A Project on

Recommendation System

Submitted for partial fulfillment of award of

MASTER OF SCIENCE

Dr. Homi Bhabha State University, Mumbai

Degree in
MATHEMATICS

By

Runali Ulhas Warge
Shweta Sanjay Suryawanshi

Under the Guidance of

Dr. Selby Jose

Department of Mathematics
The Institute of Science
Mumbai - 400032

May, 2021

Declaration

I hereby declare that the project work entitled "**RECOMMENDATION SYSTEM**" carried out at the Department of Mathematics, The Institute of Science, Mumbai, is a record of an original work done by me under the guidance of **Dr. Selby Jose**, The Institute of Science, and this project work is submitted in the partial fulfillment of the requirements for the award of the degree of Master of Science in Mathematics, Dr. Homi Bhabha State University, Mumbai. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Shweta Sanjay Suryawanshi
(Candidate)
Seat No. 1933

Runali Ulhas Wargee
(Candidate)
Seat No. 1938

Certificate

This is to certify that the project entitled **RECOMMENDATION SYSTEM**, carried out at the Department of Mathematics, The Institute of Science, Mumbai, in partial fulfillment for the award of the degree of Master of Science in Mathematics, Dr. Homi Bhabha State University, Mumbai, is a record of bonafide work carried out by **Ms. Runali Ulhas Warge** , Seat No. **1938** and **Ms. Shweta Sanjay Suryawanshi**, Seat No. **1933**, under the supervision and guidance of **Dr. Selby Jose** during the academic year 2020 – 2021.

Dr. Selby Jose	**External Examiner**	**Head of Dept.**
Project Guide		**Mathematics**

Place: Mumbai

Date:

Acknowledgment

I would like to express my special thanks of gratitude to
Dr.Selby Jose, research guide The Institute Of Science, Mumbai for his stimulating guideance, continuous encuragement and supervision throughout the project work.

Shweta Sanjay Suryawanshi
Runali Ulhas Warge

Abstract

Recommendation Systems (RS) are commonly used in many Internet operations and their significance is growing as a result of the Internet's "Information Overload" crisis. The recommended framework will be reviewed in the first part. We will explain the three major types of suggested systems in the second section: collaborative, content-based and hybrid. We'll introduce a collaborative approach in the third segment. Based on the user's previous actions or explicit input, this approach uses item features to suggest other products that are close to what they want. We're aiming to go along with e-commerce as a domain. Buying and selling items is simple on e-commerce platforms. As a result, we've chosen an example as product data. We will recommend the best product for the client based on their requirements after analysing data. In the fourth section, we will demonstrate how to use Python to implement collaborative Filtering to provide personalized recommendations to users.

Contents

Chapter 1

Introduction

Recommendation systems are machine learning systems that help users discover new product and services. Historically, the first recommendation system was the Tapestry which coming out in 1992 from Xerox PARC, then a variety of techniques and technologies of recommendation system have been produced and introduced. Recommender systems are so commonplace now that many of us use them without even knowing it. Because we can't possibly look through all the products or content on a website, a recommendation system plays an important role in helping us have a better user experience. So, we are interested to knowing that how recommendation system exactly work.

In the later chapters we will explain the three major types of suggested systems. We are using collaborative method for that we will introduce a collaborative approach. Our further aim to go along with e-commerce as a domain. Next, we will demonstrate how to use python to implement collaborative filtering. The important component of any of these systems is, it takes information about the user and predicts the product to that user.

Chapter 2

Types of recommendation system

In this chapter we discuss different types of recommendation system.

2.1 Collaborative Method

Collaborative filtering recommender makes suggestions based on how users rated in the past and not based on the product themselves. It only knows how other customers rated the product. For example, suppose user A assigned the rating to product X and Y and new user B who has assigned the same rating to product X but hasn't purchased product Y yet. So, collaborative method will recommend him the product Y. Then, the main idea that rules collaborative methods is that these past user-item interactions are sufficient to detect similar users and similar items and make predictions based on these estimated proximities. The main advantage of collaborative method is that, there is No requirement for product descriptions.

2.2 Content Based Method

It is based on the information on the contents of the item rather than on the user opinions. The main idea is if the user likes an item, then he or she will like the "other" similar item.

Content based approaches use additional information about users and/or items. If we consider the example of a movies recommender system, this additional information can be, for example, the age, the sex, the job or any other personal information for users as well as the category, the main actors, the duration or other characteristics for the movies (items).

Then, the idea of content based methods is to try to build a model, based on the available "features", that explain the observed user-item interactions. For example, to model the fact that young women tend to rate better some movies, that young men tend to rate better some other movies and so on. If we manage to get such model, then making new predictions for a user is pretty easy: we just need to look at the profile (age, sex, . . .) of this user and, based on this information, to determine relevant movies to suggest.

2.3 Hybrid Method

Recent research has demonstrated that a hybrid approach combining collaborative filtering and content-based filtering. Hybrid approaches can be implemented in several ways: by making content-based and collaborative-based predictions separately and then combining them; by adding content-based capabilities to a collaborative-based approach (and vice versa).

Examples of systems mixed content based method and collaborative method : Tapestry Fab, it suggests relevant URLs to users by combining users “ratings and Web pages” similarities and Quickstep, it supports Web page recommendation.

Netflix is a good example of the use of hybrid recommender systems because, when a new user subscribes to their service they are required to rate content already seen or rate particular genres. Once the user begins using the service, collaborative filtering is used and similar content is suggested to the customer.

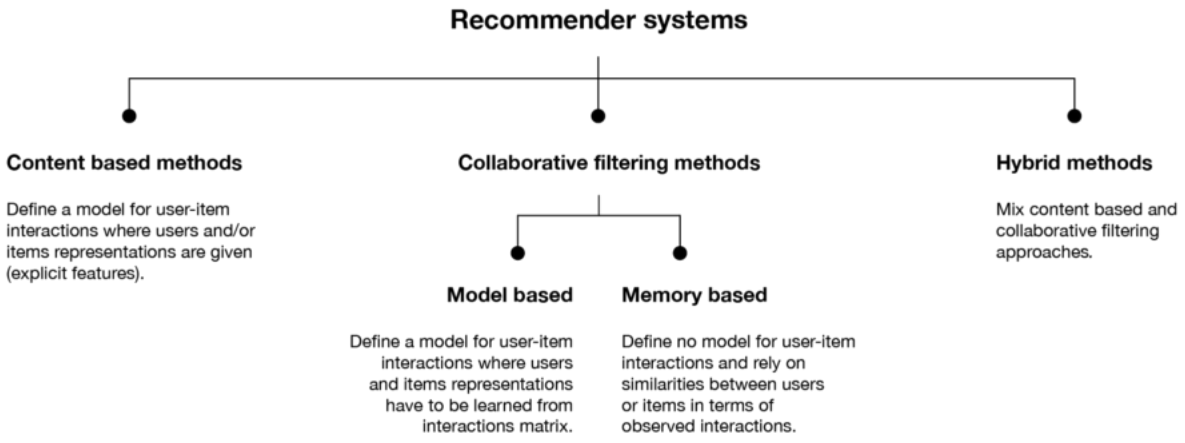

Recommender systems
Content based methods
Define a model for user-item interactions where users and/or items representations are given (explicit features).
Collaborative filtering methods
Model based
Define a model for user-item interactions where users and items representations have to be learned from interactions matrix.
Memory based
Define no model for user-item interactions and rely on similarities between users or items in terms of observed interactions.
Hybrid methods
Mix content based and collaborative filtering approaches.

