ML UL 2 Individual Assignment

Name - Shreyas Namjoshi
PG ID- 12110103
Email- shreyas_namjoshi_ampba2022s@isb.edu

1. What are the problems Bigbasket is trying to address? Please be as specific as possible.

1. Customer have to go through long list of products on a mobile app,though their requirement is only handful of those. This degrades the customer experience as the ordering time is increased.So there is great opportunity for big basket to create recommendations for the users that will help them buy the products they regularly buy and not have to scroll through huge list

2. Customer during placing the order forgets to add some or the other product to the cart. This results in either customer placing another order or buying from local store. Here there is another opportunity for big basket to give recommendations about the items which users failed to add to cart and which they have previously brought.

2. What is/are some of the fundamental differences in the recommender systems requirements between Bigbasket and other eCommerce companies such as Amazon or Flipkart?

1. The customer buying from bigbasket have different needs than customer buying from amazon/flipkart.
2. Frequency of buying on amazon/flipkart is different than that from big basket. User tends to buy from big basket more often such as daily/weekly/monthly because they normally buy daily needs.
3. The number of products customer buy from Bigbasket in a single order is very high whereas the number of products customer buy from flipkart/amazon is low.
4. Frequency of buying same products is very high on bigbasket as compared to flipkart/amazon.

Considering the above situations, the recommendation systems need to be designed differently for Big Basket.As the number of items customer buy in a single transaction is more, recommendation system should help customer with items that they are likely to buy based on past customer purchase and increase customer experience by helping customer to not go through huge list of products.This can be something such as "SMART BAKSET"

Also, since customer buy large number of items, they tend to forget few items. Thus recommendation system should also help customer by giving them recommendations if they are forgetting something from their previous behavious.This can be sometime such as "DID YOU FORGET".

3. Discuss what sort of recommendation technique(s) is/are more appropriate for Bigbasket, in general,and why? Elaborate by connecting different methods and techniques with specific use cases.

At Big Basket, since customers are frequent buyers and the frequency of buying a particular item is also high given that most customers buy daily needs from the platform. Also In a single order,number of items can be very high so it is important that customers are given recommendation about what they could buy given their earlier orders.Also it is quite possible that while customer place an order,they can forget ordering some or the other item. So to make sure that above problems are addressed below recommendation techniques can be used individually or in collaboration.

1. Association Rules- What Goes with What- Some Items goes together.For ex bread - butter/Rice- Dal etc. So if a Customer has added an item in a cart, platform can recommend another item which is frequently brought together.

2. Sequential Pattern Mining- What Goes after What- For Ex- If a customer has brought fruits in one of its orders and milkmaid in next subsequent order and if the frequency of such sequence is high, then if a customer buys fruits next time, the platform can recommend milkmaid when the customer adds fruits.

Also since its time based, its possible that a customer is buying few of the items during few months in an year. For ex user buys Maida,Curd,Baking Powder,Baking Soda,Condensed Milk,Icing Sugar etc(Ingrediants of cake making) in months such as Jan,Oct(may be some birthdays in the months) and this pattern is repeated. Then Platform can recommend these items based on these month.

3. Collaborative Filtering- User Item Collaborative Filtering- Consider that one customer is similar to another customer based on the items they both buy, then it is likely that if one customer is buying some item z,another customer will also buy that. For ex customer A buys Milk, Bread,Apple and Banana ,customer B buys Milk,Bread and Banana so we can say that they buying similar items. So customer B is likely to buy Apple also. So platform can give such recommendations using this techniques.

4. What sort of data challenges do you anticipate while building a recommendation engine for Bigbasket? Explore the data set and try applying different techniques on the dataset to identify the challenges that one would face while building the engine for Bigbasket.

For Building Recommendation for Big Basket as suggested in #3, we would be using one of the below techniques:
1. Association Rules
2. Sequential Pattern Mining
3. Collaborative Filtering- User -Item
Below are the Challenges that may appear while using above techniques for building recommendation engine.

