Implementation Documentation

Project: Intelligent Community Complaint Classification System

Author: Muhammad Affan Kabir (SAP ID: 65139)

1. Overall Working of the Program

The program is an automated classification system designed to route citizen complaints to the correct civic department (Cleaning, Water, Electricity, Inflation) while filtering out irrelevant text.

The system operates in **four sequential stages**:

1. **Data Generation:** It first generates a robust synthetic dataset of 500 samples in-memory. This dataset uses a combinatorial algorithm to mix Subjects, Actions, and shared Contexts to prevent overfitting.
2. **Training:** The text data is converted into numerical features using **TF-IDF (Term Frequency-Inverse Document Frequency)**. These features are used to train a **Multinomial Naive Bayes** classifier.
3. **Evaluation:** The model is tested on a 20% hold-out set, displaying the accuracy score and a confusion matrix to visualize performance.
4. **Hybrid Prediction (Live Demo):** The system enters an interactive loop where users can input text. The prediction logic follows a strict hierarchy:
    a. **Level 1 (Rules):** Checks for definitive keywords (e.g., "voltage"). If found, assigns category with 100% confidence.
    b. **Level 2 (AI):** If no keywords are found, the AI model predicts the category.
    c. **Level 3 (Filter):** If the prediction is "Irrelevant" or the confidence is low (< 45%), the output is labeled as "Others".

2. Functions and Modules Used

A. `generate_dataset()`

- **Purpose:** Creates the training data dynamically.

- **Logic:** Contains five lists of category-specific subjects (e.g., "Transformer" for Electricity) and one list of shared contexts (e.g., "since yesterday"). It combines these randomly to create 500 unique labeled sentences.
- **Returns:** A Pandas DataFrame with columns `Category` and `Complaint_Text`.

B. `predict_complaint(text)`

- **Purpose:** The core intelligence engine for the live demo.
- **Logic:**
    - **Keyword Search:** Scans the input string for hardcoded triggers (e.g., "garbage", "pipe"). Returns immediately if found.
    - **Probabilistic Prediction:** Calls `model.predict_proba()` to get confidence scores for all classes.
    - **Thresholding:** Checks if the max confidence is below `0.45` or if the predicted class is `Irrelevant`. If so, returns "Others".

C. `make_pipeline()`

- **Purpose:** Bundles the vectorizer and classifier into a single object.
- **Components:** `TfidfVectorizer` (Text Processor) + `MultinomialNB` (Classifier).

3. Input and Output Formats

Training Phase

- **Input:** A DataFrame of 500 text strings labeled with categories.
- **Output:** A trained Machine Learning model object.

Live Prediction Phase

- **User Input:** A raw string entered via the console (e.g., *"The voltage is fluctuating"*).
- **System Output:**
    - **Category:** The predicted department (e.g., *Electricity* or *Others*).
    - **Confidence:** A percentage score (e.g., *98.5%*) or a tag indicating the method used (e.g., *Guaranteed by Keyword*).

4. Libraries and Frameworks Required

To run this project, the following Python libraries must be installed:

- **`pandas`**: For data manipulation and DataFrame creation.
- **`scikit-learn` (sklearn)**:
    - `train_test_split`: For splitting data.
    - `TfidfVectorizer`: For NLP feature extraction.
    - `MultinomialNB`: The Machine Learning algorithm.
    - `metrics`: For calculating accuracy and generating the confusion matrix.
- **`matplotlib` & `seaborn`**: For plotting the confusion matrix heatmap.
- **`random`**: For the combinatorial data generation.

5. Execution Instructions

The code is designed to be **self-contained**. No external CSV files are required.

1. **Environment Setup:** Ensure Python 3.x is installed along with the libraries mentioned above.

```
Bash
pip install pandas scikit-learn matplotlib seaborn
```

2. **Run the Script:** Open the `.ipynb` file in Jupyter Notebook or Google Colab and execute the cells in order.
    a. **Step 1 & 2** will generate data and train the model automatically.
    b. **Step 3** will display the accuracy report.
3. **Using the Demo:**
    a. Scroll to **Step 4** (The Live Prediction Cell).
    b. When the prompt `Enter Complaint:` appears, type a sentence and press Enter.
    c. **Example 1:** Type *"My bill is too high"* ---> Result: *Inflation*.
    d. **Example 2:** Type *"I am playing cricket"* ---> Result: *Others*.
    e. Type **`exit`** or **`quit`** to stop the program.