



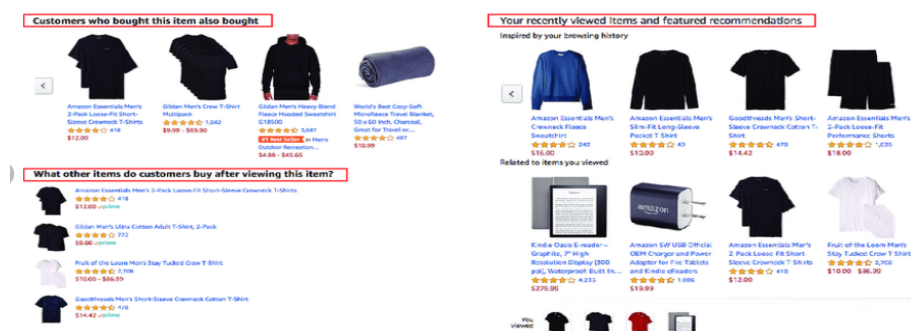
چالش دوم: Recommender system using Amazon reviews

فراهم آورنده: سینا حیدری

زمستان ۹۸

۱ در مورد این چالش

شرکت‌های E-Commerce همچون Amazon و Flipkart از سیستم‌های توصیه‌گر متفاوتی برای پیشنهاد دادن به مشتری‌های خود استفاده می‌کنند. آمازون از item-item collaborative filtering استفاده می‌کند؛ این نوع از collaborative filtering مقیاس‌پذیری بالایی دارد و یک سیستم توصیه‌گر بی‌درنگ، با کیفیت بالا فراهم می‌آورد. این سیستم یک نوع از سیستم‌های فیلترینگ می‌باشد که هدفش پیش‌بینی رتبه یا اولویت‌های مشتری است.



An example of Amazon recommender system.

۲ بررسی راه حل

انواع سیستم‌های توصیه‌گر

به‌طور کلی ۶ نوع سیستم توصیه‌گر وجود دارد:

- سیستم‌های مبتنی بر محبوبیت
- سیستم‌های مبتنی بر مدل‌های دسته‌بندی
- سیستم‌های مبتنی بر محتوا
- فیلترینگ مشارکتی
- رویکردهای ترکیبی
- کاوش قواعد وابستگی

در این رقابت، ما یک سیستم توصیه‌گر فیلترینگ اشتراکی از نوع item-item را معرفی و پیاده‌سازی می‌کنیم.

Item-Item Collaborative Filtering

این نوع از فیلترینگ اشتراکی از آن دسته از روش‌های توصیه‌ای است که به دنبال آیتم‌های مشابه بر اساس آیتم‌هایی که پیش از این مورد پسند کاربران بوده یا با آن‌ها به صورت مثبت تعامل داشته‌اند، می‌گردد. این روش توسط آمازون در سال ۱۹۹۸ توسعه داده شد و نقش مهمی در موفقیت آمازون داشته است.



IBFC بر اساس آیتم‌هایی که قبلاً توسط کاربر استفاده شده عمل می‌کند. این روش ابتدا به آیتم‌هایی که توسط کاربر خریداری شده‌اند نگاه کرده، سپس، آیتم‌های مشابه آن‌ها را پیدا می‌کند.

برای نمونه؛ فرض کنید الکس می‌خواهد یک فیلم در قالب «دی‌وی‌دی» خریداری کند. کار ما این است که به الکس یک فیلم بر اساس ترجیح‌های گذشته‌اش پیشنهاد بدهیم. ابتدا، به جست‌وجوی فیلم‌هایی که دیده یا «لایک» کرده می‌پردازیم؛ بیایید این فیلم‌ها را A، B، C بنامیم. در ادامه، ما به جست‌وجوی فیلم‌های مشابه به این سه فیلم می‌پردازیم. فرض کنید ما متوجه شدیم فیلم D مشابه با فیلم C است، در نتیجه، احتمال اینکه الکس این فیلم را تماشا کند بالاست.

اما سوال اصلی اینجاست که چگونه آیتم‌های مشابه را بیابیم؟ و اگر چندین آیتم مشابه وجود دارد، کدام آیتم را اول پیشنهاد دهیم؟ برای درک این مسئله، بیاید در ابتدا به بصیرت این روند پی ببریم. این به ما کمک خواهد کرد ریاضیاتی که IBCF بر پایه‌ی آن‌ها عمل می‌کند را بهتر بفهمیم.

پیدا کردن تشابه میان آیتم‌ها

برای نمونه، این جدول از کاربران و رتبه‌هایی که به فیلم‌ها داده‌اند را در نظر بگیرید:

User	1: Toy Story	2: Star Wars	Epi: 356: Forrest Gump	318: Shawshank Redemption	593: Silence of the Lambs	3578: Gladiator
755	2		5	2		4
5277	1				2	4
1577					5	2
4388	2		3			1
1202			3	4	1	4
3823	3		4	4	4	1
5448				3	1	4
5347	2					3
4117	4		1		4	2
2765	4		2		5	3
5450	5		1	5		
139	2		5	2		
1940	4				5	4
3118	3			3		2
4656	2		4			5

ما دو فیلم را کرده (Id 1: Toy Story and Id 2: Star Wars) و تشابه آن‌ها را محاسبه خواهیم کرد. مقصودمان مقایسه‌ی این دو بر اساس مقبولیت کاربران می‌باشد. برای محاسبه‌ی این شباهت، ابتدا رتبه‌های کاربران برای این دو فیلم را در هم ضرب کرده سپس، حاصل ضرب‌ها را با هم جمع می‌کنیم.

