**Customer Support Case Type Classification**

**REPORT**

**NAME : PRASHASTHA SINGH**

**BRANCH : CSE-AI**

**SEC : C**

**ROLL NO : 202401100300176**

****

**KIET GROUP OF INSTITUTION , GHAZIABAD**

**Customer Support Case Type Classification Report**

**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**

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')

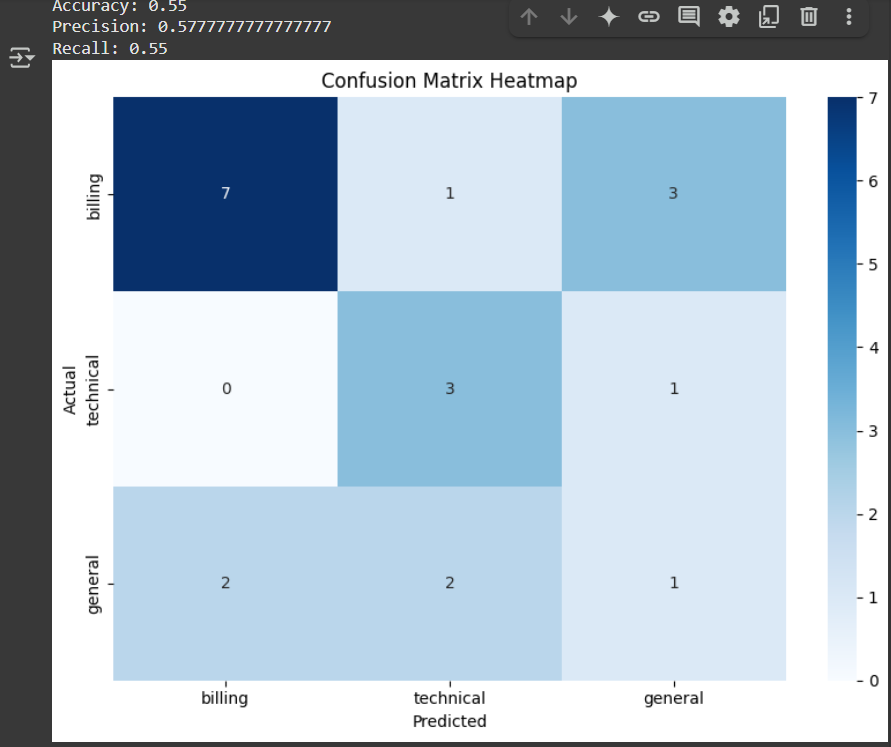
plt.ylabel('Actual')

plt.title('Confusion Matrix Heatmap')

plt.tight\_layout()

plt.show()

**4.OUTPUT**



**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.
* **Project Guide / Instructor (if applicable)**  
  *(Add your professor’s or mentor’s name here if this is part of a class project)*
* **OpenAI ChatGPT**  
  For assistance in writing, explaining code, and preparing this report.