Collaborative filtering - memory based using cosine distance and kNN

Recommender systems are an integral part of many online systems. From e-commerce to online streaming platforms. Recommender systems employ the past purchase patters on it's user to predict which other products they may in interested in and likey to purchase. Recommending the right products gives a significat advantage to the business. A mojor portion of the revenue is generated through recommendations.

The Collaborative Filtering algorithm is very popular in online streaming platforms and e-commerse sites where the customer interacts with each product (which can be a movie/ song or consumer products) by either liking/ disliking or giving a rating of sorts. One of the requirements to be able to apply collaborative filtering is that sufficient number of products need ratings associated with not them. User interaction is required.

This notebook walks through the implementation of collaborative filtering using memory based technique of distnce proximity using cosine distances and nearest neighbours.

Importing libraries and initial data checks

In [36]:
```
# import required libraries
import pandas as pd
import numpy as np
```

About the data

This is a dataset related to over 2 Million customer reviews and ratings of Beauty related products sold on Amazon's website.

It contains:

- the unique UserId (Customer Identification),
- the product ASIN (Amazon's unique product identification code for each product),
- Ratings (ranging from 1-5 based on customer satisfaction) and
- the Timestamp of the rating (in UNIX time)

In [37]:
```
# read the dataset
df = pd.read_csv('ratings_Beauty.csv')
df.shape
```

Out[37]:
```
(2023070, 4)
```

In [38]:
```python
# check the first 5 rows
df.head()
```

Out[38]:

	UserId	ProductId	Rating	Timestamp
0	A39HTATAQ9V7YF	0205616461	5.0	1369699200
1	A3JM6GV9MNOF9X	0558925278	3.0	1355443200
2	A1Z513UWSAAO0F	0558925278	5.0	1404691200
3	A1WMRR494NWEWV	0733001998	4.0	1382572800
4	A3IAAVS479H7M7	0737104473	1.0	1274227200

Check if there are any duplicate values present

In [39]:
```python
duplicates = df.duplicated(["UserId","ProductId", "Rating", "Timestamp"]).sum()
print(' Duplicate records: ',duplicates)
```

```
 Duplicate records:  0
```

See the number of unique values present

In [40]:
```python
print('unique users:',len(df.UserId.unique()))
print('unique products:',len(df.ProductId.unique()))
print("total ratings: ",df.shape[0])
```

```
unique users: 1210271
unique products: 249274
total ratings:  2023070
```

Check for null values

In [41]:
```python
df.isnull().any()
```

Out[41]:
```
UserId       False
ProductId    False
Rating       False
Timestamp    False
dtype: bool
```

Number of rated products per user

In [42]:
```python
products_user= df.groupby(by = "UserId")["Rating"].count().sort_values(ascending
products_user.head()
```

Out[42]:
```
UserId
A3KEZLJ59C1JVH    389
A281NPSIMI1C2R    336
A3M174IC0VXOS2    326
A2V5R832QCSOMX    278
A3LJLRIZL38GG3    276
Name: Rating, dtype: int64
```

Number of ratings per product

In [43]:
```python
product_rated = df.groupby(by = "ProductId")["Rating"].count().sort_values(ascend
product_rated.head()
```

Out[43]:
```
ProductId
B001MA0QY2    7533
B0009V1YR8    2869
B0043OYFKU    2477
B0000YUXI0    2143
B003V265QW    2088
Name: Rating, dtype: int64
```

Number of products rated by each user

In [44]:
```python
rated_users=df.groupby("UserId")["ProductId"].count().sort_values(ascending=False
print(rated_users)
```

```
UserId
A3KEZLJ59C1JVH           389
A281NPSIMI1C2R           336
A3M174IC0VXOS2           326
A2V5R832QCSOMX           278
A3LJLRIZL38GG3           276
                        ...
A3BQ47C773YMU1             1
A3BQ3Y37XL049D             1
A3BQ3NGQ3JJBR3             1
A3BQ3BW37JKZZ4             1
A00008821J0F472NDY6A2      1
Name: ProductId, Length: 1210271, dtype: int64
```

In [45]:
```
rated_products=df.groupby("ProductId")["UserId"].count().sort_values(ascending=Fa
print(rated_products)
```

```
ProductId
B001MA0QY2    7533
B0009V1YR8    2869
B0043OYFKU    2477
B0000YUXI0    2143
B003V265QW    2088
              ...
B005KEH11C       1
B005KECH48       1
B005KDU5XO       1
B005KDRZCS       1
0205616461       1
Name: UserId, Length: 249274, dtype: int64
```

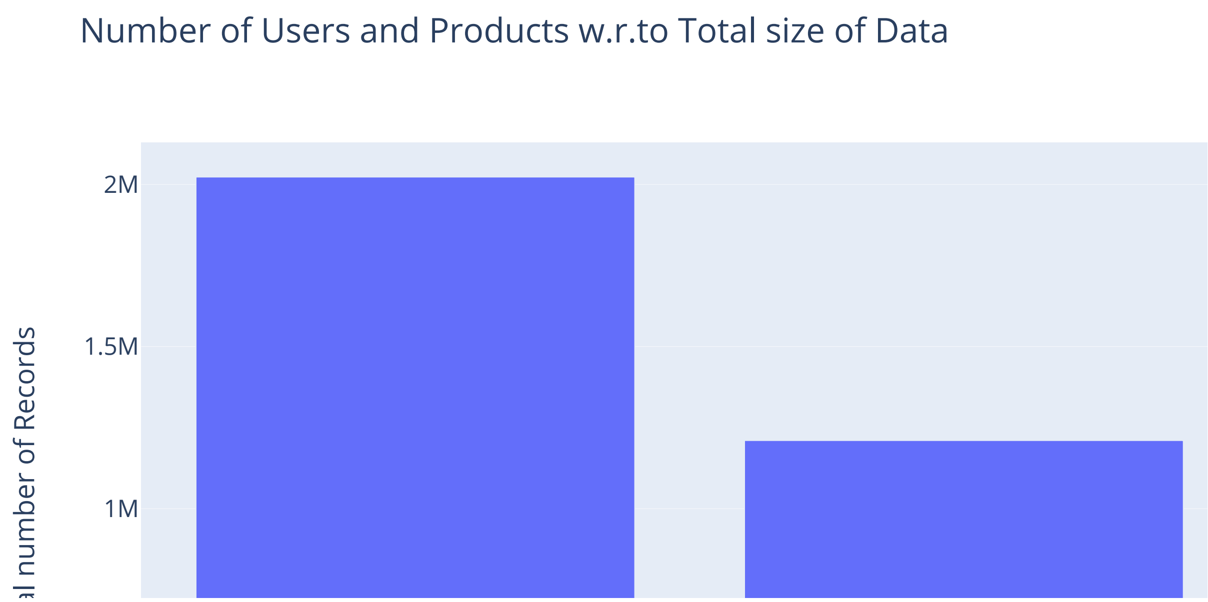
Number of products with some minimum ratings