User	1: Toy Story	2: Star Wars	Product of Movie Ratings
755	2	5	10
5277	1		0
1577			0
4388	2	3	6
1202		3	0
3823	3	4	12
5448			0
5347	2		0
4117	4	1	4
2765	4	2	8
5450	5	1	5
139	2	5	10
1940	4		0
3118	3		0
4656	2	4	8
4796			0
6037	2		0
3048	4	5	20
4790	2	1	2
4489	2	2	4
			A = 89

Product of ratings

Sum of product of ratings

در ادامه باید مربع رتبه‌های این دو فیلم را با هم جمع کنیم سپس، مجذور این دو حاصل جمع را در هم ضرب می‌کنیم. این مقدار پایانی را B می‌نامیم.

User	1: Toy Story	2: Star Wars	Product of Movie Ratings	Square of Movie 1	Square of Movie 2
755	2	5	10	4	25
5277	1		0	1	0
1577			0	0	0
4388	2	3	6	4	9
1202		3	0	0	9
3823	3	4	12	9	16
5448			0	0	0
5347	2		0	4	0
4117	4	1	4	16	1
2765	4	2	8	16	4
5450	5	1	5	25	1
139	2	5	10	4	25
1940	4		0	16	0
3118	3		0	9	0
4656	2	4	8	4	16
4796			0	0	0
6037	2		0	4	0
3048	4	5	20	16	25
4790	2	1	2	4	1
4489	2	2	4	4	4
			A = 89	B = 137.9855065	

1. Square ratings

2. Sum of values from 1

3. Square root the result of 2

Product of values from 3

در مرحله‌ی سوم، مقدار A را بر B بخش می‌کنیم، این مقدار مشخص می‌کند فیلم‌ها چقدر به هم نزدیک هستند.

User	1: Toy Story	2: Star Wars	Product of Mov	Square of Movie 1	Square of Movie 2		
755	2	5	10	4	25		
5277	1		0	1	0		
1577			0	0	0	Sum of Movie 1 Ratings	Sum of Movie 2 Ratings
4388	2	3	6	4	9	140	136
1202		3	0	0	9		
3823	3	4	12	9	16	Square Root of Sum of Movie 1 Ratings	Square Root of Sum of Movie 2 Ratings
5448			0	0	0	11.83215957	11.66190379
5347	2		0	4	0		
4117	4	1	4	16	1		
2765	4	2	8	16	4		
5450	5	1	5	25	1		
139	2	5	10	4	25		
1940	4		0	16	0		
3118	3		0	9	0		
4656	2	4	8	4	16		
4796			0	0	0		
6037	2		0	4	0		
3048	4	5	20	16	25		
4790	2	1	2	4	1		
4489	2	2	4	4	4		
			A = 89	B = 137.9855065		Similarity (Movie 1 and 2) = 0.6449952772	

similarity between movie 1 and 2

با تکرار روند بالا برای تمام فیلم‌ها، یک جدول (ماتریس) با شباهت‌های میان هر دو فیلم انتخابی خواهیم داشت. روند بالا را می‌توان با علائم ریاضی به صورت زیر نمایش داد:

$$similarity(i, j) = \frac{\sum_u r_{(u,i)} r_{(u,j)}}{\sqrt{\sum_u r_{(u,i)}^2} \sqrt{\sum_u r_{(u,j)}^2}}$$

اضافه کردن لیبل به بعضی از حروف، درک هر قسمت از معادله را آسان می‌کند.

1. Main item for which to find other similar items

2. Item that is being compared with main item 'i' to find if they are similar

3. User 'u' rating for item 'i'

4. User 'u' rating for item 'j'

5. Repeat multiplication for all users 'u'

6. Sum the results of multiplication

$$similarity(i, j) = \frac{\sum_u r_{(u,i)} r_{(u,j)}}{\sqrt{\sum_u r_{(u,i)}^2} \sqrt{\sum_u r_{(u,j)}^2}}$$

این روش را cosine similarity هم می‌نامند. cosine similarity کمک می‌کند شباهت‌های دو بردار به هم را پیدا کنیم. با اعمال این فرمول روی رتبه‌های داده شده به آیتم‌ها، یک جدول یا ماتریس تشکیل می‌شود که برای هر آیتم شباهت‌اش با آیتم‌های دیگر را نشان می‌دهد.

