Assessment Report

on

"Predict Loan Default"

submitted as partial fulfilment for the award of

BACHELOR OF TECHNOLOGY

DEGREE

SESSION 2024-25

in

CSE(AI)

By

Name: Shubham Kumar (202401100300244)

Name: Vidisha Goel (202401100300278

Name: Saumya Dubey (202401100300221)

Name: Saumya Gupta (202401100300222)

Name: Sumedha Saxena (202401100300255)

Section: D

Under the supervision of

"Abhishek Shukla Sir"

KIET Group of Institutions, Ghaziabad

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 Feature importance in the decision tree.

---

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

In this project, a Decision Tree Classifier was implemented to predict loan approval status using a publicly available dataset. The data was uploaded and extracted in Google Colab, followed by preprocessing steps such as handling missing values, label encoding of categorical features, and dropping irrelevant columns. The dataset was split into training and validation sets (80:20), and the model was trained using the training set. Evaluation was performed using accuracy, precision, recall, and F1 score metrics on the validation set. Feature importance was also visualized to understand the contribution of each variable to the model's decision-making process

---

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.

- **Feature Importance**: Feature importance in a Decision Tree indicates how much each feature contributes to the model's predictions, helping identify the most influential variables.

---

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

---

11. Codes

```
# STEP 0: Upload and Extract Dataset ZIP
from google.colab import files
import zipfile
import os
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
from sklearn.impute import SimpleImputer
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score

# Upload and extract
uploaded = files.upload()
for zip_filename in uploaded.keys():
    with zipfile.ZipFile(zip_filename, 'r') as zip_ref:
        zip_ref.extractall("loan_data")
print("Extracted files:", os.listdir("loan_data"))

# STEP 1: Load Training Data
train_df = pd.read_csv("loan_data/train_u6lujuX_CVtuZ9i.csv")

# STEP 2: Preprocess Data
train_df.drop("Loan_ID", axis=1, inplace=True)
```

```
# Fill missing values
for col in train_df.select_dtypes(include='object').columns:
    train_df[col].fillna(train_df[col].mode()[0], inplace=True)
for col in train_df.select_dtypes(include=['float64', 'int64']).columns:
    train_df[col].fillna(train_df[col].median(), inplace=True)

# Encode categorical variables
le = LabelEncoder()
for col in train_df.select_dtypes(include='object').columns:
    train_df[col] = le.fit_transform(train_df[col])

# Split features and target
X = train_df.drop('Loan_Status', axis=1)
y = train_df['Loan_Status']
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# STEP 3: Train Decision Tree
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
dt_pred = dt_model.predict(X_val)
dt_acc = accuracy_score(y_val, dt_pred)
print("\nDecision Tree Accuracy:", dt_acc)
print(classification_report(y_val, dt_pred))

# STEP 4: Train Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_val)
```

```
rf_acc = accuracy_score(y_val, rf_pred)
print("\nRandom Forest Accuracy:", rf_acc)
print(classification_report(y_val, rf_pred))

# STEP 5: Feature Importance (Random Forest)
importances = rf_model.feature_importances_
features = X.columns
feat_imp = pd.DataFrame({'Feature': features, 'Importance': importances})
feat_imp = feat_imp.sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10, 6))
sns.barplot(data=feat_imp, x='Importance', y='Feature', palette='viridis')
plt.title("Feature Importance - Random Forest")
plt.tight_layout()
plt.show()
```

---

12. Output Screenshots

- **Accuracy**

```
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_val)
print("Accuracy:", accuracy_score(y_val, y_pred))
print(classification_report(y_val, y_pred))
```

```
Accuracy: 0.6910569105691057
              precision    recall  f1-score   support

           0       0.56      0.53      0.55        43
           1       0.76      0.78      0.77        80

    accuracy                           0.69       123
   macro avg       0.66      0.65      0.66       123
weighted avg       0.69      0.69      0.69       123
```

- **Feature importance**