Samuel Artiste

Cristian Aldana

CAI 4002 - Artificial Intelligence

Project Report: Interactive Supermarket Simulation with Association Rule Mining

1. Introduction and System Design

1.1 Project Overview

This project is an interactive application that simulates a supermarket transaction system to perform Association Rule Mining on customer data. It integrates a data preprocessing pipeline with custom implementations of the Apriori and Eclat algorithms to discover and compare purchasing patterns. The tool features a user-friendly dashboard for generation transacations, visualizing algorithmic performs in the console, and querying product recommendations based on association strength.

1.2 System Architecture

The application is built using **Python** and follows a modular design pattern, separating the user interface, data processing, and algorithmic logic.

- **User Interface (UI) Layer:** Built with tkinter, this layer handles user interactions, including manual transaction creation, CSV file imports, and result visualization.
- **Data Processing Layer:** Managed by pandas, this module handles data cleaning, standardization, and transformation into formats suitable for mining (vertical TID-sets).
- **Algorithm Layer:** Custom implementations of the **Apriori** (Breadth-First Search) and **Eclat** (Depth-First Search) algorithms to extract frequent itemsets and association rules.

2. Data Preprocessing Approach

This application implements a preprocessing pipeline to handle raw transaction data from CSV files.

2.1 Cleaning and Standardization

Upon importing a dataset, the system performs the following operations:

- **Missing Value Imputation:** NaN values are converted to empty strings to prevent runtime errors during string manipulation.
- **Normalization:** All product names are stripped of leading/trailing whitespace and converted to **lowercase** to resolve inconsistencies (e.g., treating "Milk", "milk ", and "MILK" as the same item).

2.2 Validation and Filtering

- **Inventory Validation:** Items are validated against a master products.csv inventory file. Items not found in the inventory are flagged as invalid and removed.
- **Noise Reduction:**
    - **Empty Transactions:** Transactions containing no valid items are discarded.
    - **Single-Item Transactions:** Transactions with only one valid item are removed, as they cannot form associations (pairs/tuples) and inflate the dataset size without adding value to the rule generation process.

3. Algorithm Implementation Details

3.1 Apriori Algorithm

The Apriori implementation uses a **breadth-first, level-wise search**. To improve performance, a hybrid approach was used where candidate generation follows the standard Apriori logic, but support counting uses **TID-set intersections** (borrowed from vertical mining) instead of scanning the entire database.

Pseudocode:

Input: Transactions D, MinSupport s, MinConfidence c

L1 = {frequent 1-itemsets}

k = 2

While L(k-1) is not empty:

```
    Ck = Generate_Candidates(L(k-1))  // Join & Prune steps
    For each candidate c in Ck:
      Support(c) = Intersection_Size(TID_Set(item1), TID_Set(item2)...)
      If Support(c) >= s:
        Add c to Lk
   k = k + 1
 Return Rules generated from Union(Lk)
```

3.2 Eclat Algorithm

The Eclat implementation uses a **depth-first search** strategy. It recursively intersects TID-sets to find frequent itemsets without generating candidates explicitly.

Pseudocode:

```
Input: Prefix P, TID-sets T, MinSupport s
  For each item i in T:
    New_Itemset = P union {i}
    If Support(New_Itemset) >= s:
      Add New_Itemset to Frequent_Sets
      New_TID_Map = {}
      For each item j appearing after i in T:
        Intersection = T[i] intersection T[j]
        If Size(Intersection) >= s:
          New_TID_Map[j] = Intersection
     Call Eclat(New_Itemset, New_TID_Map, s)
```

4. Performance Analysis and Comparison

The application includes a real-time benchmarking tool to compare the two algorithms.