Biased- Confidence:
While using associative rules, there is possiblity that antecedent and consequent enjoys high support even though there is not specific relation as to why these items are together. Thus while using Associative Rules for recommendation, we may get irrelavant items as recommendation just because these items enjoy high support. Thus the recommendation appear to be biased.

SO Many Rules to Process:
While using association rules, it is imperative that we need to have proper values for support and confidence else we will get so many rules which may be irrelevant.This will result in recommending irrelevant items and take unnecessary computation and space.

Computationaly Intensive:
When using User -Item Collaborative filtering, the matrix can be quite large that makes this type of recommendation computationally intensive, Also since most recommendation would be real time for ex recommendation of a forgotten item based on some item in cart, recommendation calculation on real time with such sparse matrix can be time consuming.This may result in people still forgetting on buying some items, leading to another order from the customer,which will increase supply chain cost or completely losing out on revenue if the customer buys that item from local shop.

New User- Cold Start Problem:
Most of the techniques suggested here recommend product based on previous history.If the user is new then the user doesnt have any previous history. In this case it will be difficult to give any personalized recommendations to that user.

5. What are the implementation and deployment challenges of a recommendation engine for Bigbasket?

Implementation Challenges:
1. Real Time Recommendation Systems:
Recommendation on platform such as big basket has to be real time. This means that if a user has put something in his/her cart, the system should recommend basis the items in cart on What goes with what principle of association rules.
Also when using collaborative filtering(User-Item), the matrix of user and item becomes very sparse. So trying to recommend using such sparse matrix can be time consuming for the platform considering its large user base and huge product catalog.

2. Computationally Intensive:
As from #1, we know that when using collaborative filtering, considering large product catalog and huge customer base, the user-item matrix becomes very sparse,thus finding similarity between users is a computationally intestive process. It consumes huge memory and time.

3. Concurrent Users:
Because of large user base, its important that system supports concurrent users and give general and personalized real time recommendations.Creating a system with such requirements is always challenging

Deployment Challenges:
Platform Selection and Algorithm Selection:
Considering the above implementation challenges and business requirements such as quick response time and high scalability, it is important that the deployment platform needs to have features such as auto scalabiity. Also certain design patterns need to be applied while using recommendation techniques that will ensure quick response time. The Platform should be such that it can cater to concurrent users personalized recommendations.

6. Write an R script to generate five consumer-agnostic “good-quality” association rules sorted by their lift ratios? Please include the output (copy-paste/screenshot) — those five rules along with metrics that quantify the “goodness” of those rules — in your submission. What is the support of the first rule? Explain how it has been calculated for this rule. What is the confidence of the first rule? Explain how it has been calculated for this rule. What is the lift ratio of the first rule? Explain how it has been calculated for the rule. Based on the five rules you identified, suggest a couple of action plans that can benefit Bigbasket.

In [110…
```
import pandas as pd
import numpy as np
```

In [111…
```
data=pd.read_excel("IMB575_Individual_Assignmnet (2).xls",sheet_name="POS DATA")
data.head()
```

Out[111…

	Member	Order	SKU	Created On	Description
0	M09736	6468572	34993740	22-09-2014 22:45	Other Sauces
1	M09736	6468572	15669800	22-09-2014 22:45	Cashews
2	M09736	6468572	34989501	22-09-2014 22:45	Other Dals
3	M09736	6468572	7572303	22-09-2014 22:45	Namkeen
4	M09736	6468572	15669856	22-09-2014 22:45	Sugar

In [112…
```
#removing rows that has more than more products which are comma seperated
rows=np.where(data['Description'].str.contains('CFLs,cfls,7832553'))
rows
data=data.drop(rows[0][0])
data=data.drop(rows[0][1])
data=data.drop(rows[0][2])
data=data.drop(rows[0][3])
data=data.drop(rows[0][4])
len(data)

len(data)
```

Out[112… 62136

In [113…
```
data1= data.groupby("Order")["Description"].transform(lambda x: ','.join(x))
data1=data1.unique()
data2=pd.DataFrame(data1)
data2.columns=["Products"]
data2.head()
```

Out[113…

	Products
0	Other Sauces,Cashews,Other Dals,Namkeen,Sugar,...
1	Utensil Scrub Pads,Other Rice Products,Toor Da...
2	Urad Dal,Boiled Rice,Jaggery,Other Dals,Other ...
3	Sugar,Jaggery,Root Vegetables,Cakes,Urad Dal,N...
4	Banana,Cashews,Other Rice Products,Cashews,Cas...