	A	B	C	D	E	F	G	H
	1: Toy Story (19	1210: Star Wars	356: Forrest Gu	318: Shawshan	593: Silence of	3578: Gladiator	260: Star Wars:	
1: Toy Story (19	0.120246508	-0.057786711	0.051141073	0.0309007804	-0.073409437	0.022228394	-0.00727348	
1210: Star Wars	-0.057786711	0.194592873	-0.092167798	0.000652876	-0.003591667	-0.0667663164	-0.04255209	
356: Forrest Gu	0.051141073	-0.092167798	0.1718323	-0.052204922	0.003411564	0.074738591	0.017436326	
318: Shawshan	0.03090078	0.000652876	-0.052204922	0.1702385668	-0.026938857	-0.0001921535	-0.033503161	
593: Silence of	-0.073409437	-0.003591667	0.003411564	-0.0269388568	0.122963919	0.0142544368	-0.041887562	
3578: Gladiator	0.022228394	-0.066766316	0.074738591	-0.0001921535	0.014254437	0.21202139	-0.008459879	
260: Star Wars:	-0.00727348	-0.04255209	0.017436326	-0.0335031606	-0.041887562	-0.0084598785	0.173631242	
2028: Saving Pr	0.025108325	-0.02589935	0.011707254	-0.0428316094	-0.039492796	-0.0089383341	0.027342861	
296: Pulp Fictio	0.033765626	0.001971885	-0.004355224	0.0467948925	-0.04011093	-0.0568760956	0.026393982	
1259: Stand by	-0.037735175	0.067048735	0.038792456	-0.0123807533	0.019085428	0.0617988088	0.014242897	
2396: Shakespe	0.01824466	0.030893033	0.024998181	-0.0179844653	0.014070891	-0.07385593	0.024571262	
2916: Total Rec	0.045414215	-0.00114543	-0.054147028	-0.0145962562	-0.068621935	-0.0263166755	-0.066187138	
541: Blade Run	0.002185278	-0.022214854	-0.030413182	0.0205393578	0.025387197	-0.0029349887	-0.094088756	
780: Independe	-0.04089875	0.080328054	-0.02074187	0.0443460696	0.002810111	-0.0349930476	-0.003821527	
1265: Groundh	-0.014927075	-0.048826718	-0.043655518	-0.0170377626	0.001615963	-0.0241144099	-0.010784193	
2571: Matrix, T	-0.06405905	0.084506009	-0.062754925	-0.012209162	0.027796827	-0.0440632362	-0.035189064	
527: Schindler's	-0.021695816	-0.047058808	0.047985719	-0.0375066205	0.034045682	-0.0543452166	0.04101317	
2762: Sixth Sen	-0.039983077	0.003465819	-0.02377254	-0.0515570686	0.067250693	0.0289263204	-0.081295756	
1198: Raiders o	-0.037473535	0.007345507	-0.013769452	-0.0615929147	-0.00152482	0.020115641	0.033447224	
34: Babe (1995	0.079516503	-0.058905773	-0.041172289	0.0552086799	-0.066702166	-0.0343259373	0.016518825	

محاسبه‌ی امتیاز توصیه

یک جدول شباهت، پنجاه درصد کار ما را جلو برد. ما می‌دانیم چه آیتم‌هایی با هم قابل مقایسه‌اند، اما نمی‌دانیم چگونه آیتم‌ها را برای پیشنهاد به کاربر انتخاب کنیم. برای این ما باید ماتریس شباهت به‌دست آمده را با تاریخچه‌ی رتبه‌هایی که کاربر به آیتم‌ها داده ترکیب کنیم تا یک توصیه را انجام دهیم. این مقصود با اعمال معادله‌ی IBCF به‌راحتی قابل دستیابی است.

$$score(u, i) = \frac{\sum_j^I similarity(i, j)(r_{(u,j)} - \bar{r}_j)}{\sum_j^I similarity(i, j)} + \bar{r}_i$$

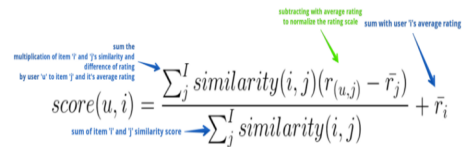
u = user for which we are generating recommendation

i = item in consideration, i.e. if this item should be recommended or not

score(u,i) = generates a score that will indicate how strong a recommendation of item 'i' is to our user 'u'.

j = items similar to main item i

طبق معادله‌ی بالا، برای محاسبه‌ی امتیاز توصیه‌ی آیتم i برای کاربر u؛ حاصل ضرب‌های شباهت i و فیلم انتخاب شده‌ی j، با اختلاف میان رتبه‌ی داده شده توسط کاربر به j و میانگین رتبه‌های داده شده به j جمع بسته سپس، این حاصل جمع را بر مجموع شباهت‌های میان آیتم‌های i و j بخش می‌کنیم. در نهایت خروجی را با میانگین رتبه‌های کاربر u جمع می‌بندیم.



The diagram shows the formula $score(u, i) = \frac{\sum_j^I similarity(i, j)(r_{(u,j)} - \bar{r}_j)}{\sum_j^I similarity(i, j)} + \bar{r}_i$ with several annotations:

- A blue arrow points to the denominator $\sum_j^I similarity(i, j)$ with the text "sum of item 'i' and 'j' similarity score".
- A green arrow points to the numerator's subtraction $(r_{(u,j)} - \bar{r}_j)$ with the text "subtracting with average rating to normalize the rating scale".
- A blue arrow points to the final addition $+ \bar{r}_i$ with the text "sum with user 'u's average rating".
- A blue arrow points to the similarity term $similarity(i, j)$ in the numerator with the text "sum the multiplication of item 'i' and 'j's similarity and difference of rating by user 'u' to item 'j' and 'j's average rating".

با اعمال این معادله، یک ماتریس از امتیازهای کاربر برای آیتم‌های در دسترس تولید خواهد شد. آیتم‌هایی که بیشترین امتیاز را دارند به کاربر پیشنهاد می‌شود.

