Project Report: Intelligent Community Complaint Classification System

Author: Muhammad Affan Kabir

SAP ID: 65139

Course: Artificial Intelligence

1. Problem Statement

In modern civic management, authorities receive thousands of citizen complaints daily regarding utilities, sanitation, and economic issues. Manually sorting these unstructured text messages is labor-intensive, slow, and prone to human error. Furthermore, existing automated systems often fail to distinguish between genuine complaints and irrelevant noise (e.g., casual conversation), or they miss obvious high-priority keywords due to probabilistic uncertainty.

There is a critical need for an intelligent, automated system capable of classifying complaints with high precision while robustly filtering out irrelevant data.

2. Project Objectives

The primary objectives of this project are:

- **Automation:** To develop an AI pipeline that automatically categorizes text inputs into specific departments (Cleaning, Water, Electricity, Inflation).
- **Noise Handling:** To implement a mechanism for identifying and discarding "Irrelevant" inputs (e.g., sports, personal updates) to prevent false positives.
- **Accuracy Guarantee:** To integrate a hybrid logic system that ensures 100% classification accuracy for critical domain-specific keywords (e.g., "voltage", "sewage") regardless of context.
- **Real-time Classification:** To provide a live interface for immediate prediction and routing.

3. Dataset Description

The project utilizes a custom-generated synthetic dataset designed to simulate real-world linguistic variety while avoiding common overfitting pitfalls.

- **Source:** Generation of Synthetic dataset using a combinatorial algorithm.
- **Size: 500 Data Samples**
- **Features:**
  - **Input:** `Complaint_Text` (Natural language sentences).
  - **Output:** `Category` (Target classification).
- **Class Distribution:** The dataset is balanced across five distinct categories:
  - **Cleaning:** (e.g., "Garbage is piling up...")
  - **Water:** (e.g., "The main pipe is leaking...")
  - **Electricity:** (e.g., "The transformer sparked...")
  - **Inflation:** (e.g., "Prices of flour have doubled...")
  - **Irrelevant:** (e.g., "I am watching cricket...")
- **Context Strategy:** To prevent the model from overfitting on time-based words (e.g., assuming "yesterday" always means "Water"), a shared bank of *Common Contexts* was distributed across all categories during generation.

4. Proposed Methodology

The solution employs a **Hybrid Classification Architecture** combining deterministic rules with probabilistic Machine Learning.

A. Feature Extraction

Unstructured text is converted into numerical vectors using **TF-IDF (Term Frequency-Inverse Document Frequency)**. This technique assigns weight to words based on their importance, filtering out common stop words while highlighting domain-specific terms like "voltage" or "leak."

B. Machine Learning Model

A **Multinomial Naive Bayes (MNB)** classifier is trained on the TF-IDF vectors. MNB is chosen for its efficiency and high performance in text classification tasks where features (word counts/frequencies) follow a multinomial distribution.

How it works:

Here is the short, simple flow of how **TF-IDF** and **Naive Bayes** work together in your specific project to classify a sentence like *"The voltage is low"*.

The Flow: From Text to Prediction

1. Input

You type: **"The voltage is low."**

2. TF-IDF (The Translator & Highlighter)

The computer cannot read English; it only understands numbers. TF-IDF converts your sentence into a list of numbers (a vector).

- **TF (Frequency):** It counts the words.
- **IDF (Rarity):** It gives a **score** to each word based on importance.
    - *"The", "is"* gives **Low Score** (Common words, ignored).
    - *"Voltage"* gives **High Score** (Rare/Unique word, highlighted).
- **Result:** Your sentence becomes a math vector like `[0.01, 0.95, 0.02, 0.5]`.

You are right. My previous explanation glossed over the specific mechanics of Naive Bayes too quickly.

Here is the corrected, step-by-step flow for how **Naive Bayes** actually thinks when it receives the numbers from TF-IDF.

3. Naive Bayes (The Probability Detective)

Once TF-IDF has highlighted the important words (like "Voltage"), Naive Bayes takes over to calculate the winner.

1. The "Naive" Setup (Treating Words as Independent Clues)

Naive Bayes assumes every word is a separate, independent piece of evidence. It doesn't care about grammar or word order. It just asks: *"Does this specific word appear?"*

2. Consulting the "Memory" (Likelihood)

During the **Training Phase** (Step 2 in your code), the model built a mental "probability table." It knows, for example:

- "Voltage" appears in **Electricity** complaints 95% of the time.
- "Voltage" appears in **Water** complaints 0.1% of the time.

3. The Calculation (Multiplying the Odds)

When you input *"The voltage is low"*, the model calculates a score for **every** category simultaneously.

- **Checking "Electricity":**
  - Start with baseline probability (25%).
  - See word "Voltage"? $\rightarrow$ *Multiply by huge boost (0.95)*.
  - See word "Low"? $\rightarrow$ *Multiply by small boost (0.40)*.
  - **Result:** A very high score.
- **Checking "Water":**
  - Start with baseline probability (25%).
  - See word "Voltage"? $\rightarrow$ *Multiply by tiny penalty (0.001)*.
  - See word "Low"? $\rightarrow$ *Multiply by small boost (0.40)*.
  - **Result:** The score crashes to near zero because "Voltage" effectively killed the possibility.

4. The Verdict

The model compares the final scores for all 4 categories.

- **Electricity:** 0.85
- **Water:** 0.02
- **Cleaning:** 0.01
- **Inflation:** 0.01

Final Decision: The system picks **Electricity**.

Summary of the Combined Flow

1. **TF-IDF** turns text into **weighted numbers** (giving high value to "Voltage").
2. **Naive Bayes** uses those numbers to **lookup probabilities** it learned during training.

3. **Multiplication** of those probabilities determines the category with the highest likelihood.

4. Output

The system returns: **Electricity**.

C. Hybrid Prediction Logic (The "Smart" Layer)

To maximize reliability, the prediction phase follows a strict hierarchical logic:

1. **Layer 1: Rule-Based Override:** The input is first checked against a dictionary of critical keywords (e.g., *electricity, voltage, sewage, pipe*). If a match is found, the system bypasses the AI model and assigns the category with **100% confidence**.
2. **Layer 2: AI Classification:** If no keywords are found, the MNB model predicts the most likely category.
3. **Layer 3: Threshold Filtering:**
    a. If the predicted category is "Irrelevant", the system outputs "Others".
    b. If the model's confidence score is below **45% (0.45)**, the prediction is deemed unsafe, and the system outputs "Others".

5. Expected Results

The implementation is expected to deliver the following outcomes:

- **High Accuracy:** The Multinomial Naive Bayes model is expected to achieve an accuracy of **>90%** on the synthetic test set due to the clear vocabulary distinction between categories.
- **Zero-Error on Critical Terms:** Thanks to the rule-based layer, inputs containing definitive triggers like "blackout" or "leakage" or “Electricity” will never be misclassified, regardless of the surrounding sentence structure.
- **Effective Noise Filtering:** The system will correctly identify non-complaint inputs (e.g., "He is eating pizza") as "Others" rather than forcing them into a civic category.