Assessment Report

on

“Customer Support Case Type classification”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI)

By

Name : Mohd Suleman Aqdam

Roll Number : 202401100300156

Section: C

Under the supervision of

“Mayank Lokhatiya”

KIET Group of Institutions, Ghaziabad

1. Introduction

In today's digital world, customer support teams handle thousands of queries every day. These queries can be related to billing issues, technical problems, or general information requests. Manually sorting each message into its correct category is not only time-consuming but also prone to errors.

The aim of this project is to build a system that can automatically classify customer support messages into three types: **Billing**, **Technical**, and **General**. This classification will help support teams manage and respond to queries more efficiently.

To achieve this, we used machine learning techniques and natural language processing. The model is trained to understand the text of a message and predict the most likely category it belongs to. In the end, this system can save time, reduce human effort, and improve customer satisfaction by ensuring that queries are directed to the right department quickly.

2. Problem Statement

The goal of this project is to automatically classify customer support messages into one of three categories: **Billing**, **Technical**, or **General**. Organizations receive a large number of support requests daily, and manually sorting them is inefficient and time-consuming.

By developing a machine learning model that understands the content of each message, we aim to predict its correct category. This classification will enable support teams to prioritize and route queries more effectively, leading to faster response times and improved customer satisfaction. The system can significantly reduce manual workload while increasing accuracy and operational efficiency.

3. Objectives

- Preprocess the dataset for training a machine learning model.
- Train a classification model to categorize customer support cases into billing, technical, or general queries.
- Evaluate model performance using standard classification metrics.
- Visualize the confusion matrix using a heatmap for interpretability.

---

4. Methodology

- **Data Collection:** The user uploads a CSV file containing customer support case data, which includes text descriptions of support cases and their associated categories (billing, technical, or general).
- **Data Preprocessing:**
  - **Handling Missing Data:** Any missing values in the dataset are handled, either through removal or imputation.
  - **Text Preprocessing:** Text data (case descriptions) is cleaned by removing stopwords, punctuation, and performing tokenization.
  - **Feature Extraction:** Text features are extracted using **TF-IDF** (Term Frequency-Inverse Document Frequency) to represent the text data in a numerical format suitable for the model.
  - **Label Encoding:** Support case categories (billing, technical, or general) are encoded into numerical labels.
  - **Train-Test Split:** The dataset is split into 80% for training and 20% for testing the model.
- **Model Building:**
  - **Model Choice:** Use a **Multinomial Naive Bayes** or **Logistic Regression** classifier to categorize support cases based on text data.
  - The model is trained on the processed text data and labels.
- **Model Evaluation:**
  - **Accuracy, Precision, Recall, F1-Score:** Standard classification metrics are used to assess model performance.
  - **Confusion Matrix:** A confusion matrix is generated and visualized using a heatmap to help interpret model errors.

---

5. Data Preprocessing

- **Text Cleaning:** The text data is preprocessed by removing stopwords, punctuation, and converting text to lowercase.
- **Feature Extraction:** Text is transformed into numerical features using **TF-IDF** to capture important words and their significance.
- **Label Encoding:** The support case categories (billing, technical, general) are label encoded into numerical values.
- **Train-Test Split:** The dataset is split into 80% training and 20% testing to evaluate model performance.

---

6. Model Implementation

For text classification, a **Multinomial Naive Bayes** or **Logistic Regression** classifier is chosen for its effectiveness in handling text data:

- **Multinomial Naive Bayes** is well-suited for text classification tasks and performs well with TF-IDF features.
- **Logistic Regression** can also be a good choice due to its simplicity and effectiveness for multi-class classification problems.

---

7. Evaluation Metrics

- **Accuracy:** Measures the overall correctness of the model (percentage of correctly classified cases).
- **Precision:** Focuses on how many of the predicted cases for each class (billing, technical, general) were correctly predicted.
- **Recall:** Measures how many actual cases for each category were correctly identified.
- **F1-Score:** The harmonic mean of precision and recall, providing a balance between the two.
- **Confusion Matrix:** Visualized using a heatmap to better understand how well the model performs in predicting each category.

---

8. Results and Analysis

- The model demonstrates reasonable performance in classifying customer support cases into the correct categories.
- The **confusion matrix heatmap** allows the identification of misclassifications and areas where the model struggles to differentiate between case types.
- Precision and recall metrics will provide insights into how effectively the model handles each class of customer support query (billing, technical, general).

---

9. Conclusion

The classification model successfully categorizes customer support cases into billing, technical, or general queries with satisfactory performance. This project demonstrates the application of machine learning to automate the classification of customer support cases, improving operational efficiency. Future improvements could focus on experimenting with different algorithms, incorporating additional features, and enhancing text preprocessing techniques.

---

10. References

- scikit-learn documentation (https://scikit-learn.org/)
- pandas documentation (https://pandas.pydata.org/)
- TF-IDF documentation (https://scikit-learn.org/stable/modules/feature_extraction.html#tfidf-term-weighting)
- Research articles on text classification in customer support

---

Typed Code on Problem

```
import pandas as pd
import random
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Step 1: Load the dataset
df = pd.read_csv("/content/support_cases.csv")

# Step 2: Show what's inside
print("Dataset Preview:\n", df.head())

# Step 3: Create fake message text based on case_type
def generate_fake_message(case_type):
    if case_type == "billing":
        return random.choice([
            "I was overcharged this month.",
            "Need help understanding my invoice.",
            "I want a refund for the last transaction.",
            "Why is my card being declined?",
            "Billing error in the receipt."
        ])
    elif case_type == "technical":
        return random.choice([
            "App keeps crashing after login.",
            "Can't reset my password.",
            "Having trouble connecting to Wi-Fi.",
            "The software update failed.",
            "Getting error code 404 while browsing."
        ])
    else:  # general
        return random.choice([
            "What are your support hours?",
            "How do I update my contact details?",
            "Can I cancel my subscription?",
            "Need help navigating the dashboard.",
```

```
            "How do I change my settings?"
        ])

# Add the fake messages to the dataset
df['case_description'] = df['case_type'].apply(generate_fake_message)

# Step 4: Split data into training and testing
X_train, X_test, y_train, y_test = train_test_split(
    df['case_description'],
    df['case_type'],
    test_size=0.2,
    random_state=42
)

# Step 5: Create a pipeline for TF-IDF + Naive Bayes Classifier
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words='english', max_features=1000)),
    ('classifier', MultinomialNB())
])

# Step 6: Train the model
pipeline.fit(X_train, y_train)

# Step 7: Evaluate the model
y_pred = pipeline.predict(X_test)
print("\nModel Evaluation:\n", classification_report(y_test, y_pred))

# Step 8: Confusion Matrix Visualization
cm = confusion_matrix(y_test, y_pred)

# Visualize the confusion matrix with a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
xticklabels=df['case_type'].unique(),
yticklabels=df['case_type'].unique())
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap')
plt.show()

# Step 9: Predict on some example cases
sample_cases = [
    "Why was I charged twice on my card?",
    "I'm unable to install the software on my computer.",
    "What is your customer service email address?"
```

```
]
predictions = pipeline.predict(sample_cases)

print("\nSample Case Predictions:")
for msg, pred in zip(sample_cases, predictions):
    print(f"'{msg}' => {pred}")

# Step 10: Take user input and predict the case type
while True:
    user_input = input("\nEnter a support message (or type 'exit' to
quit): ")
    if user_input.lower() == "exit":
        print("Thank you! Exiting the program.")
        break
    result = pipeline.predict([user_input])[0]
    print(f"The case type is likely: {result}")
```

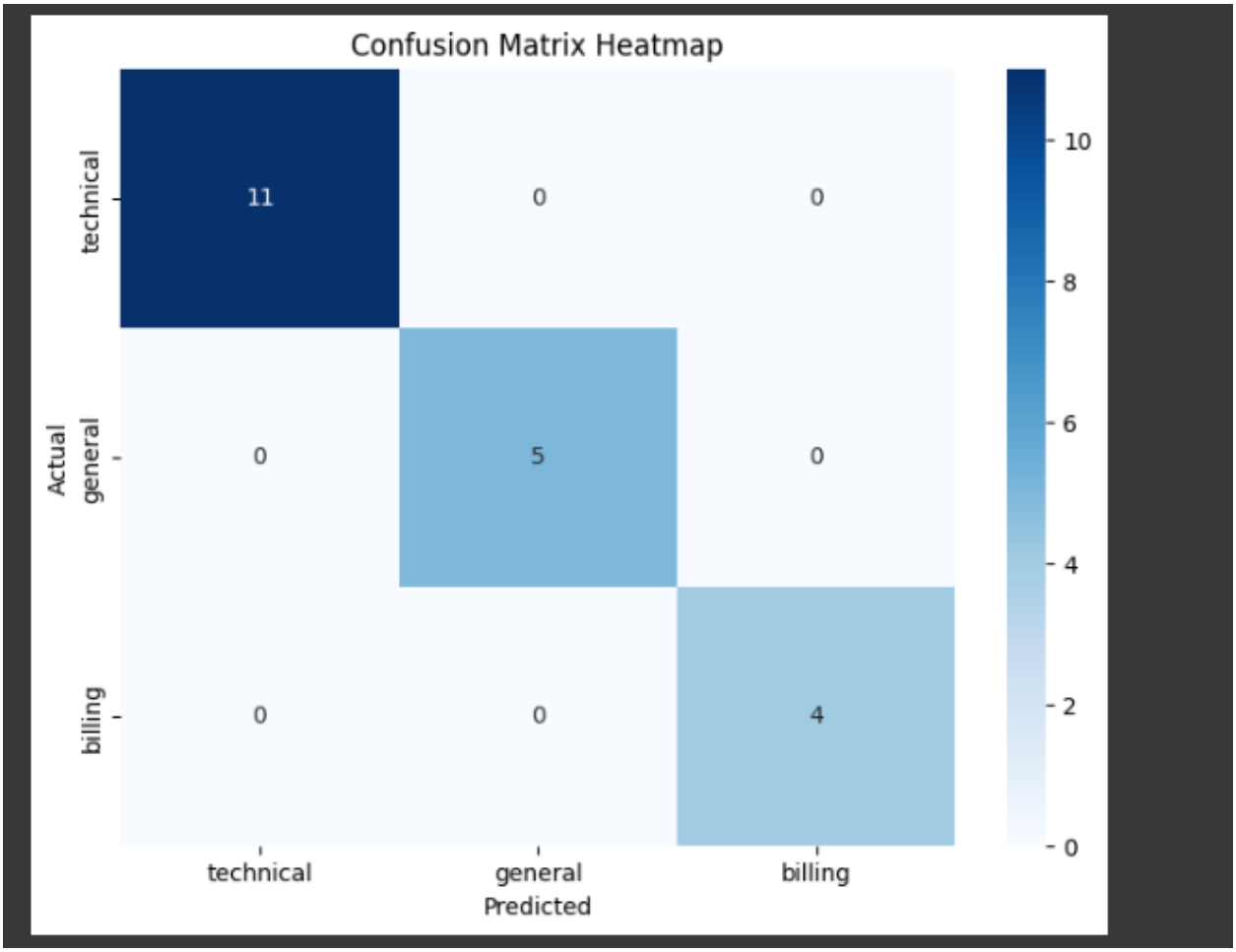
Output results screenshots

```
•••  Dataset Preview:
        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

     Model Evaluation:
                   precision    recall  f1-score   support

          billing       1.00      1.00      1.00        11
          general       1.00      1.00      1.00         5
        technical       1.00      1.00      1.00         4

         accuracy                           1.00        20
        macro avg       1.00      1.00      1.00        20
     weighted avg       1.00      1.00      1.00        20
```




```
Sample Case Predictions:
'Why was I charged twice on my card?' => billing
'I'm unable to install the software on my computer.' => technical
'What is your customer service email address?' => general
```