Customer Support Case Type Classification

A Project Report

Problem Statement - Classify support cases into billing, technical, or general queries.

Submitted by

Sanskriti Agrawal

Univ Roll No – 202401100300216

Branch – CSEAI

Section - C

in partial fulfillment for the award of the degree

of

BTech

(Computer Science & Engineering -Artificial Intelligence)

Introduction

Customer support teams deal with all kinds of questions every day—some about billing, others about technical problems, and many that are just general inquiries. Manually sorting is a tedious job.

This project aims to make that process faster and smarter by using machine learning to automatically classify support cases into three types: billing, technical, and general. It uses simple information like how long the message is and how quickly it was responded to, and train a model to learn from that. Then, we check how well the model works using common evaluation methods like accuracy, precision, and recall, and heatmap for data visualization.

Methodology

1. Data Loading:

The dataset is loaded into a pandas DataFrame using pd.read_csv(). This dataset contains the customer support cases, including features like message_length, response_time, and the target variable case_type, which classifies the cases into categories like billing, technical, and general.

2. Feature Selection And Target Definition

The features used for model training are message_length and response_time.
The target variable, case_type, is the class to be predicted (billing, technical, or general).

3. Label Encoding

Since case_type is a categorical variable (billing, technical, or general), it is encoded into numeric values using LabelEncoder from scikit-learn. This is necessary for the machine learning model to process categorical variables.

4. Train-Test Split

The dataset is split into training and testing subsets using `train_test_split()` from scikit-learn. This ensures that the model is trained on one part of the data and tested on another, preventing overfitting and providing an unbiased evaluation.

80% of the data is used for training, and 20% is used for testing.

5. Model Training

A Random Forest Classifier is chosen as the model. It is a robust, ensemble learning algorithm that works well with both classification and regression problems. The model is trained using the training data (X_train, y_train).

6. Prediction & Model Evaluation

Once the model is trained, it is used to make predictions on the test data (X_test). The model's performance is evaluated using: Accuracy, Precision & Heatmaps

CODE

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt


from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion_matrix, classification_report, accuracy_score,

precision_score, recall_score


# ==== Load dataset ====

# Replace 'your_dataset.csv' with your actual file name

df = pd.read_csv('/content/support_cases.csv')


# ==== Features and target ====

X = df[['message_length', 'response_time']]

y = df['case_type']


# ==== Encode target labels ====

le = LabelEncoder()

y_encoded = le.fit_transform(y)


# ==== Train-test split ====

X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)


# ==== Model training ====

model = RandomForestClassifier(random_state=42)
```

```
model.fit(X_train, y_train)


# ==== Predictions ====
y_pred = model.predict(X_test)


# ==== Evaluation ====
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision (macro):", precision_score(y_test, y_pred, average='macro'))
print("Recall (macro):", recall_score(y_test, y_pred, average='macro'))
print("\nClassification Report:\n", classification_report(y_test, y_pred,
target_names=le.classes_))


# ==== Confusion matrix heatmap ====
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=le.classes_,
yticklabels=le.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap')
plt.show()
```

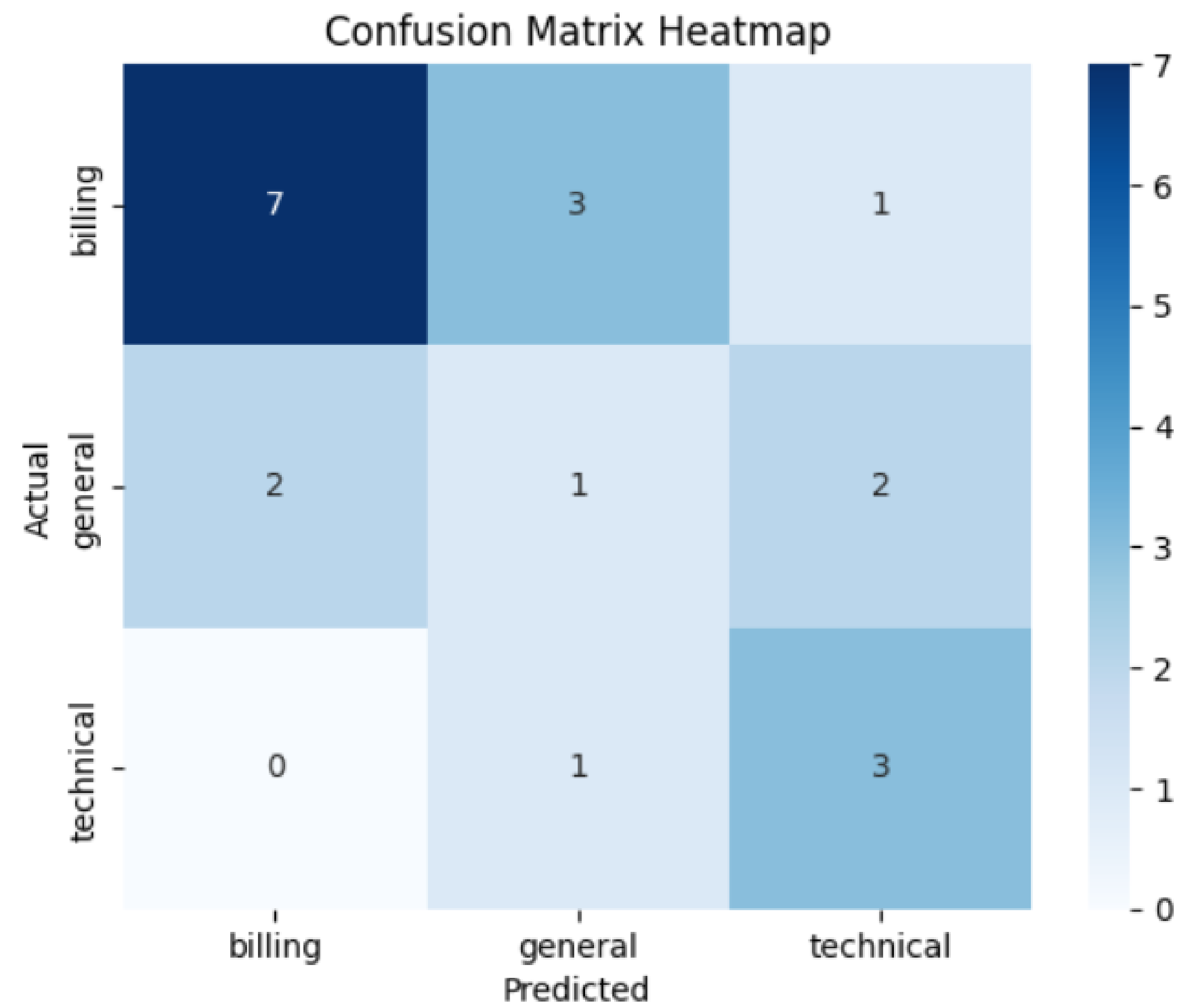
Output

```
Accuracy: 0.55
Precision (macro): 0.49259259259259264
Recall (macro): 0.5287878787878788

Classification Report:
              precision    recall  f1-score   support

     billing       0.78      0.64      0.70        11
     general       0.20      0.20      0.20         5
   technical       0.50      0.75      0.60         4

    accuracy                           0.55        20
   macro avg       0.49      0.53      0.50        20
weighted avg       0.58      0.55      0.55        20
```



References

- Scikit-learn Documentation: https://scikit-learn.org
- Matplotlib & Seaborn for visualization
- Dataset provided by Instructor/University