

## **ASSIGNMENT 3B: BUILDING A GROCERY RECOMMENDATION SYSTEM**

**Group member – Student ID: Group 27**

- Alex (Thi Ngoc Bich) Nguyen      a1848722
- Anh Tu Vu      a1911757
- Quy Duong Dang      a1893592

### **Executive Summary**

Understanding customers' behaviours is a critical factor in increasing profits. Grocery stores typically save their customers' purchase history in the database, which can be utilised to devise selling strategies. For instance, if a store manager knows which items his customers prefer, he can make the appropriate suggestions that tempt them to buy more. In this project, we perform data mining techniques to explore which items are usually purchased together. In addition, we also build a collaborative filtering model to predict a customer's preferences, thereby recommending the relevant items to them. Finally, we combine the two modules into a hybrid system that outperforms all the individual modules and reasonably scales with the data over time. The system can help grocery stores improve sales by learning customers' purchase behaviours.

# 1. Introduction

A grocery store is a retail establishment that sells various food products and household items, providing many essential products for daily living. Customers typically purchase a set of repeated items together in their baskets or transactions (itemsets). By analysing these frequent itemsets, store managers can optimise their stock and display to address consumer needs better, thereby increasing sales and reducing waste, particularly from expired products. Moreover, beyond physical stocking and display, retailers can recommend a new range of relevant products to consumers via the store's website. This report aims to evaluate the performance of well-known recommendation systems, including frequent pattern mining and collaborative filtering, combined with a hybridisation layer for enhancing grocery sales.

## 2. Exploratory Analysis

### 2.1 Data Description

In this project, we use the transactional data from loyal customers' purchase history from 2014 to 2015. The data has been split into a train set (27,000 records) and a test set (11,765 records) with no null value. Figure 1 shows a snippet of the records in the train set with a row representing a transaction of a single item bought by a customer on a specific date with the following features:

- Member\_number: The ID number of the member.
- Date: The date of the transaction.
- itemDescription: Name of the purchased item.
- year: The year of the transaction.
- month: The month of the transaction.
- day: The day of the transaction.
- day\_of\_week: The day of the week, ranging from 0 (Monday) to 6 (Sunday).

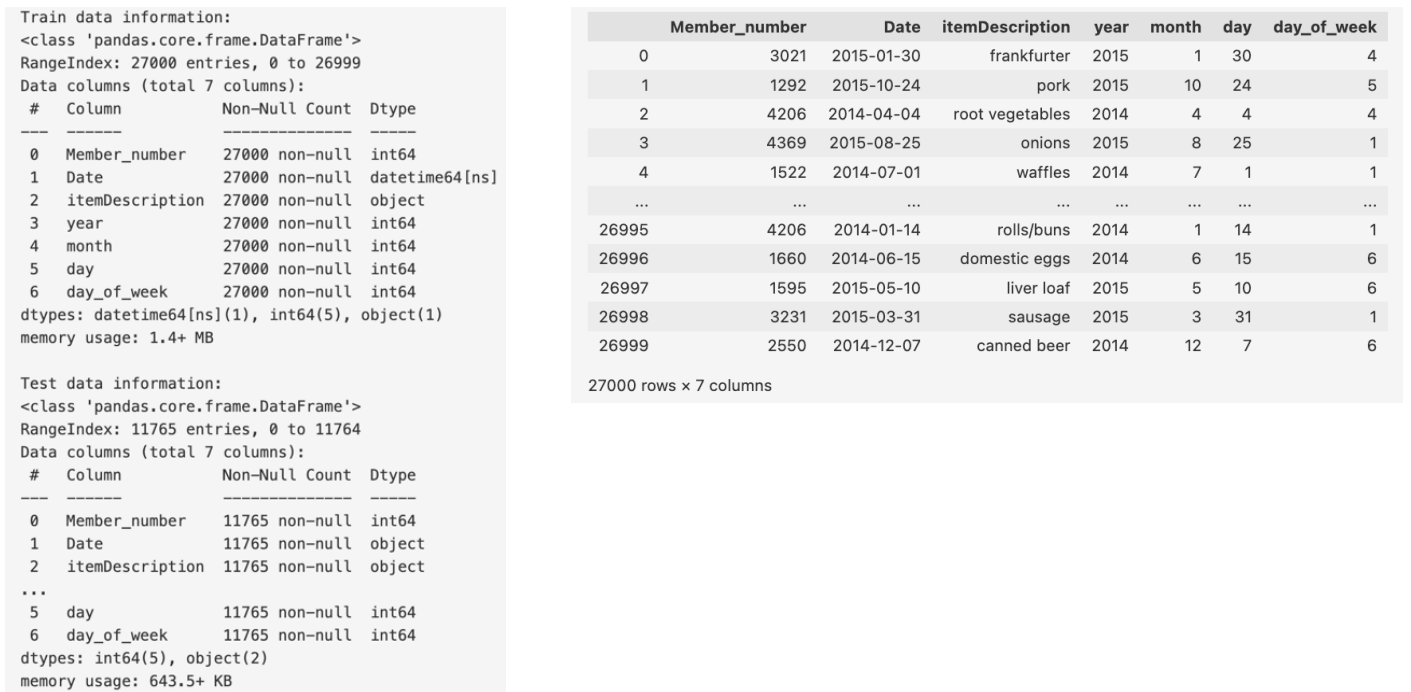


Figure 1: Summary of train and test data with a snippet from train data

## 2.2 Data Exploration

Figure 2 displays the top 10 most purchased items in the train set. We observe that the most purchased items are foods or beverages for daily consumption with specific preferences. This suggests the importance of a recommendation system that caters to the individual needs of every potential customer.

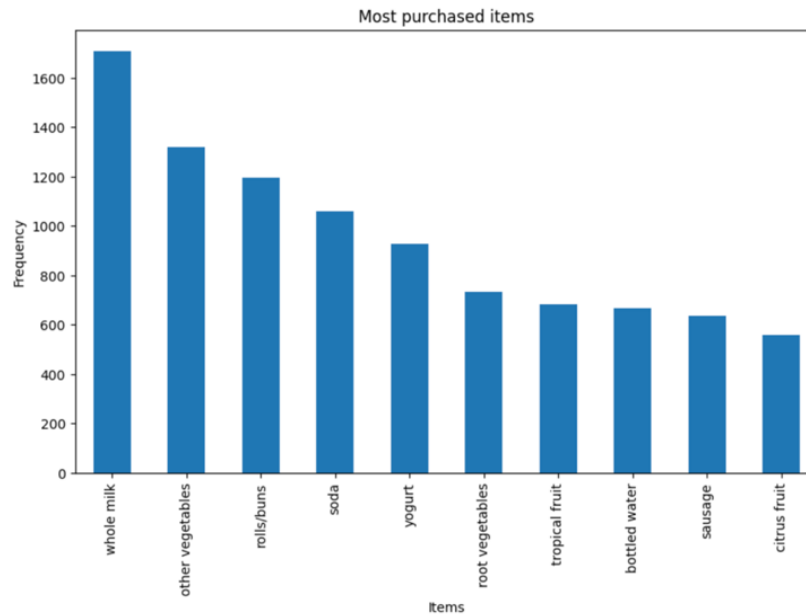


Figure 2: Most purchased items

Figure 3 demonstrates that most customers bought between 1 and 10 items within two years. The dataset has 167 unique items, suggesting that our data is very sparse, which might affect the system's training and evaluation later.

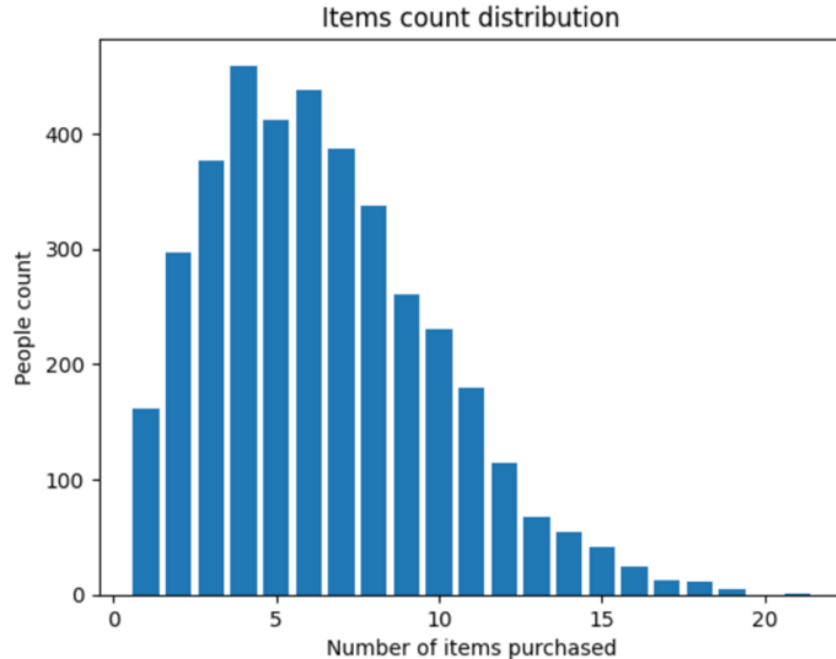


Figure 3: Items count distribution in the train set

Figure 4 offers insights into grocery shopping patterns, showing monthly and daily sales trends. Peaks in the line chart highlight seasonal or event-driven buying, which is crucial for planning promotions and inventory. The heatmap pinpoints busy days and months, helping to optimise staffing and marketing. These tools enhance recommendations and strategies by analysing user and seasonal data. However, a seasonal analysis is insufficient to make recommendations with the current small data set. It is better to start with user-based analysis and consider seasonal trends when more data is available.

