Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

Course Projects

CS7IS5-A-SEM202-201920

ADAPTIVE APPLICATIONS

Fall 2019

Adaptive Purchase Planner

Team James:

Sakina Vohra	19322611
Nikhil Girraj	19315753
Eoin Roche	15323933
Rohan Bagwe	19314431
Jiajin Zhao	19300809
Sen Yan	19315814

Trinity College Dublin, the University of Dublin

Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

Contents

Abstract:

Many people visit supermarkets at least once a week. Every supermarket tries to maintain their loyal customers by storing the history of transactions made by a customer and recommend new buys from the shop. We have come up with a solution that focuses on the products rather than the store. Our system stores overall purchases or transactions made from different supermarkets and understands the user purchasing behaviors which it uses to recommend users the products they might want to buy the moment they are in a supermarket or product that would be good to buy in advance as a shortage is expected soon. We implement this system using Collaborative filtering, Content-based filtering (Algorithmic approach and Machine Learning approach). We also considered the scenario where a user doesn't respond to a recommendation for a set period of frequency, then the product is removed from the list. This means, the suggestions adapt to user's behaviors and tries to recommend the most useful products users might want to purchase.

1 Introduction:

As the popularity of smartphones continues to increase, the number of similar applications increases exponentially. Mobile applications that adapts dynamically to user behavior, seen in the app or via sensors in the phone are becoming increasingly popular too. More and more applications are being built to structure and improve the daily mundanities of a user's life and these users are beginning to expect such applications to exist.

A survey based in the UK from 2015 suggested that 56% of people go grocery shopping at least once or twice a week [1]. We saw an opportunity in that activity for an adaptive application. The Adaptive Purchase Planner (APP) was pitched as an application that would suggest the items a user needs, when a user needs them. We would use the premise that humans are creatures of habit, how tend to follow routines, including when they purchase groceries.

That assumption allowed us to model a system for content-based filtering that would use the users purchase history to suggest future items. We also wanted our system to be able to suggest new items a user had never seen before. However, we did not want this to be a purely random exercise. So, we utilized collaborative filtering, this would allow us to divide our users into stereotypes or demographics and suggest items to users based on what group they are assigned to.

We also wanted to allow the user to have a degree of control over their model. This is done by allowing a user to explicitly reject an item they no longer want to be recommended, until they manually add it back into the system.

Recommendation Methods:

In this section we will expand on our understanding of the two methods of adaptation that our application uses - Collaborative Filtering and Content-based Filtering. We believe these two methods twin nicely, the Content-based Filtering provides day to day recommendations for regular users and the Collaborative Filtering serving as a bridge to new users as well as a mechanism for irregular users, whose model may have become stale.

- **Challenges:**

  Current recommendation systems tend to be in the same lines which is within a fixed scale or discrete, like recommending movies or music based on ratings from 0-5. However, building a recommender solution for making purchases has a continuous nature and could be without any upper bound. Shopping in a supermarket has a buy-consume-buy cycle, whose consistency and frequency may vary from User to User depending on various factors like Age, Household, Location etc. Feature Engineering is the key to better understand what all data is needed to build the User model.

- **Goals:**

  For our Adaptive Application, our goal is to come up with the best approach that can recommend users the products they might want to purchase at the given day or in coming days based on Users previous purchasing history.

  Our four main end goals are:

  1. **The recommendation of goods to buy now**

     This functionality allows the users to see the items they need to purchase now.

  2. **The list of goods to buy later**

     This functionality recommends items that they might fall short of in coming days and hence they could stock it up in advance.

  3. **The list of goods picked most times**

     This functionality shows the best buys amongst all users.

  4. **The list of missed items.**

     The items are recommended twice in the interval of frequency after which it is sent to the ignored list. The user can view the ignored products here and they have the functionality to permanently remove the products.

2 Preliminaries:

As our deliverable, we submit our MVP which will highlight the main features of our application. Our application gives suggestions to users about the daily goods they would want to buy.

At first, users would get purchase recommendations based on the similarity to other users in our database dataset. After using the application for a small period, they would receive more recommendations based on their own purchasing behaviour, which would lead to a more appropriate result than those merely based on the similarity in users' basic characteristics.

This application has laconic user interfaces, which will provide users with satisfaction when using it. All the users need to do is to enter the users ID and click the button for different functions.

2.1 Requirement Analysis:

This system is required to enable users to know the circumstances of products they have bought and give recommendations to users about what they may need to buy for the next time they go shopping.

The inputs for this system should be the ID of a certain user who have already registered and/or their purchasing records.

The outputs for this system should be items and the purchasing date for a specific user even when there is no purchasing record in the dataset.

2.2 Project Development Plan:

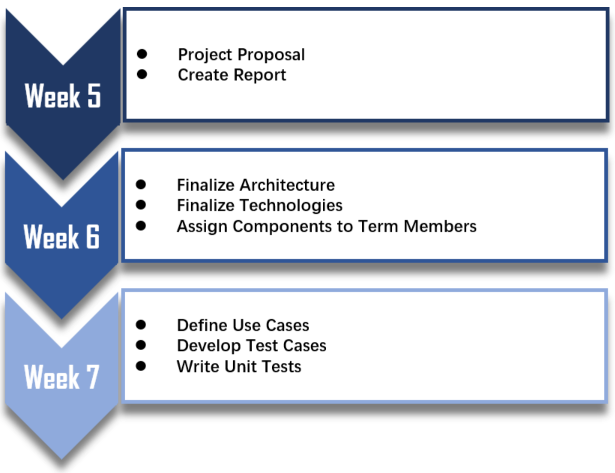


Fig 2.1 – Timeline (1)

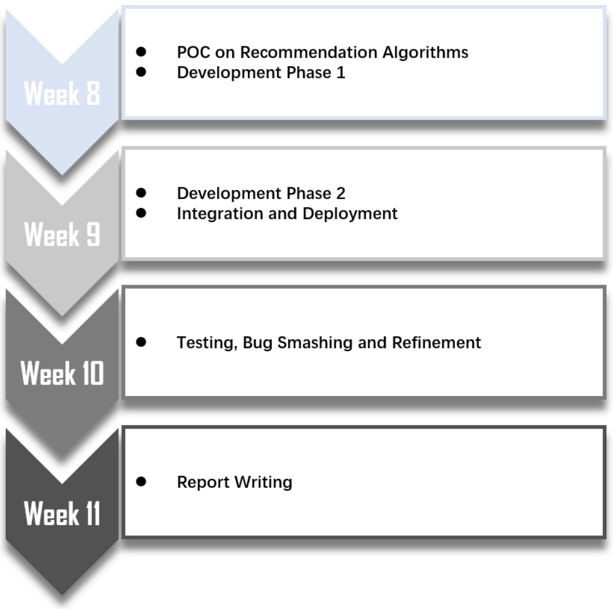

Week 8
POC on Recommendation Algorithms
Development Phase 1
Week 9
Development Phase 2
Integration and Deployment
Week 10
Testing, Bug Smashing and Refinement
Week 11
Report Writing

Fig 2.2 - Timeline (2)

The goal for our application is to make a clean and user-friendly design. We want users to have easy access to their generated shopping list with our suggestions, while giving them a significant amount of control to edit the list in real-time, while the system adapts to what the users choose to filter out. Also, crucially the application will always be keeping track of the total cost of all items on the list and will help users keep track of their spending habits over time.

2.3 Functional Architecture

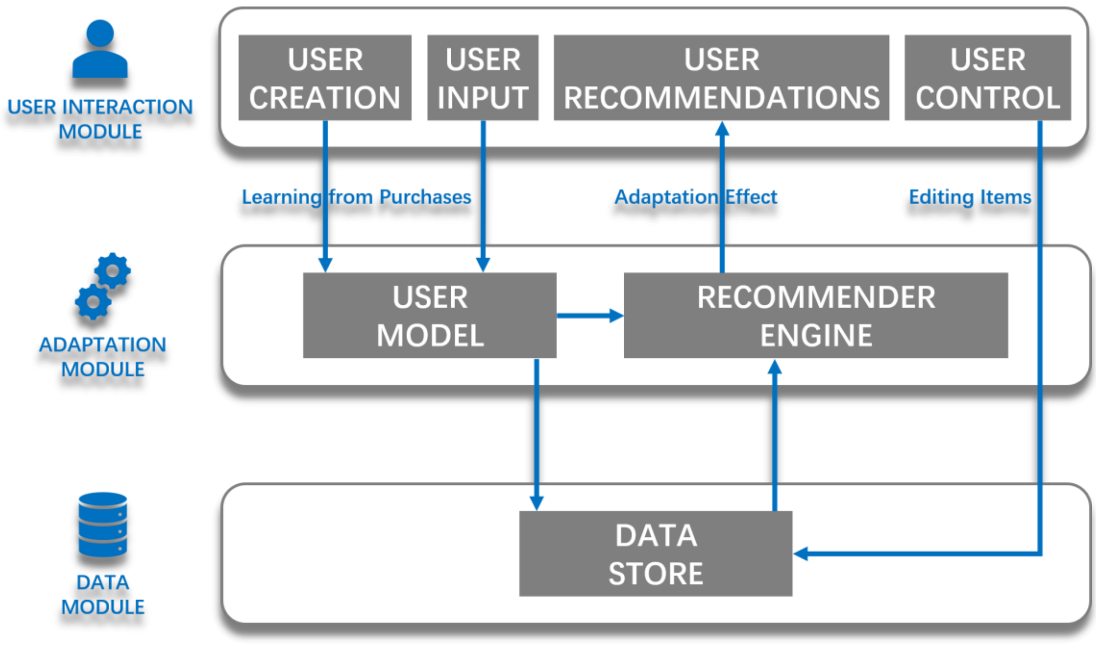


Fig 2.3 - Functional Architecture

Description:

- **User Interface:**

  In this section we will detail the initial UI we planned to build and the final UI we used for the demo, while contrasting the differences between them and explaining how they contribute to the user modeling and user control.

  - **User Model Creation:**

    Our initial plan for user creation would require a user to give us important details for their use model including their:

    - Age
    - Gender
    - Household Size

    These features would help us to classify the user into a stereotype by using Collaborative Filtering, to give the user initial recommendations.

  - **User Input:**

    In our first proposal we wanted users to be able to input data in two ways, through manual entry and through scanning of receipts. To scan receipts, we planned on using Firebase ML kit to extract all the text from a receipt and build objects representing those items from that text. The manual user input could be handled using standard forms. The receipt had one problem we say from the start. Different shops have different receipts formats and names for the same items. So, our plan was to circumvent that by using generic receipts that we made ourselves.

  - **User Recommendations:**

    We have 4 different sets of recommendations:

    - **Buy Now:**

      The recommendations users should purchase today. These are items that a user would normally buy n days after they last bought them.

    - **Buy Later:**

      The items a user should purchase soon. Includes how many days in the future a user should buy an item. This can help a user budget in the short term.

    - **Missed Items:**

      These items have been ignored, intentionally or otherwise, by the user. We assume at first that a user has simply missed this recommendation but if they ignore it multiple times, we assume the user has now rejected that item.

- **Top Picks:**

  This tab shows the user their recommendations based on our Collaborative Filtering and will always be a list of the most popular items from the user's stereotype that they have never bought before

Lastly in the demo you will see the Purchase Behavior Screen. This will show a set of algorithmic purchase frequencies and the Machine Learning purchases frequencies for a certain user.

- **User Control:**

  As a focus of our project is user control, the UI plays a vital role in that. We wanted users to be able to explicitly add an item to the system, so this can be done in the Buy Now page. That item will be added to the user model if it is new, or we will update the buy later date for that item if it has been bought before. A user must also be able to explicitly reject an item. This can be done from the missed purchases screen, hitting the 'x' under the "Not Interested" Column removes the item from the user model, and it will be filtered out of any future recommendations, unless the user manually adds the item again as above

- **Adaptiveness:**
  - It understands from user's previous purchases and recommend items.
  - It suggests items the user might need in a few days.
  - It suggests new items the user might also like based on other user's purchasing behavior.
- **Data:**

  This module will store user models, users purchasing records and the users' behaviors when using this application. Data collected from records would be used to give better suggestions, and users' behaviors are stored to adjust the frequency of recommendations for different items. At the same time, items details such as prices, quantities and locations will be stored in a database.

2.4 Project Management

2.4.1 Software and Technologies

- **Backend Technology:** Python (Flask) hosted on Microsoft Azure cloud
- **Frontend Technology:** HTML with JavaScript

- **Database**: MySQL
- **Receipt Scanning:** Firebase ML Kit for Image to Text Conversion
- **Machine Learning:** Scikit-Learn

2.4.2 Communication

Slack **WhatsApp**

3 Implementation:

Our code can be seen here:
https://github.com/rohan-tcd/TCD-Team-James-Adaptive-Purchase-Planner

3.1 Data Acquisition:

The application at its heart, is a data processing and statistical analysis tool. This directly makes the availability of data as a key requirement for development. The team conducted investigations to identify the nature of information that would be required to achieve the goal of being able to predict the purchase requirements for a large variety of users. Based on the research, "A Model of Household Grocery Shopping Behavior" [4], the team discovered that the factors that align with the purchase behaviors of a household are Shopping Trip Frequency and Expenditure per Trip. From the same research, the underlying features that contribute to those factors were found to be –

- House Hold
- Age of the Head of the Household
- Number of Stores Shopped
- Employment Status

In addition to the above, other factors like Income of the household, employment status and composition of the household are found to be contribute to their grocery shopping behavior.
The team attempted searching from a variety of data sources to find a dataset to fit our requirements. The requirements included the above-mentioned features, as well as volume of the data to be able to build a machine learning model from it. Following datasets were considered –

- The Online Retail Data Set by UCI
  http://archive.ics.uci.edu/ml/datasets/Online+Retail
- The shopping analytics datasets by Dunnhumby
  https://www.dunnhumby.com/careers/engineering/sourcefiles
- The dataset used in the research, "A Model of Household Grocery Shopping

Behavior"
(link no longer available)

These datasets could not be used because of either lack of volume, lack of the right features, availability or the complexity of the features. For future work, I would certainly choose the dataset (b), made available by Dunnhumby, which we could not utilize due to the complexity and sheer size of the data.

The current work still manages to model the shopping behavior sufficiently by generating data as described in the research mentioned above.

3.1.1 Approach to data generation:

The idea of generating the data was partially inspired from one of the Kaggle competition's synthetic datasets used for the income prediction exercise [5]. The data was generated by following a bottom up approach where the independent variables were generated first, and then correlated and randomized variables were added incrementally. Following features were generated in the dataset to model the shopping patterns –

Following features were generated in the dataset to model the shopping patterns –

- **Users –**
  - Gender - assigned from a set of 'M', 'F' and 'O' in a 3:3:1 ratio. This was however not utilized in modelling the user in the current scope.
  - Names - Random (gender wise)
  - Year of Birth - Randomly assigned to make the age fall in a range of 18 - 65 years.
  - Size of Household - Randomly assigned in a range of 1 to 10 with a skew towards smaller numbers using normal distribution and low pass filtering.

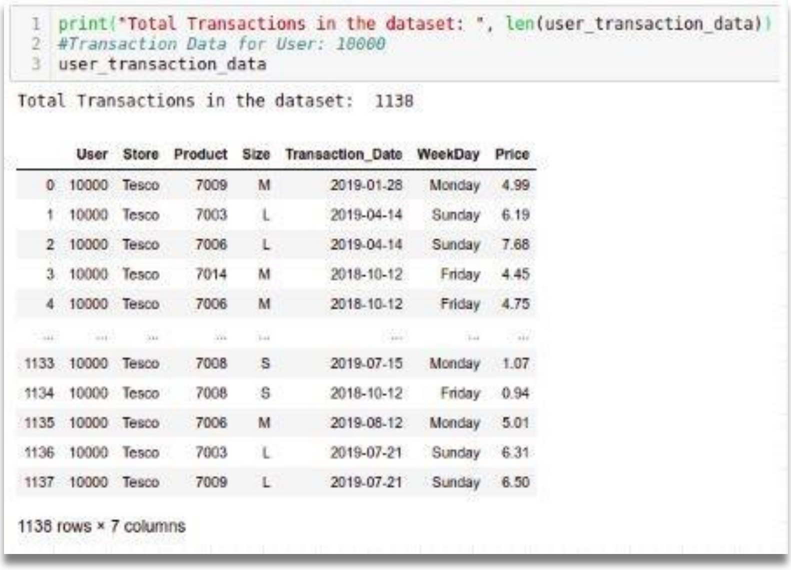
- **Products –**
  - Categories - 8 categories including Snacks, Personal Care, Cleaning, and Dairy from common grocery items were chosen to keep the model simple.
  - Items - A total of 16 items were created under these categories
  - Size - To keep the model simple, the sizes were kept as Small, Medium and Large instead of actual weights and volumes with the hypothesis that most of the goods either come in 2 or three sizes.
  - Average Price of the Product - A uniform distribution of prices was assigned to the smallest size of the product. Every larger size was assigned price higher than the smaller size by a factor of uniformly distribution between 0.3 to 0.8.

- **Stores (the names do not carry any relation to the real stores) -**
  - Name - 5 distinct names

- **Product Availability -**

  - Product - As generated before.
  - Size - As generated before
  - Store - As generated before
  - Price at Store - A 30% negative or positive variation from the average price of the product and size.
  - Is Listed - A Boolean stating whether the product is ever listed at the store or not.
- **User x Product (shared feature) -**
  - Product Affinity - Each user is assigned an affinity factor to a random number of products using multivariate beta (Dirichlet) distribution.
- **User x Stores (shared features) -**
  - General revisit tendency - In days, a Gaussian distribution with a mean of 7 and a standard deviation of 2.5. This is based on the observation from one of the Dunnhumby datasets where it is observed that transactions tend to reappear for the same customers mostly within 6-8 days with very few consumers visiting stores daily, and equally fewer with a frequency of more than 15 days.
  - Each user was also assigned a "laziness factor" which would dictate how often would they be delaying a visit to the stores for grocery. This is also like the incidents where users delay store visits due to their schedule.

- **Transactions (purchases made by users) –**
  - Day - A value between 1 to 500 with an interval of general revisit tendency plus the randomized laziness factor for each trip.
  - Store Visited - Store chosen according to store preference.
  - Product - A product chosen depending on product affinity and a visit budget. Visit budget is
  - Size - Generally larger for bigger households, directly proportional to the household size and inversely proportional to the GVT.



```
print("Total Transactions in the dataset: ", len(user_transaction_data))
#Transaction Data for User: 10000
user_transaction_data
```

```
Total Transactions in the dataset:  1138

       User  Store  Product  Size  Transaction_Date  WeekDay  Price
   0  10000  Tesco     7009     M        2019-01-28   Monday   4.99
   1  10000  Tesco     7003     L        2019-04-14   Sunday   6.19
   2  10000  Tesco     7006     L        2019-04-14   Sunday   7.68
   3  10000  Tesco     7014     M        2018-10-12   Friday   4.45
   4  10000  Tesco     7006     M        2018-10-12   Friday   4.75
 ...    ...    ...      ...   ...               ...      ...    ...
1133  10000  Tesco     7008     S        2019-07-15   Monday   1.07
1134  10000  Tesco     7008     S        2018-10-12   Friday   0.94
1135  10000  Tesco     7006     M        2019-08-12   Monday   5.01
1136  10000  Tesco     7003     L        2019-07-21   Sunday   6.31
1137  10000  Tesco     7009     L        2019-07-21   Sunday   6.50

1138 rows × 7 columns
```

Fig 3.1 - User Purchase History

3.1.2 Features for Optimal Solution:

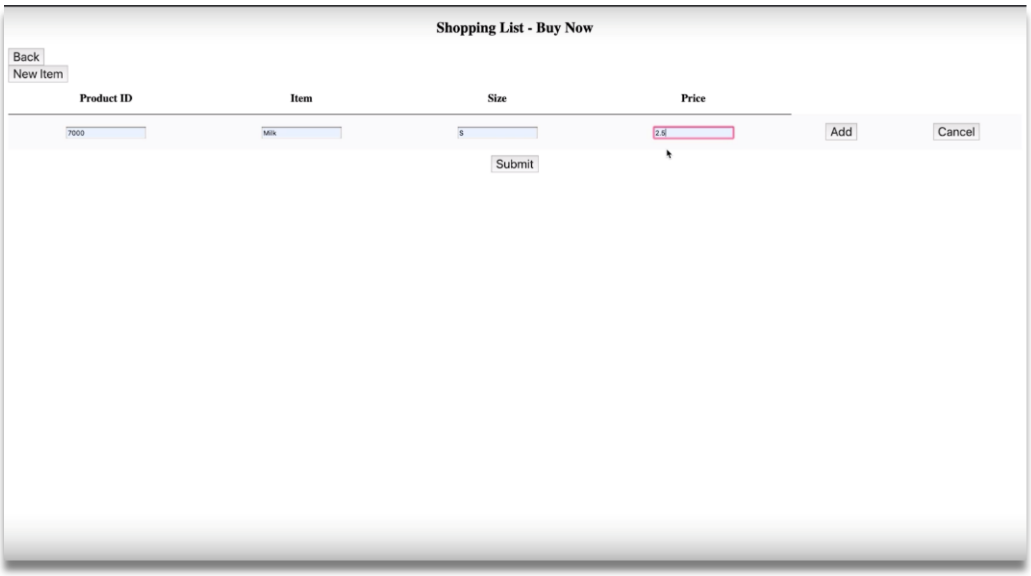
The dataset was objectively analyzed to extract the underlying user model by each team member to without the knowledge of how the data was created. This led to discovering different forms of predictions that could be made based upon how the user is modelled. Our team implemented all the ideas to observe the outcomes of applying different machine learning principles as well as general statistics to certain features of the datasets which would lead to make predictions for the user.

3.2 User Input Module:

To add an item, that is the products purchased, a user can either enter details manually or scan receipt to add an item.

3.2.1 Manual Input

To add an item a user can choose new item under the Buy now section of our application, the user can either choose the item from the list of buy now items or manually enter details of a new item.



Shopping List - Buy Now

Product ID	Item	Size	Price
7000	Milk	s	2.5

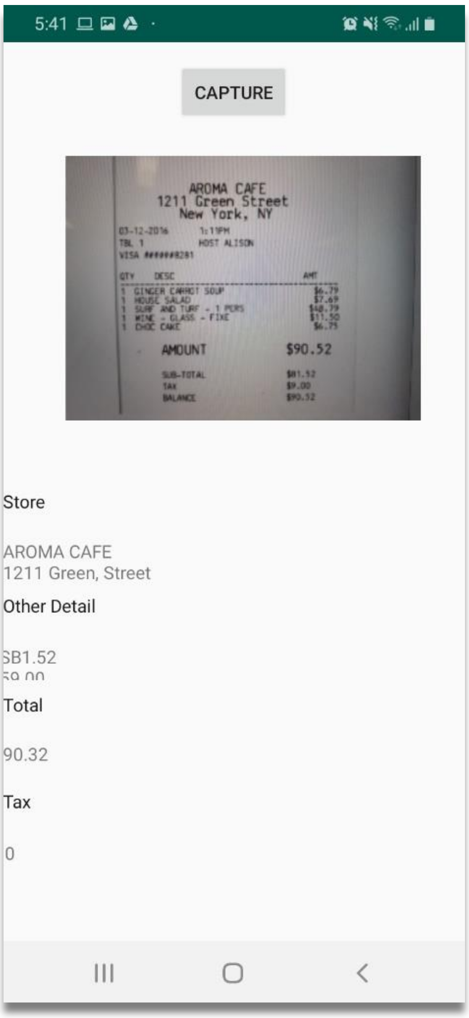
Fig 3.2 - Adding New Item

3.2.2 Scanning Receipts

For scanning receipts, we came across multiple solutions, but what worked best for us was the Firebase ML Kit for Image to Text Conversion. We created an app, that could capture an image from the phone and then detects text from this image.

How ML kit works here is, it gives us the blocks of text line by line. We used the On-device API because it is much faster. The detection feature clusters the texts under boxes per lines which we can extract with multiple loops. We were able to detect the Store, considering that it's the first text on receipt, the total amount considering that it is the highest float in the receipt and TAX. However, it didn't work for all receipts as some didn't highlight the total amount or some didn't have the store name at all.

Another challenge was retrieving each record from the receipt and converting it to quantity, product and price. Different receipts have different way of printing it. The biggest issue for retrieving records were that it was hard to analyze the end of list; there is no marking that indicated that the list has ended except for a space until total amount is printed which may or may not be the case for all receipts.
Below is the screenshot of detected receipt and output. We needed further time for research and development of this part.



```
          AROMA CAFE
       1211 Green Street
         New York, NY
03-12-2016     1:11PM
TBL 1          HOST ALISON
VISA ######8281

QTY   DESC                          AMT
---------------------------------------
1  GINGER CARROT SOUP             $6.79
1  HOUSE SALAD                    $7.69
1  SURF AND TURF - 1 PERS        $48.79
1  WINE - GLASS - FIXE           $11.50
1  CHOC CAKE                      $6.75

      AMOUNT              $90.52

      SUB-TOTAL           $81.52
      TAX                 $9.00
      BALANCE             $90.52
```

Store

AROMA CAFE
1211 Green, Street

Other Detail

SB1.52
59.00

Total

90.32

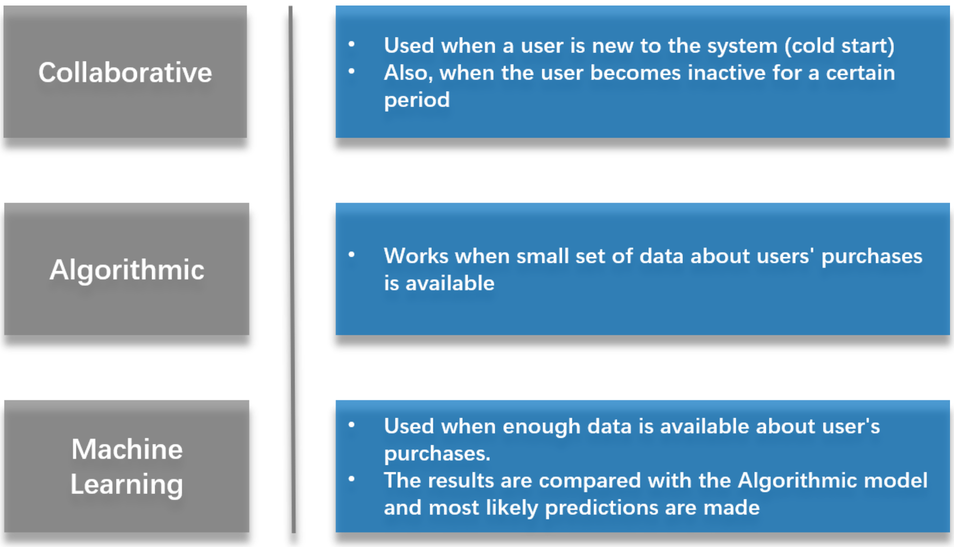
Tax

0

Fig 3.3 - Scanned Receipt

3.3 Adaptive Module:

3.3.1 User Models [2]:



- Used when a user is new to the system (cold start)
- Also, when the user becomes inactive for a certain period

- Works when small set of data about users' purchases is available

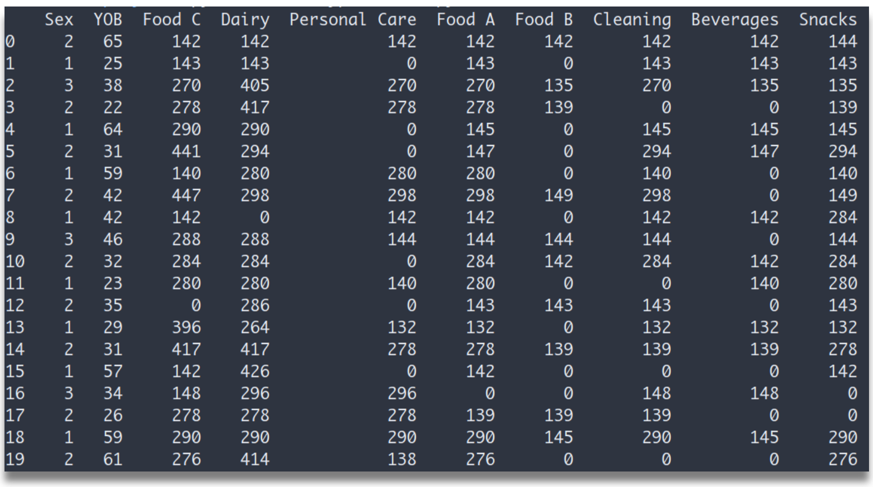
- Used when enough data is available about user's purchases.
- The results are compared with the Algorithmic model and most likely predictions are made

Fig 3.4 - User Models

3.3.1.1 Collaborative:

The system can find similar users and recommend items to a user based on those users who are similar. The System uses the following method to achieve it.

Features Selection:



```
    Sex  YOB  Food C  Dairy  Personal Care  Food A  Food B  Cleaning  Beverages  Snacks
0     2   65     142    142            142     142     142       142        142     144
1     1   25     143    143              0     143       0       143        143     143
2     3   38     270    405            270     270     135       270        135     135
3     2   22     278    417            278     278     139         0          0     139
4     1   64     290    290              0     145       0       145        145     145
5     2   31     441    294              0     147       0       294        147     294
6     1   59     140    280            280     280       0       140          0     140
7     2   42     447    298            298     298     149       298          0     149
8     1   42     142      0            142     142       0       142        142     284
9     3   46     288    288            144     144     144       144          0     144
10    2   32     284    284              0     284     142       284        142     284
11    1   23     280    280            140     280       0         0        140     280
12    2   35       0    286              0     143     143       143          0     143
13    1   29     396    264            132     132       0       132        132     132
14    2   31     417    417            278     278     139       139        139     278
15    1   57     142    426              0     142       0         0          0     142
16    3   34     148    296            296       0       0       148        148       0
17    2   26     278    278            278     139     139       139          0       0
18    1   59     290    290            290     290     145       290        145     290
19    2   61     276    414            138     276       0         0          0     276
```

Fig 3.5 - A sample output for features

This is a sample training data for our collaborative recommendation system. Every row is a record for a different user. Every column is a feature. Below are explanations of every field and why we choose these fields as our features

The data we chose as our features are as below:

- **Age**

  When a user registers in our system, the user needs to choose his/her year of birth. And our system will use it to calculate the age for the user.

  People of different ages tend to choose different items.

- **Gender**

  When a user registers in our system, the user needs to choose his/her gender. People of different genders tend to choose different items; our system will use this information to find similar users.

- **Users' purchase behavior in every category**

  Our system has predefined categories and every item can belong to at least one category. We will retrieve the shopping history of a user and group these items using their categories. As we can see from Fig 3.5, User 0 has bought food C 142 times, diary 142 times, etc.

  This is the most important feature to identify users with similar interests. We calculate how many times does a user by things in every category, then our system will use every category as a new feature for a user.

Model Creation:

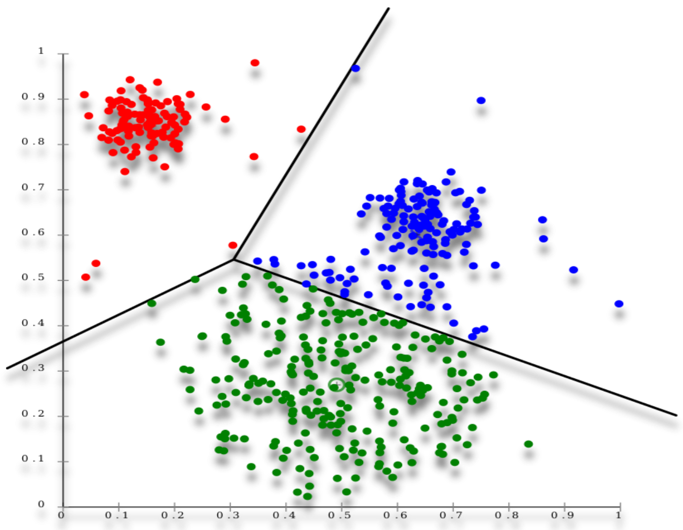


Fig 3.6 - K-means Clustering

This is a sample output for the K-means clustering algorithm. K-means clustering is a machine learning algorithm that can divide the dataset into different groups base on Euclidean distances between different instances, instances with smaller distances are more similar and tend to be classified into one cluster. Our system uses this method to find similar users.

Our training has more than two features, but we are not able to show all the features in a 2D diagram. So here we only choose two features to draw this diagram. The x-axis represents how many times does a user bought items in food C category before, the y-axis represents how many times does a user bought items in diary category before. We also scale data for all users, so they are all in the same range from 0 to 1.

K-means algorithm to find the similarity among users

- **Apply data scaling on every feature**

  Since K-means clustering use distance to cluster different entities, we must scale values of all the features into the same range in order to avoid some features will have higher priority than other features.

- **Apply K-means clustering**

  The system will use K-means clustering to divide people into different groups. people in the same group will have similar interests. Then we can use this information to recommend something that a user might like.

  The system will decide the number of clusters dynamically. We have an upper bound and a lower bound for the number of people in one group, so we will have a range for the number of clusters. Then, we will apply K-means clustering to all the values in this range. And we will choose the model using the silhouette score. The silhouette score can measure the distance between the result clusters, it has a range of [-1,1]. The model with a higher silhouette score means the distances between clusters are larger. So, we will choose the model with the highest silhouette score.

- **Divide people into different groups**

  The system will use the model and divide the people in the cluster into the same group.

Recommendation:

Prediction based on similar users

- **Find all the similar users in the same group**

  After receiving the user that needs recommendations, the system will find the group which the certain user belongs to and find all the similar users in this group.

- **Analyze users purchase behavior in the group**

  Then the system will calculate the most popular items in this group.

- **Recommend based on group purchase behavior**

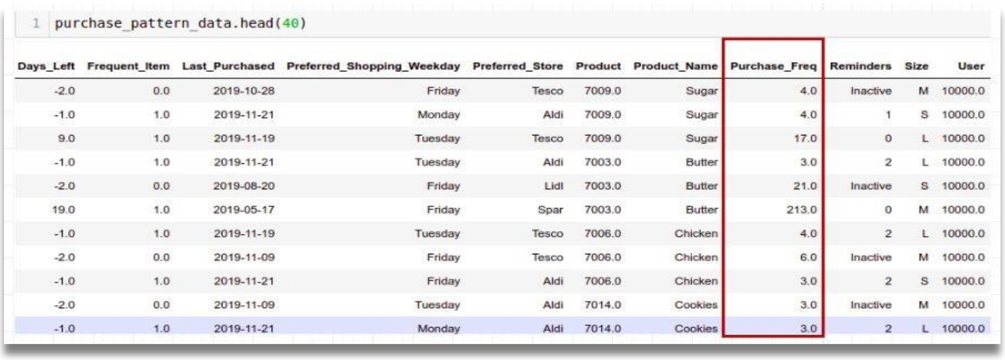
  The system will recommend these items that this user has never bought before based on the most popular items list.

3.3.1.2 Content-Based:

- **Frequency Predictions using most frequent purchase frequency (Algorithmic)**

  Here, we created an algorithm to predict the next purchase date of our product. In Our approach, we first mined the User's entire purchase history with respect to Items a User bought and their sizes. We consider a product of two different size as two different entities.

  We then calculated the difference between every purchase made and associated the most recurrent frequency to individual item. Based on this frequency we predict the next purchase date.



```
purchase_pattern_data.head(40)
```

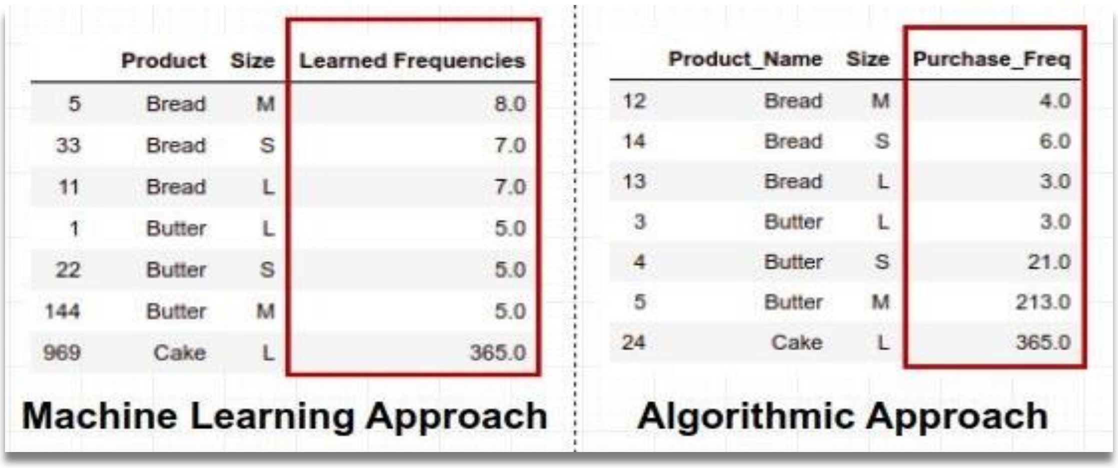
Days_Left	Frequent_Item	Last_Purchased	Preferred_Shopping_Weekday	Preferred_Store	Product	Product_Name	Purchase_Freq	Reminders	Size	User
-2.0	0.0	2019-10-28	Friday	Tesco	7009.0	Sugar	4.0	Inactive	M	10000.0
-1.0	1.0	2019-11-21	Monday	Aldi	7009.0	Sugar	4.0	1	S	10000.0
9.0	1.0	2019-11-19	Tuesday	Tesco	7009.0	Sugar	17.0	0	L	10000.0
-1.0	1.0	2019-11-21	Tuesday	Aldi	7003.0	Butter	3.0	2	L	10000.0
-2.0	0.0	2019-08-20	Friday	Lidl	7003.0	Butter	21.0	Inactive	S	10000.0
19.0	1.0	2019-05-17	Friday	Spar	7003.0	Butter	213.0	0	M	10000.0
-1.0	1.0	2019-11-19	Tuesday	Tesco	7006.0	Chicken	4.0	2	L	10000.0
-2.0	0.0	2019-11-09	Friday	Tesco	7006.0	Chicken	6.0	Inactive	M	10000.0
-1.0	1.0	2019-11-21	Friday	Aldi	7006.0	Chicken	3.0	2	S	10000.0
-2.0	0.0	2019-11-09	Tuesday	Aldi	7014.0	Cookies	3.0	Inactive	M	10000.0
-1.0	1.0	2019-11-21	Monday	Aldi	7014.0	Cookies	3.0	2	L	10000.0

Fig 3.7 - User Purchase Model: Algorithmic Approach

- **Frequency Predictions based on Machine Learning using Linear Regression (Machine Learning) [3]**

  In this approach, we first calculate the most recurrent frequency, but instead of calculating the next purchase date, we predict it using Machine Learning model. The entire data set is first split into train and test. Then performed One Hot Encoding on Product and Size, to convert the categorical values to numerical values. We then fit the model using training set and predict our next purchase dates.

  Because a new user doesn't have enough transactions of its own to train a model, we use algorithmic model. Once our user data reaches the ceiling, we switch to machine learning model and start with our predictions. We are also considering the scenario where the machine learning data isn't accurate, hence we compare it to our algorithmic model until and evaluate the error gap.



Machine Learning Approach

	Product	Size	Learned Frequencies
5	Bread	M	8.0
33	Bread	S	7.0
11	Bread	L	7.0
1	Butter	L	5.0
22	Butter	S	5.0
144	Butter	M	5.0
969	Cake	L	365.0

Algorithmic Approach

	Product_Name	Size	Purchase_Freq
12	Bread	M	4.0
14	Bread	S	6.0
13	Bread	L	3.0
3	Butter	L	3.0
4	Butter	S	21.0
5	Butter	M	213.0
24	Cake	L	365.0

Fig 3.8 – Comparisons between Two Approaches

We fixate on machine learning only once the error gap is in acceptable range and applied these approaches to give three different types of recommendations:

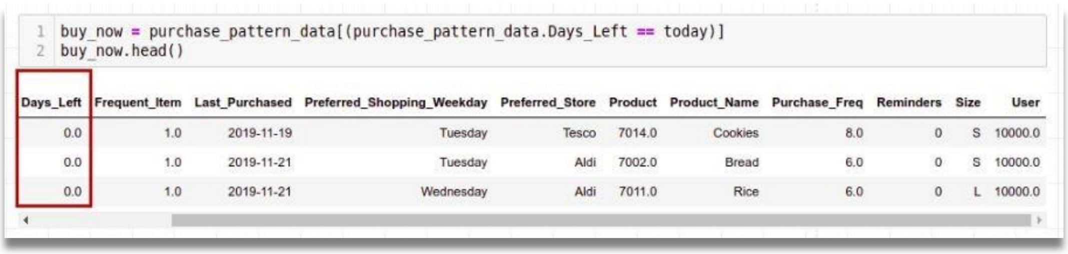
- **Buy Today:**

  We assigned the most recurrent frequency to each product from user purchase history and recommend a list of products that has -

  Frequency = Most Recurrent Frequency = (last purchase - current date)

  This means that the difference between last purchase date and current purchase date should be equal to the mode of frequency of that product of that size. That means the frequency days left is zero here and the user is expected to stock it up.