In [114…
```
fd = list(data2["Products"].apply(lambda x:x.split(",") ))
```

In [115…
```
from mlxtend.frequent_patterns import apriori, association_rules
```

In [116…
```
#Let's transform the list, with one-hot encoding
from mlxtend.preprocessing import TransactionEncoder
a = TransactionEncoder()
a_data = a.fit(fd).transform(fd)
df = pd.DataFrame(a_data,columns=a.columns_)
df = df.replace(False,0)
df = df.replace(True,1)
df.head()
```

Out[116…

	Cookies & Rusk	Foam & Gels	Glue & Tape	Marie & Milk Biscuits	Peelers & Scrapers	After Shave	Agarbatti	Almonds	Aluminium Foil & Cling Wrap	Antiseptics	...	Vanaspati	Veg & Fruit	Vermicelli	Vinegar	Wafers	Washing Bars	Whole Grains	Whole Spices	Womens Deo	Yogurt & Lassi
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 221 columns

In [117…
```
df1 = apriori(df, min_support = 0.1, use_colnames = True, verbose = 1)
df1.head()
```

```
Processing 20 combinations | Sampling itemset size 4
```

Out[117…

	support	itemsets
0	0.273034	(Banana)
1	0.425943	(Beans)
2	0.126344	(Bread)
3	0.289879	(Brinjals)
4	0.128936	(Exotic Vegetables)

In [118…
```
df_ar = association_rules(df1, metric = "confidence", min_threshold = 0.6,)
df_ar =df_ar.sort_values(by='lift',ascending=False)
df_ar.head()
```

Out[118…

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
10	(Other Vegetables, Gourd & Cucumber)	(Beans)	0.210056	0.425943	0.142931	0.680444	1.597502	0.053459	1.796422
11	(Beans, Gourd & Cucumber)	(Other Vegetables)	0.198782	0.454581	0.142931	0.719035	1.581754	0.052569	1.941237
12	(Root Vegetables, Gourd & Cucumber)	(Beans)	0.182325	0.425943	0.122716	0.673063	1.580173	0.045056	1.755866
20	(Root Vegetables, Gourd & Cucumber)	(Other Vegetables)	0.182325	0.454581	0.130491	0.715707	1.574433	0.047610	1.918512
5	(Brinjals, Gourd & Cucumber)	(Beans)	0.152650	0.425943	0.101983	0.668081	1.568477	0.036962	1.729513

1. Support of first rule is 0.142931. It is calculated as # transactions with (Other Vegetables, Gourd & Cucumber) and (Beans)/Total # of transactions. So we can say that (Other Vegetables, Gourd & Cucumber) and (Beans) appear together in 14% of the transactions

2. confidence of first rule is 0.680444 . It is calculated as # transactions with (Other Vegetables, Gourd & Cucumber) and (Beans) together/Total # of transaction of (Other Vegetables, Gourd & Cucumber). We can say that given that a customer buys (Other Vegetables, Gourd & Cucumber),the likelihood that he/she buys (Beans) is 0.68%

3. Lift Ratio for the first rule is 1.5975. Its the ratio of Confidence to benchmark confidence. Benchmark Confidence is probability of Consequent(Beans) and Confidence is calculated as above.

4. From the above rules, we can suggest that if a customer buys (Other Vegetables, Gourd & Cucumber) and if he forgets to buy beans ,it can be shown as a recommended item. Also in general for Smart Basket, since the above rules have high lift we can say that they are most frequently brought items based on given confidence and support and thus these can be given as blanket recommendations in Smart Basket.

7. Bigbasket is interested in introducing a “Did you forget?” feature to identify items a customer may have forgotten. Discuss how this feature can be created following data-driven approaches?

