

**MDB Assignment 3**  
**Pattern Mining and Recommender System**

Group 8:

Craig Atkinson a1669436

Lalitphan Sae-teoh a1932456

Pratham Maharjan a1944180

**The University of Adelaide**

4533\_COMP\_SCI\_7306

Mining Big Data

Lecturer: Dr. Alfred Krzywicki

**Table of Contents**

[1. Executive Summary 2](#_Toc1547140222)

[2. Introduction 3](#_Toc1950141282)

[3. Exploratory analysis 4](#_Toc59245996)

[4. Implementation 6](#_Toc1461018997)

[4.1 Task 1: Frequent pattern mining 6](#_Toc883904170)

[4.2 Task 2: Collaborative filtering 8](#_Toc1815892476)

[4.3 Task 3: System Integration 12](#_Toc1973716029)

[5. Testing and discussion of results on the test set 13](#_Toc1128621943)

[6. Conclusions and recommendations 15](#_Toc569936566)

[7. Contributions and Reflections 15](#_Toc775471717)

[7.1 Contributions – Group Members Task Breakdown 16](#_Toc2110516366)

[7.2 Reflections – Pratham Maharjan (a1944180) 16](#_Toc302581406)

[7.3 Reflections – Lalitphan Sae-teoh (a1932456) 17](#_Toc1283660171)

[7.4 Reflections – Craig Atkinson (a1669436) 18](#_Toc234955702)

[8. References 19](#_Toc2051819214)

[9. Appendix 19](#_Toc382662231)

# 1. Executive Summary

In this assignment, a prototype product recommendation system was created for a grocery store. The system uses a simple text-based interface and makes recommendations using two methods: the first is finding products that are often bought together (pattern-mining) and the second is recommending products based on the shopping habits of customers with similar preferences (collaborative filtering). The program was created from customer shopping data from the year 2014 to 2015 from approximately 3,200 shoppers. Early testing shows the system is able make recommendations for products, but it was found to have low overall performance due to data sparsity challenges of a grocery supermarket store, which has many different types of products for sale. It is therefore recommended that the prototype be further developed with more data sourced from multiple years of transactions, additionally, a deep learning method of product recommendation should be investigated as they may be better at dealing with the sparse data found in the grocery's dataset.

2. Introduction

The objective of this assignment was to create a prototype product recommendation system for a grocery store company. There were 3 main tasks were required for the system, frequent product pattern mining, collaborative filtering model, and the final task of system integration. Pattern mining is used to create a table of association rules that finds which products are frequently purchased together to make product recommendations, while collaborative filtering uses a machine learning model to find customers that have similar purchasing habits to recommend products.   
  
The problem the grocery store has assigned is they need an easy-to-use program that can predict items that customers are likely wanting to buy in the future, based on patterns in the historical data. The purpose of this is to create more opportunities for relevant interactions with the grocery’s stores customers, which helps the store sell more products and be more convenient to the customer as they can more easily find items that they want to buy. The purpose of the prototype program is to test the recommendation method using the real dataset that was provided by the grocery store, which will be used to find out if the recommendations are meaningful and relevant for this application.  
  
The design of the prototype system was adapted from research by Tewari and Barman (2018), who found that personalised product recommendations can be enhanced by using a combination of collaborative filtering, content-based filtering, and association rule mining which was shown to be an accurate recommendation method. This research shows that association rules derived from purchase patterns can be combined with customer behaviour modelling to create recommendations which more context. The authors found that their framework for product recommendations increased customer engagement conversion rates by more than 10% for retail stores.

Collaborative filtering is a recommendation system that predicts a user's preferences based on the tastes of similar users or items. It can be user-based or item-based filtering, leveraging interaction patterns to suggest users or items with similar features, and can use explicit feedback or implicit feedback. In recommender systems, explicit feedback involves users directly stating their preferences, such as rating a movie on Netflix or rating items on Amazon. In contrast, implicit feedback is inferred from user behaviour (Yifan Hu, Koren & Volinsky, 2008), such as watching a video on YouTube or purchasing groceries in Aldi. This assignment dataset is grocery data, which is an implicit feedback dataset. Therefore, the collaborative filtering technique used in this assignment should be suitable for the implicit dataset.

It must be acknowledged that the unique characteristics of implicit feedback limit the direct use of algorithms developed with explicit feedback in mind. (Yifan Hu, Koren, Y & Volinsky, C 2008) Firstly, it does not have negative feedback. Secondly, it might be noise. Sometimes, we cannot tell that a user bought the item because of their preference or other motives. Lastly, explicit feedback's value shows a preference, whereas implicit feedback indicates confidence.

According to research by Tanveer Et. Al. (2025), product recommendation systems can be evaluated with the following metrics: Mean Reciprocal Rank (MRR) which ranks how important a recommendation is, Hit-Rate, measures the percent of users that have at least one relevant recommendation, and F1-score which is a combination of precision and recall for measuring accurate and relevant recommendations. This assignment used these metrics to help tune parameters of the system and evaluate the final product.

# 3. Exploratory analysis

There are 2 datasets in this assignment, which is split into train and test data. They each contain roughly 1-year worth of customer purchase transactions from a grocery store. Each customer is assigned a unique user ID number, making it possible to track individual customers purchase history. The dataset column types are listed in the following dictionary:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Example** |
| User\_id | int | User id | 2226 |
| Date | Datetime | Transaction date | 1/01/2014 |
| itemDescription | string | Item name | sausage |
| year | int | Transaction year | 2014 |
| month | int | Transaction month | 1 |
| day | int | Transaction day | 1 |
| day\_of\_week | int | Transaction day of week | 2 |