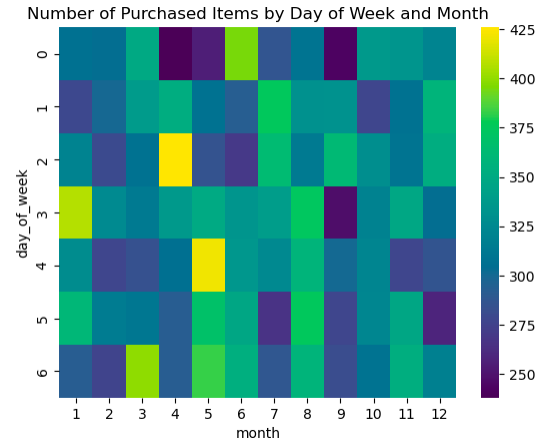
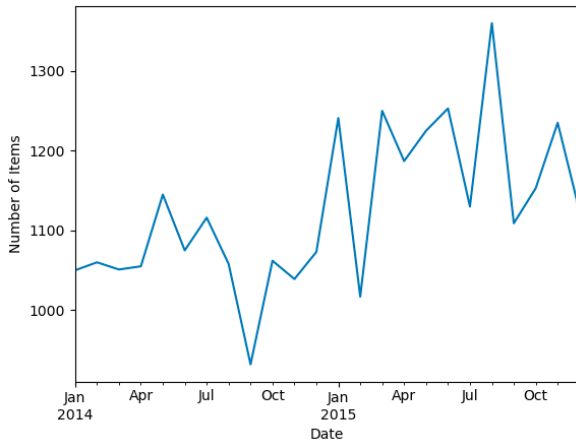


Figure 4: Items count by date, day of week, and month

## 2.3 System Architecture

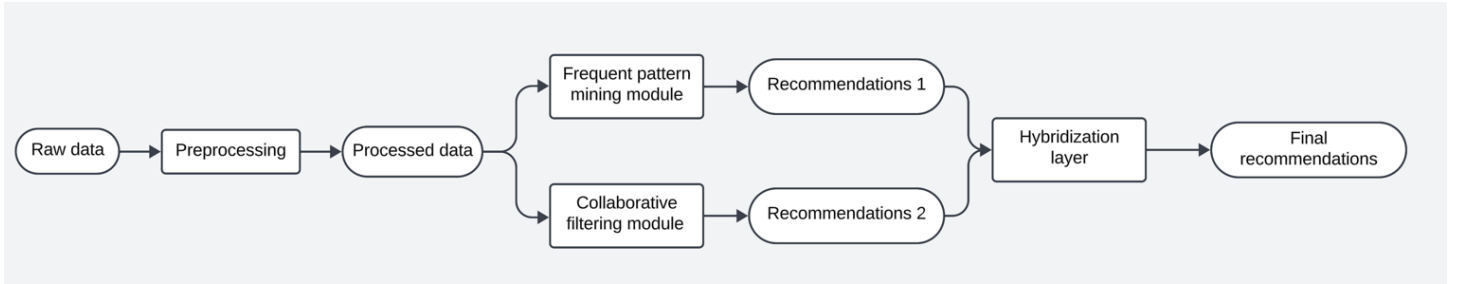


Figure 5: System Overview

Figure 5 summarises the entire architecture of our system. First, the raw data is pre-processed by renaming the items and checking for missing values. Next, the processed data is passed into two separate modules encoded in suitable formats. In the frequent pattern data mining module, all transactions made by every user are extracted to create a set of rules, which are later turned into the first list of recommendations. The techniques involved in this module are discussed in Section 5. In the second module, collaborative filtering, the data is converted to a utility matrix to produce the second list of recommendations. Finally, the two recommended lists are passed into a hybridisation layer using a weighted sum to compute the final predictions.

## 3. Frequent Pattern Mining

### 3.1 Method Description

The association analysis measures the strength of co-occurrence between items to find the pattern of item sets (Kotu and Deshpande, 2019). Association algorithms are popular for analysing retail transactions and recommendation systems to increase sales and product placement. The outcome of association analysis is a set of rules Item A  $\rightarrow$  Item B, which indicates when item A (*antecedent*) is found in a basket, there is strong evidence that item B (*consequent*) also occurs within the same transaction. Millions of customers' previous transactions would be analysed to collect these rules by prioritising computational resources and time. Sequential steps include converting data to the correct transaction format, shortlisting frequent item sets, and producing relevant association rules from the item sets based on interest measurements.

Several measurements are used to understand the strength of association rules: support, confidence, and lift.

- **Support:** Support is the relative frequency of an item's occurrence in a transaction set. The support of a rule measures how frequently all items in the rule appear together in the overall transactions, where higher support indicates more reliable and valuable rules.

- **Confidence:** Confidence measures the likelihood of *the consequent* occurring given that *the antecedent* is present in the transactions, providing a reliability measure for the rule.

$$Confidence(X \rightarrow Y) = \frac{Support(X \cup Y)}{Support(X)}$$

Equation 1: Confidence formula

- **Lift:** While confidence is commonly used, it can be misleading with infrequent items. Lift, which considers the support of both *the antecedent* and *consequent*, provides a more accurate measure of a rule's significance. Lift values close to 1 indicate that the antecedent and consequent are independent, while higher lift values suggest more significant rules.

$$Lift(X \rightarrow Y) = \frac{Support(X \cup Y)}{Support(X) * Support(Y)}$$

Equation 2: Lift formula

Generating meaningful association rules involves identifying frequent item sets by setting a minimal support threshold to filter out less common item sets and then extracting rules from these item sets with a confidence threshold. This process, visualised with an itemset tree, can produce many rules even for small datasets, so efficient algorithms like Apriori and FP-Growth are used to manage the computational resources.

### 3.1.1 Apriori Algorithm

The Apriori algorithm is famous for mining frequent item sets and generating association rules from transactional datasets. It operates on the principle that any subset of a frequent itemset must also be frequent. The process begins by identifying items that meet a minimum support threshold, the ratio of transactions containing the item to the total number of transactions.

The algorithm proceeds iteratively, generating candidate itemsets of increasing length. Each iteration eliminates itemsets that do not meet the minimum support threshold, reducing the search space. This step is crucial for efficiency, as it prevents the consideration of itemsets that cannot be frequent based on their subsets' frequencies.

Once all frequent item sets are identified, the algorithm generates association rules. Each rule implies  $A \rightarrow B$ , where A and B are itemsets. The usefulness of these rules is evaluated using metrics such as confidence, which is the ratio of the support of the combined itemsets (A and B) to the support of *the antecedent* (A).

The Apriori algorithm also uses confidence thresholds to prune low-confidence rules. If a rule does not meet the confidence threshold, any rule derived from it with a subset of the antecedent is also pruned, as it will have even lower confidence. This pruning step ensures that only the most significant rules are considered for further analysis and application in the field (Kotu and Deshpande, 2019).

### 3.1.2 FP-Growth

The Frequent Pattern (FP)-Growth algorithm efficiently mines frequent item sets and generates association rules from large datasets. It transforms the dataset into a compact data structure called the FP-Tree, which preserves the itemset ordering and frequency information.

The process begins by sorting items in each transaction based on their frequency. Items are then inserted into the FP-Tree in this sorted order, starting from a null root. If an item path already exists in the tree, its count is incremented; otherwise, new nodes are added. This tree structure effectively compresses the dataset, highlighting frequent patterns.

Frequent itemset generation in FP-Growth follows a bottom-up approach. The algorithm starts with the least frequent items found at the leaves of the FP tree and generates itemsets by traversing the tree upwards. If an item meets the minimum support threshold, all itemsets ending with that item are generated using its prefix path and conditional FP-Tree. A prefix path is a subtree containing all paths that include the item of interest, and a conditional FP-Tree is similar but excludes the target item.

Rule generation in FP-Growth is similar to the Apriori algorithm. Both methods aim to find frequently co-occurring items and generate association rules. However, FP-Growth's tree-based approach allows for greater efficiency and scalability, particularly in datasets with frequent paths. The compact FP-Tree structure reduces the need for repeated dataset scans, making the algorithm faster and more memory-efficient.

FP-Growth is widely used beyond traditional market basket analysis, including document clustering, text mining, and sentiment analysis applications. Its ability to handle large datasets and efficiently identify frequent patterns makes it a valuable tool in various data mining tasks (Kotu and Deshpande, 2019).

### 3.2 Hyper-parameters Tuning

For scaling up to 1 million customer transactions, the following hyper-parameters are recommended to ensure relevant and actionable patterns:

- *Minimum support*: After testing with values ranging from 0.001 to 0.05, the value of 0.01 is recommended. This means itemsets appearing at least 1% of the time (10,000 times in 1,000,000 transactions) will be considered, ensuring even less frequent but potentially significant itemsets are included.
- *Minimum confidence*: After testing with values ranging from 0.1 to 0.2, 0.2 is recommended. A confidence level of 20% ensures that only rules where items appear together more than 20% of the time are considered, balancing reliability and practical insights.
- *Lift value*: Greater than 1. This ensures that the identified rules show a strong positive correlation between items, guiding effective marketing strategies.

These parameters help focus on significant patterns in customer purchasing behaviour, filtering out noise. Using the FP-Growth and Apriori algorithms, the system can efficiently handle up to 1 million transactions, identifying key patterns to drive business decisions. The dual approach enhances the robustness of the findings through comparative analysis and results validation.

## 4. Collaborative Filtering

### 4.1 Method Description

The two most classic approaches in recommendation systems include content-based and collaborative filtering. Content-based filtering relies solely on an item's features to suggest new recommendations for a target user. More specifically, the method analyses which items the user prefers and makes suggestions based on similar items (Das, 2023). This allows the technique to work independently without extra data from different user profiles.

Collaborative, on the other hand, requires interactions from other users. The technique typically has two types: user-to-user or item-to-item (Evelyn, 2023). In this project, the user-to-user approach works based on the assumption that users who buy similar grocery items tend to have the same preference for other items. On the other hand, item-to-item suggests items identical to the ones preferred by users in the past. This differs from the content-based method (Das, 2023) since other users' interactions are also considered when calculating the similarity scores.

For this project, we mainly focus on the collaborative filtering technique, which leverages users' interactions to improve the quality of the recommendations. To calculate the similarity between users (user-to-user) or items (item-to-item), we can utilise either the Jaccard similarity or cosine similarity score. The Jaccard similarity measures the similarity between two vectors by considering their common elements (Jadeja, 2022). Assume we have two vectors, A and B; their similarity can be calculated by Equation 3.

$$J(A, B) = \frac{A \cap B}{A \cup B}$$

Equation 3: Jaccard similarity

In the context of our project, the Jaccard similarity is not very efficient at capturing the similarity between users or items because it completely discards the user’s rating for a particular item (how many units of that item were bought by the user) while only counting the common elements.

Alternatively, we will use the cosine similarity (Prakash, 2023), a more suitable metric demonstrated in Equation 4.

$$\cos(A, B) = \frac{A \cdot B}{||A|| \cdot ||B||}$$

Equation 4: Cosine similarity

Collaborative filtering involves two major stages: item count prediction and item recommendation. In the first stage, given a target user and an item, we can apply collaborative filtering to predict how many units of the item our user will buy. The algorithm first converts the raw transactions into a utility matrix. Figure 6 demonstrates the matrix for the user-to-user approach in the train set, where each row is a user, each column displays an item, and each cell indicates the units of the item bought by the user. The matrix is transposed for item-to-item, where each row is an item, and each column is a user. Below are the steps for the user-to-user approach (which works similarly for the transposed matrix in item-to-item):

Given a target user in the train set, calculate the cosine similarity between the target and every other user.

With  $S$  as the similarity vector and  $I$  item vector, calculate the dot product  $S \cdot I$ . The result gives us the item count prediction for that user.

In the second stage, we attempt to give our target user a list of recommended items. Given a list of new items, we sort them based on their prediction scores and recommend the top  $k$  items based on the requirement.

itemDescription	abrasive cleaner	artif. sweetener	baby cosmetics	bags	baking powder	bathroom cleaner	beef	...
Member_number								...
1000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1001	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...
1002	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1003	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1004	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
...	...	...	...	...	...	...	...	...
4996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
4997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
4998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
4999	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
5000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...

[3872 rows x 167 columns]

Figure 6: Utility matrix for user-to-user in train set

## 4.2 Evaluation Metrics

We use the Mean Average Precision at  $k$  (MAP@ $k$ ) and root mean square error (RMSE) to evaluate the system’s performance on the test set. In the context of this project, MAP@ $k$  measures how well our system finds and ranks the relevant items (EvidentlyAI, 2024). Given a target user, the appropriate items (ground truths) are defined as those that exist in the test set but not in the train set. In simple terms, they are the new items that the user has bought in the test set, which we focus on since we want to recommend new items to the user. The  $k$  term indicates the threshold of the top predicted items we evaluate. As our system produces a list of items as predictions, we can compare how many exist in the relevant list and how well they rank to evaluate our system’s performance. The score ranges between 0 and 1, with 0 indicating the worst performance and 1 the best.

For this project, we choose  $k$  with 1, 3, 5, and 10 values. Ideally, we want to calculate the MAP@ $k$  for each value of  $k$  for all users in the test set. However, due to the data sparsity explained in Section 2.2, we cannot apply the metric to all users. For instance, if a user only bought five new items in the test set, it does not make sense to include this user while evaluating MAP@10. Therefore, given a  $k$  value, we only assess the MAP@ $k$  score on users who have bought at least  $k$  new items in the test set.



Due to the sparse dataset, MAP@k might not fully describe the performance of our recommendation system. Therefore, we use the RMSE score as an extra measurement (Gupta, 2021). For every user in the test set, we gather all the new items they have bought and store the values as the ground truth. Then, we use collaborative filtering to predict each value and compare them with the ground truth using RMSE. By turning the recommendation task into item count prediction, we can use RMSE to measure the results, just like a regression problem.

## 5. Recommendation Method from Frequent Patterns

### 5.1 Recommendation From Patterns

The pattern mining techniques discussed in Section 3.1 return a ranked list of rules, each expressed as *antecedents*  $\rightarrow$  *consequents*. The rule suggests that if a user buys the items in the *antecedents*, they will likely buy those from the *consequents* list as well. Given the rules as input, we aim to indicate a list of recommended items to the target user.

We review each rule in the ranked list to perform this task and compare the *antecedents* with the user's purchase history. If the *antecedents* in a rule are a subset of the user's purchased items, we add the *consequents* to our recommended list.

As discussed in the previous section, we evaluate the technique's performance on the test set using MAP@k.

### 5.2 Hybrid Method with Collaborative Filtering

To combine the effects of frequent patterns and collaborative filtering, we build a hybrid system that utilises the results from both techniques. We denote  $R1$  as the recommended list from the frequent patterns and  $R2$  as the list from collaborative filtering. Consider an item  $i$  in our dataset, we compute the score for  $i$  based on the three following scenarios:

- If  $i$  only exists in  $R1$ :  $score(i) = 0.3$
- If  $i$  only exists in  $R2$ :  $score(i) = 0.7$
- If  $i$  exists in both  $R1$  and  $R2$ :  $score(i) = 0.3 + 0.7 = 1$

Finally, we combine all items from both  $R1$  and  $R2$ , then sort them based on the scores and return the top  $k$  items ( $k$  is the system's parameter).

The technique prioritises recommended items from the collaborative filtering module by assigning them a higher weight. If an item appears in both recommended lists, it will be given a higher score in the predictions.

## 6. Discussion of Results

### 6.1 Mining Frequent Patterns

Table 1 illustrates five examples of frequent patterns on training and test sets, showcasing that these patterns have strong relationships to uplift sales (the lift on the train set ranges from 1.90 to 2.16, while the lift on the test set ranges from 1.10 to 1.44).



Table 1: Five frequent patterns of both train and test sets

Rule	Support (Train)	Confidence (Train)	Lift (Train)	Support (Test)	Confidence (Test)	Lift (Test)
sausage, rolls/buns → other vegetables, whole milk	0.0114	0.2431	2.1638	0.0137	0.2178	1.4408
other vegetables, rolls/buns, whole milk → sausage	0.0114	0.3034	2.0470	0.0165	0.2588	1.2960
sausage, other vegetables → rolls/buns, whole milk	0.0114	0.2115	1.9929	0.0101	0.2553	1.2787
brown bread, whole milk → bottled water	0.0132	0.3148	1.9692	0.0373	0.2468	1.2358
shopping bags, whole milk → canned beer	0.0106	0.2216	1.9112	0.0104	0.2216	1.1097

We can make user recommendations based on the five frequent rules based on their purchase history. Given a rule, assuming the *antecedents* are a subset of a user's purchased list, we can recommend the *consequents* to them. Table 2 demonstrates ten examples of such recommendations.

Table 2: Ten recommendations examples based on five frequent patterns

Rule	Bought Items	Recommendations
sausage, rolls/buns → other vegetables, whole milk	rolls/buns, specialty bar, butter, butter milk, frozen vegetables	other vegetables, whole milk
	frozen dessert, sausage, cat food, coffee, canned beer, yogurt, rolls/buns, whole milk, etc.	other vegetables
other vegetables, rolls/buns, whole milk → sausage	other vegetables, rolls/buns, whole milk, brown bread, etc.	sausage
	other vegetables, rolls/buns, whole milk, domestic eggs, coffee, etc.	sausage
sausage, other vegetables → rolls/buns, whole milk	sausage, other vegetables, herbs, onions, etc.	rolls/buns, whole milk
	sausage, other vegetables, root vegetables, slice cheese	rolls/buns, whole milk
brown bread, whole milk → bottled water	brown bread, whole milk, sausage, oil, etc.	bottled water
	brown bread, whole milk, beverages, etc.	bottled water
shopping bags, whole milk → canned beer	shopping bags, whole milk, meat, brown bread, etc.	canned beer
	shopping bags, whole milk, pastry, tropical fruit	canned beer

Looking at the detailed association rules of the test set in Table 2, except for the combination of pip fruit and other vegetables, other daily consumption is bought together with the whole milk with high confidence and lift values.

Table 3: Association rules of frequent patterns on both train and test sets

Antecedents	Consequents	Support	Confidence	Lift
pip fruit	other vegetables	0.0137	0.2178	1.4408
shopping bags	whole milk	0.0166	0.2588	1.2960
beef	whole milk	0.0101	0.2553	1.2788
other vegetables	whole milk	0.0373	0.2468	1.2359
frankfurter	whole milk	0.0104	0.2216	1.1097
root vegetables	whole milk	0.0199	0.2212	1.1078
bottled beer	whole milk	0.0112	0.2151	1.0771
citrus fruit	whole milk	0.0140	0.2058	1.0305
canned beer	whole milk	0.0126	0.2055	1.0291
tropical fruit	whole milk	0.0191	0.2055	1.0289
rolls/buns	whole milk	0.0280	0.2041	1.0221
yogurt	whole milk	0.0219	0.2037	1.0100

## 6.2 Collaborative Filtering: user-to-user versus item-to-item

Table 4 summarises the results of the user-to-user and item-to-item approaches for our collaborative filtering module. The models are compared on the train and test sets using RMSE and MAP@k scores, with  $k$  taking 1, 3, 5, and 10 values. We can observe that the user-to-user approach shows clear dominance on all metrics, achieving lower RMSE scores and higher MAP@k values. As a result, we choose this approach for our hybrid models in the next section.

Table 4: User-to-user and item-to-item comparison

Dataset	Metrics	User-to-user	Item-to-item
Train	RMSE	<b>0.2913</b>	0.3484
	MAP@1	<b>0.4964</b>	0.0220
	MAP@3	<b>0.3937</b>	0.0171
	MAP@5	<b>0.2992</b>	0.0171
	MAP@10	<b>0.3075</b>	0.0085
Test	RMSE	<b>0.1831</b>	0.2506
	MAP@1	<b>0.1747</b>	0.0130
	MAP@3	<b>0.1466</b>	0.0103
	MAP@5	<b>0.1446</b>	0.0105
	MAP@10	<b>0.1748</b>	0.0153

## 6.3 Hybrid System

Our hybrid model consists of collaborative filtering and data pattern mining modules. The former is built using the user-to-user approach, while the latter will be tested using both the Apriori and FP-Growth algorithms. When run separately, we will compare our hybrid system with the Apriori, FP-Growth, and user-to-user models to see whether it has made any significant improvement.

Table 5 summarises the performance of the models. First, FP-Growth seems to perform worse than Apriori as the value of  $k$  increases. Nonetheless, both models show no significant difference in performance. Secondly, the hybrid model with user-to-user and FP-Growth achieves the best results on the test set, even though the individual user-to-user model outperforms it on the train set.

Table 5: Hybrid system comparison

Dataset	Metric	Apriori	FP-Growth	User-to-user	User-to-user + Apriori	User-to-user + FP-Growth
Train	MAP@1	0.1415	0.2629	<b>0.4964</b>	<b>0.4964</b>	<b>0.4964</b>
	MAP@3	0.1190	0.1445	<b>0.3937</b>	0.3892	0.3747
	MAP@5	0.1157	0.0981	<b>0.2992</b>	0.2840	0.2818
	MAP@10	0.1220	0.0555	<b>0.3075</b>	0.2989	0.3000
Test	MAP@1	0.0709	0.1258	<b>0.1747</b>	<b>0.1747</b>	<b>0.1747</b>
	MAP@3	0.0722	0.0718	0.1466	0.1456	<b>0.1476</b>
	MAP@5	0.0737	0.0551	0.1446	0.1346	<b>0.1462</b>
	MAP@10	0.0992	0.0667	<b>0.1748</b>	<b>0.1748</b>	<b>0.1748</b>

## 6.4 Execution Time Evaluation

Even though the hybrid system of user-to-user and FP-Growth produces the best results, it does not scale very well with the data. FP-Growth is very memory intensive and slow when the number of records is scaled to one million, especially when we have lots of distinct values in the dataset. Initial experiment also shows that the hybrid system runs very slowly with one million transactions. Due to insufficient hardware resources, we have decided to switch to user-to-user as the main model since its performance difference is insignificant while scaling better with the data size.

To see how the system’s execution time scales with the sample size, we generate dummy data with 10,000, 100,000, and 1,000,000 transactions. Each size is measured with k values of 1, 3, 5, and 10. The experiment was performed on a MacBook Pro with an M3 Max chip as follows:

- 14-core CPU with ten performance cores and four efficiency cores
- 30-core GPU
- 16-core Neural Engine
- 300GB/s memory bandwidth.

Table 6: Inference time comparison between different sample sizes (in seconds).

	size = 10000	size = 100000	size = 1000000
<b>MAP@1</b>	0.1510	0.3881	1.0432
<b>MAP@3</b>	0.1509	0.4063	1.0255
<b>MAP@5</b>	0.1507	0.4045	1.0465
<b>MAP@10</b>	0.1491	0.4141	1.0384

Table 6 shows that the user-to-user approach scales very well with the data. For one million transactions, the system only takes roughly one second to produce a recommendation. As the data size increases by 10 times, the executed time only takes roughly 2.5 – 2.7 times longer. It is also worth noting that the k value does not affect the system’s performance. We can conclude that our system is fast enough to be implemented in real-life scenarios when the number of transactions grows quickly.

## 7. Conclusion and Recommendations

The results from our experiment suggest that the hybrid system combining the user-to-user and FP-Growth produces the best results. However, its improved performance is negligible while the executed time scales very poorly with the data size. Therefore, we suggest that the user-to-user model is the best candidate for our system.

The recommendation system is only a prototype and still needs further work to improve. First, the system must consider the common cold-start problem where new users shop at the store. The system cannot make relevant recommendations for these users since their purchase history is outside the database. Secondly, we must address the data sparsity problem, which occurs when users only purchase a few unique items. One solution is to group similar items to reduce the total number of items. For example, items such as “root vegetables”, “other vegetables”, and “frozen vegetables” can be grouped as “vegetables”. We can apply a pre-trained model to embed the item names and user cosine similarity to detect similar items for grouping. Finally, we can explore some state-of-the-art techniques using neural networks to boost the performance of our current system.

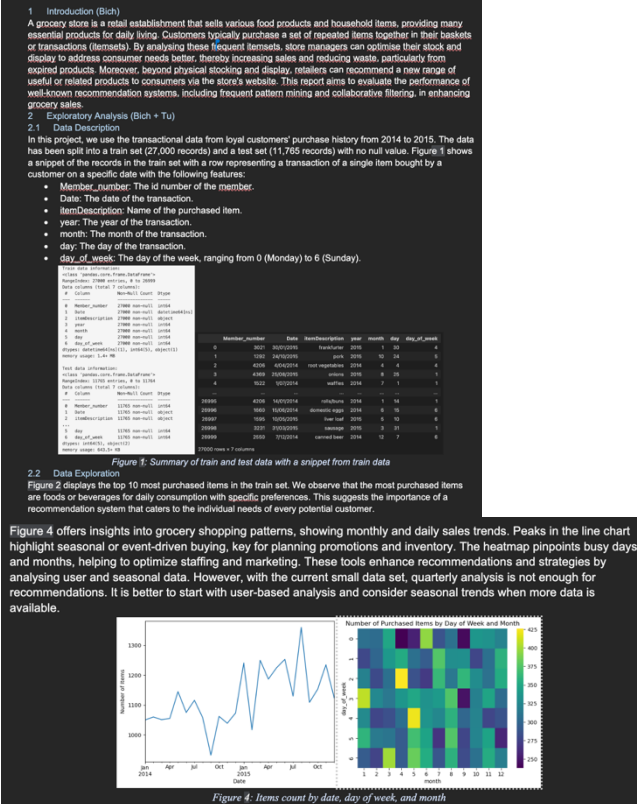
## 8. Reflection

In this project, we have learned how to work as a group to develop a recommendation system by dividing and completing the tasks simultaneously. The tasks involve implementing a data mining pattern, building a collaborative filtering system, and combining the two to create a final hybrid system. The main challenges in this project were researching the method to integrate our modules for the hybrid system and developing the proper metrics to evaluate its performance. The main area we need to improve next time is our system’s performance since its metrics are currently showing relatively poor performance. In addition, we also need to consider exceptional cases like the previous cold-start problem and accommodate new users joining the system.

## 9. References

- DAS, S. 2023. *Beginners Guide to Content Based Recommender Systems* [Online]. Available: <https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/> [Accessed May 30 2024].
- EVELYN. 2023. *Collaborative Filtering in Recommender System: An Overview* [Online]. Available: <https://medium.com/@evelyn.eve.9512/collaborative-filtering-in-recommender-system-an-overview-38dfa8462b61> [Accessed May 30 2024].
- EVIDENTLYAI. 2024. *How mean Average Precision at  $k$  ( $mAP@k$ ) can be more useful than other evaluation metrics* [Online]. Available: <https://www.evidentlyai.com/ranking-metrics/mean-average-precision-map> [Accessed May 30 2024].
- GUPTA, S. 2021. *RMSE: What does it mean?* [Online]. Available: <https://medium.com/@mygreatlearning/rmse-what-does-it-mean-2d446c0b1d0e#> [Accessed May 30 2024].
- JADEJA, M. 2022. *Jaccard Similarity Made Simple: A Beginner’s Guide to Data Comparison* [Online]. Available: <https://medium.com/@mayurdhvajsinhjadeja/jaccard-similarity-34e2c15fb524> [Accessed May 30 2024].
- KOTU, V. & DESHPANDE, B. 2019. *Data science: concepts and practice*, Morgan Kaufmann.
- PRAKASH, A. 2023. *Understanding Cosine Similarity: A key concept in data science* [Online]. Available: <https://medium.com/@arjunprakash027/understanding-cosine-similarity-a-key-concept-in-data-science-72a0fcc57599> [Accessed May 30 2024].

## Contribution Appendices

SECTION IN CHARGE	MEMBER	SCREENSHOT or LINK
Code: Task 1 & EDA	Alex (Thi Ngoc Bich) Nguyen a1848722	<a href="https://drive.google.com/drive/folders/1XGSwnerYLREAOTPWMa9IznIEt2CydAup">https://drive.google.com/drive/folders/1XGSwnerYLREAOTPWMa9IznIEt2CydAup</a>
Code: Task 2	Anh Tu Vu a1911757	<ul style="list-style-type: none"> <li>Wrote the following classes: Apriori, FPGrowth, CollaborativeFilteringUserBased, CollaborativeFilteringItemBased, HybridSystem, Utils, Time</li> <li>Converted the FP-Growth code to appropriate class</li> <li>Improved the algorithm to combine the modules for the hybrid system (HybridSystem class)</li> </ul>
Code: Task 3	Quy Duong Dang a1893592	<a href="https://drive.google.com/drive/folders/1SaLpAO82I0oEgTU9Eb7wb6E_Frzh9fV?usp=sharing">https://drive.google.com/drive/folders/1SaLpAO82I0oEgTU9Eb7wb6E_Frzh9fV?usp=sharing</a>
Report writing: 1. Introduction 2. Exploratory Analysis 3. Frequent Pattern Mining 6. Discussion of Results (frequent pattern examples and evaluation from test set)	Alex (Thi Ngoc Bich) Nguyen a1848722	 <p>1 Introduction (Bich)</p> <p>A grocery store is a retail establishment that sells various food products and household items, providing many essential products for daily living. Customers typically purchase a set of repeated items together in their baskets or transactions (baskets). By analyzing these frequent itemsets, store managers can optimise their stock and display to address consumer needs better, thereby increasing sales and reducing waste, particularly from expired products. Moreover, beyond physical stocking and display, retailers can recommend a new range of useful or related products to consumers via the store's website. This report aims to evaluate the performance of well-known recommendation systems, including frequent pattern mining and collaborative filtering, in enhancing grocery sales.</p> <p>2 Exploratory Analysis (Bich + Tu)</p> <p>2.1 Data Description</p> <p>In this project, we use the transactional data from loyal customers' purchase history from 2014 to 2015. The data has been split into a train set (27,000 records) and a test set (11,705 records) with no null value. Figure 1 shows a snippet of the records in the train set with a row representing a transaction of a single item bought by a customer on a specific date with the following features:</p> <ul style="list-style-type: none"> <li>Member_number: The id number of the member.</li> <li>Date: The date of the transaction.</li> <li>ItemDescription: Name of the purchased item.</li> <li>year: The year of the transaction.</li> <li>month: The month of the transaction.</li> <li>day: The day of the transaction.</li> <li>day_of_week: The day of the week, ranging from 0 (Monday) to 6 (Sunday).</li> </ul> <p>Figure 1: Summary of train and test data with a snippet from train data</p> <p>2.2 Data Exploration</p> <p>Figure 2 displays the top 10 most purchased items in the train set. We observe that the most purchased items are foods or beverages for daily consumption with specific preferences. This suggests the importance of a recommendation system that caters to the individual needs of every potential customer.</p> <p>Figure 4 offers insights into grocery shopping patterns, showing monthly and daily sales trends. Peaks in the line chart highlight seasonal or event-driven buying, key for planning promotions and inventory. The heatmap pinpoints busy days and months, helping to optimize staffing and marketing. These tools enhance recommendations and strategies by analysing user and seasonal data. However, with the current small data set, quarterly analysis is not enough for recommendations. It is better to start with user-based analysis and consider seasonal trends when more data is available.</p>



	<div><div>3 Frequent Pattern Mining</div><div>3.1 Method Description</div><div>The association analysis measures the strength of co-occurrence between items to find the pattern of item sets (Kotu and Deshpande, 2019). Association algorithms are popular for analysing retail transactions and recommendation systems to increase sales and product placement. The outcome of association analysis is a set of rules Item A → Item B, which indicates when Item A (antecedent) is found in a basket, there is strong evidence that Item B (consequent) also occurs within the same transaction. Millions of customers' previous transactions would be analysed to collect these rules by prioritising computational resources and time. Sequential steps include converting data to the correct transaction format, shortlisting frequent item sets, and producing relevant association rules from the item sets based on interest measurements.</div><div>Several measurements are used to understand the strength of association rules: support, confidence, and lift.</div><div><ul style="list-style-type: none"><li>Support: Support is the relative frequency of an item's occurrence in a transaction set. The support of a rule measures how frequently all items in the rule appear together in the overall transactions, where higher support indicates more reliable and valuable rules.</li><li>Confidence: Confidence measures the likelihood of the consequent occurring given that the antecedent is present in the transactions, providing a reliability measure for the rule.</li></ul></div><div><math display="block">\text{Confidence}(X \rightarrow Y) = \frac{\text{Support}(X \cup Y)}{\text{Support}(X)}</math></div><div>Equation 1: Confidence formula</div><div><ul style="list-style-type: none"><li>Lift: While confidence is commonly used, it can be misleading with infrequent items. Lift, which considers the support of both the antecedent and consequent, provides a more accurate measure of a rule's significance. Lift values close to 1 indicate that the antecedent and consequent are independent, while higher lift values suggest more significant rules.</li></ul></div><div><math display="block">\text{Lift}(X \rightarrow Y) = \frac{\text{Support}(X \cup Y)}{\text{Support}(X) \times \text{Support}(Y)}</math></div><div>Equation 2: Lift formula</div><div>Generating meaningful association rules involves identifying frequent item sets by setting a minimal support threshold to filter out less common item sets and then extracting rules from these item sets with a confidence threshold. This process, visualised with an itemset tree, can produce many rules even for small datasets, so efficient algorithms like Apriori and FP-Growth are used to manage the computational resources.</div><div>3.1.1 Apriori Algorithm</div><div>The Apriori algorithm is popular for mining frequent item sets and generating association rules from transactional datasets. It operates on the principle that any subset of a frequent itemset must also be frequent. The process begins by identifying items that meet a minimum support threshold, the ratio of transactions containing the item to the total number of transactions.</div><div>6.1 Mining Frequent Patterns</div><div>Figure 7 illustrates five examples of frequent patterns on both training and test sets, showcasing these patterns have strong relationships to uplift sales (lift values of train set 1.90-2.16 and test set 1.10-1.44).</div><div><table><tr><th>Comparison of top 5 rules between Training and Test Sets</th><th>Rule</th><th>Support (Train)</th><th>Confidence (Train)</th><th>Lift (Train)</th><th>Support (Test)</th><th>Confidence (Test)</th><th>Lift (Test)</th></tr><tr><td>frequent('sausage', 'rollbacks') → frequent('other vegetables', 'whole milk')</td><td></td><td>0.01384</td><td>0.24094</td><td>2.16353</td><td>0.01342</td><td>0.22778</td><td>1.40886</td></tr><tr><td>frequent('other vegetables', 'rollbacks', 'whole milk') → frequent('sausage')</td><td></td><td>0.01384</td><td>0.26048</td><td>2.04954</td><td>0.01246</td><td>0.26772</td><td>1.26684</td></tr><tr><td>frequent('sausage', 'other vegetables') → frequent('rollbacks', 'whole milk')</td><td></td><td>0.01384</td><td>0.21538</td><td>1.92088</td><td>0.00985</td><td>0.20539</td><td>1.27847</td></tr><tr><td>frequent('sausage', 'sausage', 'whole milk') → frequent('other vegetables')</td><td></td><td>0.01373</td><td>0.24401</td><td>1.98044</td><td>0.01297</td><td>0.24759</td><td>1.23544</td></tr><tr><td>frequent('sausage', 'whole milk') → frequent('other vegetables')</td><td></td><td>0.01009</td><td>0.22822</td><td>1.91178</td><td>0.01076</td><td>0.22037</td><td>1.18663</td></tr></table></div><div>Figure 7: Five frequent patterns on both train and test sets</div><div>Looking at detailed association rules of test set in Figure 8, except combination of pip fruit and other vegetables, other daily consume is bought together with the whole milk with high value of confidence and lift values.</div><div><table><tr><th>Association Rules on Test Data (Top10)</th><th>Antecedents</th><th>Consequents</th><th>Support</th><th>Confidence</th><th>Lift</th></tr><tr><td>1</td><td>(pip fruit)</td><td>other vegetables</td><td>0.01718</td><td>0.31778</td><td>1.44886</td></tr><tr><td>2</td><td>(shopping bag)</td><td>whole milk</td><td>0.01045</td><td>0.28772</td><td>1.29688</td></tr><tr><td>3</td><td>(other vegetables)</td><td>whole milk</td><td>0.00865</td><td>0.25559</td><td>1.21847</td></tr><tr><td>4</td><td>(other vegetables)</td><td>whole milk</td><td>0.01701</td><td>0.24873</td><td>1.23544</td></tr><tr><td>5</td><td>(other vegetables)</td><td>whole milk</td><td>0.01076</td><td>0.22037</td><td>1.18663</td></tr><tr><td>6</td><td>(other vegetables)</td><td>whole milk</td><td>0.01009</td><td>0.22184</td><td>1.18773</td></tr><tr><td>7</td><td>(other vegetables)</td><td>whole milk</td><td>0.01127</td><td>0.22084</td><td>1.18786</td></tr><tr><td>8</td><td>(other vegetables)</td><td>whole milk</td><td>0.01045</td><td>0.28772</td><td>1.29688</td></tr><tr><td>9</td><td>(other vegetables)</td><td>whole milk</td><td>0.01076</td><td>0.22037</td><td>1.18663</td></tr><tr><td>10</td><td>(other vegetables)</td><td>whole milk</td><td>0.01009</td><td>0.22184</td><td>1.18773</td></tr></table></div><div>Figure 8: Association rules of frequent patterns on test sets</div><div>Once all frequent item sets are identified, the algorithm generates association rules. Each rule implies the form A → B, where A and B are itemsets. The usefulness of these rules is evaluated using metrics such as confidence, which is the ratio of the support of the combined itemsets (A and B) to the support of the antecedent (A).</div><div>The Apriori algorithm also uses confidence thresholds to prune low-confidence rules. If a rule does not meet the confidence threshold, any rule derived from it with a subset of the antecedent is also pruned, as it will have even lower confidence. This pruning step ensures that only the most significant rules are considered for further analysis and application (Kotu and Deshpande, 2019).</div><div>3.1.2 FP-Growth</div><div>The Frequent Pattern (FP)-Growth algorithm efficiently mines frequent item sets and generates association rules from large datasets. It transforms the dataset into a compact data structure called the FP-Tree, which preserves the itemset ordering and frequency information.</div><div>The process begins by sorting items in each transaction based on their frequency. Items are then inserted into the FP-Tree in this sorted order, starting from a null root. If an item path already exists in the tree, its count is incremented; otherwise, new nodes are added. This tree structure effectively compresses the dataset, highlighting frequent patterns.</div><div>Frequent itemset generation in FP-Growth follows a bottom-up approach. The algorithm starts with the least frequent items found at the leaves of the FP tree and generates itemsets by traversing the tree upwards. If an item meets the minimum support threshold, all itemsets ending with that item are generated using its prefix path and conditional FP-Tree. A prefix path is a subtree containing all paths that include the item of interest, and a conditional FP-Tree is similar but excludes the target item.</div><div>Rule generation in FP-Growth is similar to the Apriori algorithm. Both methods aim to find frequently co-occurring items and generate association rules. However, FP-Growth's tree-based approach allows for greater efficiency and scalability, particularly in datasets with many frequent paths. The compact FP-Tree structure reduces the need for repeated dataset scans, making the algorithm faster and more memory-efficient.</div><div>FP-Growth is widely used beyond traditional market basket analysis, including document clustering, text mining, and sentiment analysis applications. Its ability to handle large datasets and efficiently identify frequent patterns makes it a valuable tool in various data mining tasks (Kotu and Deshpande, 2019).</div><div>3.2 Hyper-parameters Tuning</div><div>For scaling up to 1 million customer transactions, the following hyper-parameters are recommended to ensure relevant and actionable patterns:</div><div><ul style="list-style-type: none"><li>Minimum support: 0.01. This means itemsets appearing at least 1% of the time (10,000 times in 1,000,000 transactions) will be considered, ensuring even less frequent but potentially important itemsets are included.</li><li>Minimum confidence: 0.2. A confidence level of 20% ensures that only rules where items appear together more than 20% of the time are considered, balancing reliability and practical insights.</li><li>Lift value: Greater than 1. This ensures that the identified rules show a strong positive correlation between items, guiding effective marketing strategies.</li></ul></div><div>These parameters help focus on significant patterns in customer purchasing behaviour, filtering out noise. Using the FP-Growth and Apriori algorithms, the system can efficiently handle up to 1 million transactions, identifying key patterns to drive business decisions. The dual approach enhances the robustness of the findings through comparative analysis and results validation.</div></div>	Comparison of top 5 rules between Training and Test Sets	Rule	Support (Train)	Confidence (Train)	Lift (Train)	Support (Test)	Confidence (Test)	Lift (Test)	frequent('sausage', 'rollbacks') → frequent('other vegetables', 'whole milk')		0.01384	0.24094	2.16353	0.01342	0.22778	1.40886	frequent('other vegetables', 'rollbacks', 'whole milk') → frequent('sausage')		0.01384	0.26048	2.04954	0.01246	0.26772	1.26684	frequent('sausage', 'other vegetables') → frequent('rollbacks', 'whole milk')		0.01384	0.21538	1.92088	0.00985	0.20539	1.27847	frequent('sausage', 'sausage', 'whole milk') → frequent('other vegetables')		0.01373	0.24401	1.98044	0.01297	0.24759	1.23544	frequent('sausage', 'whole milk') → frequent('other vegetables')		0.01009	0.22822	1.91178	0.01076	0.22037	1.18663	Association Rules on Test Data (Top10)	Antecedents	Consequents	Support	Confidence	Lift	1	(pip fruit)	other vegetables	0.01718	0.31778	1.44886	2	(shopping bag)	whole milk	0.01045	0.28772	1.29688	3	(other vegetables)	whole milk	0.00865	0.25559	1.21847	4	(other vegetables)	whole milk	0.01701	0.24873	1.23544	5	(other vegetables)	whole milk	0.01076	0.22037	1.18663	6	(other vegetables)	whole milk	0.01009	0.22184	1.18773	7	(other vegetables)	whole milk	0.01127	0.22084	1.18786	8	(other vegetables)	whole milk	0.01045	0.28772	1.29688	9	(other vegetables)	whole milk	0.01076	0.22037	1.18663	10	(other vegetables)	whole milk	0.01009	0.22184	1.18773	<div>Report writing:</div> <div>4. Collaborative Filtering</div> <div>5. Recommendation Method from Frequent Patterns</div> <div>6.1 Mining Frequent Patterns</div> <div>6.2 Collaborative Filtering: user-to-user versus item-to-item</div>	<div>Anh Tu Vu</div> <div>a1911757</div>	<div>4.2 Evaluation Metrics</div> <div>We use the Mean Average Precision at k (MAP@k) and root mean square error (RMSE) to evaluate the system's performance on the test set. In the context of this project, MAP@k simply measures how well our system finds and ranks the relevant items (EvidentlyAI, 2024). Given a target user, the relevant items (ground truths) are defined as those that exist in the test set but not in the train set. In simple terms, they are the new items that user has bought in the test set which we focus on since we want to recommend new items to the user. The k term simply indicates the threshold of the top predicted items we want to evaluate. As our system produces a list of items as predictions, we can compare how many exist in the relevant list and how well they</div> <div>4 Collaborative Filtering</div> <div>4.1 Method Description</div> <div>The two most classic approaches in recommendation systems include content-based and collaborative filtering. Content-based relies solely on the items' features to suggest new recommendations for a target user. More specifically, the method analyses which items the user prefers and make suggestions based on other similar items (Das, 2023). This allows the technique to work independently without extra data needed from different user profiles.</div>
Comparison of top 5 rules between Training and Test Sets	Rule	Support (Train)	Confidence (Train)	Lift (Train)	Support (Test)	Confidence (Test)	Lift (Test)																																																																																																															
frequent('sausage', 'rollbacks') → frequent('other vegetables', 'whole milk')		0.01384	0.24094	2.16353	0.01342	0.22778	1.40886																																																																																																															
frequent('other vegetables', 'rollbacks', 'whole milk') → frequent('sausage')		0.01384	0.26048	2.04954	0.01246	0.26772	1.26684																																																																																																															
frequent('sausage', 'other vegetables') → frequent('rollbacks', 'whole milk')		0.01384	0.21538	1.92088	0.00985	0.20539	1.27847																																																																																																															
frequent('sausage', 'sausage', 'whole milk') → frequent('other vegetables')		0.01373	0.24401	1.98044	0.01297	0.24759	1.23544																																																																																																															
frequent('sausage', 'whole milk') → frequent('other vegetables')		0.01009	0.22822	1.91178	0.01076	0.22037	1.18663																																																																																																															
Association Rules on Test Data (Top10)	Antecedents	Consequents	Support	Confidence	Lift																																																																																																																	
1	(pip fruit)	other vegetables	0.01718	0.31778	1.44886																																																																																																																	
2	(shopping bag)	whole milk	0.01045	0.28772	1.29688																																																																																																																	
3	(other vegetables)	whole milk	0.00865	0.25559	1.21847																																																																																																																	
4	(other vegetables)	whole milk	0.01701	0.24873	1.23544																																																																																																																	
5	(other vegetables)	whole milk	0.01076	0.22037	1.18663																																																																																																																	
6	(other vegetables)	whole milk	0.01009	0.22184	1.18773																																																																																																																	
7	(other vegetables)	whole milk	0.01127	0.22084	1.18786																																																																																																																	
8	(other vegetables)	whole milk	0.01045	0.28772	1.29688																																																																																																																	
9	(other vegetables)	whole milk	0.01076	0.22037	1.18663																																																																																																																	
10	(other vegetables)	whole milk	0.01009	0.22184	1.18773																																																																																																																	

		<div><h2>5 Recommendation Method from Frequent Patterns</h2><h3>5.1 Recommendation From Patterns</h3><p>The pattern mining techniques discussed in Section 4 return a ranked list of rules, each of which is expressed in the form of <i>antecedents</i> <math>\rightarrow</math> <i>consequents</i>. The rule suggests that if a user buys the items in the <i>antecedents</i>, they will be likely to buy those from the <i>consequents</i> list as well. Our objective is to suggest a list of recommended items to the target user given the rules as input.</p><p>To perform this task, we go over each rule in the ranked list and compare the antecedents with the user's purchase history. If the antecedents in a rule are a subset of the user's purchased items, we add the consequents to our recommended list.</p><p>We evaluate the technique's performance on the test set using <math>MAP@k</math> as discussed in the previous section.</p><h3>5.2 Hybrid Method with Collaborative Filtering</h3><p>To combine the effects of frequent patterns and collaborative filtering, we build a hybrid system that utilises the results from both techniques. We denote <math>R1</math> as the recommended list from the frequent patterns and <math>R2</math> as the list from collaborative filtering. Consider an item <math>i</math> in our dataset, we compute the score for <math>i</math> based on the three following scenarios:</p><ul style="list-style-type: none"><li>• If <math>i</math> only exists in <math>R1</math>: <math>score(i) = 0.3</math></li><li>• If <math>i</math> only exists in <math>R2</math>: <math>score(i) = 0.7</math></li><li>• If <math>i</math> exists in both <math>R1</math> and <math>R2</math>: <math>score(i) = 0.3 + 0.7 = 1</math></li></ul><p>Finally, we combine all items from both <math>R1</math> and <math>R2</math>, then sort them based on the scores and return the top <math>k</math> items (<math>k</math> is the system's parameter).</p><p>The technique prioritises recommended item from the collaborative filtering module by assigning them with a higher weight. If an item appears in both recommended lists, they will be given a higher score in the predictions.</p><p>Based on the five frequent rules previously, we can make recommendations for users based on their purchase history. Given a rule, assuming the antecedents are the subset of a user's purchased list, we can recommend the consequents to them. Table 2 demonstrates ten examples of such recommendations.</p><p>Table 2: Ten recommendations examples based on five frequent patterns</p><table><tr><th>Rule</th><th>Bought Items</th><th>Recommendations</th></tr><tr><td>sausage, rolls/buns</td><td>rolls/buns, specialty bar, butter, butter, milk, frozen vegetables</td><td>other vegetables, whole milk</td></tr><tr><td><math>\rightarrow</math> other vegetables, whole milk</td><td>frozen dessert, sausage, cat food, coffee, canned beer, yogurt, rolls/buns, whole milk, etc.</td><td>other vegetables</td></tr><tr><td>other vegetables, rolls/buns, whole milk</td><td>other vegetables, rolls/buns, whole milk, brown bread, etc.</td><td>sausage</td></tr><tr><td><math>\rightarrow</math> sausage</td><td>other vegetables, rolls/buns, whole milk, domestic eggs, coffee, etc.</td><td>sausage</td></tr><tr><td>sausage, other vegetables</td><td>sausage, other vegetables, herbs, onions, etc.</td><td>rolls/buns, whole milk</td></tr><tr><td><math>\rightarrow</math> rolls/buns, whole milk</td><td>sausage, other vegetables, root vegetables, slice cheese</td><td>rolls/buns, whole milk</td></tr><tr><td>brown bread, whole milk</td><td>brown bread, whole milk, sausage, oil, etc.</td><td>bottled water</td></tr><tr><td><math>\rightarrow</math> bottled water</td><td>brown bread, whole milk, beverages, etc.</td><td>bottled water</td></tr><tr><td>shopping bags, whole milk</td><td>shopping bags, whole milk, meat, brown bread, etc.</td><td>canned beer</td></tr><tr><td><math>\rightarrow</math> canned beer</td><td>shopping bags, whole milk, pastry, tropical fruit</td><td>canned beer</td></tr></table><h3>6.2 Collaborative Filtering: user-to-user versus item-to-item</h3><p>Table 3 summarizes the results of the user-to-user and item-to-item approaches for our collaborative filtering module. The models are compared on the train and test sets using RMSE and <math>MAP@k</math> scores, with <math>k</math> taking the values of 1, 3, 5, and 10. We can observe that the user-to-user approach shows clear dominance on all metrics, achieving lower RMSE scores and higher <math>MAP@k</math> values. As a result, we choose this approach for our hybrid models in the next section.</p><p>Table 4: User-to-user and item-to-item comparison</p><table><tr><th>Dataset</th><th>Metrics</th><th>User-to-user</th><th>Item-to-item</th></tr><tr><td rowspan="5">Train</td><td>RMSE</td><td>0.2913</td><td>0.3484</td></tr><tr><td><math>MAP@1</math></td><td>0.4964</td><td>0.0220</td></tr><tr><td><math>MAP@3</math></td><td>0.3937</td><td>0.0171</td></tr><tr><td><math>MAP@5</math></td><td>0.2992</td><td>0.0171</td></tr><tr><td><math>MAP@10</math></td><td>0.3075</td><td>0.0085</td></tr><tr><td rowspan="5">Test</td><td>RMSE</td><td>0.1831</td><td>0.2506</td></tr><tr><td><math>MAP@1</math></td><td>0.1747</td><td>0.0130</td></tr><tr><td><math>MAP@3</math></td><td>0.1466</td><td>0.0103</td></tr><tr><td><math>MAP@5</math></td><td>0.1446</td><td>0.0105</td></tr><tr><td><math>MAP@10</math></td><td>0.1748</td><td>0.0153</td></tr></table></div>	Rule	Bought Items	Recommendations	sausage, rolls/buns	rolls/buns, specialty bar, butter, butter, milk, frozen vegetables	other vegetables, whole milk	$\rightarrow$ other vegetables, whole milk	frozen dessert, sausage, cat food, coffee, canned beer, yogurt, rolls/buns, whole milk, etc.	other vegetables	other vegetables, rolls/buns, whole milk	other vegetables, rolls/buns, whole milk, brown bread, etc.	sausage	$\rightarrow$ sausage	other vegetables, rolls/buns, whole milk, domestic eggs, coffee, etc.	sausage	sausage, other vegetables	sausage, other vegetables, herbs, onions, etc.	rolls/buns, whole milk	$\rightarrow$ rolls/buns, whole milk	sausage, other vegetables, root vegetables, slice cheese	rolls/buns, whole milk	brown bread, whole milk	brown bread, whole milk, sausage, oil, etc.	bottled water	$\rightarrow$ bottled water	brown bread, whole milk, beverages, etc.	bottled water	shopping bags, whole milk	shopping bags, whole milk, meat, brown bread, etc.	canned beer	$\rightarrow$ canned beer	shopping bags, whole milk, pastry, tropical fruit	canned beer	Dataset	Metrics	User-to-user	Item-to-item	Train	RMSE	0.2913	0.3484	$MAP@1$	0.4964	0.0220	$MAP@3$	0.3937	0.0171	$MAP@5$	0.2992	0.0171	$MAP@10$	0.3075	0.0085	Test	RMSE	0.1831	0.2506	$MAP@1$	0.1747	0.0130	$MAP@3$	0.1466	0.0103	$MAP@5$	0.1446	0.0105	$MAP@10$	0.1748	0.0153
Rule	Bought Items	Recommendations																																																																					
sausage, rolls/buns	rolls/buns, specialty bar, butter, butter, milk, frozen vegetables	other vegetables, whole milk																																																																					
$\rightarrow$ other vegetables, whole milk	frozen dessert, sausage, cat food, coffee, canned beer, yogurt, rolls/buns, whole milk, etc.	other vegetables																																																																					
other vegetables, rolls/buns, whole milk	other vegetables, rolls/buns, whole milk, brown bread, etc.	sausage																																																																					
$\rightarrow$ sausage	other vegetables, rolls/buns, whole milk, domestic eggs, coffee, etc.	sausage																																																																					
sausage, other vegetables	sausage, other vegetables, herbs, onions, etc.	rolls/buns, whole milk																																																																					
$\rightarrow$ rolls/buns, whole milk	sausage, other vegetables, root vegetables, slice cheese	rolls/buns, whole milk																																																																					
brown bread, whole milk	brown bread, whole milk, sausage, oil, etc.	bottled water																																																																					
$\rightarrow$ bottled water	brown bread, whole milk, beverages, etc.	bottled water																																																																					
shopping bags, whole milk	shopping bags, whole milk, meat, brown bread, etc.	canned beer																																																																					
$\rightarrow$ canned beer	shopping bags, whole milk, pastry, tropical fruit	canned beer																																																																					
Dataset	Metrics	User-to-user	Item-to-item																																																																				
Train	RMSE	0.2913	0.3484																																																																				
	$MAP@1$	0.4964	0.0220																																																																				
	$MAP@3$	0.3937	0.0171																																																																				
	$MAP@5$	0.2992	0.0171																																																																				
	$MAP@10$	0.3075	0.0085																																																																				
Test	RMSE	0.1831	0.2506																																																																				
	$MAP@1$	0.1747	0.0130																																																																				
	$MAP@3$	0.1466	0.0103																																																																				
	$MAP@5$	0.1446	0.0105																																																																				
	$MAP@10$	0.1748	0.0153																																																																				
Report writing:  1. Executive Summary  6.3 Hybrid System  6.4 Execution Time Evaluation  7. Conclusion and Recommendations  8. Reflection  9. Refactor documents	Quy Duong Dang a1893592																																																																						



## Executive Summary

Understanding customers' behaviours is a critical factor in increasing profits. Grocery stores typically save their customers' purchase history in the database, which can be utilised to devise selling strategies. For instance, if a store manager knows which items his customers prefer, he can make the appropriate suggestions that tempt them to buy more. In this project, we perform data mining techniques to explore which items are usually purchased together. In addition, we also build a collaborative filtering model to predict a customer's preferences, thereby recommending the relevant items to them. Finally, we combine the two modules into a hybrid system that outperforms all the individual modules and reasonably scales with the data over time. The system can help grocery stores improve sales by learning customers' purchase behaviours.

## 6.3 Hybrid System

Our hybrid model consists of collaborative filtering and data pattern mining modules. The former is built using the user-to-user approach, while the latter will be tested using both the **Apriori** and FP-Growth algorithms. When run separately, we will compare our hybrid system with the **Apriori**, FP-Growth, and user-to-user models to see whether it has made any significant **improvement**.

Table 5 summarises the performance of the models. First, FP-Growth seems to perform worse than **Apriori** as the value of k increases. Nonetheless, both models show no significant difference in performance. Secondly, the hybrid model with user-to-user and FP-Growth achieves the best results on the test set, even though the individual user-to-user model outperforms it on the train set.

Table 5: Hybrid system comparison

Dataset	Metric	Apriori	FP-Growth	User-to-user	User-to-user + Apriori	User-to-user + FP-Growth
Train	MAP@1	0.1415	0.2629	<b>0.4964</b>	<b>0.4964</b>	<b>0.4964</b>
	MAP@3	0.1190	0.1445	<b>0.3937</b>	0.3892	0.3747
	MAP@5	0.1157	0.0981	<b>0.2992</b>	0.2840	0.2818
	MAP@10	0.1220	0.0555	<b>0.3075</b>	0.2989	0.3000
Test	MAP@1	0.0709	0.1258	<b>0.1747</b>	<b>0.1747</b>	<b>0.1747</b>
	MAP@3	0.0722	0.0718	0.1466	0.1456	<b>0.1476</b>
	MAP@5	0.0737	0.0551	0.1446	0.1346	<b>0.1462</b>
	MAP@10	0.0992	0.0667	<b>0.1748</b>	<b>0.1748</b>	<b>0.1748</b>

## 6.4 Execution Time Evaluation

Even though the hybrid system of user-to-user and FP-Growth produces the best results, it does not scale very well with the data. FP-Growth is very memory intensive and slow when the number of records is scaled to one million, especially when we have lots of distinct values in the dataset. Initial experiment also shows that the hybrid system runs very slowly with one million transactions. Due to insufficient hardware resources, we have decided to switch to user-to-user as the main model since its performance difference is insignificant while scaling better with the data size.

To see how the system's execution time scales with the sample size, we generate dummy data with 10,000, 100,000, and 1,000,000 transactions. Each size is measured with k values of 1, 3, 5, and 10. The experiment was performed on a MacBook Pro with an M3 Max chip as follows:

- 14-core CPU with ten performance cores and four efficiency cores
- 30-core GPU
- 16-core Neural Engine
- 300GB/s memory bandwidth.

Table 6: Inference time comparison between different sample sizes (in seconds).

	size = 10000	size = 100000	size = 1000000
MAP@1	0.1510	0.3881	1.0432
MAP@3	0.1509	0.4063	1.0255
MAP@5	0.1507	0.4045	1.0465
MAP@10	0.1491	0.4141	1.0384

Table 6 shows that the user-to-user approach scales very well with the data. For one million transactions, the system only takes roughly one second to produce a recommendation. As the data size increases by 10 times, the executed time only takes roughly 2.5 – 2.7 times longer. It is also worth noting that the k value does not affect the system's performance. We can conclude that our system is fast enough to be implemented in real-life scenarios when the number of transactions grows quickly.

## 7. Conclusion and Recommendations

The results from our experiment suggest that the hybrid system combining the user-to-user and FP-Growth produces the best results. However, its improved performance is negligible while the executed time scales very poorly with the data size. Therefore, we suggest that the user-to-user model is the best candidate for our system.

The recommendation system is only a prototype and still needs further work to improve. First, the system must consider the common cold-start problem where new users shop at the store. The system cannot make relevant

recommendations for these users since their purchase history is outside the database. Secondly, we must address the data sparsity problem, which occurs when users only purchase a few unique items. One solution is to group similar items to reduce the total number of items. For example, items such as "root vegetables", "other vegetables", and "frozen vegetables" can be grouped as "vegetables". We can apply a pre-trained model to embed the item names and user cosine similarity to detect similar items for grouping. Finally, we can explore some state-of-the-art techniques using neural networks to boost the performance of our current system.

## 8. Reflection

In this project, we have learned how to work as a group to develop a recommendation system by dividing and completing the tasks simultaneously. The tasks involve implementing a data mining pattern, building a collaborative filtering system, and combining the two to create a final hybrid system. The main challenges in this project were researching the method to integrate our modules for the hybrid system and developing the proper metrics to evaluate its performance. The main area we need to improve next time is our system's performance since its metrics are currently showing relatively poor performance. In addition, we also need to consider exceptional cases like the previous cold-start problem and accommodate new users joining the system.