BigBasket can create "Did You Forget?" feature using below techniques:
1. Association Rules: Association Rules are based on what item goes with what. If in the entire data set, there are certain patterns such as item a goes with item b,then this rules are used. Considering that Bigbasket is platform on which user buys daily need items, it is imperative that such patterns would exist. So if such patterns are found ,they can be used to recommend products that customer is likely to buy with some or the other item that is in his/her cart.
2. Sequenctial Pattern Mining: Sequential Pattern Mining basically means what goes after what. This helps in understanding that if customer has bought one item in one order and another item in subsequent order and the frequency of such repetitions is high, then it is likely that when customer buys item 1, he/she is likely to buy item 2 also. Such patterns will also be available in the data as the frequency that customer buys items on the platform can be daily /weekly /monthly.In short such patterns are likely to be present as the frequency is orders of a customer is high.

Both the above techquies are user agnostic but can be made personalized also. Once the user logins, since platform knows the user id and the previous order history, recommendation system just need to use above tecnhique on only previous order data of that particular customer.Thus it will help in better personalized recommendations.

8. Write an R script to do the following: Pick a customer of your choice. Now, present an example of product sets A and B such that when a shopper has added product set A in his/her cart, the recommendation engine should suggest product sets B as “Did you forget items?” Support your answers using output generated by your scripts.

In [119…
```
#get the previous transaction for M09736
data_mem=data.loc[data['Member']=='M09736']
data_mem.head()
```

Out[119…

	Member	Order	SKU	Created On	Description
0	M09736	6468572	34993740	22-09-2014 22:45	Other Sauces
1	M09736	6468572	15669800	22-09-2014 22:45	Cashews
2	M09736	6468572	34989501	22-09-2014 22:45	Other Dals
3	M09736	6468572	7572303	22-09-2014 22:45	Namkeen
4	M09736	6468572	15669856	22-09-2014 22:45	Sugar

In [120…
```
mem_order= data_mem.groupby("Order")["Description"].transform(lambda x: ','.join(x))
mem_order=mem_order.unique()
mem_order=pd.DataFrame(mem_order)
mem_order.columns=["Products"]
mem_order.head()
```

Out[120…

	Products
0	Other Sauces,Cashews,Other Dals,Namkeen,Sugar,...
1	Utensil Scrub Pads,Other Rice Products,Toor Da...
2	Urad Dal,Boiled Rice,Jaggery,Other Dals,Other ...
3	Sugar,Jaggery,Root Vegetables,Cakes,Urad Dal,N...
4	Banana,Cashews,Other Rice Products,Cashews,Cas...

In [121…
```
fd1 = list(mem_order["Products"].apply(lambda x:x.split(",") ))
```

In [122…
```
b = TransactionEncoder()
b_data = b.fit(fd1).transform(fd1)
df_mem = pd.DataFrame(b_data,columns=b.columns_)
df_mem= df_mem.replace(False,0)
df_mem= df_mem.replace(True,1)
df_mem.head()
```

Out[122…

	Foam & Gels	Almonds	Avalakki / Poha	Banana	Basmati Rice	Beans	Besan	Boiled Rice	Bread	Brinjals	...	Sugar	Sugar Cubes	Sunflower Oils	Toor Dal	Urad Dal	Utensil Scrub Pads	Wafers	Washing Bars	Whole Spices	Yogurt & Lassi
0	0	0	0	1	0	0	0	0	0	0	...	1	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	...	0	0	0	1	0	1	0	0	0	0
2	0	1	0	0	0	0	0	1	0	0	...	0	0	0	0	1	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	...	1	0	0	1	1	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0

5 rows × 59 columns

In [123…
```
df_mem_1 = apriori(df_mem, min_support = 0.2, use_colnames = True, verbose = 1)
```

```
Processing 35 combinations | Sampling itemset size 5
```

In [124…
```
df_ar_1 = association_rules(df_mem_1, metric = "confidence", min_threshold = 0.8,)
df_ar_1 =df_ar_1.sort_values(by='lift',ascending=False)
df_ar_1.head()
```

