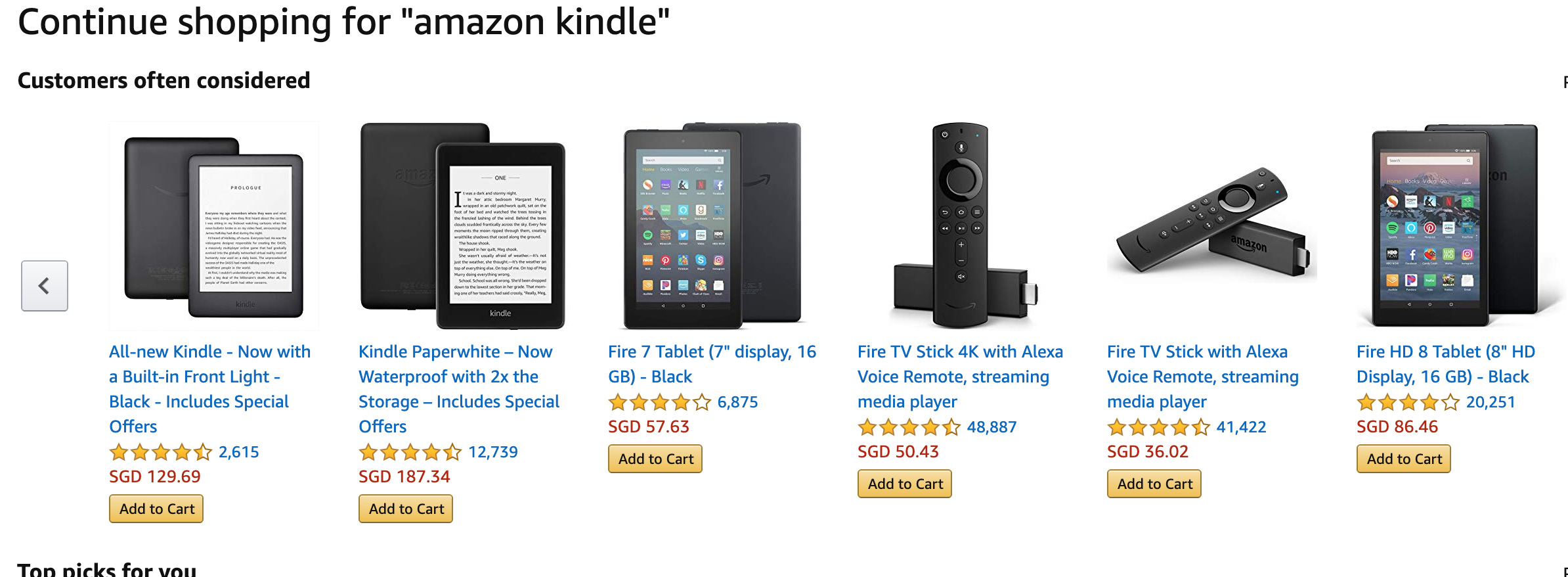
**Text Analytic & Recommender System Projects**

***Recommender System for Electronics Stores on Amazon E-commerce Platform***

|  |  |  |  |
| --- | --- | --- | --- |
| S/No | Team Member Names | Student ID | Matriculation Num. |
| 1 | Ng Soo Kuan | A0195309X | E0384940 |
| 2 | Ci Ruixiang | A0195307B | E0384938 |
| 3 | Gao Jiacheng | A0101568J | E0384320 |
| 4 | Koh Cher Guan | A0048190H | E0384217 |
| 5 | Liu Tianyu | A0100675M | E0384317 |

**Business Problems**

Amazon is a multinational e-commerce company known for its cutting-edge technologies. The company has a heavy focus on data-driven marketing which leverages on its tremendous amount of data created by billions of customers and merchants on the website. One important way of making use of the data is to recommend customers the most relevant and accurate items to each of them, by discovering patterns in the data set and producing recommendations that correlate to the interests and needs of the customers. Hence, by adopting the above mentioned method known as Recommender Systems, Amazon is able to suggest products that customers could be interested to purchase, as shown in the example below.



The benefits of adopting recommender systems include:

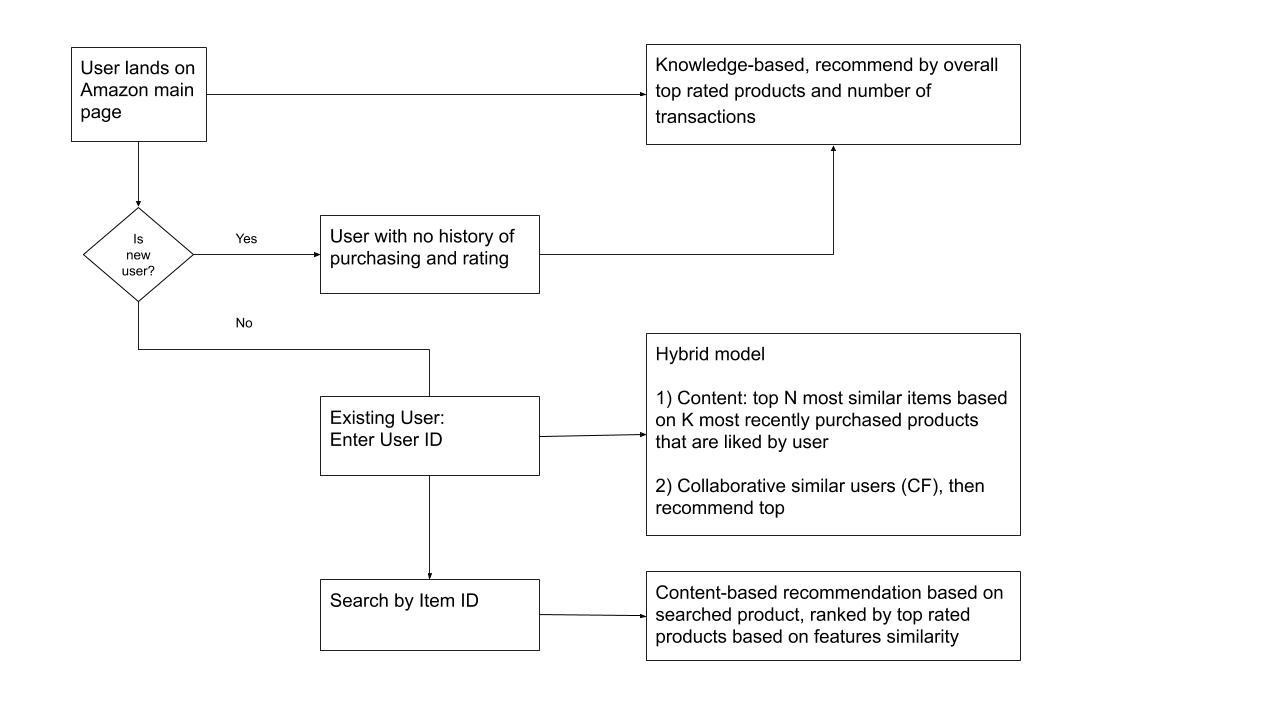
* Provide personalised recommendation to each customer
* Increase customer satisfaction for Amazon
* Increase customer loyalty
* Increase traffic to Amazon platform
* Higher chance that customers would make purchase
* Greater possibility for cross-sell and upselling
* Ultimately Increase revenue
* Try to market the long tail products to clear the existing inventory.

