Customer Support Case Type Classification

PROJECT REPORT

SUBMITTED BY:

RASHIKA

202401100300196

In partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY IN CSE AI

KIET GROUP OF INSTITUTIONS

April 2025

1. Introduction

In the era of digital transformation, customer support systems play a pivotal role in ensuring customer satisfaction and efficient service delivery. Companies receive a wide range of customer queries daily, and manually categorizing each support case can be time-consuming and prone to errors. To automate and streamline this process, we propose a machine learning model that classifies customer support cases into three categories: **Billing**, **Technical**, or **General**.

This classification not only improves response time but also ensures that the right team handles each query, thereby enhancing operational efficiency and customer satisfaction.

2. Methodology

The methodology adopted for this project includes the following steps:

1. **Data Collection**
   We used a dataset containing customer support messages. Each message was labeled as either *Billing*, *Technical*, or *General*.

2. **Text Preprocessing**
   The text was cleaned by removing unnecessary characters, converting it to lowercase, and removing common words (like "the", "is", "and").

3. **Feature Extraction**
   We used **TF-IDF (Term Frequency-Inverse Document Frequency)** to convert the text data into numbers that a machine learning model can understand.

4. **Model Training**
   We chose a **Naive Bayes classifier** because it works well for text classification. The model learned patterns from the training data.

5. **Testing and Evaluation**
   We tested the model on new data and measured how accurately it could classify the messages. We used metrics like accuracy and F1-score.

3. Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
import joblib

# === STEP 1: Load CSV ===
df = pd.read_csv('/content/support_cases.csv')  # Replace with your CSV path

# Print available columns
print("Available columns:", df.columns.tolist())

# Try to detect the correct column names
text_col = None
label_col = None

# Guess text and label columns (you can hardcode them too if you know)
for col in df.columns:
    if 'text' in col.lower() or 'message' in col.lower() or 'query' in col.lower():
        text_col = col
    if 'label' in col.lower() or 'category' in col.lower() or 'type' in col.lower():
        label_col = col

# Raise errors if not found
if not text_col or not label_col:
    raise ValueError("Could not detect 'text' or 'category' columns. Please check your CSV column
names.")
```

```
# === STEP 2: Preprocess ===
df = df[[text_col, label_col]].dropna()
df[text_col] = df[text_col].astype(str).str.lower()

# === STEP 3: Split dataset ===
X_train, X_test, y_train, y_test = train_test_split(df[text_col], df[label_col], test_size=0.2,
random_state=42)

# === STEP 4: Build Pipeline ===
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words='english')),
    ('clf', LogisticRegression(max_iter=1000))
])

# === STEP 5: Train Model ===
pipeline.fit(X_train, y_train)

# === STEP 6: Evaluate Model ===
y_pred = pipeline.predict(X_test)
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# === STEP 7: Save Model ===
joblib.dump(pipeline, 'support_case_classifier.pkl')

# === STEP 8: Predict New Cases ===
new_cases = [
    "I need help with my subscription payment.",
    "The software crashes when I try to export a file.",
    "Can you tell me your working hours?"
```

```
]

preds = pipeline.predict(new_cases)
print("\nPredictions:")
for case, pred in zip(new_cases, preds):
    print(f" - \"{case}\" → {pred}")
```

OUTPUT:

```
Available columns: ['message_length', 'response_time', 'case_type']

Accuracy: 0.3

Classification Report:
              precision    recall  f1-score   support

     billing       0.50      0.09      0.15        11
     general       0.29      1.00      0.45         5
   technical       0.00      0.00      0.00         4

    accuracy                           0.30        20
   macro avg       0.26      0.36      0.20        20
weighted avg       0.35      0.30      0.20        20


Predictions:
 - "I need help with my subscription payment." → general
 - "The software crashes when I try to export a file." → general
 - "Can you tell me your working hours?" → general
```

REFERENCES & CREDITS:

1. **Pandas**:
    - **Purpose**: Data manipulation and analysis, especially for loading, cleaning, and preprocessing CSV data.
    - **Documentation**: Pandas Documentation
    - **Source**: https://pandas.pydata.org
2. **Scikit-learn**:
    - **Purpose**: Provides tools for machine learning, including classifiers like LogisticRegression, TfidfVectorizer, and functions like train_test_split for splitting data.
    - **Documentation**: Scikit-learn Documentation
    - **Source**: https://scikit-learn.org
3. **Joblib**:
    - **Purpose**: Used to save the trained machine learning model (joblib.dump), which makes it easy to load and use the model in the future without retraining.
    - **Documentation**: Joblib Documentation
    - **Source**: https://joblib.readthedocs.io
4. **Logistic Regression Classifier**:
    - **Purpose**: The classifier used to train the model and predict case types. Logistic Regression is a widely used algorithm for classification problems.
    - **Documentation**: Logistic Regression in Scikit-learn

    - **Reference**: This model was chosen for its simplicity and effectiveness in text classification.

5. **TF-IDF Vectorization**:
    - **Purpose**: Converts the text data into numerical vectors that can be processed by machine learning algorithms. This method assigns weights to words based on their frequency in a document relative to their frequency across all documents.
    - **Reference**: This is a standard method in text mining and information retrieval.
    - **Documentation**: TF-IDF in Scikit-learn

General Approach:

- The **machine learning pipeline** used (combining vectorization and classification in one workflow) is a standard best practice for text classification tasks, especially with scikit-learn.
- **Train-test split** and **model evaluation** techniques are basic and standard in machine learning to avoid overfitting and to assess model performance.

Acknowledgements:

- This approach and code are based on standard practices commonly shared across tutorials, books, and online resources for text classification.