CUSTOMER SUPPORT CASE TYPE CLASSIFICATION REPORT

INTRODUCTION TO AI

NAME : PRASHASTHA SINGH
BRANCH : CSE-AI
SEC : C
ROLL NO : 202401100300176

KIET GROUP OF INSTITUTION , GHAZIABAD

1. Introduction

Customer support centers receive a large volume of queries that can be broadly categorized into types like **Billing**, **Technical**, or **General**. Classifying these cases automatically helps in routing them to the right department, reducing resolution time and improving customer satisfaction.

This project aims to build a **text classification model** that categorizes customer support cases into the correct category using Natural Language Processing (NLP) techniques and a **Naive Bayes classifier**.

2. Methodology

➤ **Dataset**

A mock dataset was used initially with case descriptions and their corresponding labels: Billing, Technical, or General. You can replace this with a real-world dataset for practical deployment.

➤ **Preprocessing**

- **Text Vectorization**: The case descriptions were converted into numerical features using **TF-IDF Vectorization**.
- **Label Encoding**: Labels were kept as strings, suitable for Naive Bayes classification.

➤ **Model**

- We used **Multinomial Naive Bayes**, a common algorithm for text classification tasks.
- The data was split into **training (70%)** and **testing (30%)** sets.

➤ **Evaluation Metrics**

We evaluated the model using:

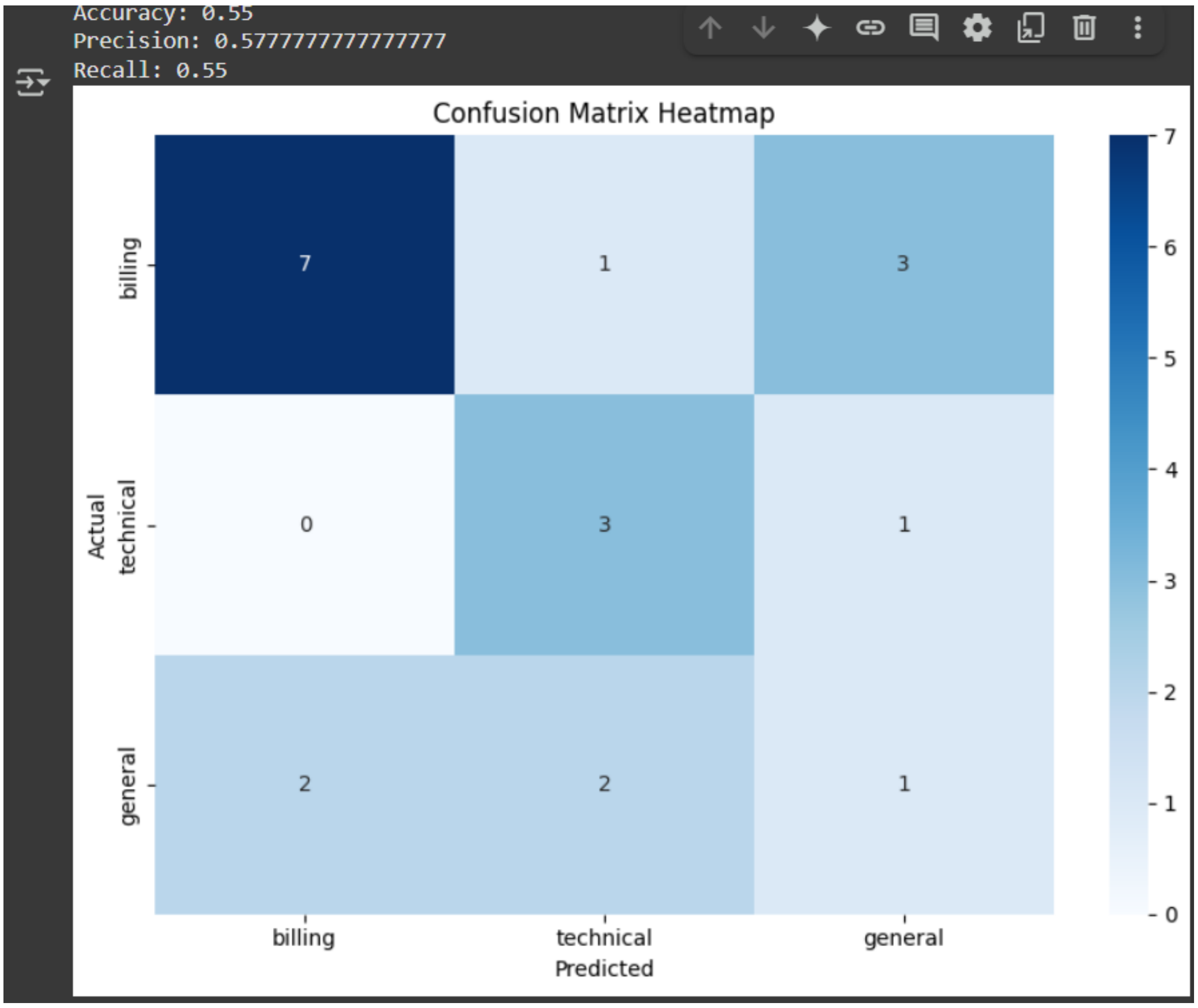
- **Accuracy**
- **Precision**
- **Recall**
- **Confusion Matrix** (visualized using a heatmap)

3. Code

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv("/content/support_cases.csv")  # Replace with your actual path if running locally

# Prepare features and target
X = df[['message_length', 'response_time']]
y = df['case_type']

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a Random Forest Classifier
clf = RandomForestClassifier(random_state=42)
```

```
clf.fit(X_train, y_train)

# Make predictions
y_pred = clf.predict(X_test)

# Compute metrics
conf_matrix = confusion_matrix(y_test, y_pred, labels=['billing', 'technical', 'general'])
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')

# Print metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)

# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
        xticklabels=['billing', 'technical', 'general'],
        yticklabels=['billing', 'technical', 'general'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap')
plt.tight_layout()
plt.show()
```

4.OUTPUT



```
Accuracy: 0.55
Precision: 0.5777777777777777
Recall: 0.55
```

5. References / Credits

- **Scikit-learn Documentation**
  https://scikit-learn.org
  For machine learning models, evaluation metrics, and utilities like TfidfVectorizer, MultinomialNB, and classification_report.

- **Pandas Documentation**
  https://pandas.pydata.org
  Used for handling and processing tabular data.

- **Seaborn & Matplotlib Documentation**
  https://seaborn.pydata.org
  https://matplotlib.org
  Used for generating the confusion matrix heatmap.

- **Google Colab**
  https://colab.research.google.com
  For running the Python code in a cloud-based Jupyter Notebook environment.

- **OpenAI ChatGPT**
  For assistance in writing, explaining code, and preparing this report.