Out[124…

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
45	(Other Rice Products, Other Dals)	(Boiled Rice)	0.262295	0.377049	0.229508	0.875000	2.320652	0.130610	4.983607
63	(Root Vegetables, Other Vegetables)	(Banana, Beans)	0.278689	0.426230	0.245902	0.882353	2.070136	0.127116	4.877049
56	(Banana, Other Dals, Beans)	(Root Vegetables)	0.262295	0.475410	0.245902	0.937500	1.971983	0.121204	8.393443
46	(Other Rice Products, Boiled Rice)	(Other Dals)	0.245902	0.475410	0.229508	0.933333	1.963218	0.112604	7.868852
59	(Root Vegetables, Other Dals)	(Banana, Beans)	0.295082	0.426230	0.245902	0.833333	1.955128	0.120129	3.442623

In [125…
```
prod=df_ar_1[['antecedents','consequents']]
prod.columns=['Set A','Set B']
prod.head()
```

Out[125…

	Set A	Set B
45	(Other Rice Products, Other Dals)	(Boiled Rice)
63	(Root Vegetables, Other Vegetables)	(Banana, Beans)
56	(Banana, Other Dals, Beans)	(Root Vegetables)
46	(Other Rice Products, Boiled Rice)	(Other Dals)
59	(Root Vegetables, Other Dals)	(Banana, Beans)

In [126…
```
def getProdList():
    i=int(input("Enter number of Products :"))
    if i<4:
        list=[]
        while i>0:
            text=input("Enter Product :")
            list.append(text)
            i-=1
        return list
    else:
        print("Please enter product number between 1 and 3")


def createSet(list):
    userSet=set()
    try:
        for i in list:
            userSet.add(i)
    except:
        pass

    return userSet

def getRecommendedProducts(prodSet,df):
    test=df.loc[df['Set A']==prodSet]
    test=test[['Set B']]
    listb=test.values.tolist()
    return listb
```

In [127…
```
prodList=getProdList()
getSet=createSet(prodList)
test1=getRecommendedProducts(getSet,prod)
print(test1)
```

```
Enter number of Products :1
Enter Product :Other Vegetables
[[frozenset({'Beans'})], [frozenset({'Banana'})]]
```

So if user adds Other Vegetables to the list, Big Basket should recommend the user, Beans and Banana

9. How do we find similarities between users based on what customers buy? Can collaborative filtering be used to create values given this dataset? Present a use case where finding similarities between user baskets can be helpful to Bigbasket.

We can find similarity between user based on what customers buy basis collaborative filtering technique,in which similarity between user is found basis the items purchased. The same is demonstrated below:

In [128…
```
len(data)
```

Out[128… 62136

In [129…
```
data.head()
```

Out[129…

	Member	Order	SKU	Created On	Description
0	M09736	6468572	34993740	22-09-2014 22:45	Other Sauces
1	M09736	6468572	15669800	22-09-2014 22:45	Cashews
2	M09736	6468572	34989501	22-09-2014 22:45	Other Dals
3	M09736	6468572	7572303	22-09-2014 22:45	Namkeen
4	M09736	6468572	15669856	22-09-2014 22:45	Sugar

In [130…
```
user_item_matrix = data.pivot_table(index='Member',columns='Description',values='Order',aggfunc='count')
user_item_matrix.head()
```

Out[130…

Description	After Shave	Agarbatti	Almonds	Aluminium Foil & Cling Wrap	Antiseptics	Avalakki / Poha	Ayurvedic	Ayurvedic Food	Baby Care Accessories	Baby Cereal	...	Vanaspati	Veg & Fruit	Vermicelli	Vinegar	Wafers	Washing Bars	Whole Grains	Whole Spices	Womens Deo	Yogurt & Lassi
Member																					
M04158	NaN	NaN	8.0	9.0	NaN	2.0	NaN	NaN	NaN	NaN	...	NaN	1.0	NaN	1.0	NaN	NaN	NaN	9.0	NaN	8.0
M08075	NaN	NaN	2.0	NaN	NaN	8.0	NaN	NaN	NaN	NaN	...	NaN	NaN	2.0	NaN	1.0	NaN	1.0	34.0	NaN	8.0
M09303	NaN	NaN	6.0	12.0	NaN	8.0	NaN	1.0	NaN	NaN	...	NaN	NaN	1.0	1.0	NaN	NaN	NaN	28.0	NaN	NaN
M09736	NaN	NaN	8.0	NaN	NaN	2.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	2.0	8.0	NaN	33.0	NaN	1.0
M12050	NaN	NaN	5.0	NaN	NaN	5.0	NaN	NaN	NaN	NaN	...	NaN	NaN	1.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN

5 rows × 215 columns

In [131…
```
user_item_matrix = user_item_matrix.applymap(lambda x: 1 if x>0 else 0)
```

In [132…
```
user_item_matrix.head()
```

Out[132…

Description	After Shave	Agarbatti	Almonds	Aluminium Foil & Cling Wrap	Antiseptics	Avalakki / Poha	Ayurvedic	Ayurvedic Food	Baby Care Accessories	Baby Cereal	...	Vanaspati	Veg & Fruit	Vermicelli	Vinegar	Wafers	Washing Bars	Whole Grains	Whole Spices	Womens Deo	Yogurt & Lassi
Member																					
M04158	0	0	1	1	0	1	0	0	0	0	...	0	1	0	1	0	0	0	1	0	1
M08075	0	0	1	0	0	1	0	0	0	0	...	0	0	1	0	1	0	1	1	0	1
M09303	0	0	1	1	0	1	0	1	0	0	...	0	0	1	1	0	0	0	1	0	0
M09736	0	0	1	0	0	1	0	0	0	0	...	0	0	0	0	1	1	0	1	0	1
M12050	0	0	1	0	0	1	0	0	0	0	...	0	0	1	0	1	0	0	0	0	0

5 rows × 215 columns

In [133…
```
from sklearn.metrics.pairwise import cosine_similarity
user_user_sim_matrix = pd.DataFrame(cosine_similarity(user_item_matrix))
user_user_sim_matrix.head()
```

Out[133…

	0	1	2	3	4	5	6	7	8	9	...	96	97	98	99	100	101	102	103	104	105
0	1.000000	0.646843	0.656967	0.589160	0.523256	0.563133	0.599667	0.590039	0.590017	0.632982	...	0.581926	0.641689	0.544581	0.560112	0.539036	0.600245	0.563602	0.554955	0.499434	0.582552
1	0.646843	1.000000	0.736024	0.617010	0.633614	0.671098	0.645458	0.636655	0.650428	0.715237	...	0.609434	0.624020	0.604888	0.513265	0.580645	0.682500	0.623035	0.528352	0.473229	0.671098
2	0.656967	0.736024	1.000000	0.678265	0.645553	0.544705	0.623009	0.687310	0.661376	0.692232	...	0.589832	0.730159	0.634358	0.581914	0.672022	0.712697	0.666865	0.611499	0.568291	0.625402
3	0.589160	0.617010	0.678265	1.000000	0.655098	0.525646	0.631271	0.677603	0.554798	0.667344	...	0.599362	0.678265	0.536056	0.505399	0.633685	0.705642	0.661112	0.546268	0.566529	0.567698
4	0.523256	0.633614	0.645553	0.655098	1.000000	0.539792	0.555651	0.675956	0.586993	0.611216	...	0.536582	0.645553	0.550482	0.441149	0.599364	0.629285	0.626680	0.579667	0.475997	0.582975