Figure 2.1: Types of recommended system

Chapter 3

Domain

Recommended Systems have been widely used in many internet activities and it is worth mentioning some examples of the current actual uses of recommended system. However, we will mention e-commerce that use one or more methods of recommended system.

3.1 E-Commerce

The e-commerce becomes an important media to exchange products and services. Within e-commerce sites, buying and selling things between client and trader is easy, especially in our e-societies age. But the growth of e-commerce sites caused the product information overload; however, recommended system are used to solve this problem and are used "to suggest products to their customers and provide consumers with information to help them decide which products to purchase. There are more and more e-commerce businesses that use one or more variations of recommended system technologies in their Web sites". Examples : Amazon.com, Flipkart.com and CdNow.com.

Chapter 4

Approach To Collaborative Recommendation System

One important thing is that in an approach purely based on collaborative filtering, the similarity is not calculated using factors like the age of users, genre of the movie, or any other data about users or items. It is calculated only on the basis of the rating a user gives to an item.

Relationships provide recommender systems with tremendous insight, as well as an understanding of customers. There are three main types that occur:

- User-Item Relationship
- Item-Item Relationship
- User-User Relationship

Here, we are using Item-Item relationship.

4.1 Item-Item Relationship

Item-Item filtering will take an item, find users who liked that item, and find other items that those users or similar users also liked. It takes items and outputs other items as recommendations.

Item-based collaborative filtering was developed by Amazon. In a system where there are more users than items, item-based filtering is faster and more stable than user-based. It is effective because usually, the average rating received by an item doesn't change as quickly as the average rating given by a user to different items.

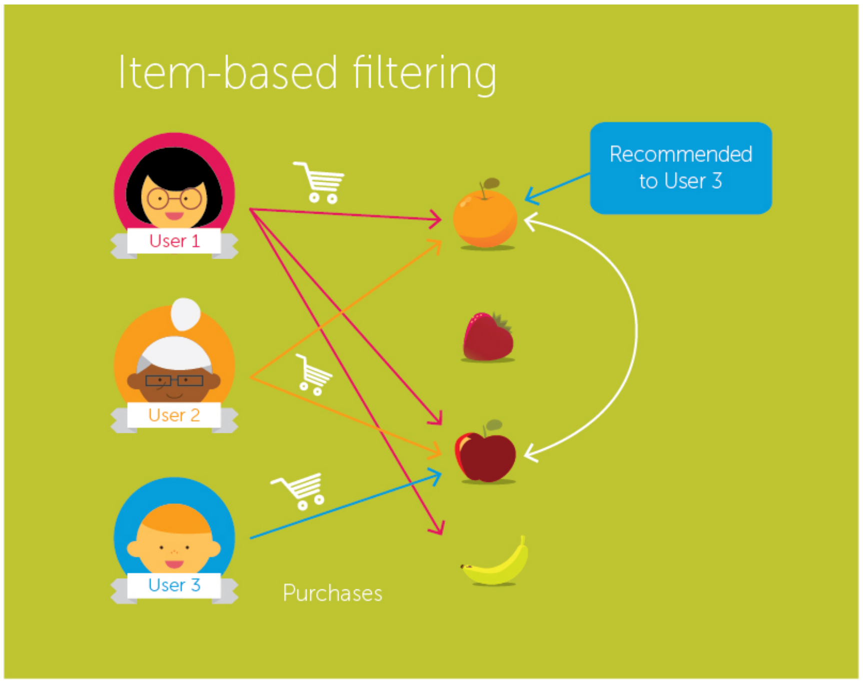


Figure 4.1: item-based-filterning

Chapter 5

Algorithm

To build a system that can automatically recommend items to users based on the rating history of the user, the first step is visualizing and preprocessing our data. The second step is to find similarity between products. So, you will require answers of following questions:

- How to determine the total number of ratings and average rating of each product?
- How to determine which products are similar to another?

The answer of first question is simple, for getting information about the data you have to analyse the data from where you'll get some basic statistical details for rating like standard deviation, minimum, maximum, etc.. So, using mean() method you'll get the avarage rating and using count() method you'll get the total number of rating of each product which are important attributes for further calculations.

There are multiple answers available for second question. Collaborative filtering is a family of algorithms where there are multiple ways to find similar products. You can use the correlation between the ratings of a product as the similarity matrix. To find the correlation between the ratings of the product, you need to create a matrix where each column is a product Id and each row contains the rating assigned by a specific user to that product.

You can take any product and correlate it with other products, then you'll get the products list which is highly correlated with that product. But correlation alone is not enough. Hence solution to this problem is to retrieve only those correlated products that have atleast more than 50 ratings. So, add the total number of rating, which will gives the list of products which are highly correlated and has high rating count.

Chapter 6

Dataset

The dataset that we are going to use for this project is the Electronic product Dataset. We download the dataset from kaggle.com namely "Rating Electronics Data" file, which contains purchase history of Electonic products and contains 999 ratings for 17 products purchase by 200 users.

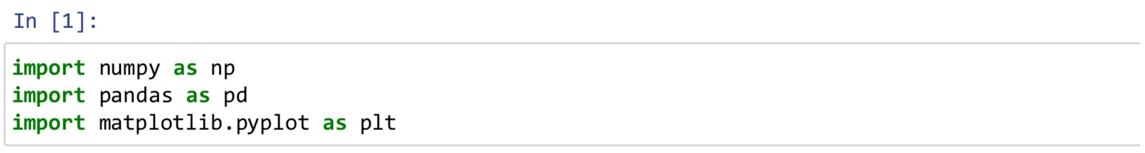
6.1 Attribute Information:

- User-Id : Every user identified with a unique id
- Product-Name : Every product identified with a name
- Brand-Name : Product with specific brand name
- Rating : Rating of the corresponding product by the corresponding user
- Review : Review of the product
- Price : Price of the product

Chapter 7

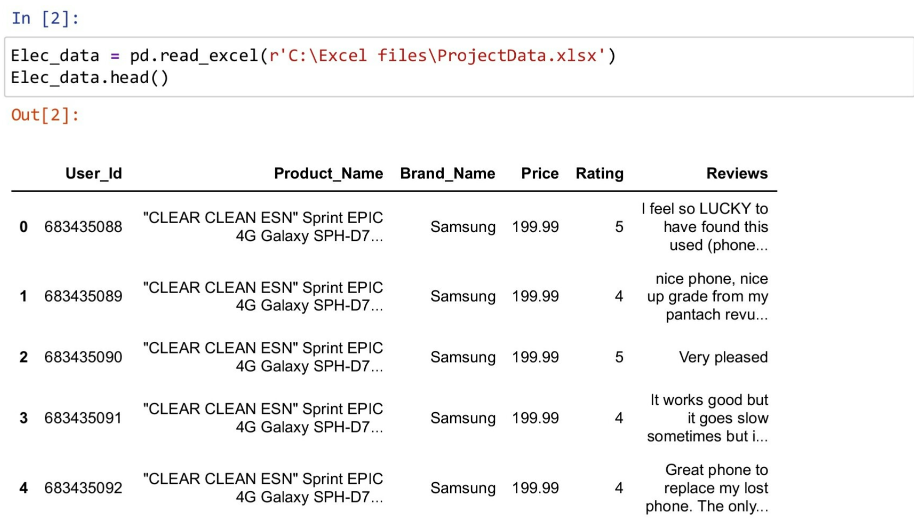
Pythone Code

let's first import the "ProjectData.xlsx" file and see what it contains.