User	1: Toy Story	2: Star Wars	356: Forrest Gu	318: Shawshan	593: Silence of	3578: Gladiator	2028: Saving Pr	260: Star Wars
755	-1	-1	-1	3.2	-1	-1	-1	-1
5277	-1	2.769231	2.769231	-1	-1	-1	-1	2.769231
1577	2.333333	2.333333	2.333333	-1	-1	2.333333	2.333333	2.333333
4388	-1	-1	2.833333	2.833333	2.833333	-1	2.833333	-1
1202	3.214286	-1	-1	-1	-1	-1	-1	-1
3823	-1	-1	-1	-1	3.071429	3.071429	-1	-1
5448	2.666667	2.666667	-1	-1	-1	-1	-1	-1
5347	-1	2.666667	2.666667	2.666667	-1	-1	-1	2.666667
4117	-1	-1	3.166667	-1	-1	-1	-1	-1
2765	-1	-1	3.181818	-1	-1	3.181818	-1	-1
5450	-1	-1	-1	2.818182	2.818182	-1	-1	2.818182
139	-1	-1	-1	2.8	2.8	2.8	-1	2.8
1940	-1	3.777778	3.777778	-1	-1	3.777778	-1	-1
3118	-1	2.636364	-1	2.636364	-1	2.636364	-1	2.636364
4656	-1	-1	3.153846	3.153846	-1	-1	-1	3.153846
4796	2.583333	2.583333	-1	2.583333	-1	-1	2.583333	-1
6037	-1	2.666667	2.666667	2.666667	2.666667	2.666667	2.666667	2.666667
3048	-1	-1	-1	-1	-1	-1	-1	3.076923
4790	-1	-1	-1	2.545455	2.545455	-1	-1	-1
4489	-1	-1	-1	-1	-1	3.117647	-1	-1

۳ پیاده‌سازی

بارگذاری پکیج‌های مورد نیاز

```

1 import numpy as np # linear algebra
2 import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
3 import os
4 from IPython.core.interactiveshell import InteractiveShell
5 InteractiveShell.ast_node_interactivity = "all"
6 import math
7 import json
8 import time
9 import matplotlib.pyplot as plt
10 import seaborn as sns
11 from sklearn.metrics.pairwise import cosine_similarity
12 from sklearn.model_selection import train_test_split
13 from sklearn.neighbors import NearestNeighbors
14 from sklearn.externals import joblib
15 import scipy.sparse
16 from scipy.sparse import csr_matrix
17 from scipy.sparse.linalg import svds
18 import warnings; warnings.simplefilter('ignore')
19 %matplotlib inline
20
21 for dirname, _, filenames in os.walk('/kaggle/input'):
22     for filename in filenames:
23         print(os.path.join(dirname, filename))
24
25 # Any results you write to the current directory are saved as output.

```

خواندن داده

```
1 electronics_data=pd.read_csv("/kaggle/input/amazon-product-reviews/  
  ratings_Electronics (1).csv",names=['userId', 'productId','Rating',  
  'timestamp'])  
2
```

```
1 electronics_data.head()  
2
```


Out[3]:

	userId	productId	Rating	timestamp
0	AKM1MP6POOYPR	0132793040	5.0	1365811200
1	A2CX7LUOHB2NDG	0321732944	5.0	1341100800
2	A2NWSAGRHCP8N5	0439886341	1.0	1367193600
3	A2WNBOD3WNDNKT	0439886341	3.0	1374451200
4	A1GI0U4ZRJA8WN	0439886341	1.0	1334707200

```
1 #Shape of the data
2 electronics_data.shape
3
```

Out[4]:

```
(7824482, 4)
```

```
1 #Taking subset of the dataset
2 electronics_data=electronics_data.iloc[:1048576,0:]
3
```

```
1 #Check the datatypes
2 electronics_data.dtypes
3
```

Out[6]:

```
userId      object
productId   object
Rating      float64
timestamp   int64
dtype: object
```

```
1 electronics_data.info()
2
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048576 entries, 0 to 1048575
Data columns (total 4 columns):
userId      1048576 non-null object
productId   1048576 non-null object
Rating      1048576 non-null float64
timestamp   1048576 non-null int64
dtypes: float64(1), int64(1), object(2)
memory usage: 32.0+ MB
```

```
1 #Five point summary
2 electronics_data.describe()['Rating'].T
3
```

```
Out[8]:
count      1.048576e+06
mean       3.973380e+00
std        1.399329e+00
min        1.000000e+00
25%        3.000000e+00
50%        5.000000e+00
75%        5.000000e+00
max        5.000000e+00
Name: Rating, dtype: float64
```

```
1 #Find the minimum and maximum ratings
2 print('Minimum rating is: %d' %(electronics_data.Rating.min()))
3 print('Maximum rating is: %d' %(electronics_data.Rating.max()))
4
```

```
Minimum rating is: 1
Maximum rating is: 5
```

رتبه‌دهی بین ۰ تا ۵ می‌باشد.

رسیدگی به مقادیر نامشخص

```
1 #Check for missing values
2 print('Number of missing values across columns: \n',electronics_data.
      isnull().sum())
3
```

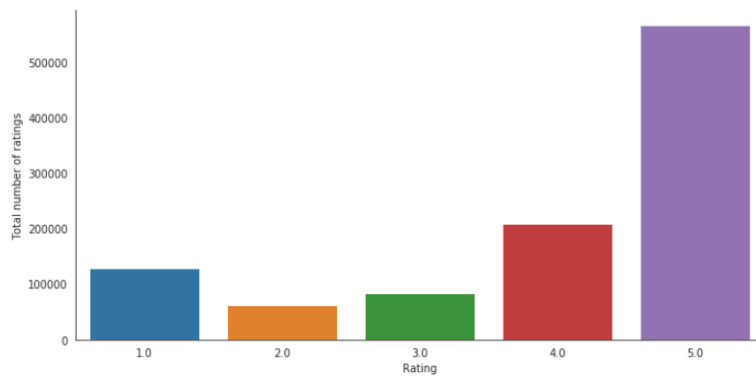
```
Number of missing values across columns:
```

```
userId      0  
productId   0  
Rating      0  
timestamp   0  
dtype: int64
```