5 rows × 106 columns

In [134…
```
user_user_sim_matrix.columns = user_item_matrix.index
```

In [135…
```
user_user_sim_matrix['Member'] = user_item_matrix.index
```

In [136…
```
user_user_sim_matrix = user_user_sim_matrix.set_index('Member')
```

In [137…
```
user_user_sim_matrix.head()
```

Out[137…

Member	M04158	M08075	M09303	M09736	M12050	M12127	M14746	M16218	M16611	M18732	...	M78720	M82651	M84827	M86304	M86572	M90375	M91098	M96365	M99030	M99206
Member																					
M04158	1.000000	0.646843	0.656967	0.589160	0.523256	0.563133	0.599667	0.590039	0.590017	0.632982	...	0.581926	0.641689	0.544581	0.560112	0.539036	0.600245	0.563602	0.554955	0.499434	0.582552
M08075	0.646843	1.000000	0.736024	0.617010	0.633614	0.671098	0.645458	0.636655	0.650428	0.715237	...	0.609434	0.624020	0.604888	0.513265	0.580645	0.682500	0.623035	0.528352	0.473229	0.671098
M09303	0.656967	0.736024	1.000000	0.678265	0.645553	0.544705	0.623009	0.687310	0.661376	0.692232	...	0.589832	0.730159	0.634358	0.581914	0.672022	0.712697	0.666865	0.611499	0.568291	0.625402
M09736	0.589160	0.617010	0.678265	1.000000	0.655098	0.525646	0.631271	0.677603	0.554798	0.667344	...	0.599362	0.678265	0.536056	0.505399	0.633685	0.705642	0.661112	0.546268	0.566529	0.567698
M12050	0.523256	0.633614	0.645553	0.655098	1.000000	0.539792	0.555651	0.675956	0.586993	0.611216	...	0.536582	0.645553	0.550482	0.441149	0.599364	0.629285	0.626680	0.579667	0.475997	0.582975

5 rows × 106 columns

In [138…
```
user_user_sim_matrix.loc['M04158'].sort_values(ascending = False)
```

Out[138…
```
Member
M04158    1.000000
M76390    0.702802
M51043    0.695725
M37253    0.682131
M33064    0.671703
            ...
M59012    0.513994
M27871    0.504184
M99030    0.499434
M36876    0.491806
M64379    0.482759
Name: M04158, Length: 106, dtype: float64
```

In [139…
```
items_bought_by_M04158 = set(user_item_matrix.loc['M04158'].iloc[user_item_matrix.loc['M04158'].to_numpy().nonzero()].index)
items_bought_by_M04158
```

Out[139…
```
{'Almonds',
 'Aluminium Foil & Cling Wrap',
 'Avalakki / Poha',
 'Baking Ingredients',
 'Banana',
 'Basmati Rice',
 'Beans',
 'Besan',
 'Boiled Rice',
 'Bread',
 'Brinjals',
 'Buns & Pavs',
 'Butter & Cream',
 'Cashews',
 'Chilli & Soya Sauces',
 'Chinese Noodles',
 'Cookies',
 'Cooking Sauce',
 'Cotton & Bandages',
 'Cream Biscuits',
 'Curd',
 'Exotic Vegetables',
 'Floor Cleaners',
 'Frozen Chicken',
 'Frozen Veg Food',
 'Ghee',
 'Glass Cleaners',
 'Glucose, Marie & Milk Biscuits',
 'Gourd & Cucumber',
 'Hand Sanitizer',
 'Hand Wash',
 'Health Drinks',
 'Healthy Snacks',
 'Honey',
 'Jaggery',
 'Liquid Detergent',
 'Maida',
 'Moong Dal',
 'Mosquito Repellent',
 'Namkeen',
 'Olive Oils',
 'Organic F&V',
 'Organic Flours',
 'Organic Rice & Rice Products',
 'Other Dals',
 'Other Dry Fruits',
 'Other Sweets',
 'Other Vegetables',
 'Pasta & Noodles',
 'Pre & Post Wash Care',
 'Raisins',
 'Raw Peanuts',
 'Ready Mix',
 'Regular Pasta',
 'Rodent Control',
 'Root Vegetables',
 'Shaving Blade & Razors',
 'Sooji & Rava',
 'Sugar',
 'Sugar Cubes',
 'Sunflower Oils',
 'Tomato Ketchup',
 'Toor Dal',
 'Urad Dal',
 'Veg & Fruit',
 'Vinegar',
 'Whole Spices',
 'Yogurt & Lassi'}
```

In [140…
```
items_bought_by_M76390 = set(user_item_matrix.loc['M76390'].iloc[user_item_matrix.loc['M76390'].to_numpy().nonzero()].index)
items_bought_by_M76390
```