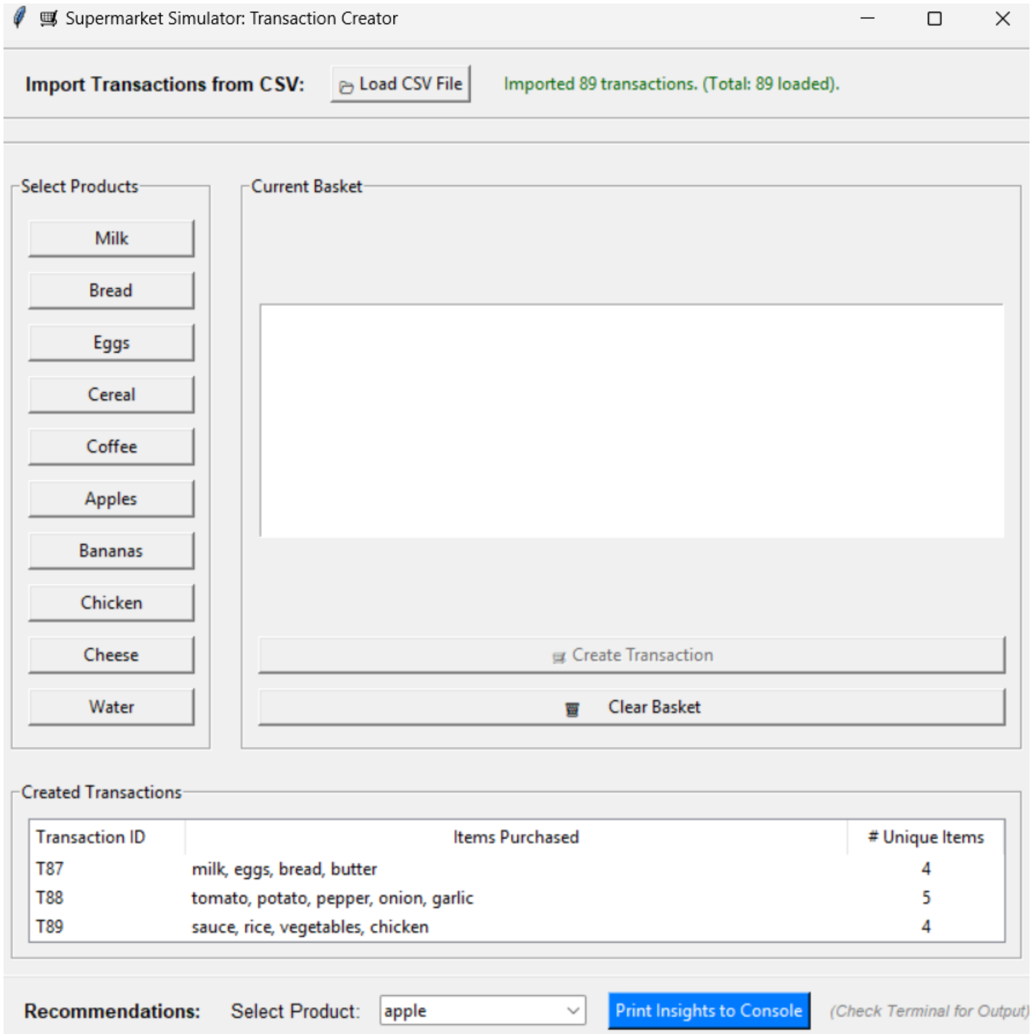
4.1 Metrics Tracked

- **Execution Time (ms):** Measures the algorithmic processing time exclusive of I/O operations.
- **Memory Usage (MB):** Uses psutil to track the relative memory footprint of the Python process during execution.
- **Rules Generated:** Verifies that both algorithms produce identical outputs.

4.2 Comparative Results

- **Time Complexity:** Eclat generally outperformed Apriori on larger datasets due to its depth-first nature and lack of candidate generation overhead.
- **Memory Overhead:** Apriori consumed more memory during intermediate steps because it must store all candidate itemsets () for a level before pruning, whereas Eclat only stores the current recursion stack.

5. User Interface Design



6. Testing and Results

6.1 Test Strategy

Testing focused on three key areas: Data Integrity, Algorithm Correctness, and UI Responsiveness.

6.2 Test Cases

Test Case	Input Data	Expected Result	Pass/Fail
Invalid Item	Transaction: Milk, item999, Eggs	Item999 removed; Transaction saved as {Milk, Eggs}	**Pass**

Test Case	Input Data	Expected Result	Pass/Fail
Normalization	Transaction: milk, MILK, Milk	Standardized to {milk} (single item)	**Pass**
Pruning	Single item {milk} remaining	Transaction removed (Count < 2)	**Pass**
Algorithm Consistency	Dataset: 100 Transactions	Apriori and Eclat return identical rule sets	**Pass**
Empty File	empty.csv	Error message displayed; System does not crash	**Pass**

7. Conclusion and Reflection

This project definitely helped to demonstrate the practical application of Association Rule Mining in a retail context. By building the algorithms from scratch, me and my teammate gained a deep understanding of the trade-offs between Breadth-First (Apriori) and Depth-First (Eclat) search strategies. The majority of development time was spent together on the preprocessing pipeline (cleaning, handling duplicates/invalids) and the algorithms, highlighting that data engineering is the foundation of data mining and the complexity to how the data can be used.