In [46]:
```
print('Number of products with minimum of 5 reviews/ratings:',rated_products[rate
print('Number of products with minimum of 4 reviews/ratings:',rated_products[rate
print('Number of products with minimum of 3 reviews/ratings:',rated_products[rate
print('Number of products with minimum of 2 reviews/ratings:',rated_products[rate
print('Number of products with minimum of 1 reviews/ratings:',rated_products[rate
```

```
Number of products with minimum of 5 reviews/ratings: 57722
Number of products with minimum of 4 reviews/ratings: 67345
Number of products with minimum of 3 reviews/ratings: 81247
Number of products with minimum of 2 reviews/ratings: 103581
Number of products with minimum of 1 reviews/ratings: 145790
```

Visualizing the data

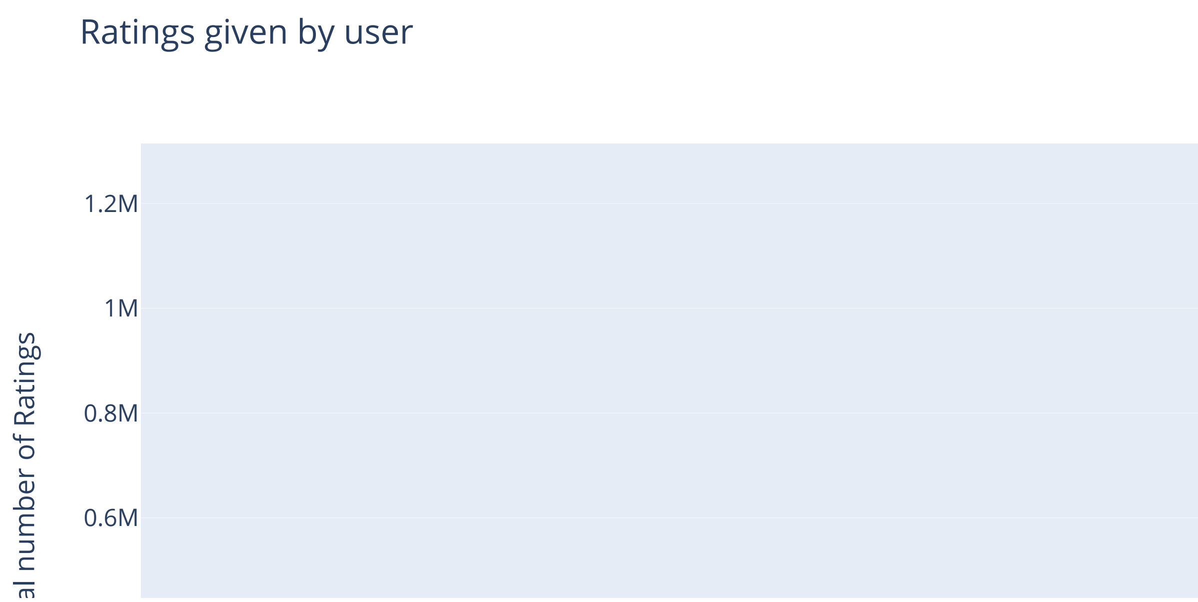
In [47]:
```
# plot the data
import plotly.graph_objects as go
index = ['Total size of records', "Number of unique users","Number of unique proc
values =[len(df),len(df['UserId'].unique()),len(df['ProductId'].unique())]

plot = go.Figure([go.Bar(x=index, y=values,textposition='auto')])
plot.update_layout(title_text='Number of Users and Products w.r.to Total size of
                   xaxis_title="Records",
                   yaxis_title="Total number of Records")

plot.show()
```



The ratings given by users

In [48]:
```
print("Range of Ratings: ", df['Rating'].value_counts())
print(list(df['Rating'].value_counts()))

values = list(df['Rating'].value_counts())

plot = go.Figure([go.Bar(x = df['Rating'].value_counts().index, y = values,textpo

plot.update_layout(title_text='Ratings given by user',
                   xaxis_title="Rating",
                   yaxis_title="Total number of Ratings")

plot.show()
```

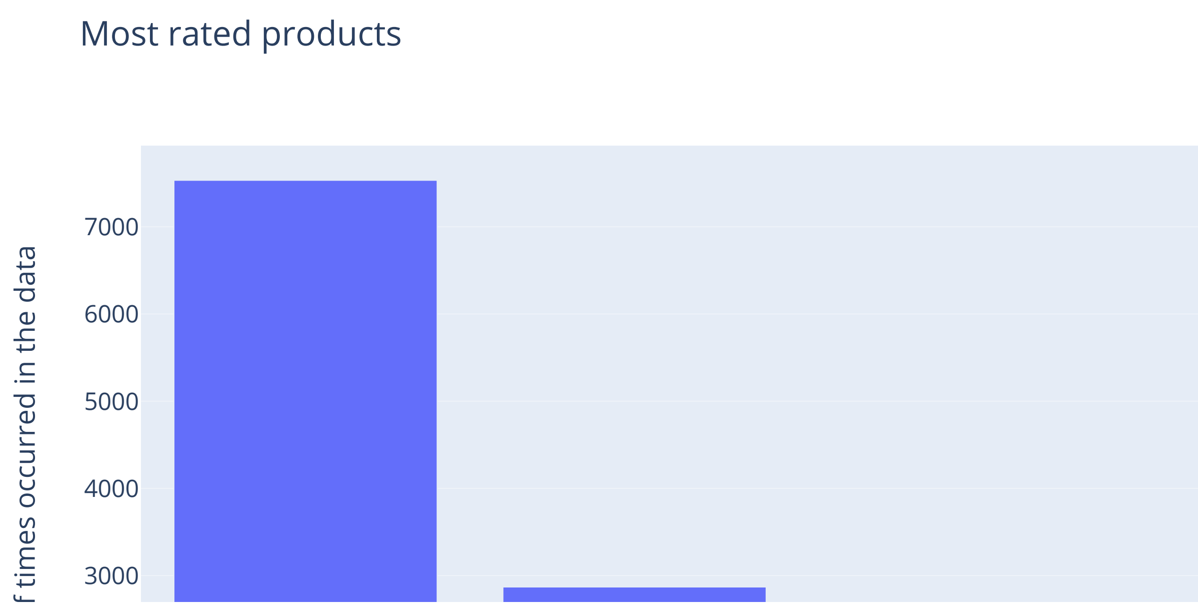
```
Range of Ratings:  5.0    1248721
4.0     307740
1.0     183784
3.0     169791
2.0     113034
Name: Rating, dtype: int64
[1248721, 307740, 183784, 169791, 113034]
```