As the newly created data science team in Amazon Singapore dedicated to the electronics market, we leverage on the large amount of data created in the past to develop and fine tune our recommender systems for electronics, so that the above mentioned benefits could be achieved.

**Business Requirements**

In view of the nature of e-commerce platform, recommendation could play its important role in several stagings while a user is browsing through the platform.

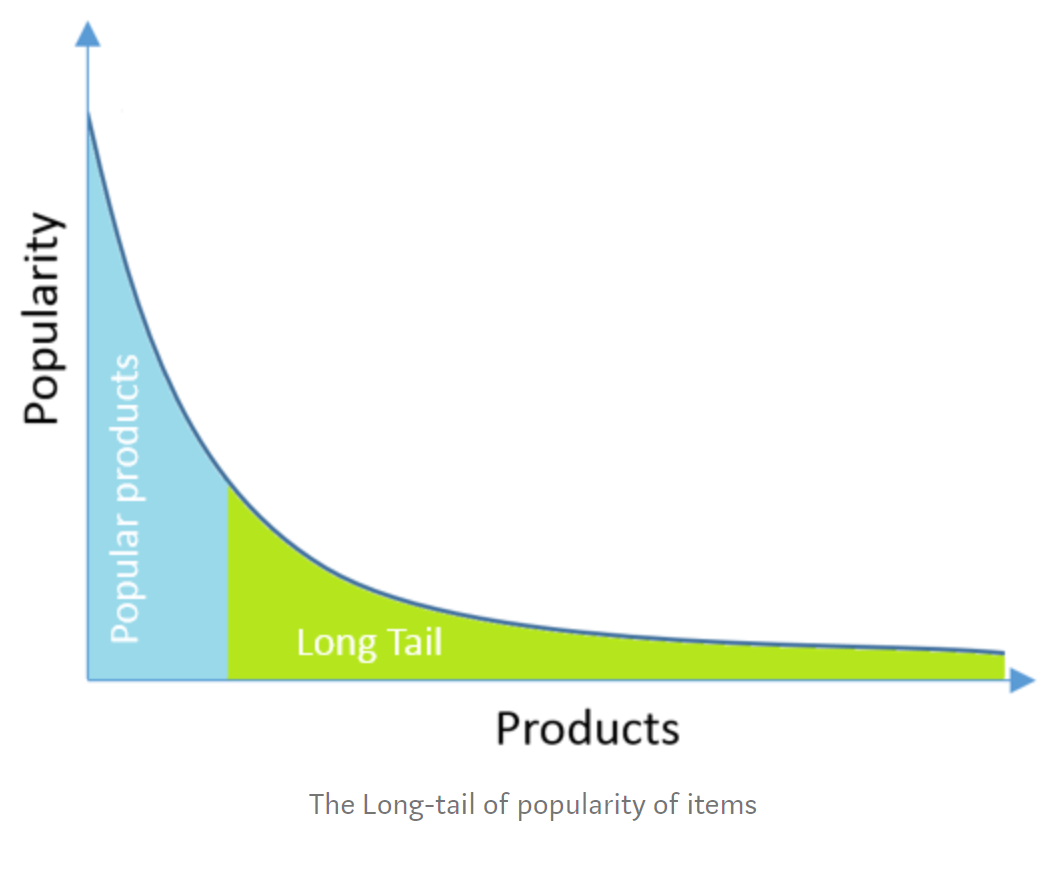
To ensure consumers can get accurate and customised product recommendations, we designed the recommender system to function at three different stages of touch points based on their profiles illustrated in the flowchart below:



**Methodology used on the Recommender System**

Knowledge-based Recommender System based on explicit knowledge is used on the new users or users without any history record in the database. We built the Knowledge-based Recommender System to market the products which are top rated and highest amount of transaction to this batch of new users.

Collaborative filtering and content-based filtering are automatic approaches for predictions about the interests of a user or similarity of the products. Collaborative filtering to recommend the products which have similarity between users and items in order to market the popular products for boosting the revenue. Moreover, collaborative filtering use rating system to filter the active users and popular items which can be used for building recommender systems (either the user-based or item-based). At the same time, using the content based approach to market the products at long tail in order to improve the business and clearing the existing inventory. At the end, both collaborative filtering and content based filtering will combine as hybrid model for prediction on users products’ preferences.



**Datasets used**

Dataset we used in this study contains customers’ transactional records with reviews and explicit ratings. In total, there are 79 listed items in this dataset, rated by 17429 different consumers, across the timeline from Feb 26, 2009 to Mar 25, 2019. Most of the items fall under the category of electronics, which fits the working purpose of our team.

Product descriptions are missing from the original dataset. We decide to manually search product description from the Internet, as it is crucial for analysis to come up with recommender systems. Upon collecting all product descriptions, we applied text mining techniques to extract and analyze product features. This is a necessary step to develop content-based recommender system.

For customer reviews, there are two ways of getting review ratings - explicit and implicit. Explicit ratings are ratings directly provided by customers after each purchase of products. They can be straightforward and easy to use, and it is a good indication of users’ preference. Therefore, we rely on only explicit ratings to build our recommender system.

The source of the dataset is as below:

<https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products>

**Data Preparation and Pre-processing**

Content Based Approach - Data Preparation and Pre-processing

After checking the dimension, structure and summary of the data frame, we decide to carry out the following steps for data pre-processing:

* Remove the missing row.
* Select the variables that require for text mining. In this case, select ***Product Title*** and ***Product Descriptions*** that consist a lot of information which required for text mining.

