Ashley Prado
PID: 6409687
Artificial Intelligence Fall 2025

Interactive Supermarket Simulation with Association Rule Mining

1. Introduction and System Design

This project implements an interactive supermarket simulation that enables users to create and analyze shopping transactions using association rule mining. The system provides a complete pipeline: users can manually build transactions or import them from a CSV file, clean the dataset through preprocessing, and apply two data mining algorithms, **Apriori** and **Eclat**, to discover patterns in product co-purchases. Finally, the system presents the discovered rules through an intuitive interface that allows querying product associations and generating recommendations.

The system is designed using a modular architecture. The key components include:

- **User Interface (Streamlit):** Allows users to interact with the system without programming knowledge.
- **Preprocessing Module:** Cleans and standardizes transaction data.
- **Mining Algorithms:** Independent implementations of Apriori and Eclat.
- **Recommendation Engine:** Converts mined rules into human-readable insights.

This design ensures clarity, modularity, ease of testing, and a smooth experience for non-technical users.

2. Data Preprocessing Approach

The dataset contains 80–100 transactions and intentionally includes dirty data to simulate real-world scenarios. The preprocessing module performs a series of cleaning steps before mining.

Issues Identified Before Cleaning

- **Total transactions:** 100
- **Empty transactions:** 5
- **Single-item transactions:** 6
- **Duplicate items:** 9 instances
- **Invalid items:** 2
- **Case inconsistencies:** present
- **Extra whitespace:** present

These issues reduce data quality and negatively affect association mining, which relies on accurate item co-occurrence patterns.

Preprocessing Steps Implemented

1. **Empty Transaction Removal**
   Transactions containing zero items are removed entirely.

2. **Single-Item Transaction Handling**
   Transactions with only one item are removed because association rules require at least two items.

3. **Duplicate Item Removal**
   Within each transaction, repeated items are converted into a set to ensure uniqueness.

4. **Case Standardization**
   All items are converted to lowercase.

5. **Whitespace Trimming**
   Leading/trailing spaces are removed (e.g., "bread " → "bread").

6. **Invalid Product Handling**
   Items not present in products.csv are removed.

Post-Cleaning Summary

- **Valid transactions:** 89
- **Total items:** 276
- **Unique products:** 30

These steps ensure the dataset is clean, consistent, and ready for accurate association rule mining.

3. Algorithm Implementation Details

The project implements **two required algorithms**: Apriori and Eclat. CLOSET is not included.

3.1 Apriori Algorithm

Apriori uses a horizontal transaction representation and generates itemsets level-by-level using the principle that:

If an itemset is frequent, all of its subsets must also be frequent.

Apriori Pseudocode

```
Apriori(transactions, min_support):
  L1 = find_frequent_1_itemsets(transactions)
  k = 2
  L = {1: L1}

  while L[k-1] not empty:
    Ck = generate_candidates(L[k-1])
    for each candidate c in Ck:
      count support of c in transactions
    Lk = filter candidates by min_support
    L[k] = Lk
    k = k + 1
    return all frequent itemsets from L
```

Rule Generation Pseudocode

```
GenerateRules(frequent_itemsets, min_confidence):
    rules = []
    for each itemset I with size ≥ 2:
        for each non-empty subset A of I:
            B = I - A
            confidence = support(I) / support(A)
            if confidence ≥ min_confidence:
                add rule A → B
    return rules
```

3.2 Eclat Algorithm

Eclat uses a vertical data representation, assigning each item to a set of transaction IDs (TID-set). Frequent itemsets are discovered through depth-first search and set intersection.

Eclat Pseudocode

```
Eclat(prefix, items, min_support):
    for each item i in items:
        new_itemset = prefix ∪ {i}
        support = size of TID-set of i
        if support ≥ min_support:
            output new_itemset
            new_items = {}
            for each item j after i:
                intersect TID-set(i) and TID-set(j)
                if intersection non-empty:
                    new_items[j] = intersection
            Eclat(new_itemset, new_items, min_support)
```

Eclat is typically faster than Apriori on sparse datasets because it avoids candidate generation.

4. Performance Analysis and Comparison

The algorithms were tested on the cleaned dataset of 89 transactions, using:

- **Minimum support:** 0.2
- **Minimum confidence:** 0.5

Results

Algorithm	Runtime (ms)	Rules Generated
Apriori	0.2400	19
Eclat	0.2200	19

Analysis

- **Eclat** performed slightly faster due to efficient TID-set intersections and depth-first exploration.
- **Apriori** required more candidate generation, causing marginally higher runtime.
- Both produced the **same number of association rules**, consistent with the dataset size.

5. User Interface Design

The full system is implemented as a Streamlit web app.

Key UI Components

- **Product Buttons:** Users can manually build transactions.
- **CSV Import:** Users load sample_transactions.csv.
- **Preprocessing Report:** Automatically displayed after cleaning.
- **Sliders:** Allow adjusting support and confidence thresholds.

- **Tabs for Apriori and Eclat:** Show frequent itemsets, rules, and performance.
- **Product Query Tool:** Users select a product to get:
    - Associated items
    - Confidence values
    - Visual strength bars
    - Business-oriented recommendations

The UI emphasizes simplicity and accessibility for non-technical users.

6. Testing and Results

Test Case 1: Dirty Data Handling

- Loaded the dataset with 5 empty transactions, 6 single-item transactions, duplicates, invalid items, and inconsistent formatting.
- Verified all issues were removed or standardized.

Test Case 2: Threshold Sensitivity Test

- Adjusted support and confidence values.
- Confirmed rules updated accordingly.

Test Case 3: Recommendation Query

- Searched for items like “milk”.
- Verified that returned associated products matched expected patterns.

Test Case 4: CSV Import Error Handling

- Uploaded malformed CSV data.
- Streamlit displayed appropriate error messages.

All features function as expected.

7. Conclusion and Reflection

This project demonstrates a complete data mining pipeline applied to a realistic supermarket simulation. Through the implementation of Apriori and Eclat, the system successfully discovers meaningful association rules and presents them through a user-friendly interface.

Key lessons learned include:

- The importance of data preprocessing for improving mining accuracy.
- Differences between horizontal and vertical mining strategies.
- The value of designing clear interfaces that make complex algorithms accessible.
- How Generative AI tools can accelerate development while still requiring human understanding and oversight.

This project reflects practical skills useful for real-world applications in retail analytics, recommendation systems, and pattern discovery.

8. AI Tool Usage

ChatGPT was used to clarify data mining concepts, structure algorithm logic, write pseudocode, suggest Streamlit UI patterns, and generate portions of the README and report. All generated material was reviewed, corrected, and customized to align with the actual implementation. No AI tool was used to generate full algorithms; they were implemented manually following learned concepts.

9. References

- Agrawal, R., & Srikant, R. (1994). *Fast Algorithms for Mining Association Rules.*
- Zaki, M. J. (2000). *Scalable Algorithms for Association Mining.*
- CAI 4002: Artificial Intelligence – Lecture Materials
- Streamlit Documentation — https://docs.streamlit.io
- pandas Documentation — https://pandas.pydata.org