In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```



In [2]:

```
Elec_data = pd.read_excel(r'C:\Excel files\ProjectData.xlsx')
Elec_data.head()
```

Out[2]:

	User_Id	Product_Name	Brand_Name	Price	Rating	Reviews
0	683435088	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	5	I feel so LUCKY to have found this used (phone...
1	683435089	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	nice phone, nice up grade from my pantach revu...
2	683435090	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	5	Very pleased
3	683435091	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	It works good but it goes slow sometimes but i...
4	683435092	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	Great phone to replace my lost phone. The only...

In the script above we use the read-excel() method of the Pandas library to read the “ProjectData.xlsx” file. Next, we call the head() method from the dataframe object returned by the read-excel() function, which will display the first five rows of the dataset.

You can see from the output that the “ProjectData.xlsx” file contains the UserId, ProductName, BrandName, Rating, Price and Reviews attributes. Each row in the dataset corresponds to one rating. The UserId column contains the ID of the user who left the rating. The ProductaName and BrandName column contains the name of the product and brand. The Price column contains the price of the product, the Rating column contains the rating left by the user. Ratings can have values between 1 and 5. And finally, the Reviews refers to the comments which the user left.

```
In [3]:
```

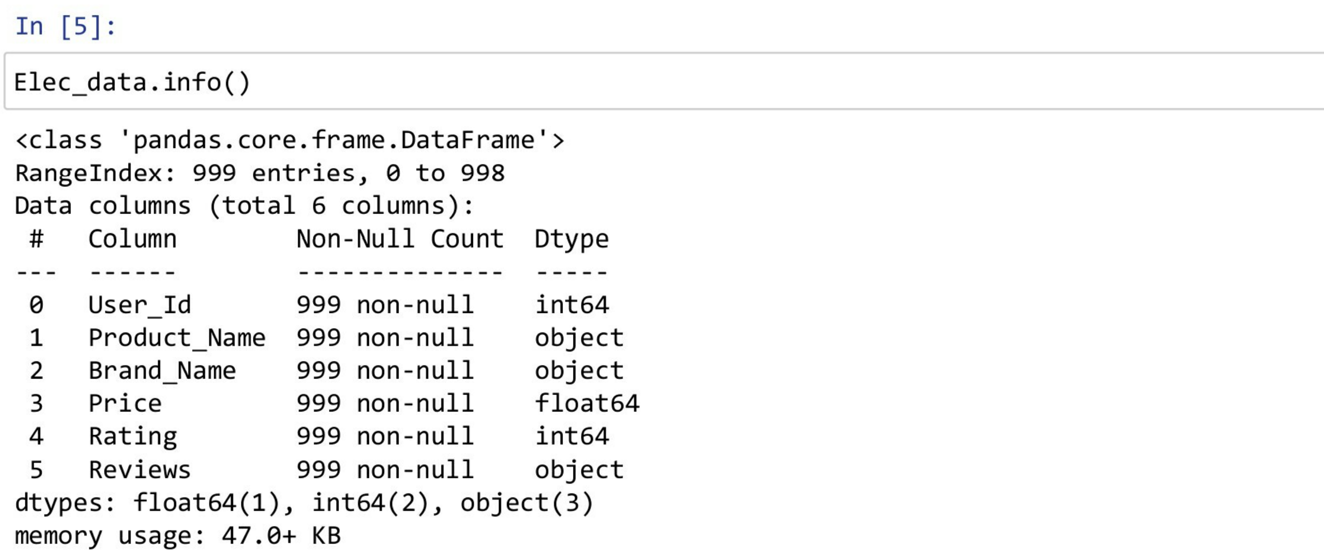
```
Elec_data.shape
```

```
Out[3]:

(999, 6)
```

```
In [4]:
```

```
#Check the datatype
Elec_data.dtypes
```

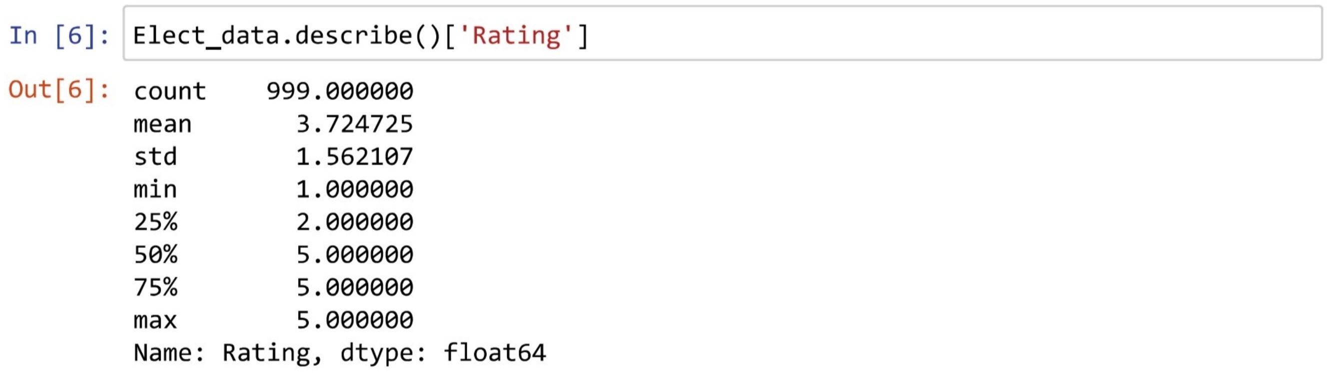
```
Out[4]:

User_Id           int64
Product_Name     object
Brand_Name       object
Price           float64
Rating            int64
Reviews          object
dtype: object
```



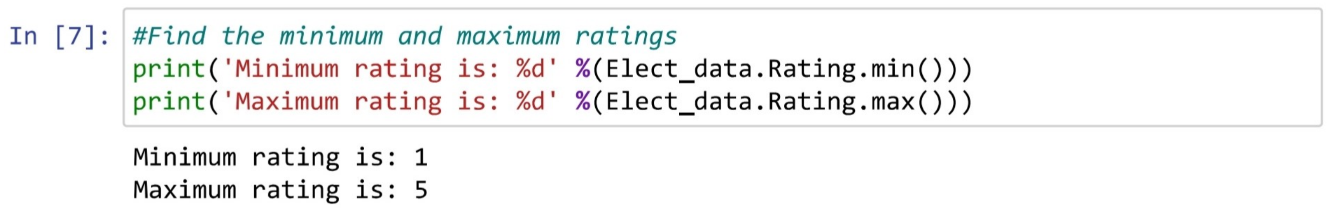
```
In [5]:
Elec_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   User_Id       999 non-null    int64  
 1   Product_Name  999 non-null    object 
 2   Brand_Name    999 non-null    object 
 3   Price         999 non-null    float64
 4   Rating        999 non-null    int64  
 5   Reviews       999 non-null    object 
dtypes: float64(1), int64(2), object(3)
memory usage: 47.0+ KB
```

We called info() method for checking any null entry in the dataset. but, here all entries are non-null so, dataset is fullfil.