توزيع رتبه‌ها

```
1 # Check the distribution of the rating  
2 with sns.axes_style('white'):  
3 g = sns.factorplot("Rating", data=electronics_data, aspect=2.0, kind='  
    count')  
4 g.set_ylabels("Total number of ratings")  
5
```

```
Out[11]: <seaborn.axisgrid.FacetGrid at 0x7fbc92751208>
```



بیشتر مشتری‌ها امتیاز ۵ را به محصولات داده‌اند.

مشتری‌ها محصولات یکتا

```
1 print("Total data ")
2 print("-"*50)
3 print("\nTotal no of ratings :",electronics_data.shape[0])
4 print("Total No of Users   :", len(np.unique(electronics_data.userId)))
5 print("Total No of products  :", len(np.unique(electronics_data.
6         productId)))
```

```
Total data
-----

Total no of ratings : 1048576
Total No of Users   : 786330
Total No of products : 61894
```

حذف ستون Timestamp:

```
1 #Dropping the Timestamp column
2 electronics_data.drop(['timestamp'], axis=1,inplace=True)
3
```

تحلیل رتبه‌ها

```
1 #Analysis of rating given by the user
2
```

```

3 no_of_rated_products_per_user = electronics_data.groupby(by='userId')['
    Rating'].count().sort_values(ascending=False)
4
5 no_of_rated_products_per_user.head()

```

```

Out[14]:
userId
A5JLAU2ARJ0B0    412
A231WM2Z2JL0U3    249
A25HBQ5V8S8SEA    164
A6FIAB28IS79      146
AT6CZDCP4TRGA     128
Name: Rating, dtype: int64

```

```

1 no_of_rated_products_per_user.describe()
2

```

```

Out[15]:
count    786330.000000
mean         1.333506
std         1.385612
min          1.000000
25%          1.000000
50%          1.000000
75%          1.000000
max         412.000000
Name: Rating, dtype: float64

```

```

1 quantiles = no_of_rated_products_per_user.quantile(np.arange
    (0,1.01,0.01), interpolation='higher')
2

```

```

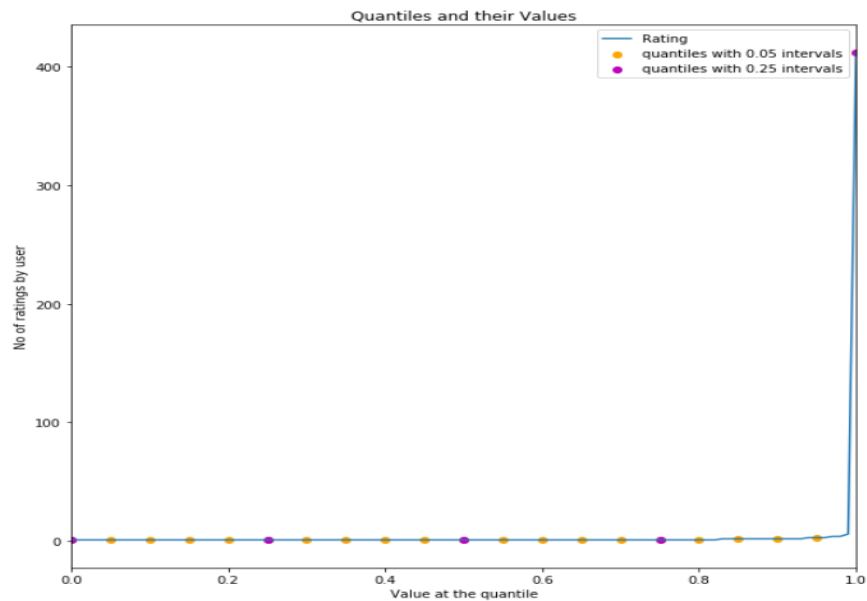
1 plt.figure(figsize=(10,10))
2 plt.title("Quantiles and their Values")
3 quantiles.plot()
4 # quantiles with 0.05 difference
5 plt.scatter(x=quantiles.index[::5], y=quantiles.values[::5], c='orange',
    , label="quantiles with 0.05 intervals")
6 # quantiles with 0.25 difference
7 plt.scatter(x=quantiles.index[::25], y=quantiles.values[::25], c='m',
    label = "quantiles with 0.25 intervals")

```

```

8 plt.ylabel('No of ratings by user')
9 plt.xlabel('Value at the quantile')
10 plt.legend(loc='best')
11 plt.show()

```



```

1 print('\n No of rated product more than 50 per user : {}'.format(sum(
    no_of_rated_products_per_user >= 50)) )

```

```

No of rated product more than 50 per user : 38

```

توصیه‌ی مبتنی بر محبوبیت

سیستم توصیه‌گر مبتنی بر محبوبیت با ترند کار می‌کند. این روش اساساً از آیتم‌هایی استفاده می‌کند که در ترند است. برای نمونه، آیتمی که معمولاً توسط کاربران تازه وارد خریداری می‌شود، به کاربرانی که تازه ثبت‌نام می‌کنند پیشنهاد می‌شود. مشکل سیستم مبتنی بر محبوبیت اینجاست که قابل شخصی‌سازی نیست؛ حتی اگر رفتار کاربر را بدانیم، کاری از دست‌مان برنمی‌آید.

```

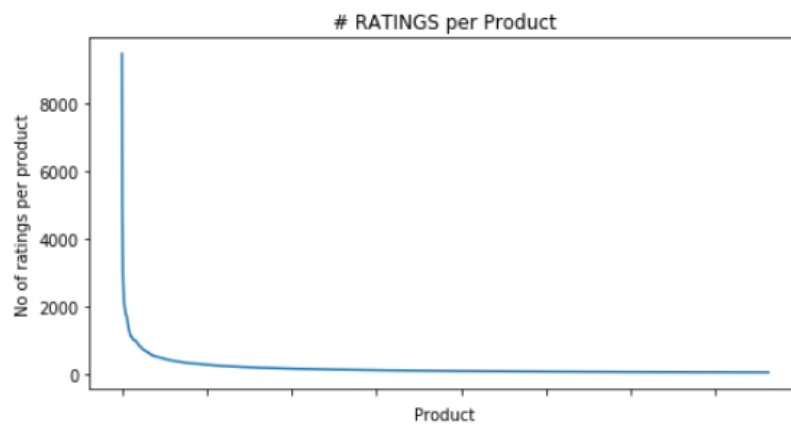
1 #Getting the new dataframe which contains users who has given 50 or
  more ratings
2 new_df=electronics_data.groupby("productId").filter(lambda x:x['Rating']
  ].count() >=50)

```

```

1 no_of_ratings_per_product = new_df.groupby(by='productId')['Rating'].
  count().sort_values(ascending=False)
2
3 fig = plt.figure(figsize=plt.figaspect(.5))
4 ax = plt.gca()
5 plt.plot(no_of_ratings_per_product.values)
6 plt.title('# RATINGS per Product')
7 plt.xlabel('Product')
8 plt.ylabel('No of ratings per product')
9 ax.set_xticklabels([])
10
11 plt.show()

```



```

1 #Average rating of the product
2 new_df.groupby('productId')['Rating'].mean().head()

```

```

Out[21]:
productId
0972683275    4.470980
1400501466    3.560000
1400501520    4.243902
1400501776    3.884892
1400532620    3.684211
Name: Rating, dtype: float64

```

```

1 new_df.groupby('productId')['Rating'].mean().sort_values(ascending=
  False).head()

```

```
Out[22]:
productId
B0000DYV9H    4.947368
B000053HC5    4.945783
B00009R96C    4.885714
B00005LE76    4.879310
B000I1X3W8    4.869565
Name: Rating, dtype: float64
```

```
1 #Total no of rating for product
2 new_df.groupby('productId')['Rating'].count().sort_values(ascending=
    False).head()
```

```
Out[23]:
productId
B0002L5R78    9487
B0001FTVEK    5345
B000I68BD4    4903
B000BQ7GW8    4275
B00007E7JU    3523
Name: Rating, dtype: int64
```

```
1 ratings_mean_count = pd.DataFrame(new_df.groupby('productId')['Rating']
    ].mean())
```

```
1 ratings_mean_count['rating_counts'] = pd.DataFrame(new_df.groupby('
    productId')['Rating'].count())
```

```
1 ratings_mean_count.head()
```

Out[26]:

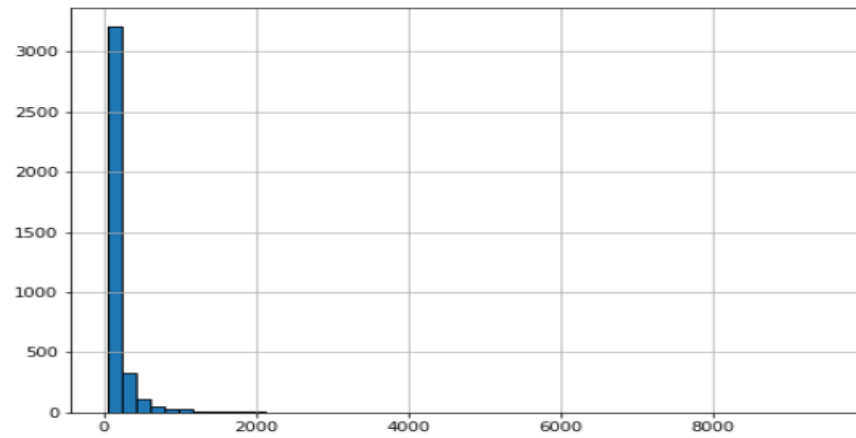
productId	Rating	rating_counts
0972683275	4.470980	1051
1400501466	3.560000	250
1400501520	4.243902	82
1400501776	3.884892	139
1400532620	3.684211	171

```
1 ratings_mean_count['rating_counts'].max()
```

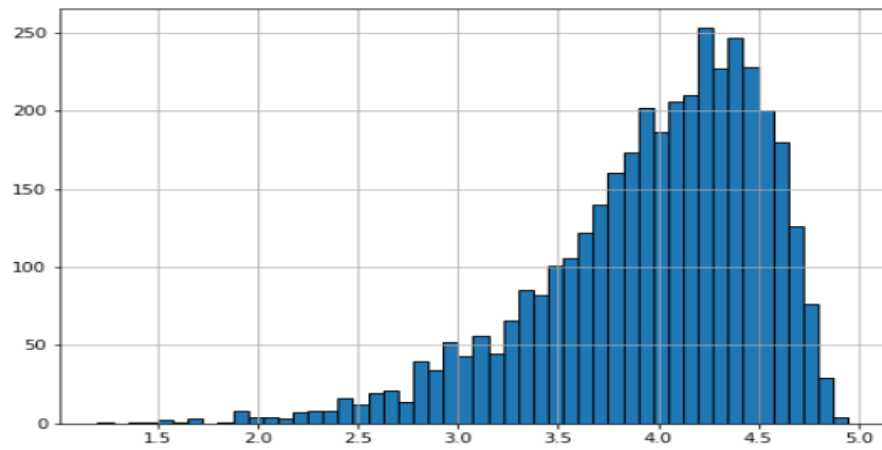
```
1 plt.figure(figsize=(8,6))
2 plt.rcParams['patch.force_edgecolor'] = True
3 ratings_mean_count['rating_counts'].hist(bins=50)
```


Out[27]:

9487



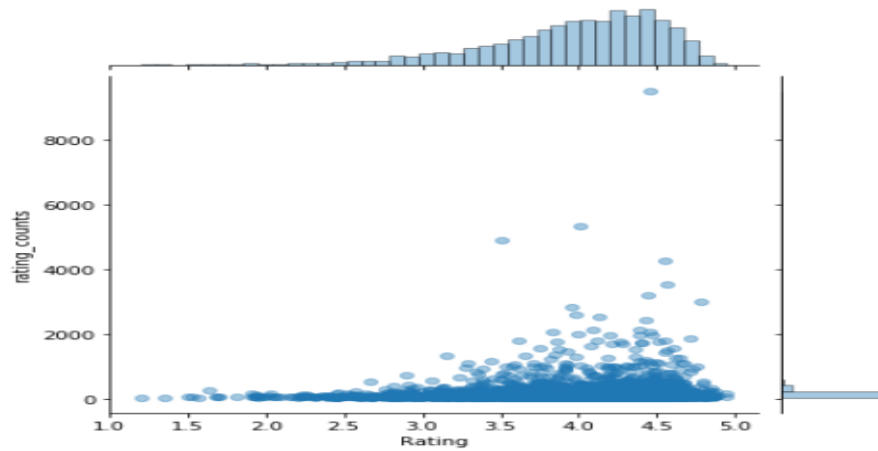
```
1 plt.figure(figsize=(8,6))
2 plt.rcParams['patch.force_edgecolor'] = True
3 ratings_mean_count['Rating'].hist(bins=50)
```



```

1 plt.figure(figsize=(8,6))
2 plt.rcParams['patch.force_edgecolor'] = True
3 sns.jointplot(x='Rating', y='rating_counts', data=ratings_mean_count,
  alpha=0.4)

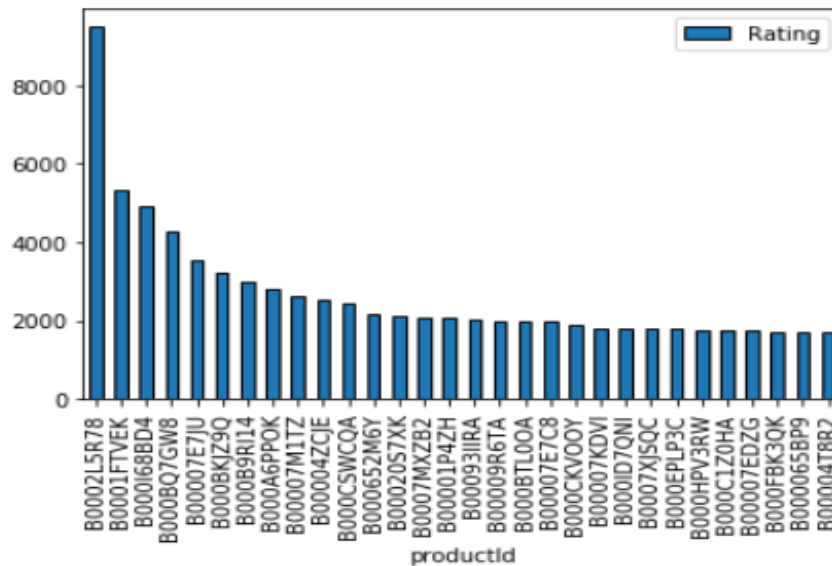
```



```

1 popular_products = pd.DataFrame(new_df.groupby('productId')['Rating'].
  count())
2 most_popular = popular_products.sort_values('Rating', ascending=False)
3 most_popular.head(30).plot(kind = "bar")

```



فیلترینگ اشتراکی (item-item recommendation)

معمولاً از فیلترینگ اشتراکی برای سیستم‌های توصیه‌گر استفاده می‌شود. مقصود از این روش‌ها، پرکردن مقادیر نامشخص برای ماتریس وابستگی item-user می‌باشد. collaborative filtering بر این پایه عمل می‌کند، که بهترین توصیه‌ها از کاربرانی نتیجه گرفته می‌شود که سلیقه‌های مشابه دارند. به عبارت دیگر، از رتبه‌ی کاربرانی که مانند کاربر مورد نظر فکر می‌کنند استفاده کرده تا رتبه‌ی کاربر را پیش‌بینی کند. فیلترینگ اشتراکی شامل دو دسته‌ی memory-based و model-based می‌شود.

```

1 from surprise import KNNWithMeans
2 from surprise import Dataset
3 from surprise import accuracy
4 from surprise import Reader
5 import os
6 from surprise.model_selection import train_test_split

1 #Reading the dataset
2 reader = Reader(rating_scale=(1, 5))
3 data = Dataset.load_from_df(new_df, reader)

1 #Splitting the dataset
2 trainset, testset = train_test_split(data, test_size=0.3, random_state
    =10)

1 # Use user_based true/false to switch between user-based or item-based
    collaborative filtering
2 algo = KNNWithMeans(k=5, sim_options={'name': 'pearson_baseline', '
    user_based': False})
3 algo.fit(trainset)

```

```

Estimating biases using als...
Computing the pearson_baseline similarity matrix...
Done computing similarity matrix.

<surprise.prediction_algorithms.knns.KNNWithMeans at 0x7fbc92233208>

```

```

1 # run the trained model against the testset
2 test_pred = algo.test(testset)

1 test_pred

1 # get RMSE
2 print("Item-based Model : Test Set")
3 accuracy.rmse(test_pred, verbose=True)

```

```

Item-based Model : Test Set
RMSE: 1.3436

Out[38]:
1.343641161111319

```

Model-based collaborative filtering system

این تکنیک‌ها بر پایه‌ی متدهای یادگیری ماشین و داده‌کاوی هستند. هدف ما، آموزش مدلی برای انجام پیش‌بینی است. برای نمونه، ما از رابطه‌ی item-user برای آموزش مدلی بهره می‌گیریم تا پنج آیتم برتر که امکان دارد مورد پسند کاربر قرار گیرد را پیش‌بینی کنیم. یکی از برتری‌های این متدها نسبت به دیگر روش‌ها-همچون روش مبتنی بر حافظه-قابلیت توصیه‌ی آیتم‌های بیشتر به تعداد بالای کاربران می‌باشد؛ این متدها پوشش بیشتری دارند، حتی برای ماتریس‌های بزرگ اسپارس.

```

1 new_df1=new_df.head(10000)
2 ratings_matrix = new_df1.pivot_table(values='Rating', index='userId',
    columns='productId', fill_value=0)
3 ratings_matrix.head()

```

همانطور که مشاهده می‌کنید، ماتریس به‌دست آمده در بالا اسپارس است. مقادیر نامشخص با ۰ پر شده‌اند.

```

1 ratings_matrix.shape

```

ترانهاده کردن ماتریس:

Out[39]:

productId	0972683275	1400501466	1400501520	1400501776	1400532620	1400532655	1400532680
userId							
A01852072Z7B68UHLI5UG	0	0	0	0	0	0	0
A0266076X6KPZ6CCHGVS	0	0	0	0	0	0	0
A0293130VTX2ZXA70JQS	5	0	0	0	0	0	0
A030530627MK66BD8V4LN	4	0	0	0	0	0	0
A0571176384K8RBNKGF8O	0	0	0	0	0	0	0

5 rows x 76 columns

Out[40]:

(9832, 76)

Out[41]:

userId	A01852072Z7B68UHLI5UG	A0266076X6KPZ6CCHGVS	A0293130VTX2ZXA70JQS	A030530627MK66BD8V4LN
productId				
0972683275	0	0	5	4
1400501466	0	0	0	0
1400501520	0	0	0	0
1400501776	0	0	0	0
1400532620	0	0	0	0

5 rows x 9832 columns

```
1 X = ratings_matrix.T
2 X.head()
```

```
1 X.shape
```

Out[42]:

(76, 9832)

محصول‌های یکتا در زیرمجموعه‌ی داده:

```
1 X1 = X
```

```

1 #Decomposing the Matrix
2 from sklearn.decomposition import TruncatedSVD
3 SVD = TruncatedSVD(n_components=10)
4 decomposed_matrix = SVD.fit_transform(X)
5 decomposed_matrix.shape

```

```

Out[44]:
(76, 10)

```

```

1 #Correlation Matrix
2 correlation_matrix = np.corrcoef(decomposed_matrix)
3 correlation_matrix.shape

```

```

Out[45]:
(76, 76)

```

```

1 X.index[75]

```

```

Out[46]:
'B00000K135'

```

```

1 # Find index of 'B00000K135'
2 i = "B00000K135"
3 product_names = list(X.index)
4 product_ID = product_names.index(i)
5 product_ID

```

```

Out[47]:
75

```

شباهت برای تمام آیتم‌های خریداری شده توسط این مشتری، بر پایه‌ی آیتم‌های رتبه‌دهی شده توسط مشتری‌هایی که محصول یکسان را خریداری کرده‌اند:

```

1 correlation_product_ID = correlation_matrix[product_ID]
2 correlation_product_ID.shape

```

توصیه کردن ۲۵ آیتم با شباهت بیشتر:

Out[48]:

```
(76,)
```

```
1 Recommend = list(X.index[correlation_product_ID > 0.65])
2
3 # Removes the item already bought by the customer
4 Recommend.remove(i)
5
6 Recommend[0:24]
```

Out[49]:

```
['3744295508',
 '9888002198',
 '9984984354',
 'B00000J1EJ',
 'B00000J1U8',
 'B00000J3NF',
 'B00000J4GE',
 'B00000J6WY',
 'B00000JBHP',
 'B00000JCT8',
 'B00000JCT0',
 'B00000JFE3',
 'B00000JHWX',
 'B00000JYWQ']
```

References

- [1] Item-based collaborative filtering recommendation algorithms
- [2] Recommender System Using Amazon Reviews
- [3] Comprehensive Guide on Item Based Collaborative Filtering