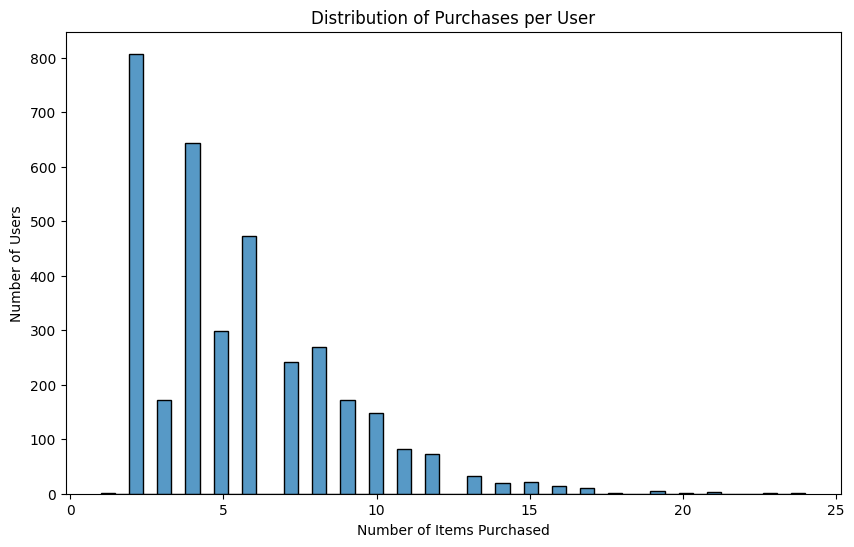
Each record in the dataset represents an item purchased by a user in that day. For example, user 2226 purchased 3 items on 1/01/2014, so it will be 3 records in the dataset.

The following tables shows statistical between train and test data sets.

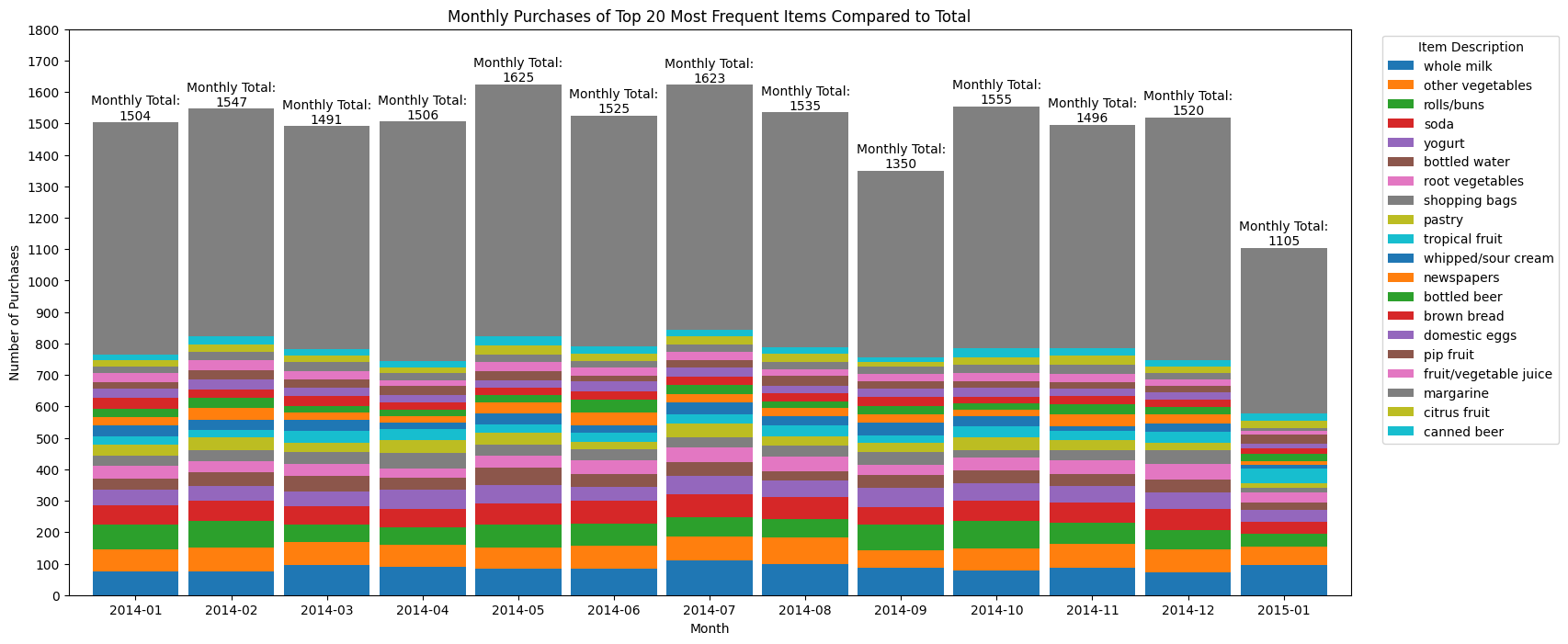
|  |  |  |
| --- | --- | --- |
| **Statistics** | **Train Dataset** | **Test Dataset** |
| Total records | 19,382 | 19,383 |
| Total transactions\* | 8,361 | 6,612 |
| Min Date | 01-01-2014 | 20-01-2015 |
| Max Date | 20-01-2015 | 30-12-2015 |
| Total unique users | 3,493 | 3,231 |
| Total unique items | 167 | 164 |
| Intersection unique users | 3,898 | |
| Intersection unique items | 167 | |

\* Transaction is combination of purchase items in a day by same user.

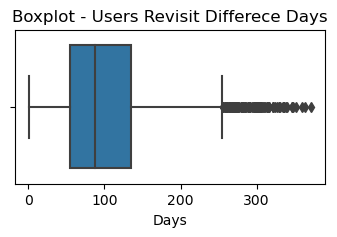
The total distribution of how many items each user purchased, can be seen in the figure below. was found to be skewed to the right, with majority of users purchasing only a few items.



The distribution of the purchases of each item every month is shown in the figure below. This shows that the number of monthly purchases remained similar each month, with similar amounts of the top 20 products being purchased. However, it also shows that they are many products categories in the dataset, many of which only have small number of recorded purchases in the dataset. This means the dataset is very sparse which poses several challenges for the recommendation system, as it may be hard to determine frequent patterns and find long term user-item relationships.



