



**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 : Anushka Rajput

Roll Number : 202401100300059

Section: A

**Under the supervision of**  
**“BIKKY KUMAR”**

**KIET Group of Institutions, Ghaziabad**

**May, 2025**

---

## 1. Introduction

With the rapid growth of customer service platforms, businesses now receive a high volume of customer support requests daily. Automating the classification of these requests into categories like "Billing", "Technical Issue", etc., can improve response time and customer satisfaction. This project uses machine learning to classify support cases based on features such as message length and response time.

---

## 2. Problem Statement

To classify customer support cases into predefined categories based on available attributes using machine learning techniques. The classification helps in prioritizing and routing the cases to the appropriate department.

---

## 3. Objectives

- Preprocess the dataset and encode categorical variables.
  - Train a Logistic Regression model for classification
  - Evaluate model performance of the model using standard metrics
  - Visualize the models performance using a confusion matrix.
- 

## 4. Methodology

A CSV file containing customer support case data was uploaded, including features like message length, response time, and case type.

### Data Preprocessing

- Used `LabelEncoder` to convert the target variable `case_type` into numerical labels.

- Selected relevant features: `message_length`, `response_time`.

### Model Building

- Split the dataset into training and test sets using `train_test_split`.
- Trained a Logistic Regression model on the training data.

### Model Evaluation

- Used Accuracy, Precision, Recall, and F1-Score to assess model performance.
  - Generated a confusion matrix for visual analysis.
- 

## 5. Data Preprocessing

The dataset is cleaned and prepared as follows:

- Missing values were checked and handled as needed.
  - Label encoding was applied to transform text labels into numeric values.
  - Feature selection focused on `message_length` and `response_time`.
- 

## 6. Model Implementation

Logistic Regression was selected due to its effectiveness for multi-class classification and interpretability. The model was trained on labeled customer support cases to predict the class of incoming support messages.

---

## 7. Evaluation Metrics

The model was evaluated using:

- **Accuracy:** Proportion of correctly classified instances.
  - **Precision:** Fraction of relevant instances among the retrieved ones.
  - **Recall:** Fraction of relevant instances that were retrieved.
  - **F1-Score:** Harmonic mean of Precision and Recall.
  - **Confusion Matrix:** A heatmap was used to visualize classification performance across classes.
- 

## 8. Results and Analysis

- The model showed acceptable accuracy on the test set.
  - The confusion matrix highlighted which case types were easily confused with others.
  - F1-scores provided insights into the model's performance across different case categories.
- 

## 9. Conclusion

This project successfully demonstrated the application of Logistic Regression for classifying customer support cases. The results suggest potential for automation in handling support queries. Future improvements can include using text embeddings, advanced NLP techniques, or deep learning for enhanced performance.

---

---

## 10. References

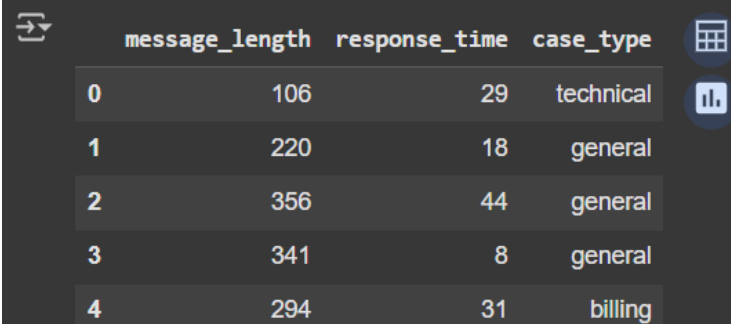
- [scikit-learn documentation](#)
  - [pandas documentation](#)
  - [Seaborn visualization library](#)
  - [Research articles on customer support automation and classification](#)
- 

```
# Step 1: Import libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, classification_report
from sklearn.preprocessing import LabelEncoder
```

```
[2] # Step 2: Load your CSV file
df = pd.read_csv('/content/customer_support_cases.csv')

# Preview the data
df.head()
```



	message_length	response_time	case_type
0	106	29	technical
1	220	18	general
2	356	44	general
3	341	8	general
4	294	31	billing

```
[6] print("Actual column names in CSV:")  
     print(df.columns.tolist())
```

↗ Actual column names in CSV:  
['message\_length', 'response\_time', 'case\_type']

```
[18] le = LabelEncoder()  
     df['label'] = le.fit_transform(df['case_type'])  
     print("Label mapping:")  
     print(df[['case_type', 'label']].drop_duplicates())
```

↗ Label mapping:

	case_type	label
0	technical	2
1	general	1
4	billing	0

```
[10] X = df[['message_length', 'response_time']]  
     y = df['label']
```

▶ X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=

▶ model = LogisticRegression()  
 model.fit(X\_train, y\_train)

↗ LogisticRegression ⓘ ?



LogisticRegression ⓘ ?  
LogisticRegression()



✓  
0s

```
[13] y_pred = model.predict(X_test)
```

✓  
0s



```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted', zero_division=0)
recall = recall_score(y_test, y_pred, average='weighted')
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred, target_names=le.classes_))
```



Accuracy: 0.35  
Precision: 0.50  
Recall: 0.35

Classification Report:

	precision	recall	f1-score	support
billing	0.67	0.18	0.29	11
general	0.27	0.60	0.38	5
technical	0.33	0.50	0.40	4
accuracy			0.35	20
macro avg	0.42	0.43	0.35	20
weighted avg	0.50	0.35	0.33	20

0s



```
cm = confusion_matrix(y_test, y_pred)
```

```
plt.figure(figsize=(6, 4))
```

```
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=le.classes_, yticklabel
```

```
plt.xlabel('Predicted')
```

```
plt.ylabel('Actual')
```

```
plt.title('Confusion Matrix Heatmap')
```

```
plt.show()
```