Now,using describe() method to view some basic statistical details for rating like count, mean, standard deviation,etc.,

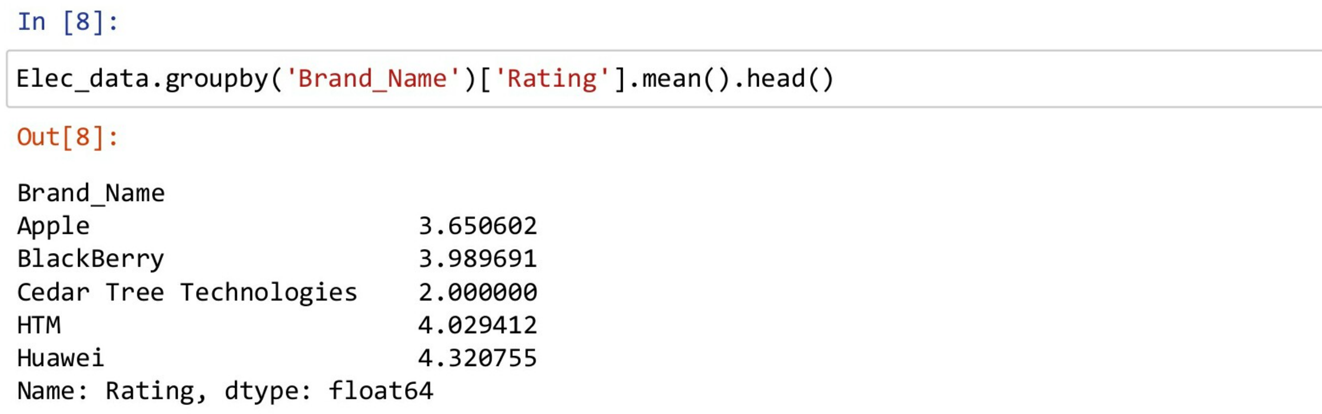


```
In [6]: Elect_data.describe()['Rating']

Out[6]: count    999.000000
        mean       3.724725
        std        1.562107
        min        1.000000
        25%        2.000000
        50%        5.000000
        75%        5.000000
        max        5.000000
        Name: Rating, dtype: float64
```



```
In [7]: #Find the minimum and maximum ratings
        print('Minimum rating is: %d' %(Elect_data.Rating.min()))
        print('Maximum rating is: %d' %(Elect_data.Rating.max()))

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

Now let's take a look at the average rating of each product. To do so, we can group the dataset by the brand of the product and then calculate the mean of the rating for each brand. We will then display the first five brand name along with their average rating using the head() method.



```
In [8]:

Elec_data.groupby('Brand_Name')['Rating'].mean().head()

Out[8]:

Brand_Name
Apple                       3.650602
BlackBerry                  3.989691
Cedar Tree Technologies     2.000000
HTM                         4.029412
Huawei                      4.320755
Name: Rating, dtype: float64
```

You can see that the average ratings are not sorted. Let's sort the ratings in the descending order of their average ratings:

```
In [9]:
```

```
rating_mean= Elec_data.groupby('Brand_Name')['Rating'].mean().sort_values(ascending=Fals
rating_mean.head()
```

```
Out[9]:
```

```
Brand_Name
Phone Baby    5.000000
Motorola      4.678571
Huawei        4.320755
OtterBox      4.233333
Nokia         4.205479
Name: Rating, dtype: float64
```

The product brand have now been sorted according to the ascending order of their ratings. However, there is a problem. A product brand can make it to the top of the above list even if only a single user has given it five stars. Therefore, the above stats can be misleading. Normally, a product brand which is really a good one gets a higher rating by a large number of users.

Let's now plot the total number of ratings for a product brand.

```
In [10]:
```

```
rating_count= Elec_data.groupby('Brand_Name')['Rating'].count().sort_values(ascending=Fa
rating_count.head()
```

```
Out[10]:
```

```
Brand_Name
Lenovo        163
Jethro        117
BlackBerry     97
LG             93
Ulefone        83
Name: Rating, dtype: int64
```

Unique users and products

```
In [11]:
```

```
print("Total data ")
print("\nTotal no of ratings :",Elec_data.shape[0])
print("Total No of Users   :", len(pd.unique(Elec_data.User_Id)))
print("Total No of products  :", len(pd.unique(Elec_data.Brand_Name)))
```

```
Total data

Total no of ratings : 999
Total No of Users   : 200
Total No of products  : 17
```

We know that both the average rating per brand and the number of ratings per brand are important attributes.

Create a new dataframe that contains both of these attributes.

```
In [12]:
```

```
ratings_mean_count = pd.DataFrame(Elec_data.groupby('Brand_Name')['Rating'].mean())
ratings_mean_count['rating_counts'] = pd.DataFrame(Elec_data.groupby('Brand_Name')['Rati
ratings_mean_count.head()
```

```
Out[12]:
```

	Rating	rating_counts
Brand_Name		
Apple	3.650602	83
BlackBerry	3.989691	97
Cedar Tree Technologies	2.000000	4
HTM	4.029412	34
Huawei	4.320755	53

```
In [13]:
```

```
rating_counts = ratings_mean_count.iloc[:,1].values
rating_counts
```

```
Out[13]:
```

```
array([ 83,  97,   4,  34,  53,   4, 117,  93, 163,  28,  73,  30,   3,
        37,  80,  83,  17], dtype=int64)
```

```
In [14]:
```

```
Rating = ratings_mean_count.iloc[:,0].values
Rating
```

```
Out[14]:
```

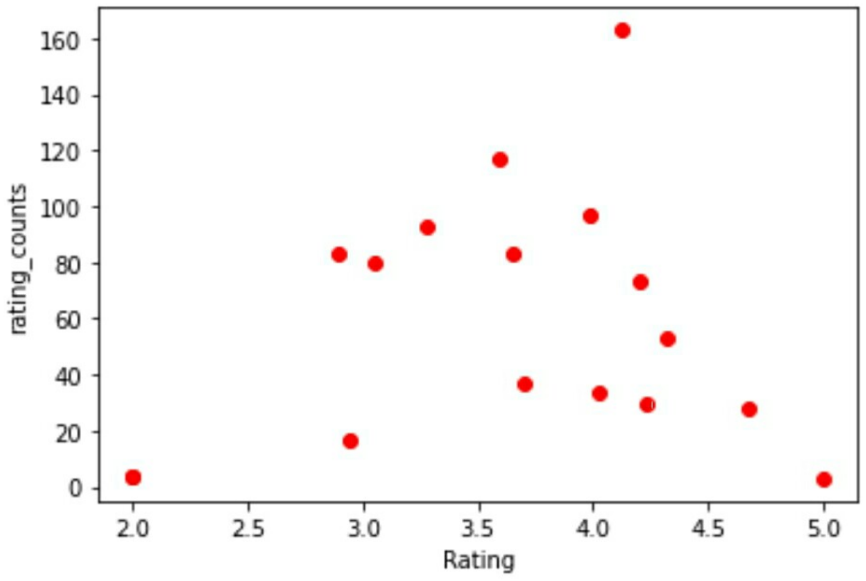
```
array([3.65060241, 3.98969072, 2.        , 4.02941176, 4.32075472,
       2.        , 3.58974359, 3.27956989, 4.12883436, 4.67857143,
       4.20547945, 4.23333333, 5.        , 3.7027027 , 3.05      ,
       2.89156627, 2.94117647])
```

We konw that brand with a higher number of ratings usually have a high average rating as well since a good brand is normally well-known and a well-known brand is purchase by a large number of people, and thus usually has a higher rating.

We will plot average ratings against the number of ratings.

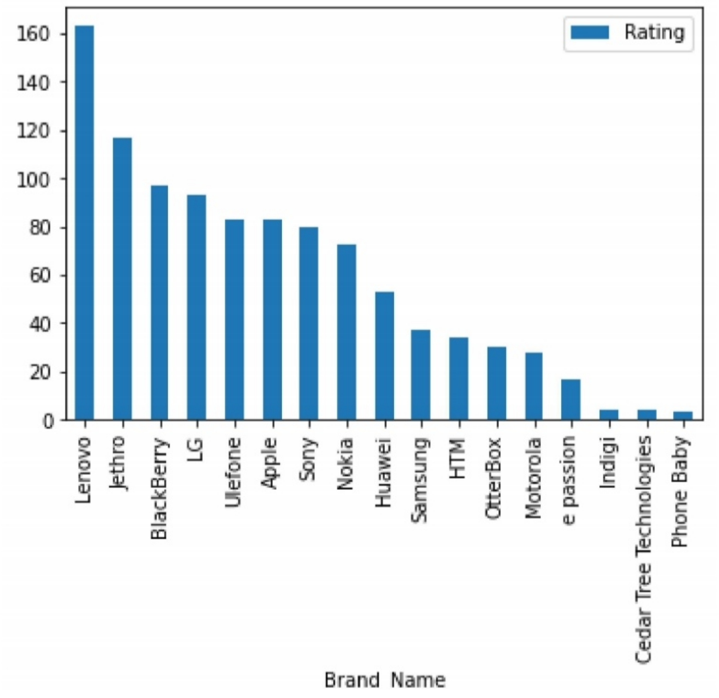
In [15]:

```
plt.xlabel('Rating')
plt.ylabel('rating_counts')
plt.scatter(Rating,rating_counts,color = 'red')
plt.show()
```



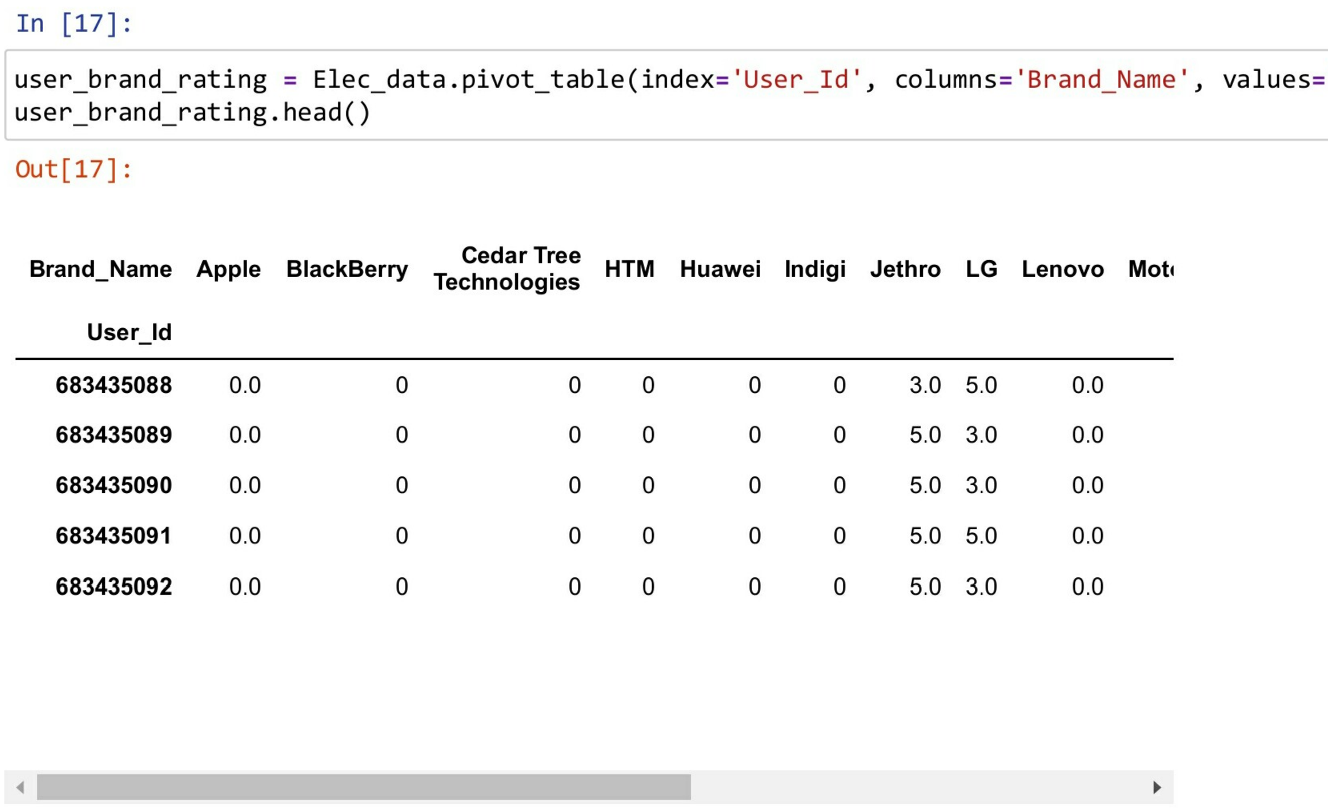
Let's plot a histogram for the number of ratings in decending order represented by the rating counts.

```
In [16]: popular_products = pd.DataFrame(Elect_data.groupby('Brand_Name')['Rating'].count
         most_popular = popular_products.sort_values('Rating', ascending=False)
         most_popular.head(30).plot(kind = "bar")

Out[16]: <AxesSubplot:xlabel='Brand_Name'>
```



We spent quite a bit of time on visualizing and preprocessing our data. Now is the time to find the similarity between products.

We will use the correlation between the ratings of a product as the similarity metric. To find the correlation between the ratings of the product, we need to create a matrix where each column is brand name and each row contains the rating assigned by a specific user to that product. Bear in mind that this matrix will have a lot of null values since every product is not rated by every user.



In [17]:

```
user_brand_rating = Elec_data.pivot_table(index='User_Id', columns='Brand_Name', values=
user_brand_rating.head()
```

Out[17]:

Brand_Name	Apple	BlackBerry	Cedar Tree Technologies	HTM	Huawei	Indigi	Jethro	LG	Lenovo	Mot
User_Id										
683435088	0.0	0	0	0	0	0	3.0	5.0	0.0	
683435089	0.0	0	0	0	0	0	5.0	3.0	0.0	
683435090	0.0	0	0	0	0	0	5.0	3.0	0.0	
683435091	0.0	0	0	0	0	0	5.0	5.0	0.0	
683435092	0.0	0	0	0	0	0	5.0	3.0	0.0	

Assuming the customer search for product “Jethro”

```
In [18]:
```

```
Jethro_ratings = user_brand_rating['Jethro']
Jethro_ratings.head()
```

```
Out[18]:
```

```
User_Id
683435088    3.0
683435089    5.0
683435090    5.0
683435091    5.0
683435092    5.0
Name: Jethro, dtype: float64
```

Now let’s retrieve all the brand name that are similar to “Jethro”. We can find the correlation between the user ratings for the “Jethro” and all the other brands using corrwith() function.

```
In [19]:
```

```
brand_like_jethro = user_brand_rating.corrwith(Jethro_ratings,axis=0,drop=False,method='
brand_like_jethro.head()
```

```
Out[19]:
```

```
Brand_Name
Apple                       -0.429777
BlackBerry                  -0.014784
Cedar Tree Technologies      0.084185
HTM                          0.361630
Huawei                       0.399466
dtype: float64
```

In the above script, we first retrieved the list of all the brand name related to “Jethro” along with their correlation value, using corrwith() function.

Next, we created a dataframe that contains brand name and correlation columns.

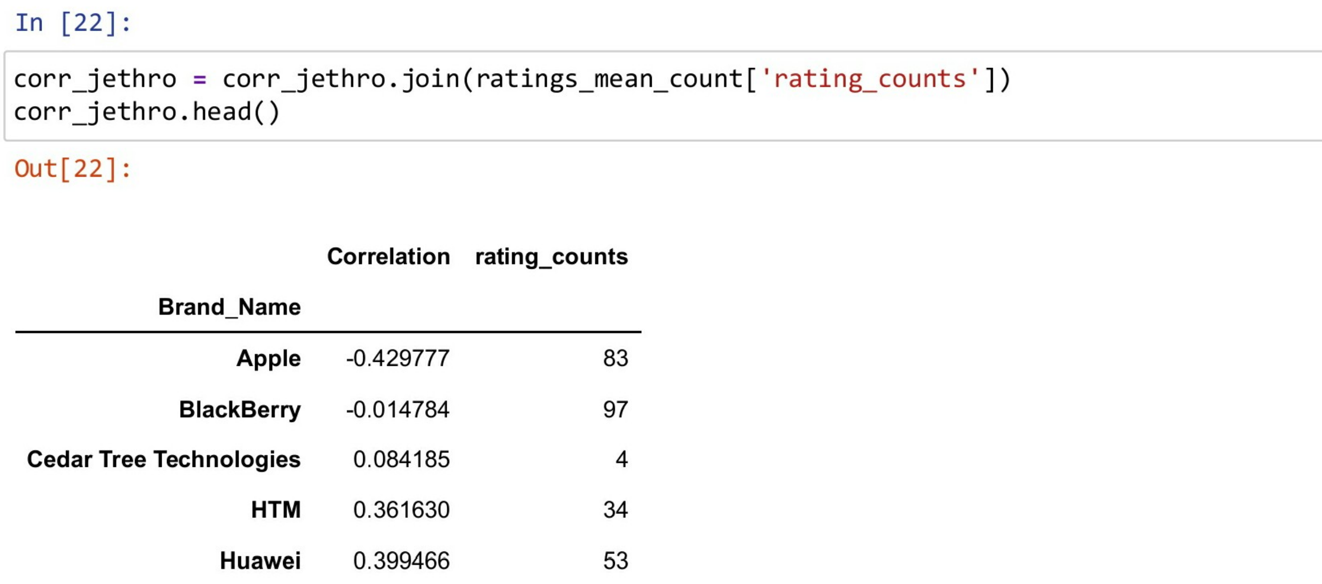
```
In [20]: corr_Jethro = pd.DataFrame(Brand_like_Jethro , columns=['Correlation'])
         corr_Jethro.dropna(inplace=True)
         corr_Jethro.head()
Out[20]:
                                  Correlation
                   Brand_Name
                        Apple       -0.429777
                   BlackBerry       -0.014784
      Cedar Tree Technologies        0.084185
                          HTM        0.361630
                       Huawei        0.399466
```

Let's sort the product brand in descending order of correlation to see highly correlated product at the top.

```
In [21]:
corr_jethro.sort_values('Correlation', ascending=False).head()
Out[21]:

             Correlation
 Brand_Name
     Jethro     1.000000
      Nokia     0.611878
     Huawei     0.399466
   OtterBox     0.369417
    Samsung     0.364072
```

From the output you can see that the brand that have high correlation with “Jethro” are not very well known. This shows that correlation alone is not a good metric for similarity because there can be a user who purchase “Jethro” and only one other product and rated both of them as 5.

A solution to this problem is to retrieve only those correlated product that have at least more than 50 ratings. To do so, will add the rating-counts column from the rating-mean-count dataframe to our corr-jethro dataframe.



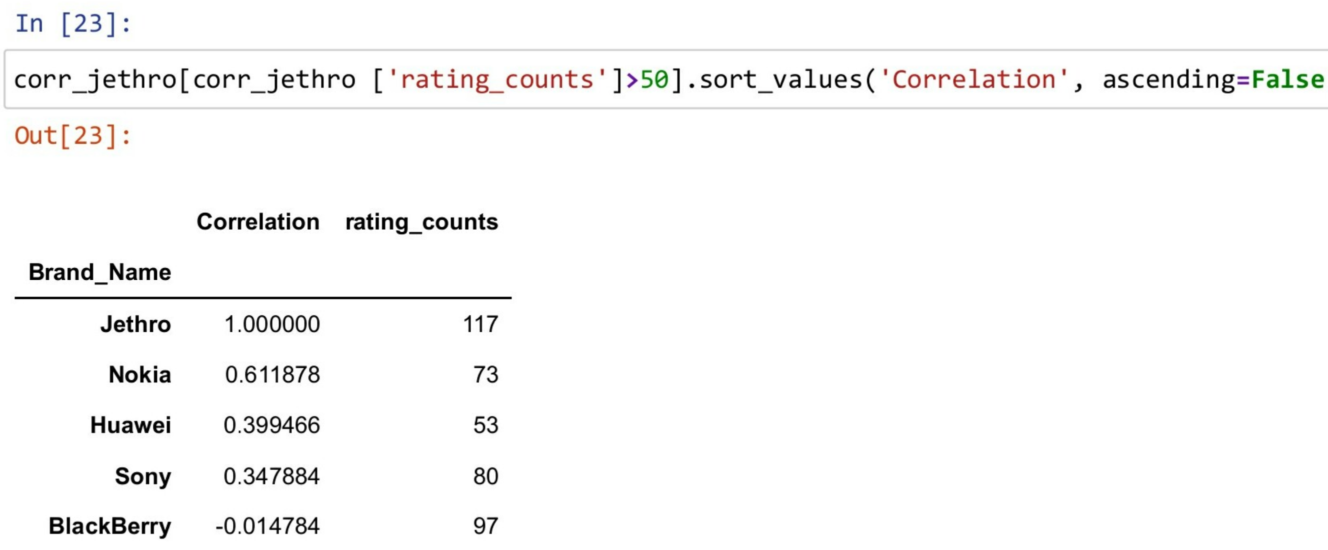
In [22]:

```
corr_jethro = corr_jethro.join(ratings_mean_count['rating_counts'])
corr_jethro.head()
```

Out[22]:

	Correlation	rating_counts
Brand_Name		
Apple	-0.429777	83
BlackBerry	-0.014784	97
Cedar Tree Technologies	0.084185	4
HTM	0.361630	34
Huawei	0.399466	53

You can see that the product “Cedar Tree Technologies”, which has the highest correlation has only four ratings. This means that only four users gave same ratings to “Jethro”, “Cedar Tree Technologies”. However, we can deduce that a product cannot be declared similar to the another product based on just 4 ratings. This is why we added “rating-counts” column. Let’s now filter product brand correlated to “Jethro”, that have more than 50 ratings.

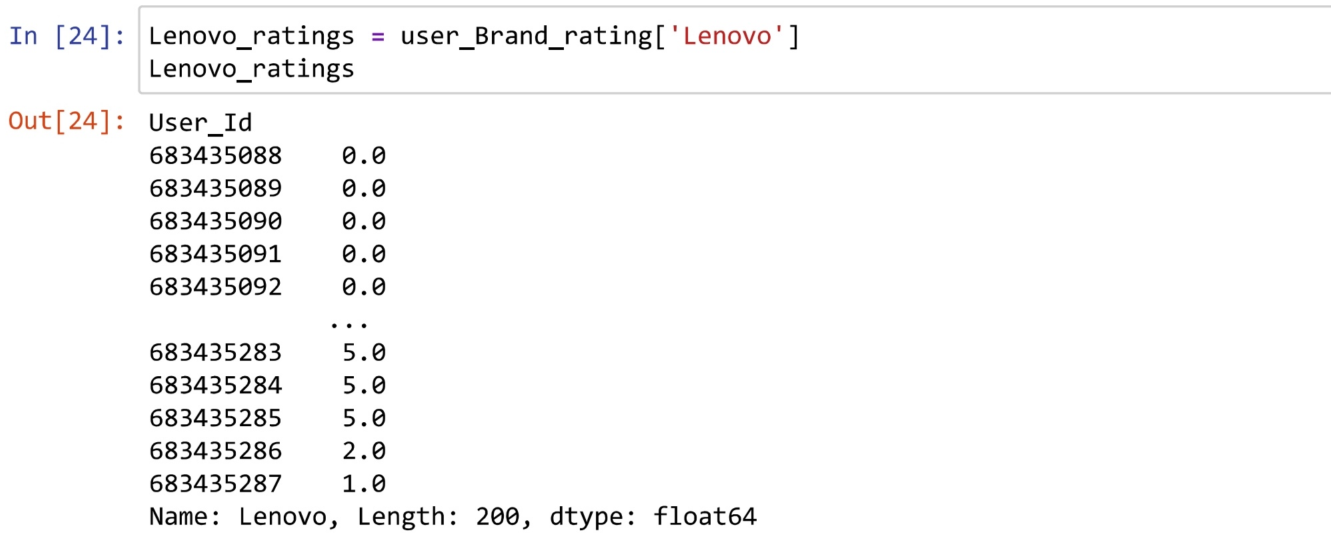


```
In [23]:
corr_jethro[corr_jethro ['rating_counts']>50].sort_values('Correlation', ascending=False
Out[23]:
```

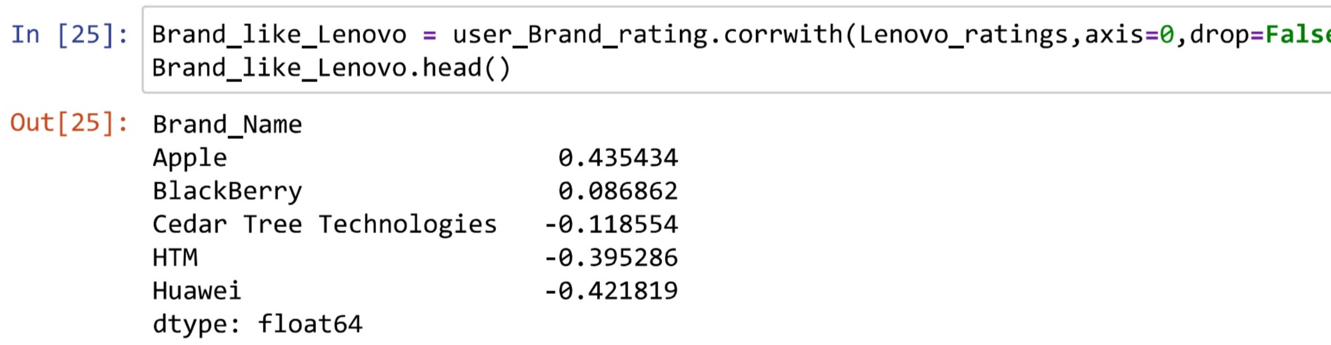
```
             Correlation  rating_counts
Brand_Name
    Jethro      1.000000            117
     Nokia      0.611878             73
    Huawei      0.399466             53
      Sony      0.347884             80
BlackBerry     -0.014784             97
```

Here are the top products to be displayed by the recommendation system which are “Nokia”, “Huawei”, “Sony” to the above customer based on high correlation and rating counts

Similarly,if you search for product “Lenovo”.



```
In [24]: Lenovo_ratings = user_Brand_rating['Lenovo']
         Lenovo_ratings

Out[24]: User_Id
         683435088    0.0
         683435089    0.0
         683435090    0.0
         683435091    0.0
         683435092    0.0
                     ...
         683435283    5.0
         683435284    5.0
         683435285    5.0
         683435286    2.0
         683435287    1.0
         Name: Lenovo, Length: 200, dtype: float64
```



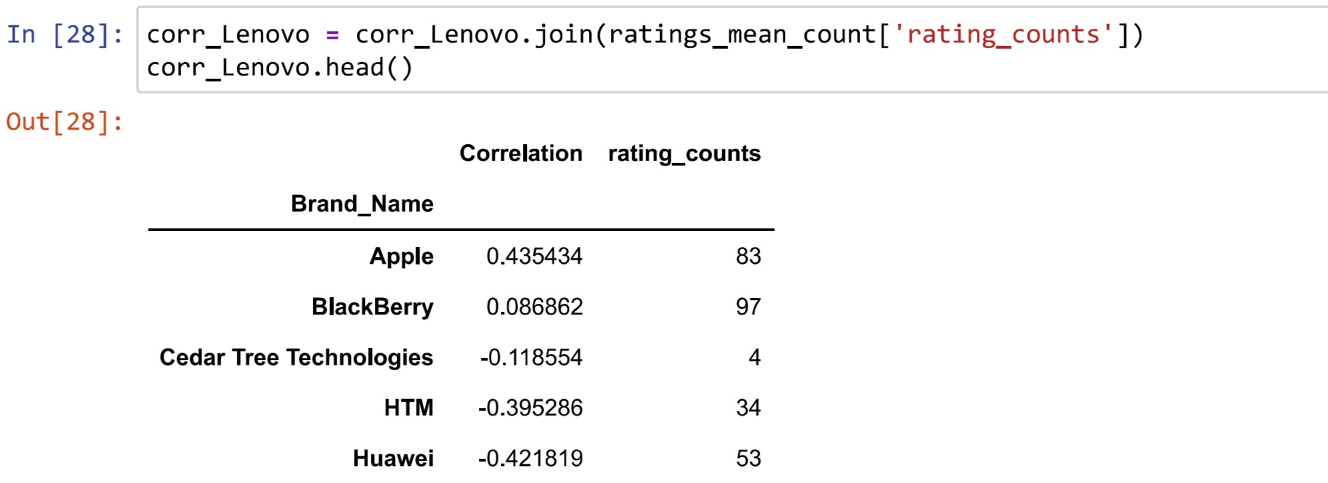
```
In [25]: Brand_like_Lenovo = user_Brand_rating.corrwith(Lenovo_ratings,axis=0,drop=False
         Brand_like_Lenovo.head()

Out[25]: Brand_Name
         Apple                      0.435434
         BlackBerry                 0.086862
         Cedar Tree Technologies   -0.118554
         HTM                       -0.395286
         Huawei                    -0.421819
         dtype: float64
```

```
In [26]: corr_Lenovo = pd.DataFrame(Brand_like_Lenovo , columns=['Correlation'])
         corr_Lenovo.dropna(inplace=True)
         corr_Lenovo.head()
```

```
Out[26]:
                                    Correlation
                     Brand_Name
                          Apple        0.435434
                     BlackBerry        0.086862
        Cedar Tree Technologies       -0.118554
                            HTM       -0.395286
                         Huawei       -0.421819
```

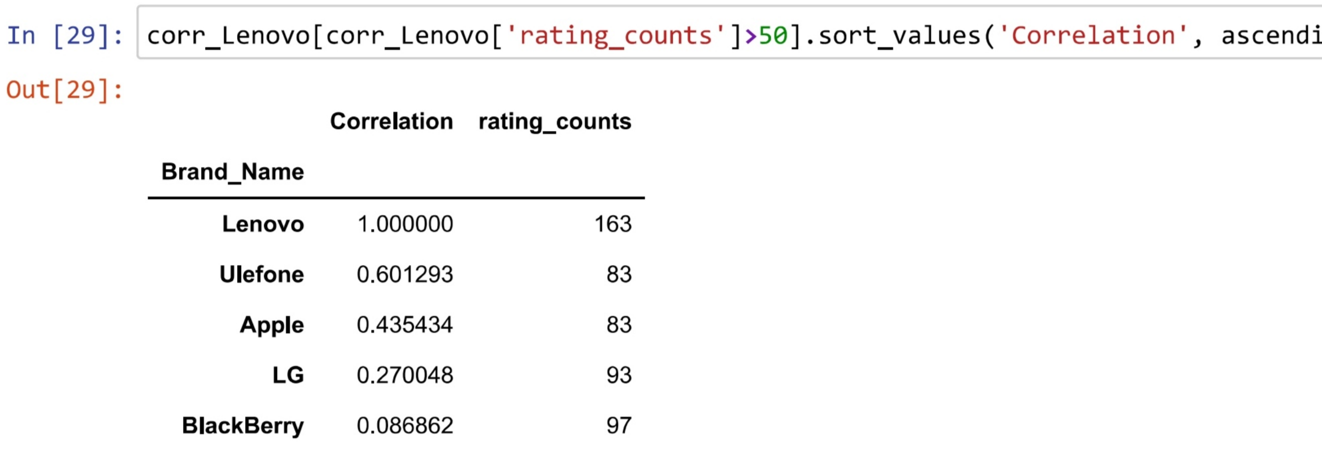
```
In [27]: corr_Lenovo.sort_values('Correlation', ascending=False).head()
```

```
Out[27]:
                     Correlation
        Brand_Name
            Lenovo      1.000000
           Ulefone      0.601293
             Apple      0.435434
          Motorola      0.387328
                LG      0.270048
```



```
In [28]: corr_Lenovo = corr_Lenovo.join(ratings_mean_count['rating_counts'])
         corr_Lenovo.head()
Out[28]:
```

Brand_Name	Correlation	rating_counts
Apple	0.435434	83
BlackBerry	0.086862	97
Cedar Tree Technologies	-0.118554	4
HTM	-0.395286	34
Huawei	-0.421819	53



```
In [29]: corr_Lenovo[corr_Lenovo['rating_counts']>50].sort_values('Correlation', ascendi
Out[29]:
```

Brand_Name	Correlation	rating_counts
Lenovo	1.000000	163
Ulefone	0.601293	83
Apple	0.435434	83
LG	0.270048	93
BlackBerry	0.086862	97

Here, you can see that recommendation system recommended “Ulefone”, “Apple”, “LG” products.

In these way, if you search any other product, then recommendation system will recommend you products, based on correlation and rating counts.

Chapter 8

Conclusion

Recommendation system help users find item they want to buy from various website. This system benefit users by enabling them to find items they like. This paper discussed the three traditional recommendation techniques and highlighted collaborative recommendation technique. Various learning algorithms used in generating recommendation models and correlation metrics used in measuring the performance of recommendation algorithms were discussed. Using collaborative method we are recommending the products to the customer based on the rating.

Bibliography

[1] https://www.kaggle.com/saurav9786/recommender-system-using-amazon-reviews

[2] https://realpython.com/build-recommendation-engine-collaborative-filtering/steps-involved-in-collaborative-filtering

[3] https://builtin.com/data-science/recommender-systems