Out[140…
```
{'Almonds',
 'Aluminium Foil & Cling Wrap',
 'Avalakki / Poha',
 'Banana',
 'Basmati Rice',
 'Beans',
 'Besan',
 'Boiled Rice',
 'Bottle & Tin Openers',
 'Bread',
 'Brinjals',
 'Cashews',
 'Chilli & Soya Sauces',
 'Chinese Noodles',
 'Chips',
 'Cookies',
 'Cream Biscuits',
 'Exotic Vegetables',
 'Gourd & Cucumber',
 'Hair Colors & Dyes',
 'Hand Wash',
 'Ice Creams & Desserts',
 'Instant Noodles',
 'Jaggery',
 'Ladles & Spatulas',
 'Liquid Detergent',
 'Liquid Soaps & Bars',
 'Maida',
 'Moong Dal',
 'Muffins, Cookies & Rusk',
 'Namkeen',
 'Olive Oils',
 'Organic F&V',
 'Organic Flours',
 'Organic Rice & Rice Products',
 'Organic Salt,Sugar & Jaggery',
 'Other Dals',
 'Other Dry Fruits',
 'Other Rice Products',
 'Other Sweets',
 'Other Vegetables',
 'Packaged Drinking Water',
 'Powdered Spices',
 'Raisins',
 'Raw Peanuts',
 'Raw Rice',
 'Ready Mix',
 'Regular Pasta',
 'Root Vegetables',
 'Shampoo',
 'Shaving Blade & Razors',
 'Snacky Nuts',
 'Sooji & Rava',
 'Sugar',
 'Sugar Cubes',
 'Sunflower Oils',
 'Toor Dal',
 'Urad Dal',
 'Vermicelli',
 'Vinegar',
 'Whole Grains',
 'Whole Spices',
 'Yogurt & Lassi'}
```

In [141…
```
items_recommend_M76390 = items_bought_by_M04158 - items_bought_by_M76390
items_recommend_M76390
```

Out[141…
```
{'Baking Ingredients',
 'Buns & Pavs',
 'Butter & Cream',
 'Cooking Sauce',
 'Cotton & Bandages',
 'Curd',
 'Floor Cleaners',
 'Frozen Chicken',
 'Frozen Veg Food',
 'Ghee',
 'Glass Cleaners',
 'Glucose, Marie & Milk Biscuits',
 'Hand Sanitizer',
 'Health Drinks',
 'Healthy Snacks',
 'Honey',
 'Mosquito Repellent',
 'Pasta & Noodles',
 'Pre & Post Wash Care',
 'Rodent Control',
 'Tomato Ketchup',
 'Veg & Fruit'}
```

User-Item similarity can be found by using cosine similarity/correlation based similarity. User-Item based collaborative filtering works on the principle of finding similar users based of their buying preferences based on previous transactions history.Once the similar users are found,and the preferences are known, recommendations can be made by finding differences between the list of users.

This feature can be used in SMART BASKET along with personalised association rule as an hybrid model. This will not only help customer in user experiences by making them scroll less through so many products but it will also help Big Basket sell/cross sell some additional products to the users which they have not bought. Thus finding user-item similarity benefit consumers as well as Big Basket.

10. How do you propose to build a Smart Basket for a customer on Bigbasket?

Smart Basket is proposed to help users with creation of basket automatically as soon as the cutomer logins thus reducing burden on user that he/she has to go by scrolling through huge list of products, thus improving customer experience.

Smart Basket thus can also personalized recommendation system that gives user the products that they are most likely to buy based on their previous transactions.

So to create a SMART BASKET, I suggest a hybrid approach of combining both collaborative filtering (user-item) and personalized association rules.

Using Collaborative Filtering: Based on the users preferences,similarity between users can be found and then list of products for similar users can be taken and then taking the difference between lists will give the products that the user is likely to buy.

Personlized Association Rules: Based on users previous history, association rules can be found out which gives idea about which product goes with which another, thus giving list of products that are brought by user quiet frequently.

Combining both the list can create a very helpful basket for the user which will lessen the users burden of scrolling through long list.This will enhance user experience on the platform and will reduce order time drastically.

In [ ]: