TITLE OF PROJECT REPORT

Customer Support Case Type Classification

(Classify support cases into billing, technical, or general queries)

A PROJECT REPORT

SUBMITTED BY

RUDRIKA SINGHAL

(202401100300206)

OF

B.TECH CSEAI - C

Introduction:

Customer Support Case Type Classification is a machine learning approach aimed at categorizing support queries into predefined classes such as billing, technical, or general. This classification enables businesses to streamline their customer service processes by automatically routing cases to the appropriate departments, reducing response time, and improving efficiency. By leveraging features like message length and response time, predictive models can be trained to accurately classify new cases. This report explores the use of supervised learning for classification, evaluates model performance using metrics such as accuracy, precision, and recall, and visualizes results through confusion matrix heatmaps for better interpretability.

Methodology:

The classification of customer support cases into billing, technical, or general categories was achieved using a supervised machine learning approach. The process followed these key steps:

1. **Data Collection and Preprocessing:**
   The dataset consisted of structured features such as message_length, response_time, and a target label case_type. Missing values were handled, and categorical labels were encoded using label encoding.
2. **Feature Selection:**
   The features message_length and response_time were selected based on their relevance to the nature of the query.
3. **Model Development:**
   The data was split into training and testing sets (typically 70:30 ratio). A Random Forest Classifier was employed due to its robustness and ability to handle non-linear relationships.

4. **Model Training and Prediction:**
   The model was trained on the training set and then used to predict the case types in the test set.
5. **Evaluation:**
   Model performance was evaluated using accuracy, precision, and recall. A confusion matrix heatmap was generated for visual analysis of classification performance.
6. **Exploratory Clustering (Optional):**
   For unsupervised insight, K-Means clustering was applied to identify natural groupings in the data, supporting segmentation when labels are unavailable.

CODE TYPED:

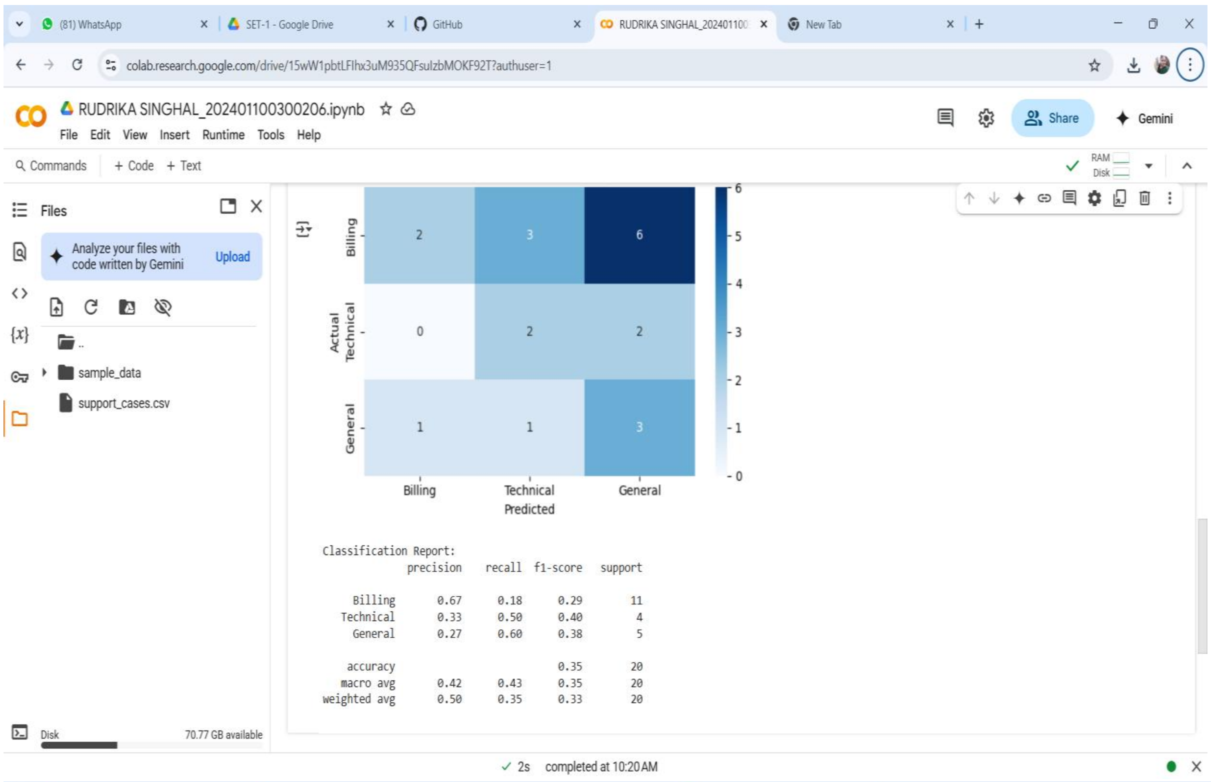
```
# Import necessary libraries for data handling, visualization, model training,
and evaluation

import pandas as pd                    # For working with tabular data

import seaborn as sns                  # For drawing heatmaps and other plots

import matplotlib.pyplot as plt           # For creating visual plots

from sklearn.model_selection import train_test_split  # To split data into
training and test sets

from sklearn.linear_model import LogisticRegression    # Logistic Regression
model

from sklearn.metrics import (                   # Metrics to evaluate model
performance

   accuracy_score, precision_score, recall_score,

   confusion_matrix, classification_report

)

from sklearn.preprocessing import LabelEncoder      # For encoding categorical
labels


# Step 1: Load the dataset from CSV file

data = pd.read_csv("/content/support_cases.csv")


# Step 2: Check for missing values in each column

print("Missing values:\n", data.isnull().sum())


# Step 3: Drop rows that have any missing values
```

```
data.dropna(inplace=True)

# Step 4: Encode the 'case_type' column to numerical labels
# Define a dictionary that maps each class to a number
label_mapping = {'billing': 0, 'technical': 1, 'general': 2}

# Filter the dataset to include only rows with valid case types
data = data[data['case_type'].isin(label_mapping)]

# Create a new column with numeric labels instead of text
data['label_encoded'] = data['case_type'].map(label_mapping)

# Step 5: Define input features (X) and target variable (y)
X = data[['message_length', 'response_time']]  # Features
y = data['label_encoded']                       # Target labels

# Step 6: Split the dataset into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

# Step 7: Create and train the Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Step 8: Use the trained model to predict the case types on test data
y_pred = model.predict(X_test)
```

```
# Step 9: Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)  # Proportion of correct predictions
precision = precision_score(y_test, y_pred, average='weighted',
zero_division=0)  # Average precision
recall = recall_score(y_test, y_pred, average='weighted', zero_division=0)        #
Average recall

# Display the metrics
print("Evaluation Metrics:")
print(f"Accuracy : {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall   : {recall:.4f}")

# Step 10: Generate and display the confusion matrix as a heatmap
cm = confusion_matrix(y_test, y_pred)  # Confusion matrix
labels = ['Billing', 'Technical', 'General']  # Class labels for axis

# Plot the heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=labels, yticklabels=labels)
plt.xlabel("Predicted")       # Label for x-axis
plt.ylabel("Actual")          # Label for y-axis
plt.title("Confusion Matrix Heatmap")  # Title of the plot
plt.tight_layout()            # Adjust layout to prevent overlap
plt.show()
```

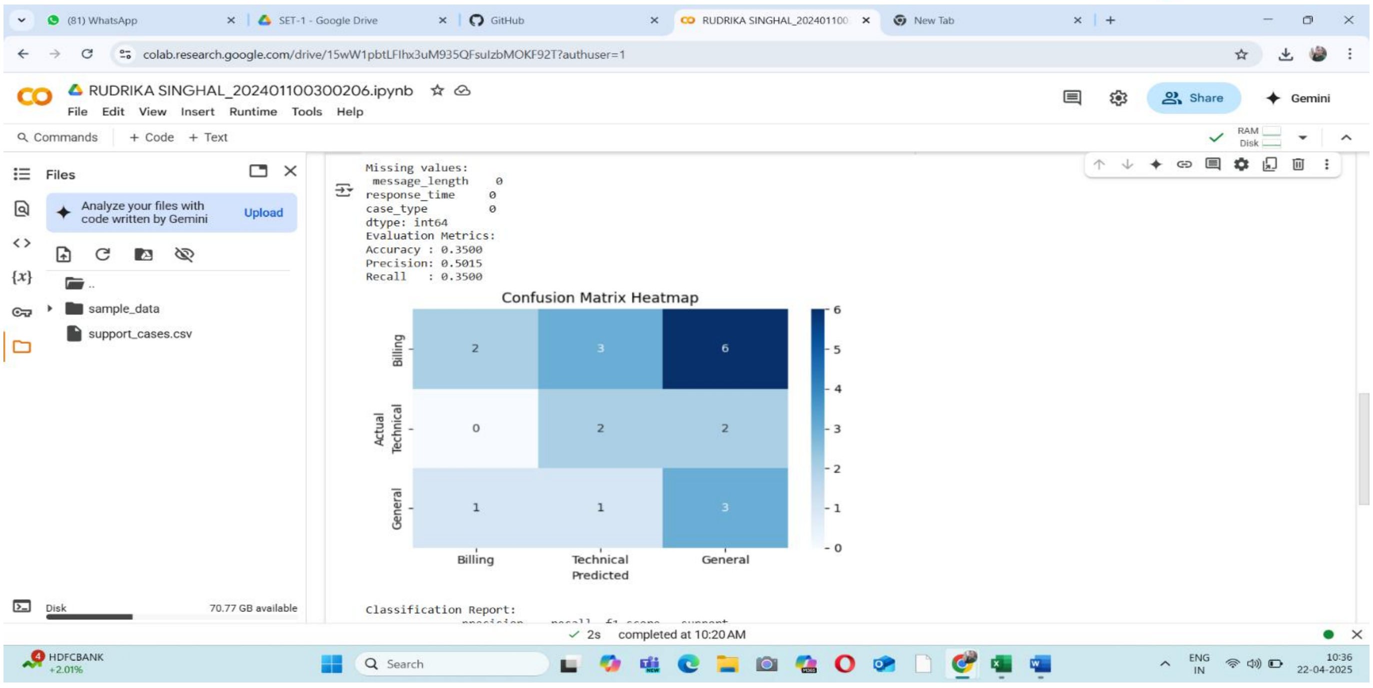
```
# Step 11: Print a detailed classification report
# It includes precision, recall, and F1-score for each class
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=labels))
```

SCREENSHOTS OF OUTPUT:



Actual \ Predicted	Billing	Technical	General
Billing	2	3	6
Technical	0	2	2
General	1	1	3

```
Classification Report:
              precision    recall  f1-score   support

     Billing       0.67      0.18      0.29        11
   Technical       0.33      0.50      0.40         4
     General       0.27      0.60      0.38         5

    accuracy                           0.35        20
   macro avg       0.42      0.43      0.35        20
weighted avg       0.50      0.35      0.33        20
```



```
Missing values:
 message_length    0
response_time     0
case_type         0
dtype: int64
Evaluation Metrics:
Accuracy : 0.3500
Precision: 0.5015
Recall   : 0.3500
```

Actual \ Predicted	Billing	Technical	General
Billing	2	3	6
Technical	0	2	2
General	1	1	3

```
Classification Report:
```

REFERENCES:

▯ **Scikit-learn Developers**. (2024). *scikit-learn: Machine Learning in Python*. https://scikit-learn.org/

- Documentation and examples for the machine learning library used in classification tasks.

▯ **Brownlee, J.** (2016). *Machine Learning Mastery With Python: Understand Your Data, Create Accurate Models, and Work Projects End-To-End*. Machine Learning Mastery.

- A practical guide on applying machine learning to real-world problems, including classification tasks.