Assessment Report

on

"Predict Loan Default"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI)

By

Name : PRIYESH KUMAR

Roll Number : 202401100300186

Section: C

Under the supervision of

"MAYNAK SIR"

KIET Group of Institutions, Ghaziabad

May, 2025

---

1. Introduction

Efficiently categorizing support cases can significantly improve response times and resource allocation within customer service teams. This project focuses on using a machine learning approach—specifically, a Random Forest Classifier—to automatically classify support cases based on measurable attributes such as message length and response time.

---

2. Problem Statement

he goal of this project is to build a classification model that can predict the type of a support case using features from case metadata. By automating this process, organizations can streamline operations and ensure cases are routed to the appropriate team with minimal manual effort.

---

3. Objectives

- Load and preprocess the support case dataset
- Train a Random Forest Classifier on relevant features
- Evaluate the model using standard classification metrics
- Visualize class distribution and model performance
- Enable real-time classification of new cases

---

4. Methodology

Data Collection

The dataset was loaded from a CSV file and contains the following key columns:

- `message_length`: Number of characters in the customer's message
- `response_time`: Time (in minutes) taken to respond
- `case_type`: The category of support required (target variable)

Feature Selection

- Features: `message_length`, `response_time`
- Target: `case_type`

Data Splitting

The dataset was split into training (80%) and testing (20%) sets using `train_test_split`.

Model Building

A Random Forest Classifier was trained with default parameters and a fixed random seed for reproducibility.

-

---

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

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,
accuracy_score, confusion_matrix, ConfusionMatrixDisplay
```

```
# Step 1: Load your data
df = pd.read_csv("/content/support_cases.csv")  # Make sure
your file is in the same folder or give the correct path

# Step 2: Feature and target selection
X = df[['message_length', 'response_time']]
y = df['case_type']

# Step 3: Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Step 4: Train the model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Step 5: Evaluate the model
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n",
classification_report(y_test, y_pred))

# Step 6: Plot 1 - Distribution of Case Types
plt.figure(figsize=(6, 4))
sns.countplot(data=df, x='case_type', palette='Set2')
plt.title('Distribution of Case Types')
plt.xlabel('Case Type')
plt.ylabel('Count')
plt.tight_layout()
plt.show()

# Step 7: Plot 2 - Confusion Matrix
cm = confusion_matrix(y_test, y_pred,
labels=model.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=model.classes_)
```

```
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.tight_layout()
plt.show()

# Step 8: Optional - Predict new case
def classify_case(message_length, response_time):
    input_data = pd.DataFrame([[message_length,
response_time]], columns=['message_length',
'response_time'])
    prediction = model.predict(input_data)[0]
    return prediction

# Example prediction
print("Prediction for a new case (length=250,
response_time=20):", classify_case(250, 20))
```