# Association rule/pattern mining for recommender system report

Group 6:

Yipeng Ma (a1846354)

Kaini Chang (a1866621)

Lin Cen Lin Jia Yi (a1807559)

# 1. Executive summary

This report outlines a recommendation system for a grocery store that can identify frequently bought item sets from customer transaction data and generate personalized recommendations for customers. The project uses frequent data mining and recommendation systems. By strategically placing frequently bought items close together and providing personalized recommendations, the system can help customers find what they need quickly and efficiently. Implementing this system can provide significant benefits for the company, including increased customer satisfaction, improved customer loyalty, and ultimately increase sales revenue for the grocery store. The system is scalable to one million user transactions, making it suitable for handling increasing volumes of data in the future. Our overall results show that both recommendation using frequent patterns method and recommendation using data method have low performance and accuracy. This might be due to the small dataset, so further actions are recommended to improve the recommendation system.

# 2. Introduction

Recommendation systems have transformed online businesses by providing personalized recommendations that increase customer engagement and satisfaction. The purpose of this report is to present a data mining project aimed at developing a recommendation system for grocery stores using transaction data from a loyalty program for the stores. Our primary objective was to increase sales by providing relevant product recommendations to customers to increase sales. In order to accomplish this, we explored various pattern extraction and recommendation techniques, ultimately implementing a recommendation system based on collaborative filtering and frequent patterns. The performance of both methods was evaluated using precision, recall, and F1 score metrics.

In this report, we provide a detailed description of the data mining process, including the methods used to extract patterns and make recommendations. We present experimental results, including examples of frequent patterns, recommendations from these patterns, and their evaluation metrics.

Our findings suggest that while the recommendation system with frequent patterns is not very accurate, the collaborative filtering method performs better, recommending relevant items with a higher degree of precision. We also estimated the system timing if the dataset was scaled up to one million transactions. This shows that our system can handle large datasets efficiently.

Overall, this report highlights how data-driven solutions can increase sales in grocery stores. We provide insights and recommendations that can help businesses achieve similar success in their operations, but further improvement is needed due to the limitation of the present dataset.

# 3. Exploratory analysis

The project's train and test datasets have similar structures, but the test dataset is smaller than the train dataset. Both datasets contain sales transactions for a retail shop from January 2014 to December 2015, involving 5,000 members who made purchases during that time, each with a unique ID between 1000 and 5000. This dataset provides valuable insights into consumer purchase behaviours and trends. The following columns are included in the dataset:

- Member_number: unique identifier is assigned to each customer.
- Date: the date on which the transaction occurred
- itemDescription: the name of the item that was bought
- year: the year in which the transaction occurred
- month: the month in which the transaction occurred
- day: the day in which the transaction occurred
- day_of_week: the day of the week the transaction occurred, represented as a number from 0 (Monday) to 6 (Sunday).

In the data visualization process, it is evident that whole milk, various vegetables, rolls/buns, soda, and yoghurt are some of the most purchased items. These products have consistently high annual sales volumes, indicating a steady demand for them. Sales of seasonal goods, such as tropical fruits and root vegetables, are higher in the fall and winter, indicating that people are more likely to purchase these items when they are in season and the weather is cooler. Additionally, sales of other products see sporadic upswings, including seasonal and specialized goods, such as prosecco and specialty chocolate. These increases may reflect demand during holidays or special events.

Further analysis of the dataset revealed potential issues with the long tail and cold start problems that can arise in recommendation systems and data analysis. The dataset contains records of purchases made by 3,872 unique users and 167 unique items that have been bought. While each user has made at least one purchase, many items have not been purchased yet. Some items have been purchased frequently by many users, while others have been purchased only a few times. As the data visualization output shows, there is a long tail distribution that can make it challenging to make accurate recommendations for niche items with limited purchase data. Techniques like collaborative filtering and content-based filtering can be used to address this problem by leveraging additional information about the items and users.

Moreover, since there are only 167 unique items in the dataset, it is likely that some items have very few purchases, while others are very popular. This represents a cold start problem with the dataset, where there may not be enough data about the preferences of new users to make accurate recommendations. One way to address this issue is to use pattern-based recommendation methods or carefully use collaborative filtering techniques.

Finally, the structure of the dataset is different from a typical grocery store receipt, where all items bought in a single transaction are listed together. Instead, each row in both the train and test data represents a single item purchased. This is unusual since customers typically purchase multiple items in one transaction. In the train data, we can see duplicate

Member_numbers with the same Date, indicating that transactions are stored as single rows with one item per row. This makes it challenging to extract complete basket sets, as we need to identify all the items purchased by a customer in a single transaction. To address this issue, the “groupby” method can be used to group the data by Member_number and Date. However, we noticed that in the test data, Member 1000 also bought whole milk and semi-finished bread on 15/03/2015. This presents a potential problem because even though we used “groupby” on both the train and test data, the transactions are still incomplete since they are split into two separate datasets, which could result in a sample size that is too small to draw meaningful insights or may not accurately represent all customer purchasing behaviours.

Here is a diagram of the proposed system that we have designed for implementation:

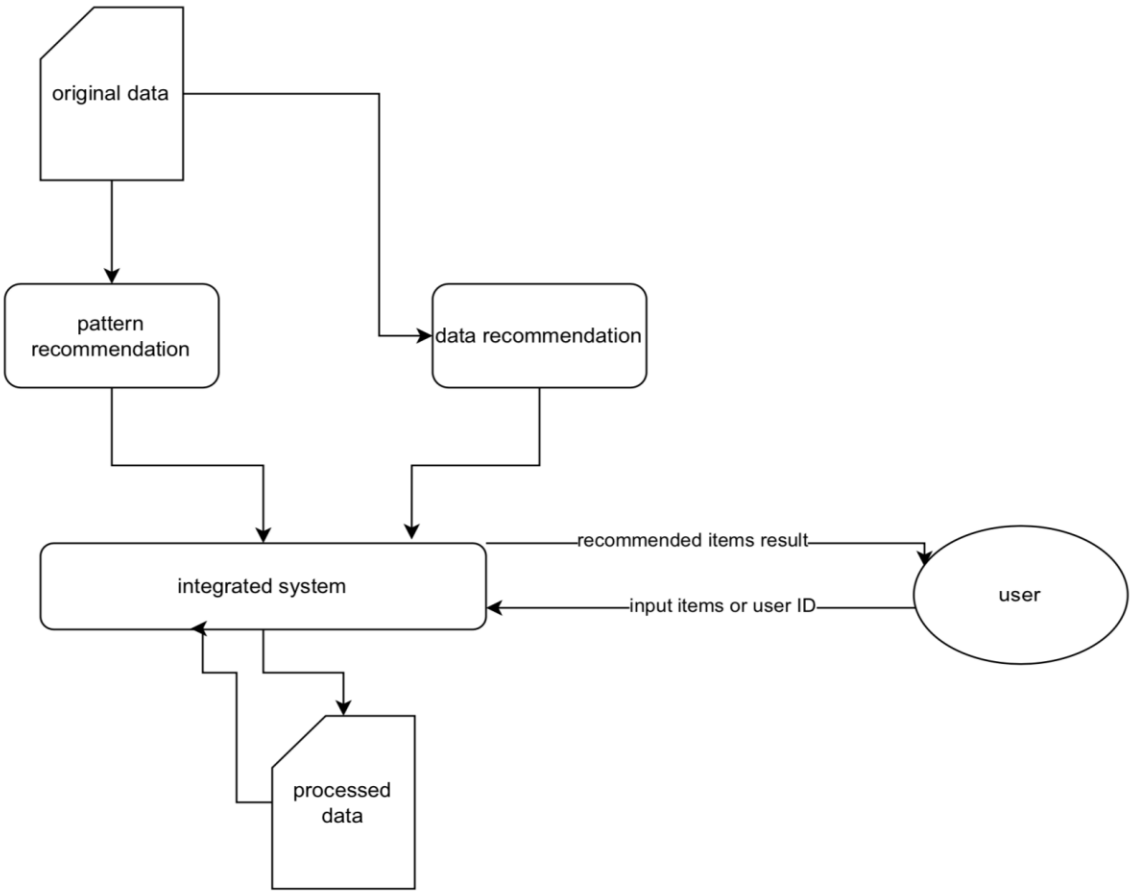


Figure 1. Diagram of the proposed system

For addressing the potential problems that we found, such as long tail and cold start issues, the pattern mining approach can be applied to solve these problems. However, if a user hasn’t purchased anything for a current blanket, we consider using the collaborative filtering approach.

# 4. Frequent pattern mining

A method called frequent itemset mining uses groupings of items that frequently appear together to look for patterns in transactional data. The Apriori algorithm was utilised in this project as a common itemset mining strategy to create candidate item sets of increasing length and remove the item sets that did not satisfy the minimal support criterion. This method was chosen because association rule mining and market basket analysis activities commonly use it. It also offers simple implementation, and there are numerous libraries that offer user-friendly Apriori implementations in well-known programming languages like Python.

Step 1. Generate association rules with Apriori algorithm. The TransactionEncoder module was used to convert the transaction data into a binary matrix format, and after that, the Apriori algorithm was used to produce frequent item sets and association rules based on the support and confidence values of the transactions. Based on the specific project goals and the features of the dataset, its hyperparameters were chosen. Apriori's two hyperparameters are use_colnames and min_support. The value of 0.001 was chosen to ensure that only frequent item sets are considered, which could help in identifying the most well-liked items and their links. The parameter min_support determines the minimum support threshold for an item set to be considered frequent. The use_colnames argument was also set to True to improve output readability, which is helpful when presenting the results.

Furthermore, to ensure that only strong relationships between item sets are taken into account, the minimum confidence criterion of 0.05 was chosen for association rules. This indicates that a rule must be true for at least 5% of transactions containing the antecedent in order to be considered strong. Then the association rules are ready to be used to predict possible items that a member might purchase.

Step 2. Extract purchased item set. We define the extract_sets() function to extract the test and train sets for the current sample. The function extracts the shopping trip ID (Combination of member number and the transaction date) , the purchased items from the test set, and the item descriptions from the train set using the extracted shopping trip ID as the key. If the shopping trip is present in the train dataset, the function splits the item descriptions into a list of items for both datasets. If the shopping trip ID is not present in the train dataset, the function returns an empty train set. The function then returns the purchased item set from train and test as lists.

Step 3. Generate predict item list using the association rules generated. To obtain the results, we used the predict_items() function to generate a list of predicted items based on the test set or train set. The predict_items() function takes a single argument, purchased, which is a set of items that the user has already purchased, it can be test set or train set. The function iterates over each row of the rules from train data, which contains the association rules that were generated during the training phase of the algorithm. For each row, the antecedent and consequent items are extracted from the "antecedents" and "consequents" columns, respectively. The function checks if the antecedent items are a subset of the purchased items. If they are, then the consequent items are added to the predicted items list. This is because the association rule suggests that customers who purchase the antecedent items are likely to purchase the consequent items as well. Once all the association rules have been checked, the predicted items list contains the predicted items that the user is likely to purchase based on the association rules. Finally, the

function formats the predicted items into a comma-separated string using the join() method and returns the predicted items as a list.

# 5. Collaborative filtering

## 5.1 Brief description of the method

We implement the collaborative filtering method based on user similarity by following the steps showed below:

Step 1. Extract the shopping history for the given user. This step is necessary because we need to know what items the user has already purchased so that we can recommend new items that the user has not yet bought.

Step 2. Group the transaction history by the member number and shopping date. This step is necessary to create a matrix where rows represent the users and columns represent the items, with the cell values representing how many times a user has bought a particular item.

Step 3. Calculate the cosine similarity between the given user and all other users. This step is necessary to find the users who are most similar to the given user based on their purchase history. By identifying the most similar users, we can use their purchase history to recommend new items to the given user. To calculate cosine similarity between two members, we first represent their purchase histories as vectors. Each item is a feature and the number of times an item has been purchased by a member is the corresponding value in the vector(image1). Then, we take the dot product of the two vectors and divide it by the product of their magnitudes. This gives us the cosine of the angle between the two vectors, which is a measure of similarity.

| Member_number | Instant food products | UHT-milk | abrasive cleaner | artif. sweetener | baby cosmetics | bags | baking powder | bathroom cleaner | beef | berries | beverages | bottled beer | bottled |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 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 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | |
| 1002 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 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 | 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 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 1005 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 1006 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | |
| 1008 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 1009 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 1010 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |

Table 1. Vectors visualized by spreadsheet

A cosine similarity of 1 means the vectors are identical, while a cosine similarity of 0 means the vectors are orthogonal (have no similarity).

Step 4. Extract the groceries that the given user has not purchased. This step is necessary to exclude the items that the user has already bought from the recommended list.

Step 5. Calculate the weighted average of the ratings of the top 10 most similar users for each grocery. This step is necessary to give more weight to the purchase history of users

who are more similar to the given user. By doing this, we ensure that the recommendations are personalized to the given user.

Step 6. Scale the scores to a range between 0 and 1. This step is necessary to make sure that the scores for different items are comparable and fall within a common range. By scaling the scores, we ensure that the items with the highest scores are the most recommended ones.

Step 7. Sort the recommendations by the scaled score in descending order and return the recommendation with a score greater than 0.5. This step is necessary to give the user a manageable number of recommendations and ensure that the most highly recommended items are presented first. By returning only the high score recommendations, we also reduce the user's cognitive load and make it easier for them to make a decision.

### 5.2 Metrics used for evaluation on test dataset

To evaluate the recommendations from collaborative filtering method, we choose

Precision, recall, and F1 score to compare the recommended item set and actual purchased item set.

Precision, recall, and F1 score are commonly used evaluation metrics for recommendation systems. These metrics are typically used in a binary classification context, where the system is trying to classify whether a user will be interested in an item or not.

Precision: Precision is the percentage of items that are truly purchased in the test set by the user among all recommended items.

Recall: Recall is the ratio of the number of items correctly recommended to the total number of items the user purchased in the test set.

F1 score: F1 score is the harmonic mean of precision and recall. It provides a single metric that balances both precision and recall.

## 6. Recommendation method from frequent patterns.

### 6.1 Brief description of the method

Step 1. Initialize an empty list for predicted items. This list will be used to store the consequents of the rules that are triggered by the purchased items.

Step 2. Iterate over the rules we extracted from the train data set to check which rules are triggered by the purchased items.

Step 3. Get the antecedent and consequent items from the current tested rule of all rules extracted from the test data set.

Step 4. Check if the antecedent is a subset of the purchased items. This step checks if the antecedent items from the current rule are present in the purchased items list. If the antecedent is a subset of the purchased items, then the rule is triggered.

Step 5. If the antecedent is present in the purchased items, add the consequent to the predicted items list. If the antecedent is present in the purchased items list, it means that the rule is triggered, and the consequent items are added to the predicted items list.

Step 6. Format the predicted items list: The predicted items list is formatted to display the items as a comma-separated string.

Step 7. Return the predicted items list: The function returns the predicted items list, which contains the consequent items of all the rules that were triggered by the purchased items.

### 6.2 Association Rules as Input: Understanding the Method and Output

The steps above allow us to identify which rules are relevant to a given set of purchased items, and then use those rules to make recommendations for additional items that the user may be interested in purchasing(figure2). By iterating over the rules and checking if the antecedent is a subset of the purchased items, we can identify which rules are relevant. By adding the consequent items to a list, we can make recommendations for those items. Finally, by formatting the list, we can display the recommended items in a user-friendly format.

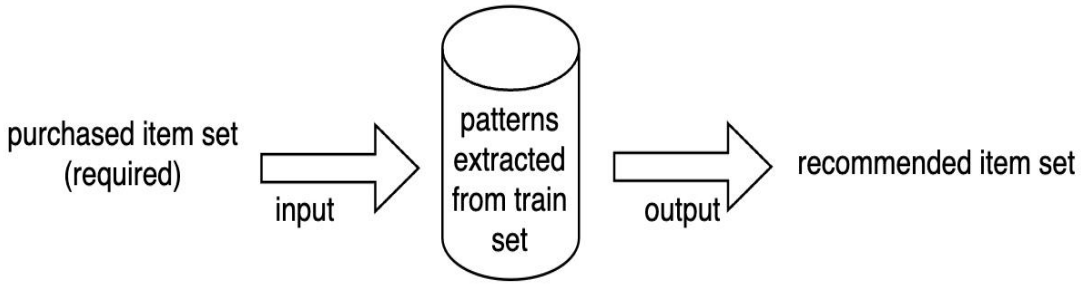


Figure 2. input and output of recommendation based on frequent patterns

## 7. Experimental results

### 7.1 Obtaining Results

Functions from the mlxtend library, such as ‘TransactionEncoder’, ‘apriori’ and ‘association_rules’ are used to get the ideal format of association rules, which has been explained in session 4.

Functions from the pandas library, such as ‘to_csv’ are used to export all the medium and final results which we gained through the above recommendation methods.

### 7.2 Evaluation Metrics and Ranking of Patterns and Recommendations

Patterns and recommendations were ranked using metrics: precision, recall, f1 score. Since we’re using the same metrics for both the two recommendation methods (frequent

pattern and collaborative filtering), the part has been explained in session 5.2 works for both.

The recommendation with a higher  f1 score is ranked as a better result.

## 7.3 Frequent Patterns: Examples and Metrics on Training and Test Sets

Here are five examples of frequent patterns with their confidence and support on both training and test sets.

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction | zhangs_metric |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | (UHT-milk) | (other vegetables) | 0.015034 | 0.092792 | 0.001223 | 0.081340 | 0.876577 | -0.000172 | 0.987533 | -0.125071 |
| 1 | (UHT-milk) | (tropical fruit) | 0.015034 | 0.048410 | 0.001007 | 0.066986 | 1.383706 | 0.000279 | 1.019909 | 0.281536 |
| 2 | (UHT-milk) | (whole milk) | 0.015034 | 0.118041 | 0.001583 | 0.105263 | 0.891754 | -0.000192 | 0.985719 | -0.109717 |
| 3 | (beef) | (whole milk) | 0.026399 | 0.118041 | 0.002446 | 0.092643 | 0.784841 | -0.000670 | 0.972009 | -0.219711 |
| 4 | (berries) | (other vegetables) | 0.016616 | 0.092792 | 0.001079 | 0.064935 | 0.699789 | -0.000463 | 0.970208 | -0.303743 |

Table 2. Examples of patterns from Train sets

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction | zhangs_metric |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | (beef) | (whole milk) | 0.016526 | 0.089013 | 0.001140 | 0.068966 | 0.774780 | -0.000331 | 0.978468 | -0.228141 |
| 1 | (bottled water) | (other vegetables) | 0.030317 | 0.065307 | 0.001596 | 0.052632 | 0.805915 | -0.000384 | 0.986621 | -0.198945 |
| 2 | (frankfurter) | (rolls/buns) | 0.019261 | 0.058924 | 0.001026 | 0.053254 | 0.903780 | -0.000109 | 0.994011 | -0.097924 |
| 3 | (frozen vegetables) | (root vegetables) | 0.013563 | 0.038409 | 0.001026 | 0.075630 | 1.969080 | 0.000505 | 1.040267 | 0.498915 |
| 4 | (other vegetables) | (whole milk) | 0.065307 | 0.089013 | 0.003305 | 0.050611 | 0.568578 | -0.002508 | 0.959551 | -0.448059 |

Table 3. Examples of patterns from Test sets

As we can see from the examples, training and test sets share some common patterns. However, it's hard to tell by the eyes. Therefore, the testing patterns we extract from the train set on the test set is necessary, which is shown in the 7.5 session.

## 7.4 Recommendation Examples from Frequent Patterns

| Train set | Test set | Predicted items | Precision | Recall | FI |
|---|---|---|---|---|---|
| ['UHT-milk'] | ['other vegetables'] | ['other vegetables', 'tropical fruit', 'whole milk'] | 0.333333 | 1 | 0.5 |
| ['UHT-milk'] | ['tropical fruit'] | ['other vegetables', 'tropical fruit', 'whole milk'] | 0.333333 | 1 | 0.5 |
| ['UHT-milk'] | ['tropical fruit'] | ['other vegetables', 'tropical fruit', 'whole milk'] | 0.333333 | 1 | 0.5 |
| ['UHT-milk'] | ['whole milk'] | ['other vegetables', 'tropical fruit', 'whole milk'] | 0.333333 | 1 | 0.5 |
| ['beef'] | ['whole milk'] | ['whole milk'] | 1 | 1 | 1 |
| ['beef'] | ['whole milk'] | ['whole milk'] | 1 | 1 | 1 |
| ['berries'] | ['other vegetables'] | ['other vegetables', 'whole milk'] | 0.5 | 1 | 0.6667 |
| ['berries'] | ['other vegetables'] | ['other vegetables', 'whole milk'] | 0.5 | 1 | 0.6667 |
| ['beef'] | ['whole milk'] | ['whole milk'] | 1 | 1 | 1 |
| ['beef'] | ['whole milk'] | ['whole milk'] | 1 | 1 | 1 |
| ['bottled water'] | ['other vegetables'] | ['other vegetables', 'rolls/buns', 'soda', 'whole milk'] | 0.25 | 1 | 0.4 |
| ['bottled water'] | ['other vegetables'] | ['other vegetables', 'rolls/buns', 'soda', 'whole milk'] | 0.25 | 1 | 0.4 |
| ['frankfurter'] | ['rolls/buns'] | ['other vegetables', 'rolls/buns', 'soda', 'whole milk'] | 0.25 | 1 | 0.4 |
| ['frankfurter'] | ['rolls/buns'] | ['other vegetables', 'rolls/buns', 'soda', 'whole milk'] | 0.25 | 1 | 0.4 |
| ['frozen vegetables'] | ['root vegetables'] | ['other vegetables', 'rolls/buns', 'root vegetables', 'sausage', 'soda', 'whole milk'] | 0.166667 | 1 | 0.2857 |
| ['whole milk'] | ['other vegetables'] | ['other vegetables', 'rolls/buns', 'soda', 'yogurt'] | 0.25 | 1 | 0.4 |
| ['whole milk'] | ['other vegetables'] | ['other vegetables', 'rolls/buns', 'soda', 'yogurt'] | 0.25 | 1 | 0.4 |

Table 4. Examples of recommendations from above patterns

## 7.5 Testing Frequent Patterns: Metrics Table for Results

| antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | precision |
|---|---|---|---|---|---|---|---|---|
| **frozenset({'beef'})** | frozenset({'whole milk'}) | 0.0165261 | 0.089012993 | 0.001139731 | 0.068965517 | 0.774780344 | -0.000331307 | 0.089012993 |
| **frozenset({'bottled water'})** | frozenset({'other vegetables'}) | 0.030316845 | 0.065306588 | 0.001595623 | 0.052631579 | 0.805915312 | -0.000384266 | 0.065306588 |
| **frozenset({'frankfurter'})** | frozenset({'rolls/buns'}) | 0.019261454 | 0.058924094 | 0.001025758 | 0.053254438 | 0.903780344 | -0.000109206 | 0.058924094 |
| **frozenset({'frozen vegetables'})** | frozenset({'root vegetables'}) | 0.013562799 | 0.038408935 | 0.001025758 | 0.075630252 | 1.96907962 | 0.000504825 | 0.038408935 |
| **frozenset({'other vegetables'})** | frozenset({'whole milk'}) | 0.065306588 | 0.089012993 | 0.00330522 | 0.05061082 | 0.568577896 | -0.002507915 | 0.089012993 |
| **frozenset({'pastry'})** | frozenset({'whole milk'}) | 0.025757921 | 0.089012993 | 0.001367677 | 0.053097345 | 0.5965123 | -0.000925112 | 0.089012993 |
| **frozenset({'tropical fruit'})** | frozenset({'whole milk'}) | 0.039548667 | 0.089012993 | 0.002051516 | 0.051873199 | 0.582759855 | -0.001468829 | 0.089012993 |

Table 5. Testing frequent patterns of train set on the test set

As we can see from the result, the precision of the rules generated from the test set is calculated using the frequent item sets from the train set. For each rule, the precision is calculated as the support of the consequent in the test set if the antecedent of the rule is present in the train set, otherwise, the precision is set to 0. After adjusting the parameters of the Apriori approach, the current result is the best result we gained. Even though it doesn't seem great, we continue to use it anyway.

## 7.6 Frequent Pattern-Based Recommendations: Metrics Table

Here are 10 examples of recommendations from frequent patterns:

| recommended set | actual set | score_precision | score_recall | score_f1 |
|---|---|---|---|---|
| ['other vegetables', 'other vegetables', 'rolls/buns', 'rolls/buns', 'soda', 'whole milk', 'sausage', 'yogurt', 'whole milk'] | ['whole milk', 'semi-finished bread'] | 0.1111 | 0.5000 | 0.1818 |
| ['rolls/buns', 'soda', 'whole milk'] | ['salty snack', 'whole milk'] | 0.3333 | 0.5000 | 0.4000 |
| ['other vegetables', 'rolls/buns', 'soda', 'whole milk', 'yogurt'] | ['hygiene articles'] | 0.0000 | 0.0000 | 0.0000 |
| ['other vegetables', 'other vegetables', 'rolls/buns', 'soda', 'whole milk', 'whole milk', 'yogurt'] | ['frankfurter'] | 0.0000 | 0.0000 | 0.0000 |
| ['other vegetables', 'rolls/buns', 'root vegetables', 'sausage', 'soda', 'whole milk'] | ['other vegetables'] | 0.1667 | 1.0000 | 0.2857 |
| [] | ['specialty chocolate'] | 0.0000 | 0.0000 | 0.0000 |
| [] | ['frozen meals'] | 0.0000 | 0.0000 | 0.0000 |
| ['other vegetables', 'rolls/buns', 'soda', 'whole milk', 'yogurt'] | ['rolls/buns'] | 0.2000 | 1.0000 | 0.3333 |
| ['other vegetables', 'soda', 'whole milk', 'yogurt'] | ['rolls/buns'] | 0.0000 | 0.0000 | 0.0000 |
| ['other vegetables', 'rolls/buns', 'soda', 'yogurt'] | ['packaged fruit/vegetables', 'chocolate', 'rolls/buns'] | 0.2500 | 0.3333 | 0.2857 |

Table 6. Examples of recommendations from frequent patterns

Based on our testing results, we found that the average precision of the system was only about 3.958%, meaning that only a small fraction of the recommended items were actually purchased by customers. On the other hand, the average recall showed that the system was able to recommend 13.96% of relevant items. By combining the precision and recall metrics, we calculated an F1 score of 5.733%, indicating a low overall accuracy in recommending items to customers. This suggests that the system may require further improvement to enhance its accuracy and effectiveness.

| Metric | Value |
|---|---|
| Average Precision | 0.03958 |
| Average Recall | 0.13955 |
| Average F1 Score | 0.05733 |

Table 7. Average metrics results

## 7.7 Data-Based Recommendations: Metrics Table

Here are 5 examples of recommendations from data:

| recommended set | actual set | precision | recall | f1_score |
|---|---|---|---|---|
| {'whole milk', 'soda'} | {'whole milk', 'onions', 'cling film/bags', 'long life bakery product'} | 0.5 | 0.25 | 0.33333 |
| {'domestic eggs', 'yogurt', 'rolls/buns'} | {'yogurt', 'soda'} | 0.333333 | 0.5 | 0.4 |
| {'rolls/buns', 'other vegetables'} | {'photo/film', 'beef', 'domestic eggs', 'root vegetables', 'ice cream', 'soda', 'flour', 'pip fruit', 'fruit/vegetable juice', 'whole milk', 'frankfurter', 'whipped/sour cream', 'other vegetables'} | 0.5 | 0.077 | 0.13333 |
| {'whole milk', 'soda'} | {'UHT-milk', 'pip fruit', 'semi-finished bread', 'margarine'} | 0 | 0 | 0 |
| {'other vegetables', 'root vegetables'} | {'bottled beer', 'tropical fruit', 'pip fruit', 'softener', 'pastry', 'other vegetables'} | 0.5 | 0.167 | 0.25 |

Table 8. Examples of recommendations from data

We then used the same metrics as the recommendation with frequent patterns method: average precision, average recall and average F1 score.

| Metric | Value |
|---|---|
| Average Precision | 0.10497 |
| Average Recall | 0.07459 |
| Average F1 score | 0.07559 |

Table 9. Average metrics results

Based on the results, the average precision of the recommendation system is 0.10497, meaning 10.497% of the recommended items were actually purchased by customers. The average recall of 0.07459 means that the system was able to recommend 7.459% of relevant items. Then, we calculated an F1 score of 7.559%, indicating a low overall accuracy in recommending items to customers. Even though it's still low, the F1 score is higher than that of recommendation with frequent patterns used (5.733%).

### 7.8 System Timing Estimation

Our system time estimation shows that the Apriori algorithm used for pattern mining is the main determinant of the overall run time of the recommendation system. The experiments we conducted demonstrate that the system takes an average of 0.095882 seconds to recommend products to users based on their input. The majority of the time taken is for pattern mining, which takes an average of 1.387622 seconds to generate frequent item sets and association rules. Therefore, the overall system time is 1.483504 seconds.

To test the scalability of the system, the team duplicates the original dataset to create train sets that are two and four times larger. The time taken for pattern mining is recorded for each set. Using linear regression, the team predicts that the system will take around 59.45 seconds to process one million records.

| Data scale | 27000 | 54000 | 108000 | 162000 |
|---|---|---|---|---|
| Time used | 1.4 | 2.7 | 6.0 | 9.4 |
| Predicted value of time used fro scale 1 milion | | | | |
| 59.4496547394852 | | | | |

Table 10. Values used for predicted value of time used

Additionally, we have implemented a personalized recommendation function that uses collaborative filtering and frequent patterns to generate recommendations. However, the time taken for this function is negligible compared to the time taken for pattern mining.

Overall, our recommendation system is efficient for small to medium-sized datasets, but may require optimization for larger datasets. Factors such as hardware and data structure can also affect the actual run time of the system.

## 8. Conclusion and Recommendations

Based on the entire analysis, it is concluded that the dataset is too small and incomplete to draw meaningful insights or accurately represent all customer purchasing behaviours, even though the Apriori algorithm is known as a highly effective frequent pattern mining method for small datasets. Considering the benefits for the company, it is recommended that actions be taken to improve the recommendation system in the future.

One recommendation is to explore other recommendation methods such as content-based filtering or hybrid approaches to improve the system's accuracy. These methods take into account user preferences and item characteristics, which can provide more accurate and personalised recommendations.

Moreover, further development could include incorporating more detailed user preference and demographic data to further personalize the recommendations. This could involve

collecting more data about customers, such as their age, income, and other preferences, to improve the accuracy of the recommendations.

# 9. Reflection

The  main takeaway from this project is the importance of integrating various components such as data processing, selecting frequent pattern mining algorithms, recommending from patterns, and selecting evaluation metrics to create an effective system. Through this project, our team has gained a better understanding of how these components work together to generate meaningful insights and improve business operations.

Furthermore, this project has shown us how analysing customer data can help businesses make informed decisions and increase profits. By implementing the recommendation system, our team can continually monitor customer feedback, collect data on customer behaviours, and implement new features or updates based on customer needs and preferences.

Overall, this project has provided valuable insights into the importance of integrating various components to create effective systems and the benefits of using customer data to improve business operations. By continuing to monitor and improve the recommendation system, our team can help businesses provide better recommendations and ultimately increase customer satisfaction and loyalty.

# 10.References

Lee, C.H., Kim, Y.H. and Rhee, P.K., 2001. Web personalization expert with combining collaborative filtering and association rule mining technique. Expert Systems with Applications, 21(3), pp.131-137.https://www.sciencedirect.com/science/article/pii/S0957417401000343/pdfft?casa_token=68QbbDO8URAAAAAA:cmETEPCp3pOkzxZC_vF_dKDgU41RwIyj1ZpQgGwkvNjIMVyCCZLGNYWUVvPrHZCNwSDrAhGGIA&md5=c9cc1e9206467a1b3df9d70ad12f93b7&pid=1-s2.0-S0957417401000343-main.pdf

Parvatikar, S. and Joshi, B., 2015, December. Online book recommendation system by using collaborative filtering and association mining. In 2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC) (pp. 1-4). IEEE. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7435717

pandas.pydata.org. (n.d.). pandas.DataFrame.sort_values — pandas 1.4.1 documentation. [online] Available at: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sort_values.html.

pandas.pydata.org. (n.d.). pandas.to_numeric — pandas 1.4.2 documentation. [online] Available at: https://pandas.pydata.org/docs/reference/api/pandas.to_numeric.html.

pandas.pydata.org. (n.d.). pandas.pivot_table — pandas 1.4.2 documentation. [online] Available at: https://pandas.pydata.org/docs/reference/api/pandas.pivot_table.html.