  The user model develops more as the User responds to recommended items.



```
buy_now = purchase_pattern_data[(purchase_pattern_data.Days_Left == today)]
buy_now.head()
```

Days_Left	Frequent_Item	Last_Purchased	Preferred_Shopping_Weekday	Preferred_Store	Product	Product_Name	Purchase_Freq	Reminders	Size	User
0.0	1.0	2019-11-19	Tuesday	Tesco	7014.0	Cookies	8.0	0	S	10000.0
0.0	1.0	2019-11-21	Tuesday	Aldi	7002.0	Bread	6.0	0	S	10000.0
0.0	1.0	2019-11-21	Wednesday	Aldi	7011.0	Rice	6.0	0	L	10000.0

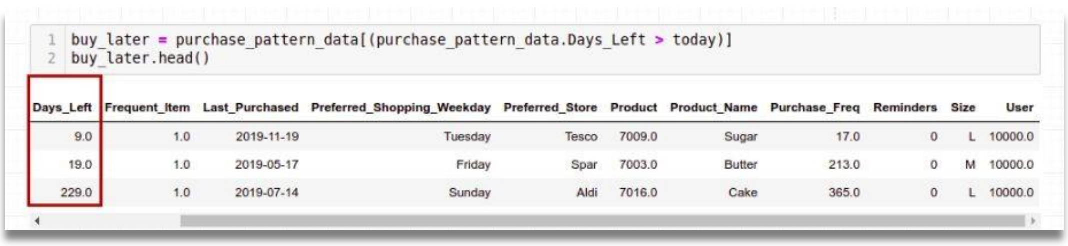
Fig 3.9 - Buy Today Recommendations

- **Buy Later:**

  This feature recommends products which a user is predicted to fall short of in near future based on their purchase history. This is done by evaluating by following -

  Remainder = Most Recurrent Frequency - (Last purchase - Current date)

This feature is provided so that the user can stock it up in advance. If the user continuous to follow the updated frequency, then the user model adapts to it and refreshes it recommended list.



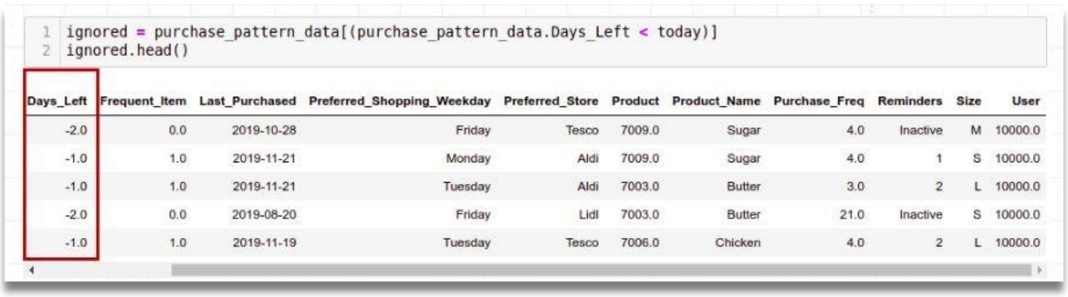
```
buy_later = purchase_pattern_data[(purchase_pattern_data.Days_Left > today)]
buy_later.head()
```

Days_Left	Frequent_Item	Last_Purchased	Preferred_Shopping_Weekday	Preferred_Store	Product	Product_Name	Purchase_Freq	Reminders	Size	User
9.0	1.0	2019-11-19	Tuesday	Tesco	7009.0	Sugar	17.0	0	L	10000.0
19.0	1.0	2019-05-17	Friday	Spar	7003.0	Butter	213.0	0	M	10000.0
229.0	1.0	2019-07-14	Sunday	Aldi	7016.0	Cake	365.0	0	L	10000.0

Fig 3.10 - Buy Later Recommendations

- **Missed Products:**

  Our model will recommend the users a product twice in the intervals of associated frequency and if the product isn't responded to, it is added to the Missed Products List. To provide User Control, we give an option to the user to view all ignored or missed products and decide to keep or completely remove that item from the list. These products will be deleted permanently and if purchased in future, they will be treated as a new purchase. Here we have used an implicit methodology where if a product isn't purchased when suggested (twice), it goes to the missed list and explicit methodology that is providing the user an option to tell the system that they aren't interested in the product.



```
ignored = purchase_pattern_data[(purchase_pattern_data.Days_Left < today)]
ignored.head()
```

Days_Left	Frequent_Item	Last_Purchased	Preferred_Shopping_Weekday	Preferred_Store	Product	Product_Name	Purchase_Freq	Reminders	Size	User
-2.0	0.0	2019-10-28	Friday	Tesco	7009.0	Sugar	4.0	Inactive	M	10000.0
-1.0	1.0	2019-11-21	Monday	Aldi	7009.0	Sugar	4.0	1	S	10000.0
-1.0	1.0	2019-11-21	Tuesday	Aldi	7003.0	Butter	3.0	2	L	10000.0
-2.0	0.0	2019-08-20	Friday	Lidl	7003.0	Butter	21.0	Inactive	S	10000.0
-1.0	1.0	2019-11-19	Tuesday	Tesco	7006.0	Chicken	4.0	2	L	10000.0

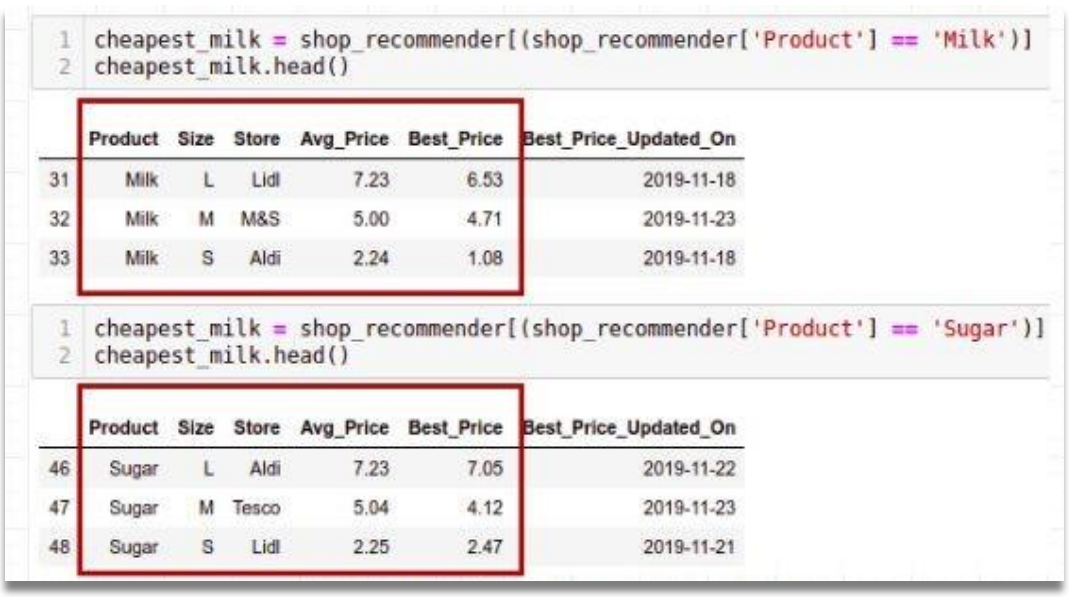
Fig 3.11 - Missed Recommendations

Note:

- '-1' indicates it is a missed item.
- ' -2' indicates the non-frequent items (False Positives), Hence Reminders are inactive.

- **Grocery Shop Recommender:**

  Our Application has an additional feature that recommends the latest minimum prices of all products in nearby stores.



```
cheapest_milk = shop_recommender[(shop_recommender['Product'] == 'Milk')]
cheapest_milk.head()
```

	Product	Size	Store	Avg_Price	Best_Price	Best_Price_Updated_On
31	Milk	L	Lidl	7.23	6.53	2019-11-18
32	Milk	M	M&S	5.00	4.71	2019-11-23
33	Milk	S	Aldi	2.24	1.08	2019-11-18

```
cheapest_milk = shop_recommender[(shop_recommender['Product'] == 'Sugar')]
cheapest_milk.head()
```

	Product	Size	Store	Avg_Price	Best_Price	Best_Price_Updated_On
46	Sugar	L	Aldi	7.23	7.05	2019-11-22
47	Sugar	M	Tesco	5.04	4.12	2019-11-23
48	Sugar	S	Lidl	2.25	2.47	2019-11-21

Fig 3.12 - Grocery Shop Recommender

Note:

- APP is also capable of identifying non-frequent products for example if the product was purchased only once or the product was never purchased within a general aging period of the product's category.
- The algorithmic approach is applied to user modeling until the time the user has sufficient purchase history to train a machine learning model.
- APP also employs on machine learning to predict purchase frequencies of the product of a given size. In the ML-based user modeling, certain purchase parameters are considered (e.g. Product ID, size, price, etc.) to predict the product's purchase frequency.
- Our system evaluates the accuracy of both approaches and updates the user model with maximum accuracy.

4 Future Works:

Our application has achieved most functions we have planned, but there is still something could be improved. Although our application has finished gathering data and understanding the features required, the dataset is unrepresentative and relatively small for machine learning. We have noted the error gaps between our Algorithmic and Machine learning models provided a larger dataset to train upon, a smaller gap is achievable however, we couldn't generate that much data to train upon, but we aim to work on it in future.

We used Firebase ML Kit, and we could scan store name and total amount from receipts although we faced many challenges in parsing each record. Also to note, our model worked in few receipts, for few it wasn't accurate. However, it is inconvenient for users to upload the image of their receipts through our application now, so we will consider making a phone application in the future for easier use. We started building the application and hope to work on it more in the future.

5 Demo

We will give an in-depth walkthrough of our application; the demo can be seen here:

https://drive.google.com/file/d/10831LwLYQaKu__veePYLC_cGlartAUS2/view

Our demo that was submitted alongside this report has 5 use cases:

- **Case 1 (0:05):**

  The user purchases an item from their Buy Now list that we recommended. The user then confirms the items appears in their Buy Later list, and then confirms that the item does not reappear in their Buy Now page.

- **Case 2 (0:45):**

  The user manually enters in an item and purchases it. Again, the user confirms that the item has been added to the buy later screen and therefore is a part of the user model.

- **Case 3 (1:36):**

  The user goes to the Missed Items page and decides they no longer want to be recommended rice and they remove it and then confirms that it does not reappear in the Missed Items page.

- **Case 4 (2:10):**

  The user views their purchase behavior which shows both our algorithmic and machine learning models and compares some suggested frequencies.

- **Case 5 (3.13):**

  We show two different users and compare their "Top Pick" pages that display recommendations based on our Collaborative Filtering.

6 Conclusion:

With a well-planned blueprint, we accomplished our task without difficulty. In our demo, you would see a usable application which provides the access for users to add an item to the shopping list manually. Later, we improved the function with another input method – scanning receipts. As can be seen in the demo, users would receive recommendations based on both collaborative and content-based filtering (i.e. algorithmic approach and machine learning approach). At the same time, users could edit their own shopping list (i.e. adding, deleting, postponing and calling back). However, there is still something we could work on in the future (e.g. adjusting our user model for dealing with real world data for ML-based recommendation). We could also consider the improvement in the usability and user-friendliness of our application.

Our solution is hoped to work better in the real world as the data on which the model is tested was generated by us and may or may not reflect the scenarios of the real world. We also need a large amount of data set for our machine learning approach to work however, because we have also incorporated algorithmic approach, we are maintaining our accuracy levels. Machine learning approach is helpful if there are more added features values in the future hence making our solution more scalable.

This project taught us how to structure a development over a very short space of time within a large team. We were given full control of all stages of development, which often is not the case with college project. This means we learned to build an app from the conception of an idea, to the planning of technologies and infrastructure, to the API implementation, to the building of the front-end application. We learned how and when to priorities different aspects of development and when to concede that something would require future work if we wanted to finish them. It gave us an opportunity to explore different methods for adaptation to find the algorithms that fit our needs, without feeling pressured to commit large amounts of time to aesthetics.

References:

[1] Statista; How many times a week do you shop at a supermarket? https://www.statista.com/statistics/495444/weekly-frequency-of-supermarket-shopping-united-kingdom-uk/, lasted accessed 27/11/2019

[2] Brusilovsky, P., Millan, E.: User Models for Adaptive Hypermedia and Adaptive Educational Systems, LNCS 4321 (2007).

[3] Van Meteren, R., van Someren, M.: Using Content-Based Filtering. In Proceedings of the Machine Learning in the New Information Age, pp. 47-56, MLnet(2000).

[4] Bawa, Kapil & Ghosh, Avijit. (1999). A Model of Household Grocery Shopping Behavior. Marketing Letters. 10. 149-160. 10.1023/A:1008093014534. https://www.researchgate.net/publication/227082840_A_Model_of_Household_Grocery_Shopping_Behavior

[5] TCD ML Comp. 2019/20 - Income Pred. (Group). https://www.kaggle.com/c/tcd-ml-comp-201920-income-pred-group/