In order to analyze and extract features in ***Product Title*** and ***Product Descriptions***, we need to apply certain text mining techniques. First step is to perform tokenization which breaks a steam of characters into tokens. Next we need to remove English stop words, convert to lower case, remove numbers, whitespace,punctuation and stemming. We also referred to the WordCloud to further remove those stopwords that do not contribute value to the analysis. After removing the unnecessary text, we converted the bag of words into document term frequency matrix. This is done by firstly converting to term frequency matrix, followed by transforming into Tf-Idf (Term Frequency-Inverse Document Frequency) matrix with normalization. Finally the words are ready for further processing to compute similarity distances and product recommendation.

Collaborative filtering - Data Preparation and Pre-processing

One thing we notice in the raw dataset is there are many transactions with reviewer named as ‘Anonymous’. By checking the review patterns and purchasing records by ‘Anonymous’ user, we realize it should refer to a group of different users who opts to not show their actual user name. As such, it does not make sense to keep records from this username, since different users could behave very differently in terms of purchasing needs and preferences.

For modeling purposes, we selected the ***reviews.username***, ***name***, ***reviews.rating*** variables from raw dataset. To focus on active users, we only keep those who have purchased and rated at least 3 items in the past. At the same time, we also removed those super long-tailed items with less than 3 ratings in the records. Transactions in the document are reduced from 33332 to 14887.

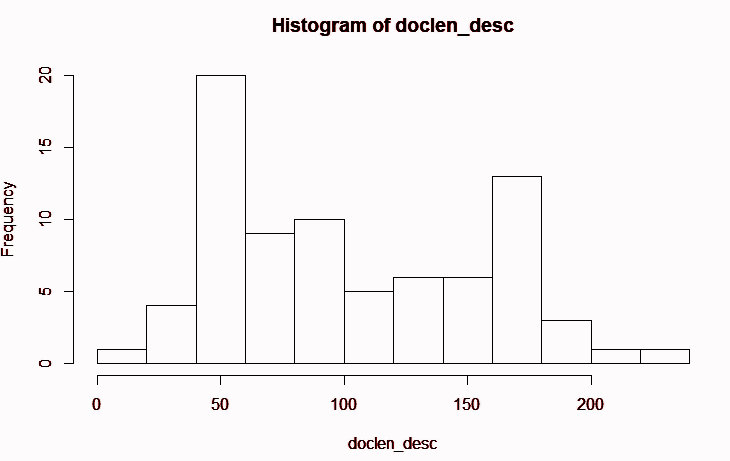
We use the ‘acast’ function to modify the dataset to cater for the requirements for similarity calculation and rating prediction. We selected 100 users randomly as test set and the rest of data as train set for model creation.

As we can see from data exploration, majority of the review ratings are more than 4 stars. To counter to this imbalanced dataset issue, we decide to use pre-normalization technique by minusing each user’s rating by their average ratings.

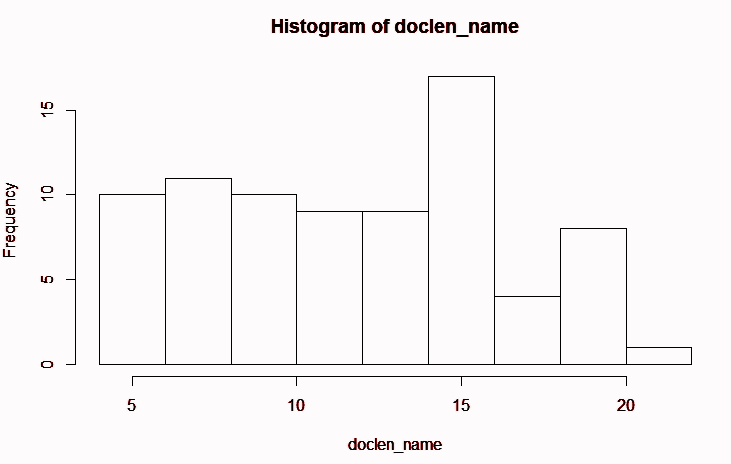
**Exploratory Data Analysis**

Content Based Approach - Data Exploration

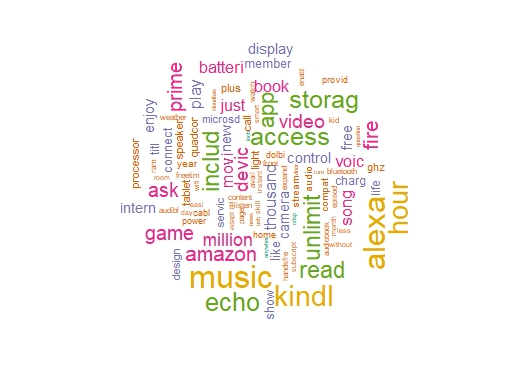
We check the document length from ***Product Description*** by splitting the sentences into a single word. Then plot histogram to visualize the distribution of word with frequency. Document length = 50 appear to be highest frequency. Majority of document length of ***Product Description*** are between 50 - 175.

`

We performed the same actions on ***Product Title*** which consists product features. Document length = 15 appear to be highest frequency. The document length of ***Product Title*** are between 0 - 20 only .



After creating the Tf-IDF, normalizing and processing the text (details steps can refer to process of Content Analyzer), we created the WordCloud to visualize the words in ***Product Description*** which have a higher frequency. Based on the WordCloud, music, Kindle, Alexa which are very popular among the product offering. We included more types of stopwords by referring WordCloud.



