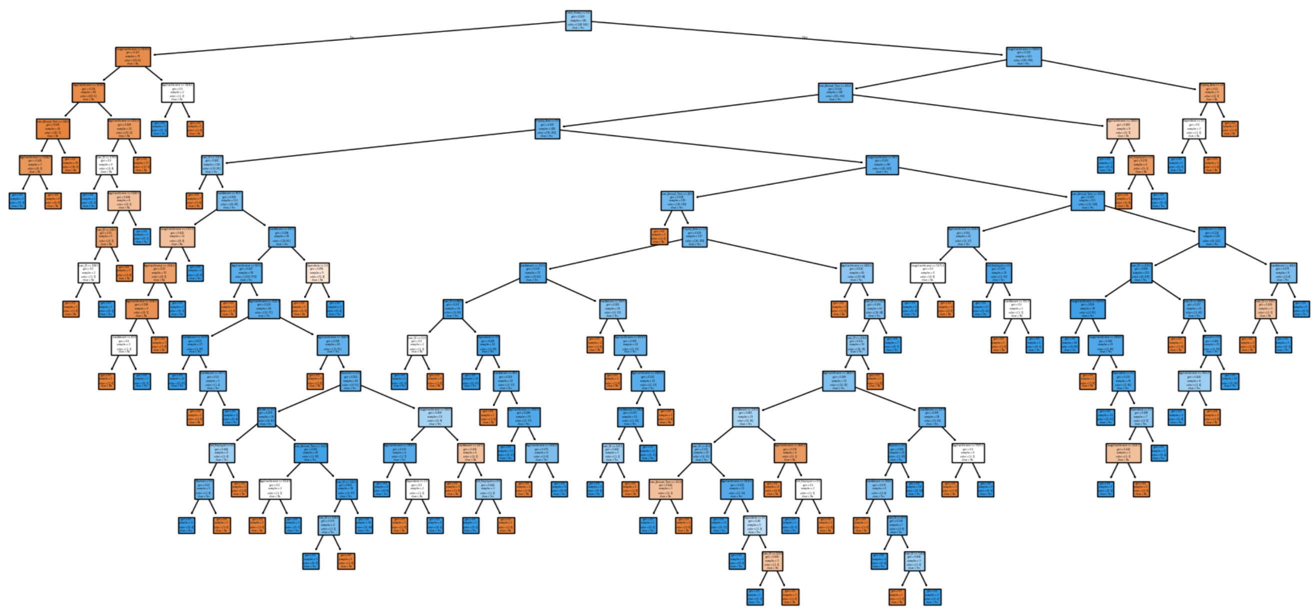
```
# Step 1: Import Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Step 2: Load Data

train = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/train_u6lujuX_CVtuZ9i.csv')

test = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/test_Y3wMUE5_7gLdaTN.csv')

# Step 3: Check and Clean Column Names

train.columns = train.columns.str.strip()   # remove any extra spaces

test.columns = test.columns.str.strip()

# Step 4: Explore Data

print(train.columns)                  # check column names

print(train['Loan_Status'].value_counts())  # ensure target column is present

# Step 7: Encode Categorical Columns
```

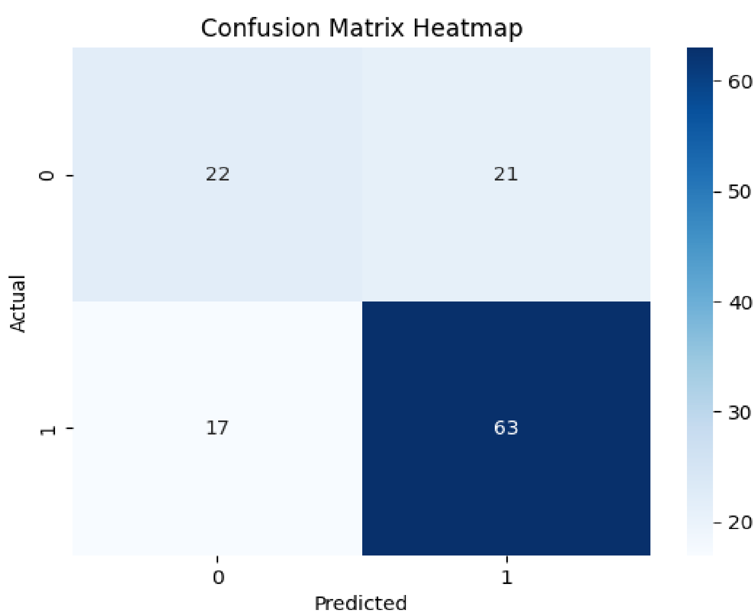
```
le = LabelEncoder()

for col in train.columns:

    if train[col].dtype == 'object':

        train[col] = le.fit_transform(train[col])


for col in test.columns:

    if test[col].dtype == 'object':

        test[col] = le.fit_transform(test[col])  # Caution: should be fit from train ideally

# Step 8: Prepare Features and Target

X = train.drop('Loan_Status', axis=1)  # features

y = train['Loan_Status']              # target

# Step 9: Split Train and Validation Set

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)


# Step 10: Train Decision Tree Classifier

model = DecisionTreeClassifier(random_state=42)

model.fit(X_train, y_train)

# Step 11: Evaluate the Model

y_pred = model.predict(X_val)


print("Accuracy:", accuracy_score(y_val, y_pred))
```

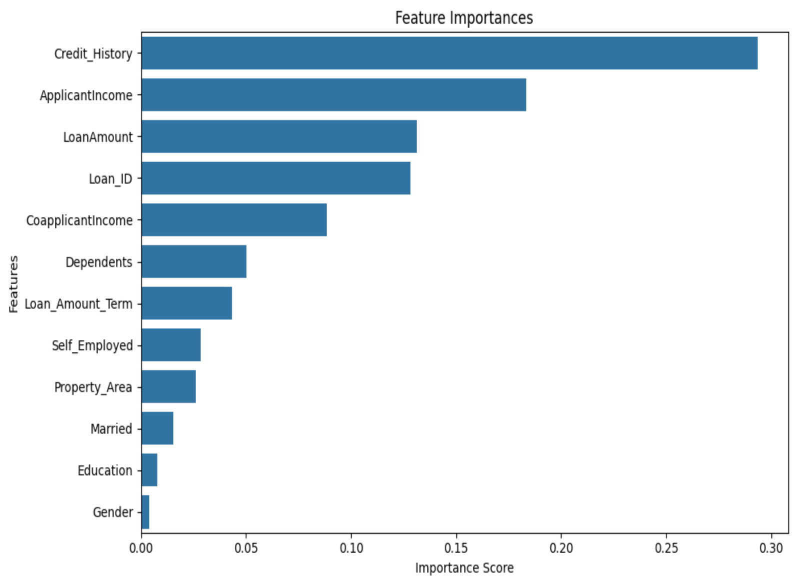
```python
print("Confusion Matrix:\n", confusion_matrix(y_val, y_pred))

print("Classification Report:\n", classification_report(y_val, y_pred))

# Step 12: Visualize Feature Importance

importances = model.feature_importances_

features = X.columns

indices = np.argsort(importances)[::-1]


plt.figure(figsize=(10, 6))

sns.barplot(x=importances[indices], y=features[indices])

plt.title('Feature Importances')

plt.xlabel('Importance Score')

plt.ylabel('Features')

plt.tight_layout()

plt.show()

#Confusion Matrix Heatmap

sns.heatmap(confusion_matrix(y_val, y_pred), annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix Heatmap')

plt.show()

#Decision Tree Plot
```

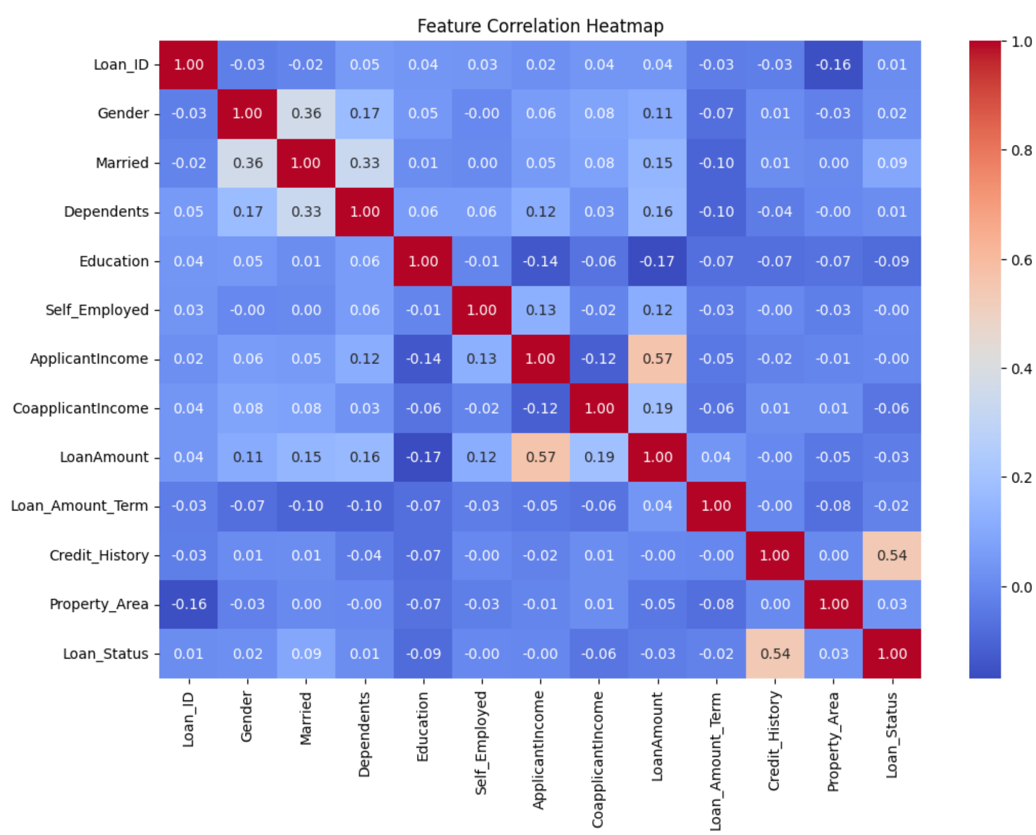
```
plt.figure(figsize=(20, 10))

plot_tree(model, filled=True, feature_names=X.columns, class_names=['No', 'Yes'])

plt.show()

#Correlation Heatmap

plt.figure(figsize=(12, 8))

sns.heatmap(train.corr(), annot=True, cmap='coolwarm', fmt=".2f")

plt.title("Feature Correlation Heatmap")

plt.show()
```

SCREENSHOTS






Feature Importances
Credit_History
ApplicantIncome
LoanAmount
Loan_ID
CoapplicantIncome
Dependents
Loan_Amount_Term
Self_Employed
Property_Area
Married
Education
Gender
Features
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Importance Score


Feature Correlation Heatmap
Loan_ID 1.00 -0.03 -0.02 0.05 0.04 0.03 0.02 0.04 0.04 -0.03 -0.03 -0.16 0.01
Gender -0.03 1.00 0.36 0.17 0.05 -0.00 0.06 0.08 0.11 -0.07 0.01 -0.03 0.02
Married -0.02 0.36 1.00 0.33 0.01 0.00 0.05 0.08 0.15 -0.10 0.01 0.00 0.09
Dependents 0.05 0.17 0.33 1.00 0.06 0.06 0.12 0.03 0.16 -0.10 -0.04 -0.00 0.01
Education 0.04 0.05 0.01 0.06 1.00 -0.01 -0.14 -0.06 -0.17 -0.07 -0.07 -0.07 -0.09
Self_Employed 0.03 -0.00 0.00 0.06 -0.01 1.00 0.13 -0.02 0.12 -0.03 -0.00 -0.03 -0.00
ApplicantIncome 0.02 0.06 0.05 0.12 -0.14 0.13 1.00 -0.12 0.57 -0.05 -0.02 -0.01 -0.00
CoapplicantIncome 0.04 0.08 0.08 0.03 -0.06 -0.02 -0.12 1.00 0.19 -0.06 0.01 0.01 -0.06
LoanAmount 0.04 0.11 0.15 0.16 -0.17 0.12 0.57 0.19 1.00 0.04 -0.00 -0.05 -0.03
Loan_Amount_Term -0.03 -0.07 -0.10 -0.10 -0.07 -0.03 -0.05 -0.06 0.04 1.00 -0.00 -0.08 -0.02
Credit_History -0.03 0.01 0.01 -0.04 -0.07 -0.00 -0.02 0.01 -0.00 -0.00 1.00 0.00 0.54
Property_Area -0.16 -0.03 0.00 -0.00 -0.07 -0.03 -0.01 0.01 -0.05 -0.08 0.00 1.00 0.03
Loan_Status 0.01 0.02 0.09 0.01 -0.09 -0.00 -0.00 -0.06 -0.03 -0.02 0.54 0.03 1.00
Loan_ID
Gender
Married
Dependents
Education
Self_Employed
ApplicantIncome
CoapplicantIncome
LoanAmount
Loan_Amount_Term
Credit_History
Property_Area
Loan_Status
1.0
0.8
0.6
0.4
0.2
0.0

Model Implementation

 Used **DecisionTreeClassifier** with random_state=42.

 Model trained on the 80% training set.

 Prediction evaluated on the 20% validation set.

 Visualized:

- Confusion Matrix as heatmap.
- Feature importance using barplot.
- Correlation heatmap of input features.
- Decision tree structure plotted.

Evaluation Metrics

 Accuracy Score

 Confusion Matrix

 Classification Report: Precision, Recall, F1-score

 Feature Importance Plot

 Correlation Heatmap

Conclusion

The Decision Tree classifier provides explainable and efficient predictions for loan default risk. With improvements like better encoding strategies or ensemble models (e.g., Random Forest) performance can be further enhanced

---

10. References

- pandas, numpy – data manipulation
- seaborn, matplotlib – data visualization
- scikit-learn – machine learning library