Introduction

In the assignment, we basically designed and implemented an interactive application that mimics that of the typical supermarket shopping experience. We performed various association rules mining, to uncover purchasing patterns, and recommend them to employers so that they may put those frequent item purchases near one another. In the application you're able to select from various products and save them as transaction. Afterwards you can use preprocessing to clean the transactions and make sure there aren't any duplicates, inconsistences, blanks, etc. From there you would go to the association mining tab and generate the number of rules and frequent item sets found within the transaction using either the Apriori algorithm or the Eclat algorithm. Finally, you can see the product recommendations, using either the Apriori algorithm or the Eclat algorithm, and see which items should be relatively bundled together.

Data Preprocessing Approach

First, we had to make sure all the product names were standardized. Meaning we had to convert all letters to lowercase and remove any spaces or gaps between them. Then the items would be checked to see if they were an actual valid product in the given products.csv. So, for instance, even though "cereal" is standardized, it isn't an actual valid product in the products.csv, so it gets dropped. We then remove transactions that become empty after the earlier check and transactions that become single items bundles. This can either happen because other items in that bundle were removed, leaving only a single item, or the bundle item set only had one item to begin with. Finally, we store the remaining items into a new set.

Algorithm Implementation

Apriori is implemented using a horizontal data format. The algorithm checks the database level by level and then builds candidates, starting from 1-itemset, by putting together frequent item sets. Candidates that aren't frequent get pruned.

Simple Pseudocode for Apriori (help from ChatGPT)

Input are the Transactions and Min Support

1.Count how many times single item appears

2.Keep items who support => min_support

3.For i = 2, 3, 4, ...

-Build candidate itemset by joining item sets

-Each candidate, count how many transaction contain it

-Keep candidate whose support => min_support

-Stop when empty

4. All item sets here, are the frequent item sets

Eclat is implemented using a vertical data format. The algorithm enhances item sets with Depth-first by intersecting TID-sets. The size is divided by the total number of transactions.

Simple Pseudocode for Eclati (help from ChatGPT)

Input are the Transactions and Min Support

1.For each item, build the set of transaction ID's that have it

2.Let min_sup_count = ceil(min_support * number_of_transactions).

3. Recursively grow itemsets:

- Start with each single item and its TID-set.

- Extend an itemset, by intersecting its TID-set with another item's TID-set.

- If the intersection size ≥ min_sup_count, the new itemset is frequent, and we can try to extend it again.

4. All item sets found this way are the frequent item sets.

Performance and Analysis Comparison

Given the same dataset plus the same minimum support and minimum confidence, the Eclat algorithm and Apriori method do and should produce the same rules and frequent item sets. For execution time, both numbers typically show up as 0ms because their values are less than 0.1ms. They are nearly identical in execution time it seems. And while we didn't test it ourselves, based off the information from ChatGPT and PowerPoint slides, it seems the Apriori algorithm needs less memory usage compared to the Eclat algorithm since the Eclat algorithm uses TID-sets,

User interface Design

The User interface was made simply with the aid and use of ChatGPT. We encouraged it to make a simple and flexible UI that anyone could maneuver without much understanding of the concepts we used. It also took inspiration from little images in the assignment description.pdf provided and from the charts in the provided read.me

Testing and Results

With strenuous testing we have come to various conclusions. However, we believe we have designed a system that encapsulates the correct results regardless of the number of transactions or the transaction itself. We have the needed performance measurements, the correct algorithm implementation, CSV imports, etc. Below are 2 test cases:

Test case 1: Using default sample_transactions with no additions and a minimum support of 0.2 and a minimum confidence of 0.5 we get

- 7 frequent itemsets for Apriori and Eclat
- 11 rules generated for Apriori and Eclat

- Test case 2: Using default sample_transactions with no additions and a minimum support of 0.1 and a minimum confidence of 0.5 we get

- 25 frequent itemsets for Apriori and Eclat
- 46 rules generated for Apriori and Eclat

Conclusion and Reflection

Overall, this assignment taught us various concepts and ideas such as preprocessing implementation and Apriori and Eclat algorithm implementations. We got to see the differences in Horizontal data sets compared to Vertical date sets. It also taught us how to interact with ChatGPT and get the proper and efficient results/code. In fact, ChatGPT was used to write the code to run the supermarket shopping application. We also used ChatGPT various times to fix any errors that the code would run into, and also to explain how the code works and functions. We also used ChatGPT to explain the implementation of both the Apriori and Eclat algorithms such as the data structure, candidate generation, and pruning. ChatGPT has helped us fill out various things within the readme and report

including testing, algorithms implementation, pseudocode, performance analysis, etc. For this assignment we used both the help of ChatGPT and the AI PowerPoint slides. We also learned how to transfer files from VScode straight to a GitHub repository.