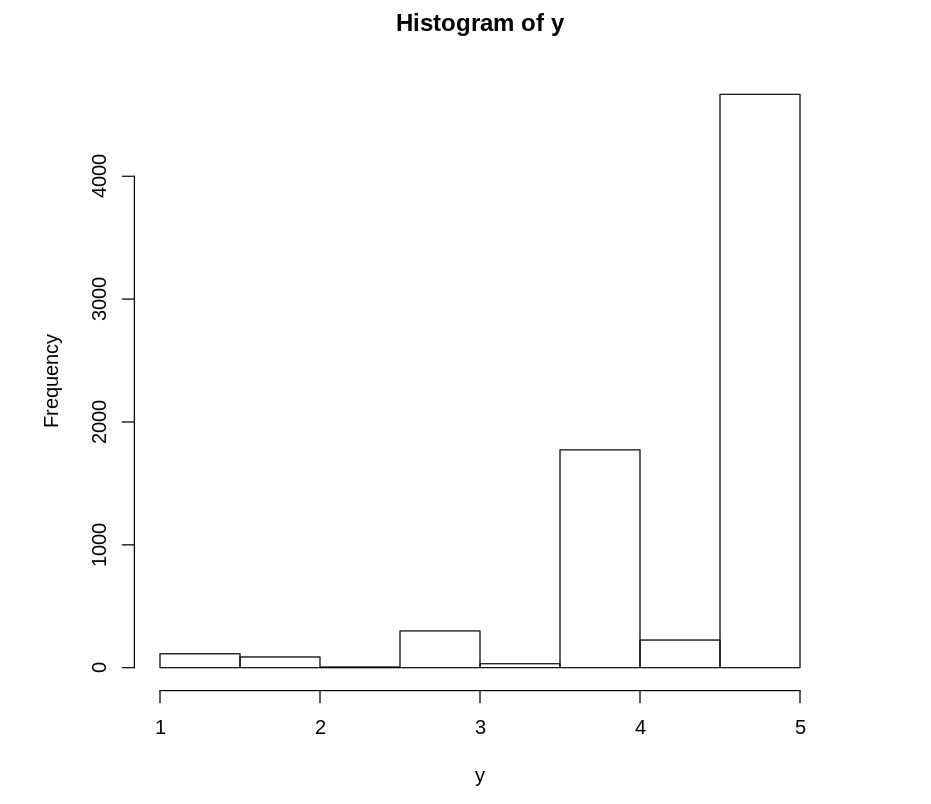
We extracted the product features from ***Product Title*** to understand the product features offer in Amazon. Amazon offers the product mostly consists of keywords like “display”, “kindle”, “fire”, “tablet”, “alexa” and others. These product features keywords can combine with keywords from product description and use as recommender system later on.



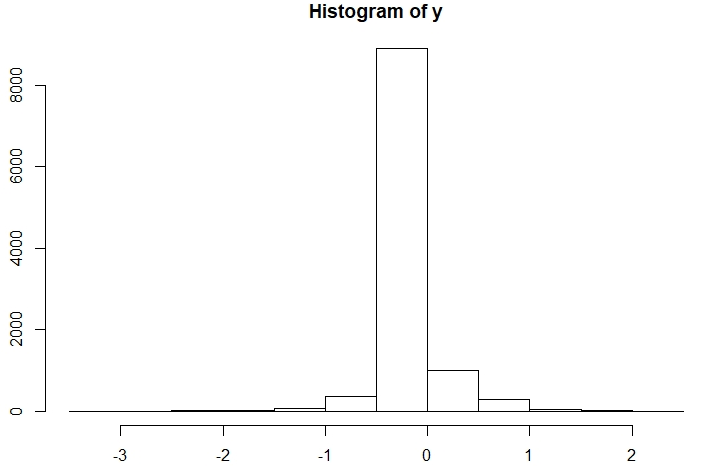
Collaborative filtering Approach - Data Exploration

Before starting the user-based or item-based collaborative filtering, we started to examine whether there are ratings on multiple items by multiple users. There are 79 unique items and 17429 unique users. After removing the inactive users and long-tail products, we left with 52 unique products and 3107 unique users.

After creating the format that ready for similarity calculation and rating prediction, we check the fill rate of users. Only 6.64% filled with product’s rating by the users. Based on the rating distribution histogram chart, majority of users rated 5 for the products offers on website. There are 92.70% users were rated 4 and above.



We normalized the rating data by minusing each user’s rating by the average in order to fix the imbalance data issue.



**Solution approaches**

1. Knowledge-based
2. Content-based
3. Collaborative Filtering
4. Hybrid

**Knowledge-based approach**

The knowledge-based recommender system simply gives a general recommendation of the products based on domain knowledge on item assortment, user preference, etc. It can be applied to cold-start scenarios where information about users’ profiles and ratings is missing. In our recommendation system, we apply the approach to recommend when users have not logged in, or when they have logged in but there is no history of purchasing or rating. We simply recommend items with the highest ratings and most number of transactions among all items. This can help us with the cold-start problem.

**Content Based Approach**

For content based approach the most important data is the data on the content of the product. For our approach, we will be using text analytics methods to extract suitable product features from ***Product Descriptions*** and ***Product Title*** text. We will also be creating user preferences by using some very simple rules based on past transaction data with explicit ratings. Finally, based on item similarity, we are able to get a list of recommended items for the user. By randomly choosing a few from this list, we are able to avoid the issue of always recommending the same category/product to the user.

Our approach will be following closely on the 3 steps recommendation process:

1. Content Analyzer
2. Profile Learner
3. Filtering Component

**Content Analyzer**

In our dataset, we have the product description. We will be using the vector space model.

The process will be as follows:

1. First we will perform unigram model tokenization on the text of ***Product Descriptions*** to break the paragraph of text to individual word called token.
2. Then, we will convert all tokens to lower case to ignore case sensitivity to reduce distinct token.
3. Then we will remove all numbers found in the tokens as numbers without context are of limited use in text analytics application.
4. Then remove any token that is a English stopword. Stop word is defined as words that are extremely common which appear in almost all documents and carry little meaning.
5. Then remove punctuation to again reduce distinct token.
6. Then perform stemming which is to convert a word to its base form using a crude heuristic process. This is also to again reduce distinct token.
7. Then perform stopword removal again due to any token previously not removed because it contains punctuation.
8. Then remove any whitespace found in each token.
9. Using the tokens, we can create a DocumentTermMatrix, which is vector representation of the product descriptions by tokens on the occurrence frequency. By using the above processing effort, this vector has a reduced number of distinct tokens which will result in increased frequency of occurrences of individual token.
10. But instead of using the frequency directly, we will be using Term Frequency - Inverse Document Frequency. TF-IDF is a measure used to evaluate how important a word is to a document in a document corpus. The importance of the word increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

We also observed that the ***Product Titles*** is quite lengthy (4 - 22 words based on our acquired dataset). The product titles itself gives a good description of the product so we will be using it in our content based approach to extract the products’ features, using the same text analytics methods above.

**Profile Learner**

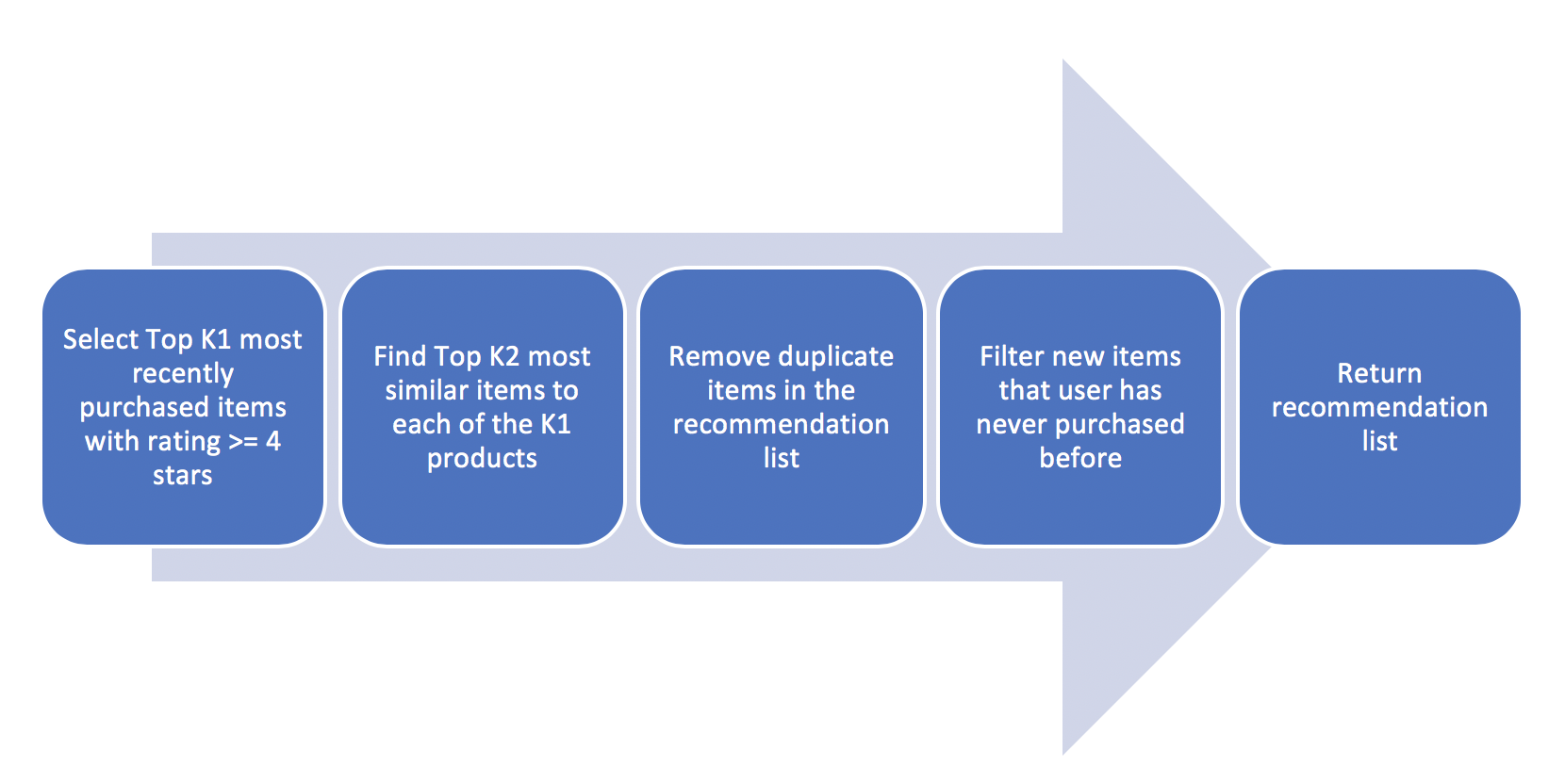
The given dataset includes transactions that a user has completed on the platform with review and explicit ratings on the item. Profile Learner leverages description and feature attributes from those items that the user has purchased and rated with at least 4 stars to recommend similar items. It depends only on users’ previous transactions, making this method robust to avoid the cold-start problem. Based on historical records, we are able to select the 3 most recently purchased items with at least a 4-star rating. This method is our approach to define user preferences and identify products which user preferred to find similar products as recommendations.

**Filtering Component**

We use DocumentTermMatrix to compute the similarity metrics using Cosine Similarity between two products for all the product.

Since we are using both ***Product Descriptions*** and ***Product Titles*** in our recommender system, we will have two similarity metrics (one for each DocumentTermMatrix). We then do a simple addition of the two similarity metrics to combine into one similarity metrics which by assumptions the description and title has equal weightage.

Then based on the product the user selected, we will sort the similarity metrics of the selected products with the rest of the products in descending order. This will become a ranked list of products the system will recommend.



**Collaborative filtering**

There are three famous similarity functions:

* Euclidean distance which is used in classification model K-nearest-neighbors
* Cosine similarity which measured by the rating vectors’ angles
* Pairwise pearson correlation which is similar to cosine similarity, but de-meaned the ratings by subtracting their averages before calculation.

Euclidean distance normally has the worst performance, as the distance measure was often affected by factors which does not reflect the similarity of users or items. The often used example is the document comparison. If a document contains a word A for 10 times and word B for 100 times, this document might still be similar to another document which contains word A for 100 times and word B for 1000 times. While is this also applied to our system? If user rated item A for 1.5 stars and item B for 2.5 stars, is he similar to another user which rated item A for 3 stars and item B for 5 stars? Maybe the first user is a picky one and his 1.5-star is similar to second user’s 3-star? Or maybe these two users are simply so different? Does the rating magnitude matter here? Indeed, we cannot tell for now, and we will test it out in the following analysis.

Pairwise pearson correlation should have slightly better performance than cosine similarity, as different users might have different standards on rating items. For instance, a picky user which rate an item with 3-star might be liking the item as much as another user which rate with 3.5 star. Pairwise pearson correlation remove this bias by deduct the mean from the ratings.

Also, the idea of pearson correlation is quite similar to normalization. What if we apply pre-normalization before we apply euclidean distance and cosine similarity? Will they improve the performance and would it be better compare with pearson correlation? To be honest there is no proof on that and the outcome might differ for different dataset. In the following model building, we will apply all there similarity function, and pre-normalization to find out the best model.

The Collaborative filtering process is as following:

1. Split data into train and test dataset. The seed is fixed here as we want to evaluate model performance based on the same data. The number of test customers is set to 300, which is around 10% of the total 3107 users. The remaining users are used for training.
2. Calculate the user-user similarity matrix (test user’s similarity against train user), using euclidean similarity function.
3. Calculate weighted mean rating for test users, based on the training user’s actual rating and the similarity matrix from step2. This rating is the predicted rating for the test user.
4. Calculate MAE based on the prediction and the actual rating of the test users.
5. Repeat step 2-4 with cosine and pearson similarity functions.
6. Compare the resulting MAE and find out the best similarity function to use.
7. Set the like threshold, in our case, set it 4.
8. Generate the confusion matrix from the like threshold, prediction and the actual ratings, so as to get model accuracy, precision and recall.
9. Perform pre-normalization on the training data and test data and then repeat step 2-8.
10. Evaluate the MAEs and confusion matrix.

**Hybrid Solution**

A hybrid modeling of both content based and collaborative filtering solutions is used when an existing user has login to his or her account. To create the hybrid model, we assembled the results of a recommender system which learns content-based item features from product description, and a deep entity enabling automatic predictions which learns collaborative-based item profile from explicit ratings data. With the User ID provided, we are able to trace back the historical purchasing records, preference on items and item features. Passing the user information to both content based and collaborative filter, we will get two list of recommended items, which will finally be assembled into one list being recommended to users.

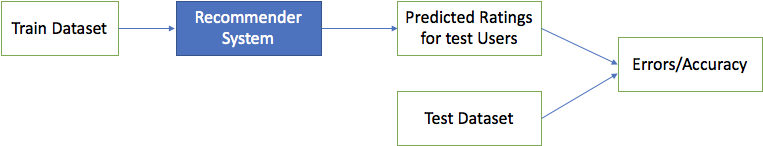
In the context of an e-commerce recommender, a collaborative filter answers the question: “What items have a similar user-rating profile?”, and a content filter answers the question: “What items have similar features?”. By creating a hybrid recommender solution, we’ve attempted to create a system that recommends products that other users rated in a similar manner, while still making on-topic recommendations based on the features of that product.

**Test Frame**

There are two ways to test a recommender system: offline evaluation and online A/B test scoring. In this section, we focus mainly on evaluating our recommender system using offline indicators, including prediction accuracy, precision rate, and Mean Absolute Error (MAE). However, it is never a victory until the recommender system shows positive impact on real customers in a live environment. As users behavior is changing from time to time, positive offline indicators may not necessary lead to positive outcomes in reality. Further information on A/B testing will be discussed in the Limitation section.

While developing a recommender system, it is necessary to measure model performance using offline indicators since preferring to go with online metrics to collect user behavior and scoring your system is expensive and time-consuming. Moreover, when continuous feedback are asked from users, they might become more hesitant to use our platform and not use it at all. Good accuracies in offline metrics followed by good online A/B scores are what we are aiming to achieve.

Fig x.xx. Evaluation Framework



We choose train-test-split approach to test the offline performance of our recommender system. Train dataset, which includes users’ average rating on each item, is used as input to develop the model which generates predicted ratings of the test users on a specified number of items. Errors and accuracy are then used to evaluate through comparing the predicted results versus actual ratings of test users.

The following offline metrics are used to evaluate the performance of each model:

1. Mean Absolute Errors (MAE) is measured as the difference between the actual value(rating) and the predicted value from model output. The lower the MAE is, the better and closer the predicted outcome is.
2. Accuracy is also measured as the overall correct prediction on whether users like or dislike the items.
3. Precision shows when model predicts user will like the item, how often is it correct.

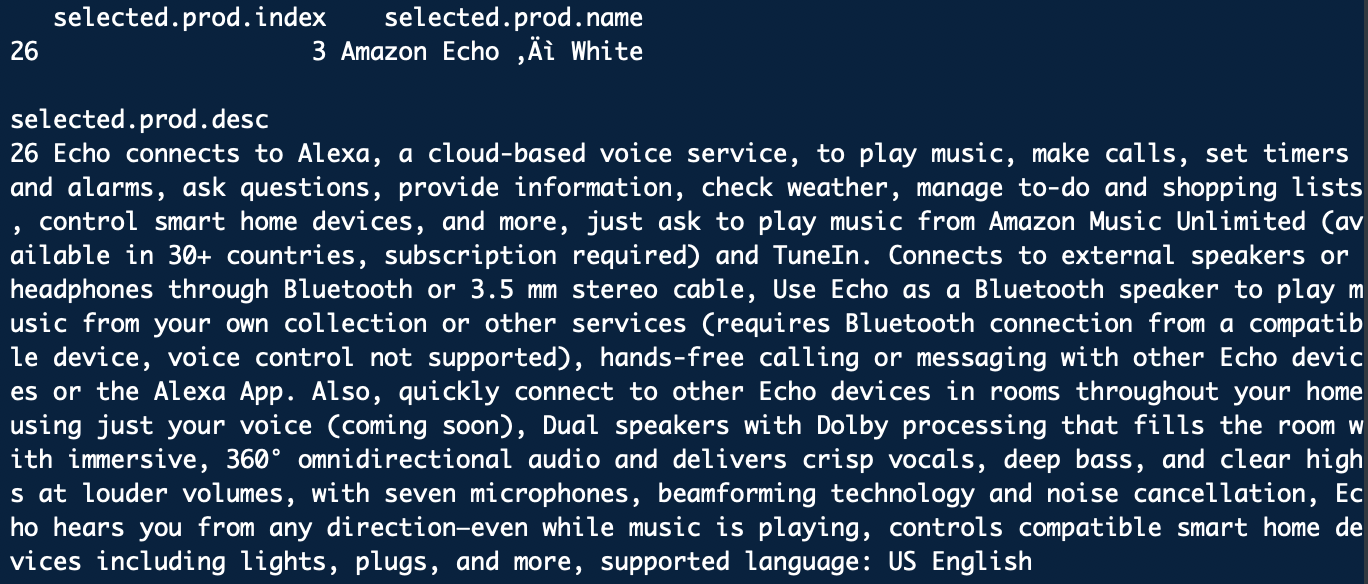
**Model Evaluation and Testing Results**

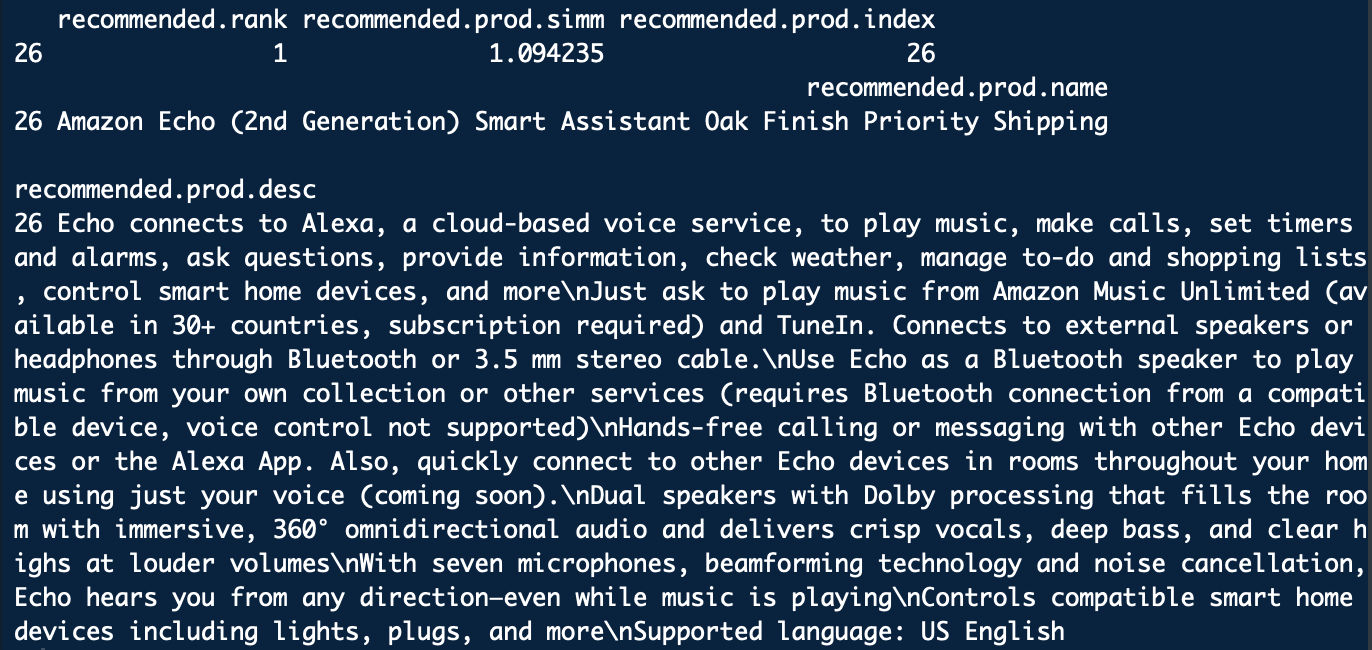
Content Based Model Testing

As we are unable to perform any statistical test to check the accuracy of the model with this approach, we select a random product and go through the information of the recommended products, in order to check if we can find similar keywords which caused the high similarity metrics returned by the cosine similarity algorithm.

Below are some examples of our testing:

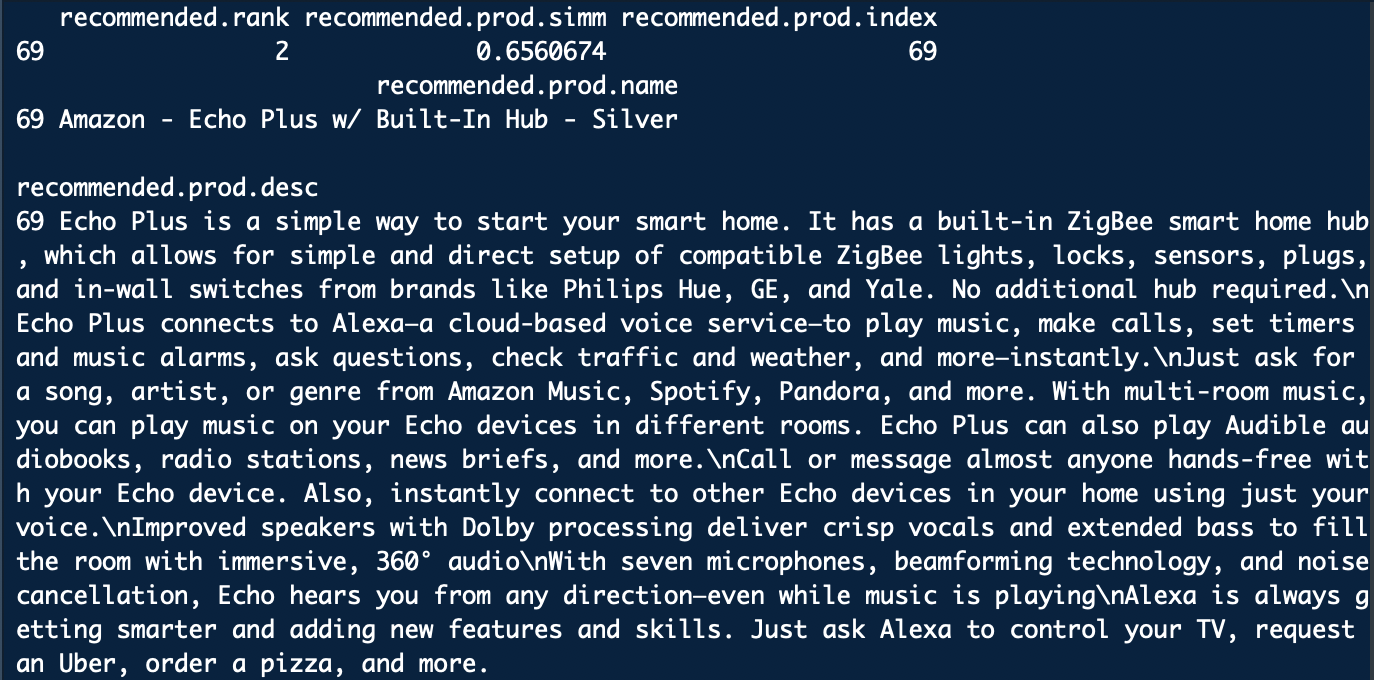
We selected a product and ranked 1 of the recommendations actually having the exact same product description only with a different product title.





From the ranked 2 of the recommendations, we can see common words/phrases between it and the selected products which means the two documents are very similar.

E.g: Echo, Alexa, a cloud-based voice service, smart home, play music



Collaborative Filtering Model Testing

From the testing results on user-user collaborative filtering model with raw dataset, it is very interesting to see that euclidean similarity function is having the best MAE value of 0.35, which is much better than the cosine similarity with MAE of 0.52. The explanation here can only be that the magnitude of the rating really matters in this model, which cause Cosine similarity performed the worst as it completely ignore the magnitude.

While the ratings was so skewed that Pearson Correlation with MAE of 0.43 out-performed a lot than cosine similarity as it de-meaned the ratings before calculation. This results also implied that after pre-normalization, we should see significant improvement on Cosine Similarity, and make it very close to Pearson Correlation.

After applying the pre-normalization on raw dataset, the new MAE for cosine similarity is 0.39, which is actually improved a lot, and better than Pearson Correlation without normalization. Also as expected, pearson correlation did not change much after pre-normalization, as it had already done something similar. Nevertheless, euclidean similarity function still performed the best with a MAE of 0.14. The confusion matrix for euclidean similarity is as following (with a normalized like threshold at 0.001):

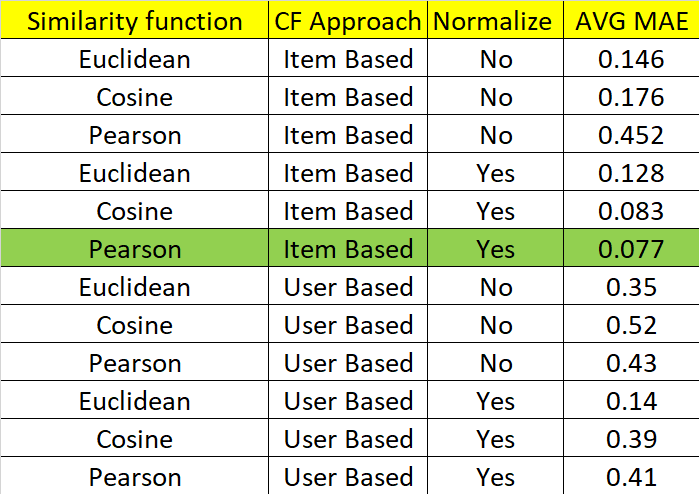
* Accuracy = 73.5% implying that out of 100 predictions made on whether the user will like or dislike the product, around 73 are correct.
* Precision = 31.0% implying that out of 100 “like” predictions, around 31 of them are actually like by the user.
* Recall = 64.4% implying that out of 100 “dislike” user reviews, 64 of them were identified correctly.

Now we have finished the user-user collaborative filtering approach, and we have learnt that utilizing pre-normalization with euclidean distance as the similarity function will yield the best model.

Nevertheless, as we can see from the dataset, there are 52 unique items and 3107 unique reviewers. It actually makes more sense to use item-based collaborative filtering approach rather than user-based, in order to achieve better operating performance. Item-item based approach is very similar to user-user based, the only difference was that instead of calculating user similarities, we calculate item similarity. The best item-item based model showed a MAE of 0.077 with pre-normalization and pearson correlation, significantly better than the previous user-user based model. The confusion matrix also indicated very good performance with a normalized like threshold at 0.001 :

* Accuracy = 96.1% implying that out of 100 predictions made, around 96 are correct.
* Precision = 88.8% implying that out of 100 “like” predictions, around 88 of them are actually like by the user.
* Recall = 84.4% implying that out of 100 “dislike” user reviews, 84 of them were identified correctly.

With the above result, we can conclude that for collaborative filtering method, Item-Item based approach with pre-normalization and pearson correlation worked the best.



*Summarize of CF Models Evaluation*

**Limitations**

1) Lack of online performance evaluation

In some cases, a recommender system can perform well in terms of offline performance measurements, but very badly once it is productionised. This is because the model is trained based on historical data and user preference. However, user behaviour change over time. Therefore it is necessary to measure the online performance of a recommender system. One way is to do online A/B test scoring evaluation.

During the pilot stage, instead of integrating the recommendation system into all potential candidates, we should just sample a small group of users and divide them into A/B test groups. Measuring the conversion rate of both groups would give a sense of the efficiency of recommender system.

2) Lack of uniqueness in item level data

In the Amazon dataset, each unique model is given a specific item ID. This means that the same item with different colors and different storage will have multiple item IDs. In our model, those will be treated as different items. As a result, it is possible that the top recommendations are all Amazon Kindle, but just different models or specifications.

3) Cold start problem

This is a typical problem in recommender systems. This problem is often caused by the lack of information, on users or items. There is relatively little information about a new user, which results in an inability to draw inferences to recommend items to users. No historical information on the new user’s preference, ratings or view history are available. Therefore collaborative filtering cannot calculate similarity among the users. However, content based methods can provide recommendation in case of new item as they do not depend on any previous rating information of other users to recommend the item.

4) Imbalanced dataset

Majority of the transactional records have good ratings (more than 4 stars). This imbalanced data could lead to high accuracy shown in the offline testing. However, when it goes on to production, the Recommender System may not show promising results. To combat the imbalanced data issue, on top of pre-normalization, one way we can do is to use k-fold cross validation method when splitting train and test. By repeating the resampling several rounds, we could combine the results from each round.

**Conclusions and Future Plans**

Based on business requirements, we have built three recommender systems with combination of different methodologies, including Knowledge-based for general product recommendation, Content-based method using product features and Collaborative Filtering method using customer explicit ratings. The systems built are a series of recommender systems for various stages (including before login, after login, searching, etc) using different methods, so that accurate and customised recommendation of products can always be made at different touch points.

Testing results for CF model give very low MAE and very high accuracy, which indicates the model performs well to provide customised recommendations based on accurate prediction on user rating. As a future plan, we can evaluate online performance of this model by conducting A/B testing to measure the real impact on the business.

As the dataset itself lack some important data such as customer view history, it brings challenges to recommend accurate and customised items to new users. In addition, the number of items are also rather limited. Hence, we could work with other teams to retrieve more data, so that we can further improve our models and have a wider variety of recommendations. With view history information, we may also turn the view history into implicit ratings to further aid collaborative filtering recommendation.

More methodologies could also be applied to further enhance and improve the performance of our recommender system. For instance, Model-based approach like Matrix Factorisation could be a supplement to the above-mentioned methods. We may load the rating data into sparse matrix and create recommender instance in recosystem library to do factorisation, and search for the optimal latent features with the best MAE value. The items recommended by the Matrix Factorisation can thus further hybrid with other CF models as well as CB results to give better recommendations.