Products which are most popular

```
In [49]: print("Products with occurred the most: \n",df['ProductId'].value_counts().nlarge

values = list(df['ProductId'].value_counts())


plot = go.Figure([go.Bar(x = df['ProductId'].value_counts().nlargest(5).index, y

plot.update_layout(title_text='Most rated products',
                    xaxis_title="ProductID",
                    yaxis_title="Number of times occurred in the data")

plot.show()
```

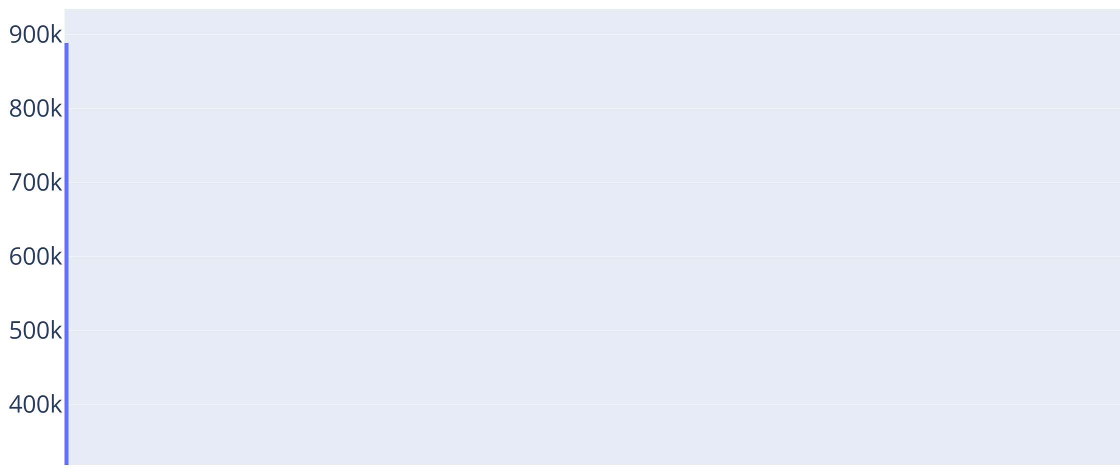
```
Products with occurred the most: 
 B001MA0QY2    7533
B0009V1YR8    2869
B0043OYFKU    2477
B0000YUXI0    2143
B003V265QW    2088
Name: ProductId, dtype: int64
```



Average rating given by each user

In [50]:
```python
ratings_per_user = df.groupby('UserId')['Rating'].count().sort_values(ascending=F
print("Average rating given by each user: ",ratings_per_user.head())

plot = go.Figure(data=[go.Histogram(x=ratings_per_user)])
plot.show()
```

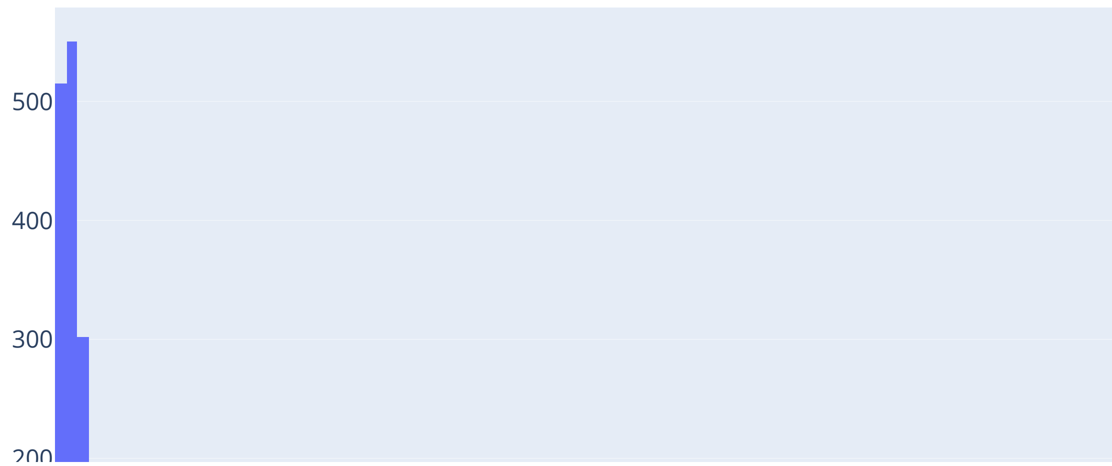
```
Average rating given by each user:  UserId
A3KEZLJ59C1JVH    389
A281NPSIMI1C2R    336
A3M174IC0VXOS2    326
A2V5R832QCSOMX    278
A3LJLRIZL38GG3    276
Name: Rating, dtype: int64
```



In [51]:

```
ratings_per_product = df.groupby('ProductId')['Rating'].count().sort_values(ascen
# print("Average rating given by each user: ",ratings_per_user.head())

plot = go.Figure(data=[go.Histogram(x=ratings_per_product)])
plot.show(title_text='Number of ratings per product',
                    xaxis_title="Product",
                    yaxis_title="Number of ratings")
```

```
In [52]: ratings_per_product = df.groupby('ProductId')['Rating'].count().sort_values(ascen
         # print("Average rating given by each user: ",ratings_per_user.head())

         plot = go.Figure(data=[go.Histogram(x=ratings_per_product.nlargest(2000))])
         plot.show(title_text='Number of ratings per product',
                             xaxis_title="Product",
                             yaxis_title="Number of ratings")
```



Products with very less ratings

In [53]:

```python
rating_of_products = df.groupby('ProductId')['Rating'].count()
# convert to make dataframe to analyse data
number_of_ratings_given = pd.DataFrame(rating_of_products)
print("Products with ratings given by users: \n",number_of_ratings_given.head())

less_than_ten = []
less_than_fifty_greater_than_ten = []
greater_than_fifty_less_than_hundred = []
greater_than_hundred = []
average_rating = []

for rating in number_of_ratings_given['Rating']:
    if rating <=10:
        less_than_ten.append(rating)
    if rating > 10 and rating <= 50:
        less_than_fifty_greater_than_ten.append(rating)
    if rating > 50 and rating <= 100:
        greater_than_fifty_less_than_hundred.append(rating)
    if rating > 100:
        greater_than_hundred.append(rating)

    average_rating.append(rating)

print("Ratings_count_less_than_ten: ", len(less_than_ten))
print("Ratings_count_greater_than_ten_less_than_fifty: ", len(less_than_fifty_gre
print("Ratings_count_greater_than_fifty_less_than_hundred: ", len(greater_than_fi
print("Ratings_count_greater_than_hundred: ", len(greater_than_hundred))
print("Average number of products rated by users: ", np.mean(average_rating))
```

```
Products with ratings given by users:
             Rating
ProductId
0205616461       1
0558925278       2
0733001998       1
0737104473       1
0762451459       1
Ratings_count_less_than_ten:  215395
Ratings_count_greater_than_ten_less_than_fifty:  27082
Ratings_count_greater_than_fifty_less_than_hundred:  4110
Ratings_count_greater_than_hundred:  2687
Average number of products rated by users:  8.115848423822781
```

In [54]:
```
x_values = ["Ratings_count_less_than_ten","Ratings_count_greater_than_ten_less_th
            "Ratings_count_greater_than_fifty_less_than_hundred","Ratings_count_gr
y_values = [len(less_than_ten),len(less_than_fifty_greater_than_ten),len(greater_
            len(greater_than_hundred)]


plot = go.Figure([go.Bar(x = x_values, y = y_values, textposition='auto')])

plot.add_annotation(
        x=1,
        y=100000,
        xref="x",
        yref="y")

plot.update_layout(title_text='Ratings Count on Products',
                    xaxis_title="Ratings Range",
                    yaxis_title="Count of Rating")
plot.show()
```

In [55]:
```
from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()
```

To convert alphanumeric data to numeric

In [56]:
```
dataset = df
dataset['user'] = label_encoder.fit_transform(df['UserId'])
dataset['product'] = label_encoder.fit_transform(df['ProductId'])
dataset.head()
```

Out[56]:

	UserId	ProductId	Rating	Timestamp	user	product
0	A39HTATAQ9V7YF	0205616461	5.0	1369699200	725046	0
1	A3JM6GV9MNOF9X	0558925278	3.0	1355443200	814606	1
2	A1Z513UWSAAO0F	0558925278	5.0	1404691200	313101	1
3	A1WMRR494NWEWV	0733001998	4.0	1382572800	291075	2
4	A3IAAVS479H7M7	0737104473	1.0	1274227200	802842	3

In [57]:

```python
# average rating given by each user
average_rating = dataset.groupby(by="user", as_index=False)['Rating'].mean()
print("Average rating given by users: \n",average_rating.head())
print("----------------------------------------------------------\n")


# let's merge it with the dataset as we will be using that later
dataset = pd.merge(dataset, average_rating, on="user")
print("Modified dataset: \n", dataset.head())
print("----------------------------------------------------------\n")

# renaming columns
dataset = dataset.rename(columns={"Rating_x": "real_rating", "Rating_y": "average
print("Dataset: \n", dataset.head())
print("----------------------------------------------------------\n")
```

```
Average rating given by users:
    user  Rating
0     0     5.0
1     1     5.0
2     2     3.0
3     3     5.0
4     4     5.0
----------------------------------------------------------

Modified dataset:
             UserId   ProductId  Rating_x   Timestamp    user  product  Rating_y
0  A39HTATAQ9V7YF  0205616461       5.0  1369699200  725046        0      4.25
1  A39HTATAQ9V7YF  B002OVV7F0       3.0  1369699200  725046    81854      4.25
2  A39HTATAQ9V7YF  B0031IH5FQ       5.0  1369699200  725046    89013      4.25
3  A39HTATAQ9V7YF  B006GQPZ8E       4.0  1369699200  725046   154092      4.25
4  A3JM6GV9MNOF9X  0558925278       3.0  1355443200  814606        1      3.50
----------------------------------------------------------

Dataset:
             UserId   ProductId  real_rating   Timestamp    user  product  \
0  A39HTATAQ9V7YF  0205616461          5.0  1369699200  725046        0
1  A39HTATAQ9V7YF  B002OVV7F0          3.0  1369699200  725046    81854
2  A39HTATAQ9V7YF  B0031IH5FQ          5.0  1369699200  725046    89013
3  A39HTATAQ9V7YF  B006GQPZ8E          4.0  1369699200  725046   154092
4  A3JM6GV9MNOF9X  0558925278          3.0  1355443200  814606        1

   average_rating
0            4.25
1            4.25
2            4.25
3            4.25
4            3.50
----------------------------------------------------------
```

Certain users tend to give higher ratings while others tend to gibve lower ratings. To negate this bias, we normalise the ratings given by the users.

In [58]:
```
dataset['normalized_rating'] = dataset['real_rating'] - dataset['average_rating']
print("Data with adjusted rating: \n", dataset.head())
```

```
Data with adjusted rating:
            UserId   ProductId  real_rating   Timestamp    user  product  \
0  A39HTATAQ9V7YF  0205616461          5.0  1369699200  725046        0
1  A39HTATAQ9V7YF  B002OVV7F0          3.0  1369699200  725046    81854
2  A39HTATAQ9V7YF  B0031IH5FQ          5.0  1369699200  725046    89013
3  A39HTATAQ9V7YF  B006GQPZ8E          4.0  1369699200  725046   154092
4  A3JM6GV9MNOF9X  0558925278          3.0  1355443200  814606        1

   average_rating  normalized_rating
0            4.25               0.75
1            4.25              -1.25
2            4.25               0.75
3            4.25              -0.25
4            3.50              -0.50
```

Cosine Similarity

We use a distance based metric - cosine similarity to identify similar users. It is important first, to remove products that have very low number of ratings.

Filter based on number of ratings available

In [59]:
```
rating_of_product = dataset.groupby('product')['real_rating'].count() # apply gro
ratings_of_products_df = pd.DataFrame(rating_of_product)
print("Real ratings:\n",ratings_of_products_df.head()) # check for real rating fo
```

```
Real ratings:
          real_rating
product
0                  1
1                  2
2                  1
3                  1
4                  1
```

In [60]:
```
filtered_ratings_per_product = ratings_of_products_df[ratings_of_products_df.real
print(filtered_ratings_per_product.head())
print(filtered_ratings_per_product.shape)
```

```
         real_rating
product
704              558
719              377
754              288
834              412
843              313
(934, 1)
```

In [61]:
```
# build a list of products to keep
popular_products = filtered_ratings_per_product.index.tolist()
print("Popular product count which have ratings over average rating count: ",len(
print("-----------------------------------------------------------------------

filtered_ratings_data = dataset[dataset["product"].isin(popular_products)]
print("Filtered rated product in the dataset: \n",filtered_ratings_data.head())
print("-----------------------------------------------------------------------

print("The size of dataset has changed from ", len(dataset), " to ", len(filtered
print("-----------------------------------------------------------------------
```

```
Popular product count which have ratings over average rating count:  934
--------------------------------------------------------------------------------
-
Filtered rated product in the dataset:
             UserId   ProductId  real_rating   Timestamp     user  product  \
1   A39HTATAQ9V7YF  B002OVV7F0          3.0  1369699200   725046    81854
18   AKJHHD5VEH7VG  B0000UTUVU          5.0  1232323200  1073169     2237
20   AKJHHD5VEH7VG  B000F8HWXU          5.0  1379721600  1073169    16510
45   AKJHHD5VEH7VG  B001LF4I8I          4.0  1232841600  1073169    65074
47   AKJHHD5VEH7VG  B001OMI93S          5.0  1236643200  1073169    67333

    average_rating  normalized_rating
1         4.250000          -1.250000
18        4.222222           0.777778
20        4.222222           0.777778
45        4.222222          -0.222222
47        4.222222           0.777778
--------------------------------------------------------------------------------
--
The size of dataset has changed from  2023070  to  370511
--------------------------------------------------------------------------------
--
```

Creating the User-item matrix

In [62]:
```
similarity = pd.pivot_table(filtered_ratings_data,values='normalized_rating',inde
similarity = similarity.fillna(0)
print("Updated Dataset: \n",similarity.head())
```

```
Updated Dataset: 
 product                704     719     754     834     843     858     861  \
UserId                                                                         
A0010876CNE3ILIM9HV0      0.0     0.0     0.0     0.0     0.0     0.0     0.0
A0011102257KBXODKL24I     0.0     0.0     0.0     0.0     0.0     0.0     0.0
A00120381FL204MYH7G3B     0.0     0.0     0.0     0.0     0.0     0.0     0.0
A00126503SUWI86KZBMIN     0.0     0.0     0.0     0.0     0.0     0.0     0.0
A001573229XK5T8PI0OKA     0.0     0.0     0.0     0.0     0.0     0.0     0.0

product                873     944     981     ...  241604  242018  242048  \
UserId                                         ...                          
A0010876CNE3ILIM9HV0      0.0     0.0     0.0  ...     0.0     0.0     0.0
A0011102257KBXODKL24I     0.0     0.0     0.0  ...     0.0     0.0     0.0
A00120381FL204MYH7G3B     0.0     0.0     0.0  ...     0.0     0.0     0.0
A00126503SUWI86KZBMIN     0.0     0.0     0.0  ...     0.0     0.0     0.0
A001573229XK5T8PI0OKA     0.0     0.0     0.0  ...     0.0     0.0     0.0

product                243416  244376  244448  245600  247603  249109  249211
UserId                                                                        
A0010876CNE3ILIM9HV0      0.0     0.0     0.0     0.0     0.0     0.0     0.0
A0011102257KBXODKL24I     0.0     0.0     0.0     0.0     0.0     0.0     0.0
A00120381FL204MYH7G3B     0.0     0.0     0.0     0.0     0.0     0.0     0.0
A00126503SUWI86KZBMIN     0.0     0.0     0.0     0.0     0.0     0.0     0.0
A001573229XK5T8PI0OKA     0.0     0.0     0.0     0.0     0.0     0.0     0.0

[5 rows x 934 columns]
```

As you can see, this is a very sparse matrix

In [63]:
```
from sklearn.metrics.pairwise import cosine_similarity
import operator
```

In [64]:
```python
selecting_users = list(similarity.index)
selecting_users = selecting_users[:100]
print("You can select users from the below list:\n",selecting_users)
```

```
You can select users from the below list:
 ['A0010876CNE3ILIM9HV0', 'A0011102257KBXODKL24I', 'A00120381FL204MYH7G3B', 'A0
0126503SUWI86KZBMIN', 'A001573229XK5T8PI0OKA', 'A00203203EBR4E6BIUOKF', 'A00222
842T0ZYI86C9LHU', 'A00258542AL4VKETFLGIJ', 'A00259242VSCRZPGIWP0M', 'A00262022J
QPXX5SXEVJR', 'A00275441WYR3489IKNAB', 'A00328401T70RFN4P1IT6', 'A00349462AOAVU
UPEJNQZ', 'A00370223FX3K9TUF1QCL', 'A00407141VL6SB77B1GGG', 'A00414041RD0BXM6WK
0GX', 'A00426443G4MEWS3K1XFA', 'A004511036AHSSV5O4SBY', 'A00454102SR84NOYTI0J
S', 'A00463203QYS5I5X6MMXW', 'A00473363TJ8YSZ3YAGG9', 'A00491723IYKW5UI74AEX',
'A0058336347PC7BSR0UJC', 'A00612582Z6ZU2SDMRQ07', 'A00615442TZG6MHZXJOIZ', 'A00
627983P6OGUFJ3IW8H', 'A006502622TE53S3J9W6H', 'A00656692CXO0VGF00V9I', 'A006680
338J29DP17XALU', 'A00669491O55AKJ5QVH9L', 'A00679332RYOO5406ARSG', 'A00700212KB
3K0MVESPIY', 'A0072717335KA6520NEMI', 'A0074075A8TZJIPLGZEK', 'A00773851NXKGCZR
Y43PG', 'A0078719IR14X3NNUG0F', 'A00802872RVW2KLY6DAL0', 'A008374338GH2TUB0S8K
P', 'A00852491YPMY2HLYZ52N', 'A0086401DFJEZA4RT4OL', 'A0090635250IP002KMMIX',
'A00995931BE16NG4F52QC', 'A01026292DKV5RYUH42C9', 'A01032093UTJ2SF3EQFS1', 'A01
0356935P3D9IEDEUIN', 'A010399725YK04VCLI8KI', 'A010407538LRAQYK3G2RZ', 'A010924
53CM8U3BVJHXJH', 'A01100491G7V0HTBV9WNO', 'A0116143FGG14B3OZ7UG', 'A0116899HIQE
DWSBJJG9', 'A01184631PAAXN2HOZGBY', 'A01198201H0E3GHV2Z17I', 'A012050730XENR11I
5PFP', 'A01235573820O0EST7S2UG', 'A01247753D6GFZD87MUV8', 'A01254212HTK2F3B304M
8', 'A01254332UU57MKWKP4VI', 'A012668725TCXOBEMGHBA', 'A01270511XXRGIDK72VQ',
'A0127703SEG0Z39MBNUL', 'A01288351ESHZ2KNAXBJ7', 'A01290231HW9YARUTSI41', 'A013
18121KA8R6FZSG436', 'A0136234302D2DRZLC42E', 'A013805820H8FMU1TKEK4', 'A0138754
2H6B05F44MHEG', 'A0140494QSPWAFGBI083', 'A01415083E93PJ008V99K', 'A01416443H8V3
1K9RB4GJ', 'A01456542S5QPYUEGJXR8', 'A01458343GQCWL1L33TC', 'A01476221J13M6NSDW
ZD0', 'A014943835U2Y0QOX9Q42', 'A01504262HV6PBSBTE8L', 'A0152362223269105BGJE',
'A01524401TZMB1ZBFP908', 'A015565634RZNSDLJBE5M', 'A01571181KF75C2M4GMGB', 'A01
614701G606FLBUNKML', 'A01643342TLO8AB9ZXLR4', 'A0166376211UTK5BSWM6W', 'A016810
8D3MS83PBNHSZ', 'A0171438140M13IHT8N6X', 'A01720702S8VXTOFEVMJ5', 'A01721922VEZ
1YPPY536P', 'A01729543L79FIQFOLPX0', 'A01762922WHF1Q6CPN7F6', 'A01836141R5V4GOT
IUND1', 'A01862021NSU0BBVBENPH', 'A01862461GV7VHBX35NBK', 'A01895041IWQIA3MJC6E
W', 'A01907982I6OHXDYN5HD6', 'A019259031LZJU6HN9ZQ6', 'A0194386SN4D2X61X94I',
'A02041131H1NHGGGF4AMW', 'A0207511NKFVPWSTQEOY', 'A0215307EJR86RMIPQZH', 'A0215
5413BVL8D0G7X6DN', 'A02157553CY714JSIXQMJ']
```

In [65]:
```
def getting_top_5_similar_users(user_id, similarity_table, k=5):
    '''


    :param user_id: the user we want to recommend
    :param similarity_table: the user-item matrix
    :return: Similar users to the user_id.
    '''

    # create a dataframe of just the current user
    user = similarity_table[similarity_table.index == user_id]
    # and a dataframe of all other users
    other_users = similarity_table[similarity_table.index != user_id]
    # calculate cosine similarity between user and each other user
    similarities = cosine_similarity(user, other_users)[0].tolist()

    indices = other_users.index.tolist()
    index_similarity = dict(zip(indices, similarities))

    # sort by similarity
    index_similarity_sorted = sorted(index_similarity.items(), key=operator.itemg
    index_similarity_sorted.reverse()

    # take users
    top_users_similarities = index_similarity_sorted[:k]
    users = []
    for user in top_users_similarities:
        users.append(user[0])

    return users
```

In [66]:
```
user_id = "A0010876CNE3ILIM9HV0"
similar_users = getting_top_5_similar_users(user_id, similarity)
```

In [67]:
```
print("Top 5 similar users for user_id:",user_id," are: ",similar_users)
```

```
Top 5 similar users for user_id: A0010876CNE3ILIM9HV0  are:  ['AXNF1BLDR4P47',
'ARTHT19OB79VZ', 'ARQ9I3Y0VPB6N', 'AOXEXSN7M9ENJ', 'AN0AO97264HP4']
```

Recommend products based on these top similar users

In [68]:
```
def getting_top_5_recommendations_based_on_users(user_id, similar_users, similari
    '''

    :param user_id: user for whom we want to recommend
    :param similar_users: top 5 similar users
    :param similarity_table: the user-item matrix
    :param top_recommendations: no. of recommendations
    :return: top_5_recommendations
    '''

    # taking the data for similar users
    similar_user_products = dataset[dataset.UserId.isin(similar_users)]
#     print("Products used by other users: \n", similar_user_products.head())
#     print("-------------------------------------------------------------------

    # getting all similar users
    similar_users = similarity_table[similarity_table.index.isin(similar_users)]

    #getting mean ratings given by users
    similar_users = similar_users.mean(axis=0)


    similar_users_df = pd.DataFrame(similar_users, columns=['mean'])

    # for the current user data
    user_df = similarity_table[similarity_table.index == user_id]


    # transpose it so its easier to filter
    user_df_transposed = user_df.transpose()


    # rename the column as 'rating'
    user_df_transposed.columns = ['rating']

    # rows with a 0 value.
    user_df_transposed = user_df_transposed[user_df_transposed['rating'] == 0]


    # generate a list of products the user has not used
    products_not_rated = user_df_transposed.index.tolist()
#     print("Products not used by target user: ", products_not_rated)
#     print("------------------------------------------------------------------'

    # filter avg ratings of similar users for only products the current user has
    similar_users_df_filtered = similar_users_df[similar_users_df.index.isin(proc

    # order the dataframe
    similar_users_df_ordered = similar_users_df_filtered.sort_values(by=['mean'],



    # take the top products
    top_products = similar_users_df_ordered.head(top_recommendations)
    top_products_indices = top_products.index.tolist()
```

```
    return top_products_indices
```

In [69]:
```
print("Top 5 productID recommended are: ",
      getting_top_5_recommendations_based_on_users(user_id, similar_users, simila
```

```
Top 5 productID recommended are:  [704, 122630, 119407, 119506, 119742]
```

In [70]:
```
filtered_ratings_data.shape
```

Out[70]:
```
(370511, 8)
```

In [71]:
```
filtered_ratings_data.head()
```

Out[71]:

	UserId	ProductId	real_rating	Timestamp	user	product	average_rating	nc
1	A39HTATAQ9V7YF	B002OVV7F0	3.0	1369699200	725046	81854	4.250000	
18	AKJHHD5VEH7VG	B0000UTUVU	5.0	1232323200	1073169	2237	4.222222	
20	AKJHHD5VEH7VG	B000F8HWXU	5.0	1379721600	1073169	16510	4.222222	
45	AKJHHD5VEH7VG	B001LF4I8I	4.0	1232841600	1073169	65074	4.222222	
47	AKJHHD5VEH7VG	B001OMI93S	5.0	1236643200	1073169	67333	4.222222	

In [72]:
```
filtered_ratings_data[filtered_ratings_data['UserId']=="A0010876CNE3ILIM9HV0"]
```

Out[72]:

	UserId	ProductId	real_rating	Timestamp	user	product	average_rat
1160176	A0010876CNE3ILIM9HV0	B0055MYJ0U	1.0	1390521600	11	136012	

In [73]:
```
from sklearn.model_selection import train_test_split
train_data, test_data = train_test_split(filtered_ratings_data,test_size=0.2)

train_data = pd.DataFrame(train_data)
test_data = pd.DataFrame(test_data)
```

In [74]:
```
similarity = pd.pivot_table(train_data,values='normalized_rating',index='UserId',
similarity = similarity.fillna(0)
print("Updated Dataset: \n",similarity.head())
```

```
Updated Dataset:
 product                704    719    754    834    843    858    861  \
UserId
A0011102257KBXODKL24I    0.0    0.0    0.0    0.0    0.0    0.0    0.0
A00126503SUWI86KZBMIN    0.0    0.0    0.0    0.0    0.0    0.0    0.0
A001573229XK5T8PI0OKA    0.0    0.0    0.0    0.0    0.0    0.0    0.0
A00203203EBR4E6BIUOKF    0.0    0.0    0.0    0.0    0.0    0.0    0.0
A00222842T0ZYI86C9LHU    0.0    0.0    0.0    0.0    0.0    0.0    0.0

product                873    944    981    ...  241604  242018  242048  \
UserId                                      ...
A0011102257KBXODKL24I    0.0    0.0    0.0  ...     0.0     0.0     0.0
A00126503SUWI86KZBMIN    0.0    0.0    0.0  ...     0.0     0.0     0.0
A001573229XK5T8PI0OKA    0.0    0.0    0.0  ...     0.0     0.0     0.0
A00203203EBR4E6BIUOKF    0.0    0.0    0.0  ...     0.0     0.0     0.0
A00222842T0ZYI86C9LHU    0.0    0.0    0.0  ...     0.0     0.0     0.0

product                243416  244376  244448  245600  247603  249109  249211
UserId
A0011102257KBXODKL24I    0.0    0.0    0.0    0.0    0.0    0.0    0.0
A00126503SUWI86KZBMIN    0.0    0.0    0.0    0.0    0.0    0.0    0.0
A001573229XK5T8PI0OKA    0.0    0.0    0.0    0.0    0.0    0.0    0.0
A00203203EBR4E6BIUOKF    0.0    0.0    0.0    0.0    0.0    0.0    0.0
A00222842T0ZYI86C9LHU    0.0    0.0    0.0    0.0    0.0    0.0    0.0

[5 rows x 934 columns]
```

In [75]:
```
similarity.shape
```

Out[75]:
```
(251692, 934)
```

In [76]:
```python
selecting_users = list(similarity.index)
selecting_users = selecting_users[:100]
print("You can select users from the below list:\n",selecting_users)
```

```
You can select users from the below list:
 ['A0011102257KBXODKL24I', 'A00126503SUWI86KZBMIN', 'A001573229XK5T8PI0OKA', 'A
00203203EBR4E6BIUOKF', 'A00222842T0ZYI86C9LHU', 'A00258542AL4VKETFLGIJ', 'A0027
5441WYR3489IKNAB', 'A00328401T70RFN4P1IT6', 'A00349462AOAVUUPEJNQZ', 'A00370223
FX3K9TUF1QCL', 'A00407141VL6SB77B1GGG', 'A00414041RD0BXM6WK0GX', 'A00426443G4ME
WS3K1XFA', 'A004511036AHSSV5O4SBY', 'A00463203QYS5I5X6MMXW', 'A00473363TJ8YSZ3Y
AGG9', 'A00491723IYKW5UI74AEX', 'A0058336347PC7BSR0UJC', 'A00612582Z6ZU2SDMRQ0
7', 'A00615442TZG6MHZXJOIZ', 'A00627983P6OGUFJ3IW8H', 'A006502622TE53S3J9W6H',
'A00656692CXO0VGF00V9I', 'A006680338J29DP17XALU', 'A00669491O55AKJ5QVH9L', 'A00
700212KB3K0MVESPIY', 'A0072717335KA6520NEMI', 'A00773851NXKGCZRY43PG', 'A007871
9IR14X3NNUG0F', 'A00802872RVW2KLY6DAL0', 'A008374338GH2TUB0S8KP', 'A00852491YPM
Y2HLYZ52N', 'A0090635250IP002KMMIX', 'A00995931BE16NG4F52QC', 'A01026292DKV5RYU
H42C9', 'A010356935P3D9IEDEUIN', 'A010399725YK04VCLI8KI', 'A010407538LRAQYK3G2R
Z', 'A01092453CM8U3BVJHXJH', 'A01100491G7V0HTBV9WNO', 'A0116143FGG14B3OZ7UG',
'A0116899HIQEDWSBJJG9', 'A01184631PAAXN2HOZGBY', 'A01198201H0E3GHV2Z17I', 'A012
050730XENR11I5PFP', 'A0123557382O0EST7S2UG', 'A01247753D6GFZD87MUV8', 'A0125421
2HTK2F3B304M8', 'A01254332UU57MKWKP4VI', 'A012668725TCXOBEMGHBA', 'A01270511XXR
GIDK72VQ', 'A01288351ESHZ2KNAXBJ7', 'A01290231HW9YARUTSI41', 'A01318121KA8R6FZS
G436', 'A0136234302D2DRZLC42E', 'A0140494QSPWAFGBI083', 'A01415083E93PJ008V99
K', 'A01456542S5QPYUEGJXR8', 'A01458343GQCWL1L33TC', 'A01476221J13M6NSDWZD0',
'A01504262HV6PBSBTE8L', 'A01524401TZMB1ZBFP908', 'A015565634RZNSDLJBE5M', 'A016
14701G606FLBUNKML', 'A01643342TLO8AB9ZXLR4', 'A0166376211UTK5BSWM6W', 'A0172070
2S8VXTOFEVMJ5', 'A01721922VEZ1YPPY536P', 'A01729543L79FIQFOLPX0', 'A01762922WHF
1Q6CPN7F6', 'A01836141R5V4GOTIUND1', 'A01862021NSU0BBVBENPH', 'A01862461GV7VHBX
35NBK', 'A01895041IWQIA3MJC6EW', 'A01907982I6OHXDYN5HD6', 'A019259031LZJU6HN9ZQ
6', 'A0194386SN4D2X61X94I', 'A02041131H1NHGGGF4AMW', 'A0207511NKFVPWSTQEOY', 'A
0215307EJR86RMIPQZH', 'A02155413BVL8D0G7X6DN', 'A02157553CY714JSIXQMJ', 'A02238
611KADHRFBPKAUK', 'A0224753305ZV8QY08RNV', 'A022567136584HD6WMJEI', 'A022740531
9CQ7DRWE1NW', 'A02305199NGR9IPD7HQF', 'A02334792IGUVYEPAUILV', 'A02339192NYXG9Z
QAG8NF', 'A0234489KXEW3JGS5XOF', 'A023535518XEETBJH6M1H', 'A02354170VQ79DHUZH3
9', 'A02467692DUHXNZCBP4Q8', 'A02532953BCG7KE24PIKE', 'A025502934P1K8T62GFM0',
'A0258781CVLYI3ZR4YQW', 'A0265436JMR91F9LHBXT', 'A0266076X6KPZ6CCHGVS', 'A02669
30CH3HIT4CKLDG', 'A02680541GF3IVW82HUBK']
```

In [77]:
```python
user_id = "A02720223TDVZSWVZYFN7"
similar_users = getting_top_5_similar_users(user_id, similarity)
```

In [78]:
```python
print("Top 5 similar users for user_id:",user_id," are: ",similar_users)
```

```
Top 5 similar users for user_id: A02720223TDVZSWVZYFN7  are:  ['AZZZRS1YZ8HVP',
'AZZZLM1E5JJ8C', 'AZZYW4YOE1B6E', 'AZZWMH759YWOO', 'AZZWJ3LICUEKJ']
```

In [79]:
```python
print("Top 5 productID recommended are: ",
      getting_top_5_recommendations_based_on_users(user_id, similar_users, simila
```

```
Top 5 productID recommended are:  [27327, 149282, 704, 119506, 119742]
```

In [80]:
```
test_data.shape
```

Out[80]:
```
(74103, 8)
```

In [81]:
```
len(test_data.user.unique())
```

Out[81]:
```
70016
```

In [82]:
```
test_data.UserId
```

Out[82]:
```
920862     A2EOX5XAVNXE2O
578494     A3U518UQGVSDTB
1471347    A1X47M5ITWUFA6
1763334    A3GPPUDOMRRQ16
1957830    A2QRRJEVHRLO9Y
                ...
1103762    A1SD8BC4FHX61P
741602     A32E7GHA5VYSVZ
1049934    A26EKYLBPGN3ZB
708504     A18I4EONOE4YMA
256298     A2ANBZ5YDZXG32
Name: UserId, Length: 74103, dtype: object
```

In [83]:
```
test_data.head()
```

Out[83]:

	UserId	ProductId	real_rating	Timestamp	user	product	average_rati
920862	A2EOX5XAVNXE2O	B0010XUU9M	4.0	1400544000	451075	42130	3.8947
578494	A3U518UQGVSDTB	B000G666HE	5.0	1363305600	907709	18032	5.000(
1471347	A1X47M5ITWUFA6	B003UH0528	5.0	1374537600	295358	104577	3.500(
1763334	A3GPPUDOMRRQ16	B006U95N34	5.0	1385424000	788952	157463	5.000(
1957830	A2QRRJEVHRLO9Y	B00CXADQ4M	5.0	1397347200	558611	219568	5.000(

In [84]:
```
def recommend_products_for_user(userId, similarity_matrix):
    similar_users = getting_top_5_similar_users(user_id, similarity_matrix)
#     print("Top 5 similar users for user_id:",user_id," are: ",similar_users)
    product_list = getting_top_5_recommendations_based_on_users(user_id, similar_
#     print("Top 5 productID recommended are: ", product_list)
    return product_list
```

In [85]:
```
recommend_products_for_user("A2XVNI270N97GL", similarity)
```

Out[85]:
```
[27327, 149282, 704, 119506, 119742]
```

Conclusion

Recommender systems are a powerful technology that adds to a businesses value. Some business thrive on their recommender systems. It helps the business by creating more sales and it helps the end user buy enabling them to find items they like.