As the recency of user purchasing history is considered in this assignment, the analysis of user revisit is performed. The average revisit from overall transactions is approximately 104 days, which is around 3.5 months. This analysis will include finding the best parameter for decay rate when applying recency weight to the user-item matrix in collaboration filtering.



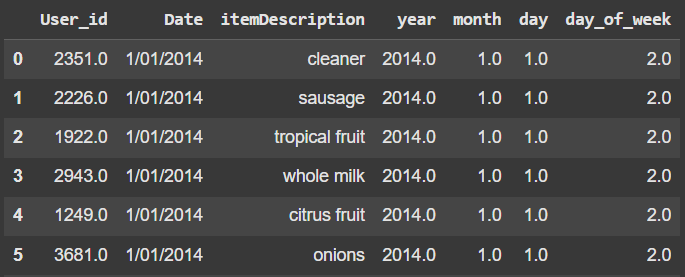
# 4. Implementation

## 4.1 Task 1: Frequent pattern mining

Frequent pattern mining is one of the most widely researched areas in data mining because of its relevance to real life applications. There are various common algorithms used in mining frequent patterns in data, and they can have differing results based on the dataset used. For this assignment, the algorithms Apriori and FP-Growth were investigated as they are both very popular algorithms that are commonly used for pattern mining. The differences between them are listed below.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.N** | **Parameters** | **Apriori** | **FP Growth** |
| 1 | Storage structure | Array based | Tree based |
| 2 | Search type | BreadthFirst Search | Divide and conquer |
| 3 | Technique | Join and prune | Constructs conditional frequency pattern treewhich satisfy minimum Support |
| 4 | Number of Database scans | K+1 Scans | 2 scans |
| 5 | Memory utilization | Largememory (candidate generation) | Less memory (No candidate generation). |
| 6 | Database | Sparse/dense datasets | Large and medium data sets |
| 7 | Run time | More time | Less time |

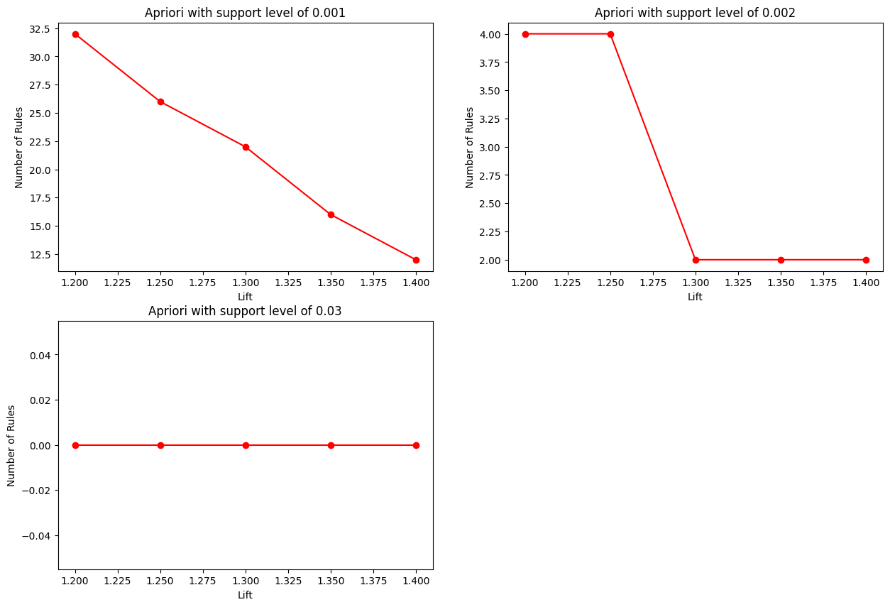
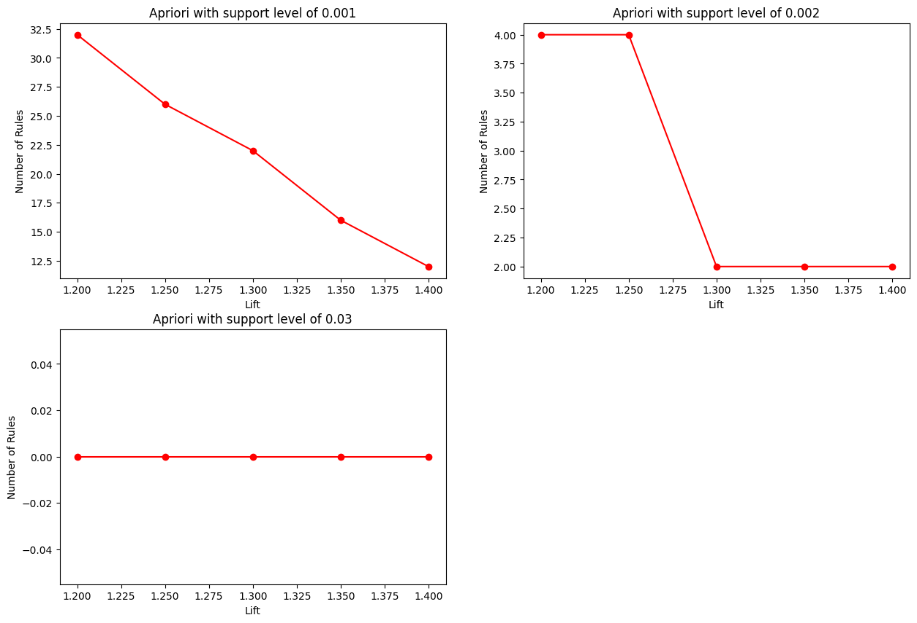
For this task, the training dataset was used, which contained 26,985 product transactions. The null values from the dataset were removed and preprocessing was done, and it was found out that the average number of items per transaction was 2.32 which presented challenges in pattern discovery.



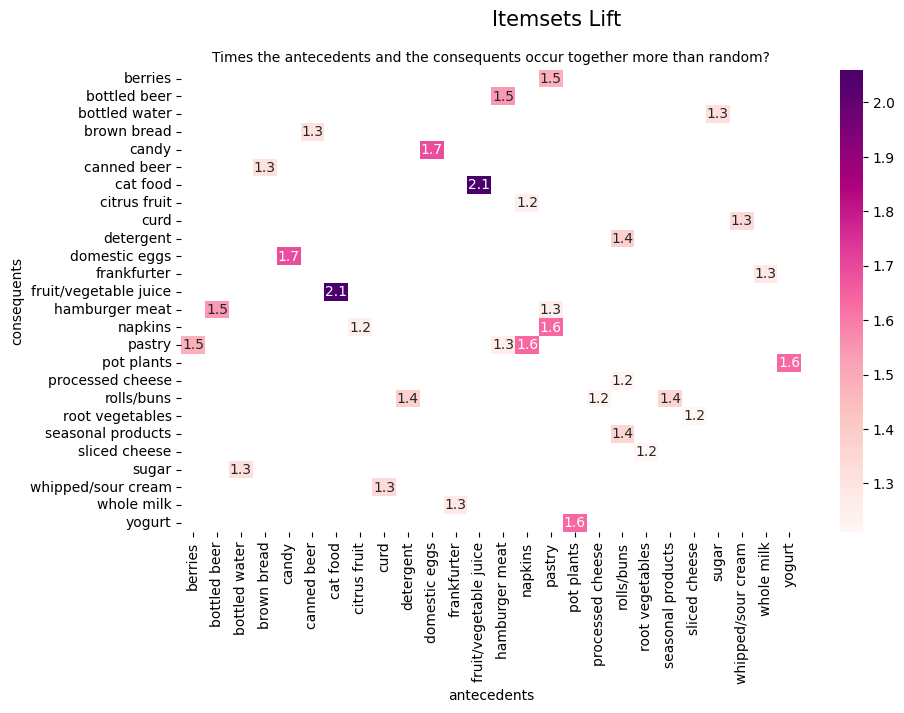
As there was no transaction id, the dataset was grouped together based on user\_id and date which gave us the total number of items per transactions of each user. The total amount of data in this newly formed dataset was 8361.



One hot encoding was applied to the transaction data where each row represented a transaction, and true and false Boolean values represented whether a particular item was bought during a transaction. After this process, the Apriori algorithm was applied using various values for min support and different metrics were experimented on to generate rules. As the average number of transactions were less than 3, a min support value of 0.001 was chosen to capture rare but meaningful item combinations. The metric lift used for generating rules as confidence can be very low in case of data with low average no of items per transaction which can be misleading. Therefore, lift value greater than 1.2 was used to generate association rules. The same was done for FP growth algorithm as well and the output were compared.



Both Apriori and FP growth algorithms generated the same number of item sets and rules which was 462 item sets and 31 rules. In the generated rules confidence was low, however lift value ranged from 1.2 to 2 which indicated that non-random and meaningful associations were able to be captured.



# 4.2 Task 2: Collaborative filtering

This assignment used the Alternating Least Squares (ALS) algorithm from the implicit Python library to implement collaborative filtering. The goal of this algorithm is to learn whether each user has interacted with each item. However, it weights each binary interaction according to a confidence value based on the confidence in user/item interaction, where the confidences are represented in the values of a sparse matrix. Therefore, the non-zero entries indicate whether the user has interacted with the item.

During the data preprocessing, the dataset is transformed into a user-item matrix, where the confidence values of user-item interaction are represented by the frequency or quantity of that item being purchased by the user. The missing interaction between user items will be imputed to 0. The recency weights are added by multiplying by frequency as a requirement to consider the recency of the user’s purchasing. The exponential decay function for recency weight is applied, as it is commonly used in several works (Larrain, Trattner, Parra, Graells-Garrido & Nørvåg, 2015). The decay rate (time constant), a parameter in decay function, to controls how quickly the influence of older interactions drops off. It’s a tuneable parameter, it has evaluated and chosen from the rate that is given the highest hit rate and precision from test dataset evaluation on base model of recommendation system without frequent dataset which is 365 days.

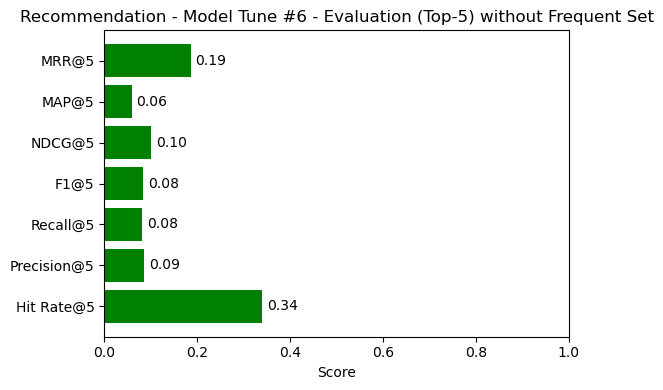
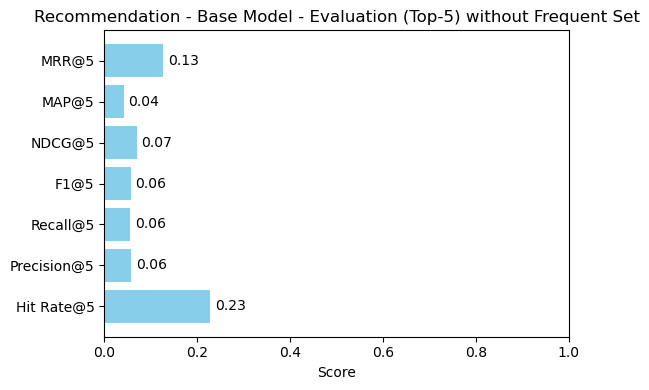
|  |  |  |  |
| --- | --- | --- | --- |
| **Decay Rate** | **Description** | **Hit Rate @ k** | **Precision @ k** |
| 30 days | Short-term (1 month) | 0.189 | 0.050 |
| 90 days | Quarter (3 months) | 0.202 | 0.053 |
| 104 days | Mean value from revisit analysis | 0.203 | 0.053 |
| 180 days | Half year (6 months) | 0.215 | 0.056 |
| 365 days | 1 year | 0.230 | 0.058 |

The model's input is the transformation of the user-item matrix into a sparse matrix form, which was weighted by the BM25 function, which helps to reduce the impact of repeated items that users have purchased and the weight of popular items. The BM25 is a method for ranking documents based on the words in a query. It looks at how often the query terms appear in each document, ignoring where they are located (Wikipedia, n.d.). The base model of ALS collaborative filtering settings are 50 factors, 0.2 regularization, 20 iterations.

The following table shows the evaluation metrics that used to evaluate the model performance in this assignment.

|  |  |  |
| --- | --- | --- |
| **Evaluation Metrics** | **Description** | **Formula** |
| Hit Rate @ k | Measure how many recommended items in top k occur in relevant list. |  |
| Precision @ k | Measure how many relevant items in top k. |  |
| Recall @ k | Measure how many relevant items found in top k out of all relevant items. |  |
| F1 Score @ k | Combination of Precision and Recall, for comparison with other methods. |  |
| Mean Average Precision (MAP) @ k | Measure how good top k recommended ranked in list. |  |
| Mean Reciprocal Rank (MRR) @ k | Measure the first position the relevant item discovers in top k. |  |
| Normalized Cumulative Discount Gain (NCDG) @ k | Measure how good the ranked list is. | DCG @ k =  IDCG @ k = The highest possible DCG that can be attained using the same set of relevance scores, provided they are arranged in the optimal ranking order.  NCDG @ k = DCG @ k / IDCG @ k |

K is the number of top recommended items that generated for users.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Model Parameters** | **Hit Count @ 5** | **F1 @ 5** | **MAP @ 5** | **MRR @ 5** | **NCDG @ 5** |
| Base model | 50 factors, 0.2 regularization, 20 iterations | 0.227 | 0.057 | 0.041 | 0.127 | 0.069 |
| Model Tune #1 | 40 factors, 0.2 regularization, 20 iterations | 0.315 | 0.079 | 0.056 | 0.172 | 0.095 |
| Model Tune #2 | 30 factors, 0.2 regularization, 20 iterations | 0.332 | 0.083 | 0.059 | 0.182 | 0.101 |
| Model Tune #3 | 20 factors, 0.2 regularization, 20 iterations | 0.314 | 0.078 | 0.052 | 0.164 | 0.091 |
| Model Tune #4 | 30 factors, 0.2 regularization, 50 iterations | 0.336 | 0.083 | 0.059 | 0.186 | 0.101 |
| Model Tune #5 | 30 factors, 0.5 regularization, 50 iterations | 0.338 | 0.083 | 0.058 | 0.183 | 0.100 |
| Model Tune #6 | 30 factors, 0.01 regularization, 50 iterations | 0.340 | 0.084 | 0.059 | 0.185 | 0.101 |

After tuning hyperparameter of model, it results that Model Tune #6 with settings of 30 factors, 0.01 regularization and 50 iterations has the best results with highest hit rate 0.340, F1 score 0.084, MAP 0.059, MRR 0.185 and NCDG 0.101. From the experiment results above, it shows that the optimal factors, regularization and iterations can obtains better results.

Apart from collaborative filter approaches, this assignment also allows system to recommend items based on frequent items set by the input of “with” frequent items set option. Several rules from task 1 were experimented with collaborative filtering best model, which is Model Tune #6, and evaluated on test dataset to find the best rules for recommendation. The table below shows the results of the experiments.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Rules Name** | **Association Rules** | **Parameters** | **Total rules** | **Hit Rate@5** | **F1@5** | **MRR@5** |
| Base Rules | Apriori | Min support 0.001, Lift 1.2 | 32 | 0.341 | 0.084 | 0.183 |
| Rules #1 | Apriori | Min support 0.001, Lift 1.25 | 26 | 0.341 | 0.084 | 0.183 |
| Rules #2 | Apriori | Min support 0.001, Lift 1.3 | 22 | 0.339 | 0.083 | 0.179 |
| Rules #3 | Apriori | Min support 0.001, Lift 1.35 | 16 | 0.338 | 0.083 | 0.179 |
| Rules #4 | FP Growth | Min support 0.001, Lift 1.2 | 32 | 0.341 | 0.084 | 0.183 |
| Rules #5 | FP Growth | Min support 0.001, Lift 1.25 | 26 | 0.341 | 0.084 | 0.183 |
| Rules #6 | FP Growth | Min support 0.001, Lift 1.3 | 22 | 0.339 | 0.083 | 0.179 |
| Rules #7 | FP Growth | Min support 0.001, Lift 1.35 | 16 | 0.338 | 0.083 | 0.179 |

The rules experiment above uses “life” as a metric to boost the recommendation score. The performance of recommendations by integrating association rules could be affected by the selection of metrics. The table below shows the comparison of the performance of collaborative filtering with association rules (“lift” and “confidence” metrics) and without rules. The performance between them was found to very similar, especially the F1 score. However, the collaborative filtering with “confidence” metrics in association rules has a better MRR and hit rate of 0.188 and 0.342. Collaborative filtering with frequent MRR and hit rate is 0.185 and 0.340, respectively, which means the confidence metric in association rules helps the recommender system rank better items.

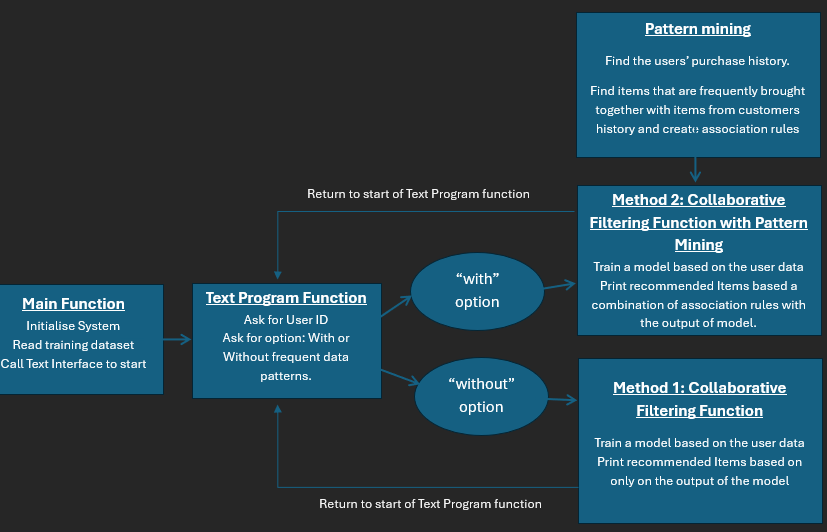
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **With/ Without Frequent Set** | **Association Rules Metrics** | **Hit Rate@5** | **F1@5** | **MRR@5** |
| Without Frequent Set | - | 0.340 | 0.084 | 0.185 |
| With Frequent Set | Lift | 0.341 | 0.084 | 0.183 |
| With Frequent Set | Confidence | 0.342 | 0.084 | 0.188 |

New users are handled by recommending top global popular items. The recommendation to cold-start users who had purchased less than 2 items was tested by recommending similar items from previously purchased items. However, the results were worse than the score from the final tuning model. Therefore, the recommendation for similarity approaches for those users was removed, and the recommendation approach remained the same as that of other users.

## 4.3 Task 3: System Integration

The system design was required to use a terminal-based text interface, which makes the use of task 1 and task 2 to recommend products for specific users. The text interface was made using a simple python program, that first loads the training dataset and then waits for a user ID number to be input. When a user ID number is provided, the system presents two options, with frequent item set or without frequent item set. If the without option is selected, the system will only call the collaborative filtering function, which establishes the user-item relationships based on the patterns of other users. The function then returns the list of recommended items. For the option with frequent item set, the pattern mining collaborative filtering function will also call the patterning mining functions. The pattern mining function will either Apriori algorithm or FP-growth algorithm to create the item-item association rules, which it returns to the collaborative filtering function. The collaborative filtering function then trains an Alternating Least Squares model on the dataset and saves it to the class global variable, which ensures it only trains the model once while running and saves execution time for the new user ID. The model is then used to generate the user-item matrix, which is combined with the item-item association rules to create the list of 5 recommended items for the provided user ID. Once the recommendations have been printed in the terminal, the system then prompts the input of the next user ID. The program can be quit by inputting the letter “q” instead of a user ID.

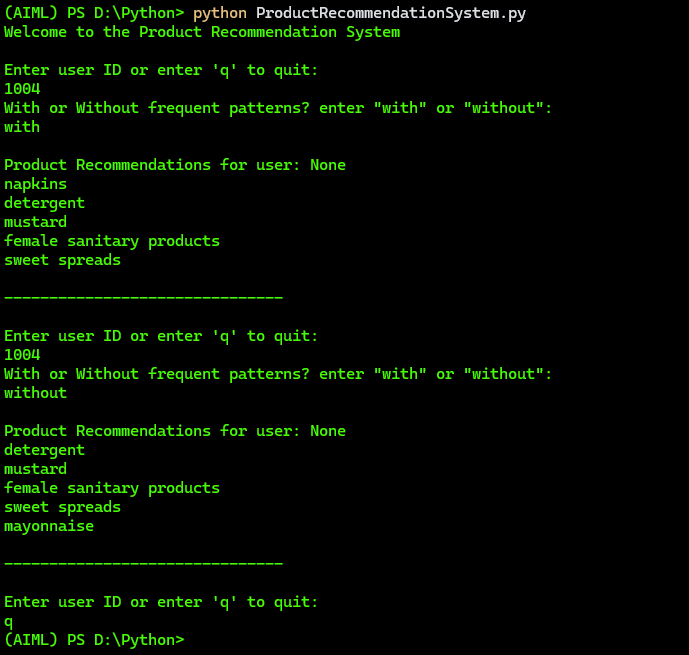
Additionally, if the user ID is a new user, which no purchase history in training dataset, the system returns a list of the top 5 most popular items instead. Below is the diagram of the flow of the final system.



**Figure 1: Overview of the system design for the Product Recommendation program.**

# 5. Testing and discussion of results on the test set

The Product Recommendation Program was run in the terminal environment using python. The initial testing shows the program output of 5 five recommended items for the User ID: 1004. The “with frequent items” method produced the recommended items: detergent, hamburger meat, processed cheese, photo/film, and waffles while the “without frequent items” method produced the recommended items: hamburger meat, photo/film, waffles, female sanitary products and herbs. This shows that both methods were able to recommend products for the customer, and the methods produced different results, although 3 out of 5 of the recommended items were the same.



**Figure 2: Output of the Python Program running the Product Recommendation System.**

The testing dataset called “Groceries data testing.csv” was then used evaluate the results the Recommendation System. The testing dataset contains the customer item transactions entire year of 2015, which is the same structure as the training dataset which was for the year of 2014. The goal of the evaluation method was to find out if the system trained on customer data from 2014 will accurately recommend products that customer ended up purchasing in 2015. This method of evaluation will be measuring if the product recommendations were actual products the customer wanted to purchase in the future. One weakness of this evaluation method is it is only considering what each customer purchased in the future and doesn't consider the system's ability to help customers find products they wouldn't have otherwise purchased.

The testing of the final version of the prototype, where without the frequent set is the collaboration filtering model from Alternating Least Squares algorithm with settings of 50 factors, 0.01 regularization, and 50 iterations and with the frequent set is the same setting of collaboration filtering model with confidence-boosting from Apriori association rules with settings of min support 0.001 and lift 1.2, the following evaluation metrics were calculated when comparing the systems user-recommendations which the actual purchases found in the testing dataset (simulated future data).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **With/ Without Frequent Set** | **Hit Rate@5** | **F1@5** | **MRR@5** | **Time to Run** |
| Without Frequent Set | 0.340 | 0.084 | 0.185 | 2.1 seconds |
| With Frequent Set | 0.342 | 0.084 | 0.188 | 7.5 seconds |

The results of the best Recommendation System had a MRR of 0.188, f1-score of 0.084 and a hit rate of 0.342. The evaluation metrics were roughly the same for both methods of recommendation, with the frequent item set or without the frequent item set. This indicates the system has relatively low performance in predicting the future purchases of individual customers. One reason for this may be the sparsity of the item purchase data, where individual users only purchase a small percentage of the available items, which makes it challenging to create consistent patterns.

The limitations of this assignment was the sparse dataset that had many items which only a few purchases and many users that only had 2 recorded transactions, which was challenging to create meaningful patterns in the historical data. Another limitation of this assignment was the data only covered 1 year of purchases, which would not have enough depth to cover patterns over seasons and would not be able to learn how users' preferences change over time including how customer trends evolve. Research by Tanveer Et. Al. (2025), suggests that a recommendation system that uses a Graph Neural Network (GNN) may be more effective finding relevant patterns in spare of noisy data, as it is able to use selective attention, which is valuable insight for future development of the prototype.

The top frequent patterns found in the training dataset using pattern mining were:

|  |  |  |
| --- | --- | --- |
| **Items Pattern** | **Support** | **Confidence** |
| Frankfurter -- > Whole Milk | 0.002 | 0.167 |
| Detergent -- > rolls/buns | 0.001 | 0.141 |
| Seasonal Products -- > rolls/buns | 0.001 | 0.138 |
| Pot Plants -- > Yogurt | 0.001 | 0.129 |
| Processed Cheese -- > rolls/buns | 0.001 | 0.125 |

# 6. Conclusions and recommendations

In this assignment a product recommendation system for a grocery store customers were created, which used frequent pattern mining and collaborative filtering to make personalised product recommendations for each customer. While the system was found to capable of generating personalised recommendations, it was found to have weak performance in predicting future purchases habits of customers, as indicated by low evaluation metrics for 5 recommendations including a prediction hit rate of only 34%. It was found that the Apriori algorithm and FP-growth algorithm produced similar performance in pattern mining, however it was found that FP-growth was faster to run, so it would be more ideal for use in a production version Product Recommendation System.

It is therefore recommended that the prototype be further developed with more data that is sourced from more years of transactions. Additionally, a Graph Neural Network (GNN) model may provide more meaningful results from the grocery's dataset, therefore, it is recommended that future prototypes be developed that test this kind of product recommendation system.

# 7. Contributions and Reflections

## 7.1 Contributions – Group Members Task Breakdown

|  |  |
| --- | --- |
| **Task** | **Group Member Assigned** |
| Exploratory Data Analysis | Craig, Lalitphan & Pratham |
| Implementation: Task1 Frequent Pattern Mining | Pratham |
| Implementation: Task2 Collaborative Filtering | Lalitphan |
| Implementation: Task3 System Integration | Craig |
| Testing and discussion of results | Craig, Lalitphan & Pratham |
| Conclusions and recommendations | Craig, Lalitphan & Pratham |

## 7.2 Reflections – Pratham Maharjan (a1944180)

**Code Version:** version01

* **Stages:** Setup the code, implemented pattern mining
* **Reflection:** The first part of the project was pattern mining. I ran a very basic setup of code for pattern mining with little data preprocessing. I ran through the data and grouped them together with user\_id and date to build transactions. After encoding and running both Apriori and FP growth algorithms for the first time, I was not able to generate any association rules with the confidence metric. It became clear to me that some more data preprocessing was needed. I researched on other metrics instead of confidence.

**Code Version:** version02

* **Stages:** EDA, data preprocessing, experimenting evaluation metrics
* **Reflection:** I faced challenges such as dealing with data where the average number of items per transaction is very low i.e 2.3 in this case. This made it difficult to capture frequent item sets without lowering the min support value. I generated charts to understand the data better. After some trials 0.001 was used as the min support. The metric was changed from confidence to lift as lift would be the better fit to decide items that occur more frequently together. In this version 462 item sets were generated and rules were also generated in both Apriori and FP growth algorithms.

**Code Version:** version03

* **Stages:** EDA, experimenting min support and lift value, generating rules
* **Reflection:** I tried different values for min support and lift and analysed it to figure out the most suitable value that would generate a good number of rules. After experimenting with a few values, I picked the most suitable one and researched on other metrics that I would be able to use to evaluate the generated rules.

## 7.3 Reflections – Lalitphan Sae-teoh (a1932456)

**Code Version:** version01

* **Stages:** Analyst, Visualized data, Pre-processed data and implemented base model.
* **Progress & Results:** Performed analysis, researched collaborative filtering methods and implemented the first base model.
* **Reflection:** I learned how to analyse transaction data. During data preprocessing, I found that the user-item matrix had many missing values, so I researched the method of handling them. As the dataset is implicit feedback, the missing value can be treated as zero to indicate that there is zero purchasing on that item (Yifan Hu, Koren, Y & Volinsky, C 2008). The first base model used the Alternating Least Squares approach. I learned the difference between explicit and implicit feedback, the characteristics of implicit feedback and the collaborative filtering approach on implicit feedback.

**Code Version:** version02

* **Stages:** Implemented recommendation model.
* **Progress & Results:** Implemented collaborative filtering with frequency dataset and added recency weight.
* **Reflection:** One of the assignment requirements indicates that the recommendation items should base on the recency of users purchased. Therefore, I added the recency weight and multiplied it by the quantity for the Users-Items matrix, which will be input for the collaborative filtering model. Then, I learned how to integrate association rules into collaborative filtering. The rules in this code version are dummy, so I couldn't compare the results with/without frequency datasets. I also cleaned the code to function that makes it easier to integrate in task.

**Code Version:** version03

* **Stages:** Analyst and Evaluation recommendation model
* **Progress & Results:** Performed more analysis on user's recency and Added evaluation metrics.
* **Reflection:** I added hit rate, precision, recall and f1 score for evaluation metrics. I did some analysis of users' recency; I thought that the revisit pattern might affect the recency weight. I used the exponential decay function for recency weight, which is quite commonly used in several works (Larrain S, Trattner C, Parra D, Graells-Garrido E & Nørvåg, K 2015). This assignment's decay rate or time constant is the revisit days pattern. From analysis, I found that the average revisit days is around 104. I applied this value to check whether it is improved from previous settings, which is 120 days (approximate from the medium-term revisit pattern). However, in the results from several settings, I found that the higher the decay rate, the better the precision and hit rate results.

**Code Version:** version04

* **Stages:** Model Tuning and Evaluation model
* **Progress & Results:** Improved the performance of recommendation model and evaluated the model with different parameters, association rules, etc.
* **Reflection:** I tested several parameters for collaboration filtering models such as number of factors, regularization and iteration. The results show that the optimal factors, regularization and iterations can obtains better results. Then, I tested the different association rules and found that the results are not different between Apriori and FP growth algorithm.

## 7.4 Reflections – Craig Atkinson (a1669436)

**Code Version:** version01

The structure of the recommendation system was created as a python program. The program is made up of a main routine and an object class called "Recommendations". The object "Recommendations" has three main functions, "pattern\_mining", "collaborative\_filtering", and "text\_program". In this version of the program, "pattern\_mining", "collaborative\_filtering" are placeholder functions for the work of the other group members (group task 1 and 2). The function "text\_program" is what the program user interacts with to get the results. It first prompts the user for to input a user ID number and ask for the "with or without" input. The function then called the "pattern\_mining", "colabrative\_filtering functions and then return the product recommendations. The program will then continue to loop until the user inputs the 'q' command to quit.

**Code Version:** version02

An error was found if the user inputs a non-numerical input string when asked for user ID, this was fixed by validating the input and outputting a try again message to the user, which avoids the program crashing with an error message. Additionally, some EDA statistics and graphs were generated in the jupyter notebook, which will help the group better understand the dataset and be used in the group report.

**Code Version:** version03

The first versions of task 1 and task 2 were added to the Reccommendations functions "pattern\_mining" and "collaborative\_filtering". The "collaborative\_filtering" method is working using the "implicit" python module, I added the ignore warnings and “show\_progress=false” so that the user does not see unnecessary warnings while using the program. The output of the "collaborative\_filtering" function is providing recommended items based on similar users as expected. However, the 'with frequent patterns' option has not yet been implimented due to issues with the "pattern\_mining" function. The current implementation of the "pattern\_mining" function is not providing the patterns and is returning an empty DataFrame.

**Code Version:** version04

The second versions of task 1 and task 2 were added to the Recommendations functions "pattern\_mining" and "collaborative\_filtering". The "with or without" frequent items option correctly implemented, where the "with" option reccommends items using the association rules from the "pattern\_mining" function and the results of the "collaborative\_filtering" model. I also noticed that if the user ID input is not in the training data it will crash the program with an error.

**Code Version:** version05

The next version of the collaborative filtering function (task 2 code) was added to the final product code. Additionally, the FP-growth was tested for the pattern mining association rules, which was found to give very similar results to the Apriori algorithm, which both had 32 association rules. Following the evaluation and tuning of the collaborative filtering function and pattern mining function by the other members of the group, the parameters of each function were changed. The final parameters were chosen because they gave the highest hit-rate when comparing product recommendations with the "future" transactions in the testing dataset. The parameters for the FP-growth function were chosen to be using the confidence metric, minimum support value of 0.001 and threshold of 1.2. For the collaborative filtering the Alternating Least Squares model parameters were chosen to be 50 factors a regularization rate of 0.01 and using 30 iterations.

# 8. References

* Tewari, AS & Barman, AG 2018, ‘Sequencing of items in personalized recommendations using multiple recommendation techniques’, Expert Systems with Applications, vol. 97, pp. 70–82.
* Tanveer, J, Lee, S-W, Rahmani, AM, Aurangzeb, K, Alam, M, Zare, G, … Hosseinzadeh, M 2025, ‘PGA-DRL: Progressive graph attention-based deep reinforcement learning for recommender systems’, Information Fusion, vol. 121, pp. 103167-.
* Yifan Hu, Koren, Y & Volinsky, C 2008, ‘Collaborative Filtering for Implicit Feedback Datasets’, in 2008 Eighth IEEE International Conference on Data Mining, IEEE, pp. 263–272.
* Larrain, S, Trattner, C, Parra, D, Graells-Garrido, E & Nørvåg, K 2015, ‘Good Times Bad Times: A Study on Recency Effects in Collaborative Filtering for Social Tagging’, in Proceedings of the 9th ACM Conference on Recommender Systems, ACM, New York, NY, USA, pp. 269–272.
* Wikipedia n.d., ‘Okapi BM25’, Wikipedia the free encyclopedia, Accessed 27 April 2025, <https://en.wikipedia.org/wiki/Okapi\_BM25>
* Pratik Aher 2023, 'Evaluation Metrics for Recommendation Systems – An Overview', medium, accessed 26 April 2025, <https://towardsdatascience.com/evaluation-metrics-for-recommendation-systems-an-overview-71290690ecba/>
* Garg, K & Kumar, D 2013, ‘Comparing the Performance of Frequent Pattern Mining Algorithms’, International Journal of Computer Applications, vol. 69, no. 25, pp. 21–28.

# 9. Appendix

Github Link to Project Files: <https://github.com/lalitphan-rainie/COMPSCI7306_MBD_A3>

Example of user-item matrix from the collaborative filtering function:

