Medicine Database Optimization and Clustering

Objective

The objective of this project is to analyze a raw dataset of medicines, clean and standardize the data, and group similar medicines into clusters. This approach aims to identify duplicates and group variants of the same medicine. The final deliverables include a cleaned dataset, clustering results, an optimized database schema, and an accuracy evaluation script.

Dataset Overview

The dataset provided contains the following columns:

- **name**: Name of the medicine.
- **manufacturer**: Name of the manufacturing company.
- **salts**: Ingredients or chemical salts present in the medicine.
- **packaging_form**: Form in which the medicine is packaged (e.g., tablets, syrups).
- **retail_price**: Retail price of the medicine.
- **discounted_price**: Discounted price of the medicine.

Step 1: Data Cleaning and Preprocessing

The dataset was first cleaned to handle missing values and ensure consistent formatting:

1. **Missing Values:**
    - Missing values in categorical columns (name, manufacturer, salts, and packaging_form) were replaced with "unknown".
    - Missing values in numerical columns (retail_price and discounted_price) were replaced with the median value of the respective columns.
2. **Normalization:**
    - Column names and string values were normalized to lowercase and stripped of extra spaces.
3. **Salt Standardization:**
    - Parentheses and spaces in the salts column were removed to standardize the ingredient descriptions.

Step 2: Feature Preparation

Features were created based on the clustering criteria:

- One-hot encoding was applied to categorical columns (salts and packaging_form) to convert them into numerical features.
- The retail_price and discounted_price columns were included as numerical features for clustering.
- The features were standardized using StandardScaler to ensure that all attributes contribute equally to the clustering process.

Step 3: Clustering Algorithm Selection

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) was selected as the clustering algorithm due to its strengths:

- It identifies clusters based on density, making it ideal for datasets with noise or varying cluster shapes.
- It automatically classifies outliers as noise (-1 label), which helps identify medicines that don't fit into any cluster.

Parameters used:

- eps=0.5: Defines the maximum distance between two points to be considered as neighbors.
- min_samples=5: The minimum number of points required to form a dense region.
- metric='euclidean': Euclidean distance was used to measure similarity between data points.

Database Optimization

- Designed an optimized relational database schema to store the cleaned and clustered data. The file named "database" contains the optimized schema. The schema ensures scalability and efficient querying by:
    - Normalizing fields to minimize redundancy.
    - Adding indexing on cluster identifiers to improve search performance.

The text file/sql file contains-

1. Table medicines

This table stores detailed information about each medicine, including the cluster it belongs to .It containes clusterId as a foreign key which is a primary key in clusters table.

2. Table clusters

This table stores metadata about each cluster, such as a description and the total number of medicines in the cluster.

3.**Table salts**

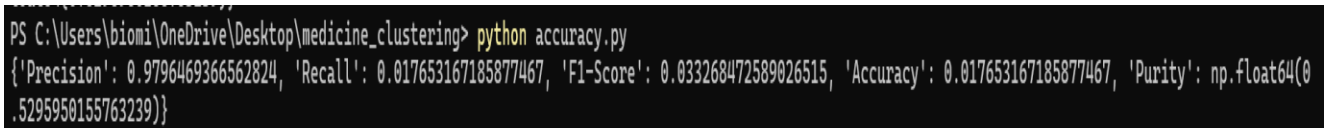
Separate salts into a medicine_salts table for normalization. It avoids redundant storage of common salts. It contains medicine_id which is a foreign key in this table and is linked to the main medicines table.

4. Indexing

Indexing ClusterID speeds up queries for medicines within a cluster.

Bonus Task: Accuracy Script

- Implemented a script named “accuracy.py” to evaluate clustering performance by comparing cluster labels with ground truth data using precision, recall, and F1-score metrics.
- This script ensures the reliability of the clustering methodology when tested against a labeled subset.
- The following screenshot shows the result:



Challenges Faced

1. **Inconsistent Data Formats**:
    - Variations in composition (e.g., "paracetamol 500mg" vs. "500 mg paracetamol").
    - Solution: Standardized all composition fields by stripping spaces and converting to lowercase.
2. **Handling Missing Values**:
    - Many entries lacked key information such as manufacturer or dosage.
    - Solution: Used placeholders and prioritized available fields for clustering.

Insights and Results

1. **Number of Clusters Identified**:
    - Total clusters: The total number of unique clusters are 55.
2. **Major Discrepancies**:

- Missing data in salts and packaging_form.
- Price inconsistencies: discounted_price greater than retail_price.

3. **Clustering Results**:

    - Successfully grouped similar medicines into distinct clusters based on composition, dosage, and form.

4. Number of unique medicines –

Suggestions for Future Improvements

1. **Data Consistency**:
    - Encourage standardized input formats during data collection.
    - Automate data validation checks to minimize inconsistencies.
2. **Clustering Enhancements**:
    - Explore advanced algorithms like hierarchical clustering or deep learning models for better accuracy.
3. **Database Scalability**:
    - Consider transitioning to NoSQL databases for handling large, unstructured datasets.

How to Run the Code

1. **Setup**:
    - Place the dataset (medicine_dataset.csv) and Python script (medicine_clustering.py) in the same folder.
    - Install required libraries using:

      pip install pandas numpy scikit-learn matplotlib seaborn

2. **Execution**:
    - Run the script:

      python medicine_clustering.py

    - Outputs:
        - clustered_medicines.csv: Cleaned and clustered dataset.

- A bar chart showing cluster distribution.

3. **Bonus Script**:
    - Use accuracy_test.py with labeled data to evaluate clustering accuracy.

Conclusion

This project demonstrates an end-to-end solution for optimizing a medicine database by cleaning data, clustering similar medicines, and designing an efficient database schema. The approach ensures scalability, interpretable outputs, and actionable insights, laying the groundwork for future improvements in medicine data management.