Building Recommendation Engine Using Amazon Review Dataset [Grocery and Gourmet]

February 8, 2020

1 Importing the Libraries

[1]:
```
#import the reqired libraries
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
import glob
import numpy as np
import pandas as pd
import math
import json
import time
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.model_selection import train_test_split
from sklearn.neighbors import NearestNeighbors
from sklearn.externals import joblib
import scipy.sparse
from scipy.sparse import csr_matrix
from scipy.sparse.linalg import svds
import warnings; warnings.simplefilter('ignore')
%matplotlib inline
```

```
C:\Users\KIIT\Anaconda3\lib\site-
packages\sklearn\externals\joblib\__init__.py:15: DeprecationWarning:
sklearn.externals.joblib is deprecated in 0.21 and will be removed in 0.23.
Please import this functionality directly from joblib, which can be installed
with: pip install joblib. If this warning is raised when loading pickled models,
you may need to re-serialize those models with scikit-learn 0.21+.
  warnings.warn(msg, category=DeprecationWarning)
```

[33]:
```
file=glob.glob('Grocery_and_Gourmet_Food.json')
```

[34]:
```
# Reading a multiple json files from a single json file␣
↪'Grocery_and_Gourmet_Food.json'.
```

```
review=[]
with open(file[0]) as data_file:
    data=data_file.read()
    for i in data.split('\n'):
        review.append(i)

# Making a list of Tuples containg all the data of json files.
reviewDataframe=[]
for x in review:
    try:
        jdata=json.loads(x)
        reviewDataframe.
 ↪append((jdata['reviewerID'],jdata['asin'],jdata['reviewerName'],jdata['reviewText'],jdata['
    except:
        pass

# Creating a dataframe using the list of Tuples got in the previous step.
dataset=pd.
 ↪DataFrame(reviewDataframe,columns=['Reviewer_ID','Asin','Reviewer_Name','Review_Text','Rati
```

[35]:
```
groceryandgourmet_df=dataset
groceryandgourmet_df.head()
```

[35]:

	Reviewer_ID	Asin	Reviewer_Name	Review_Text	Rating	Summary	Unix_Review_Time	Review_Time
0	ALP49FBWT4I7V	1888861614	Lori	Very pleased with my purchase. Looks exactly l…	5.0	Love it	1370304000	06 4, 2013
1	A1KPIZOCLB9FZ8	1888861614	BK Shopper	Very nicely crafted but too small. Am going to…	4.0	Nice but small	1400803200	05 23, 2014
2	A2W0FA06IYAYQE	1888861614	daninethequeen	still very pretty and well made…i am super p…	4.0	the "s" looks like a 5, kina	1399593600	05 9, 2014
3	A2PTZTCH2QUYBC	1888861614	Tammara	I got this for our wedding cake, and it was ev…	5.0	Would recommend this to a friend!	1397952000	04 20, 2014
4	A2VNHGJ59N4Z90	1888861614	LaQuinta Alexander	It was just what I want to put at the top of m…	4.0	Topper	1397606400	04 16, 2014

[22]:
```
groceryandgourmet_df.drop('Reviewer_Name',axis=1,inplace=True)
groceryandgourmet_df.drop('Review_Text',axis=1,inplace=True)
groceryandgourmet_df.drop('Summary',axis=1,inplace=True)
```

```
groceryandgourmet_df.drop('Unix_Review_Time',axis=1,inplace=True)
groceryandgourmet_df.drop('Review_Time',axis=1,inplace=True)
```

[23]:
```
groceryandgourmet_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5069736 entries, 0 to 5069735
Data columns (total 3 columns):
Reviewer_ID    object
Asin           object
Rating         float64
dtypes: float64(1), object(2)
memory usage: 116.0+ MB
```

[24]:
```
#Check the number of rows and columns
rows,columns=dataset.shape
print('Number of rows: ',rows)
print('Number of columns: ',columns)
```

```
Number of rows:  5069736
Number of columns:  3
```

[25]:
```
#Check the datatypes
groceryandgourmet_df.dtypes
```

[25]:
```
Reviewer_ID     object
Asin            object
Rating         float64
dtype: object
```

[ ]:
```

```

[ ]:
```

```

[26]:
```
#Taking subset of the dataset
groceryandgourmet_df1=groceryandgourmet_df.iloc[:50000,0:]
```

[27]:
```
groceryandgourmet_df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 3 columns):
Reviewer_ID    50000 non-null object
Asin           50000 non-null object
Rating         50000 non-null float64
dtypes: float64(1), object(2)
memory usage: 1.1+ MB
```

[28]:
```
#Summary statistics of rating variable
groceryandgourmet_df1['Rating'].describe().transpose()
```

[28]:
```
count    50000.000000
mean         4.410560
std          1.164932
min          1.000000
25%          4.000000
50%          5.000000
75%          5.000000
max          5.000000
Name: Rating, dtype: float64
```

[29]:
```
#Find the minimum and maximum ratings
print('Minimum rating is: %d' %(groceryandgourmet_df1.Rating.min()))
print('Maximum rating is: %d' %(groceryandgourmet_df1.Rating.max()))
```

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

[30]:
```
#Check for missing values
print('Number of missing values across columns: \n',groceryandgourmet_df.
 ↪isnull().sum())
```

```
Number of missing values across columns:
 Reviewer_ID    0
Asin            0
Rating          0
dtype: int64
```

Converting the data type of 'Review_Time' column in the Dataframe 'dataset' to datetime format.

[36]:
```
groceryandgourmet_df['Review_Time']= pd.
 ↪to_datetime(groceryandgourmet_df['Review_Time'])
```

[37]:
```
#Creating an Addtional column as 'Month' in Datatframe 'dataset' for Month by␣
 ↪taking the month part of 'Review_Time' column.

groceryandgourmet_df['Month']=groceryandgourmet_df['Review_Time'].dt.month
```

[38]:
```
#Creating an Addtional column as 'Year' in Datatframe 'dataset' for Year by␣
 ↪taking the year part of 'Review_Time' column.
groceryandgourmet_df['Year']=groceryandgourmet_df['Review_Time'].dt.year
```

2 NUMBER OF REVIEWS OVER THE YEARS.

Grouping by year and taking the count of reviews for each year.

[39]:
```
Yearly=dataset.groupby(['Year'])['Reviewer_ID'].count().reset_index()
```

[40]:
```
#Renaming the column 'Reviewer_ID' to 'Number_Of_Reviews'
Yearly=Yearly.rename(columns={'Reviewer_ID': 'Number_Of_Reviews'})
```

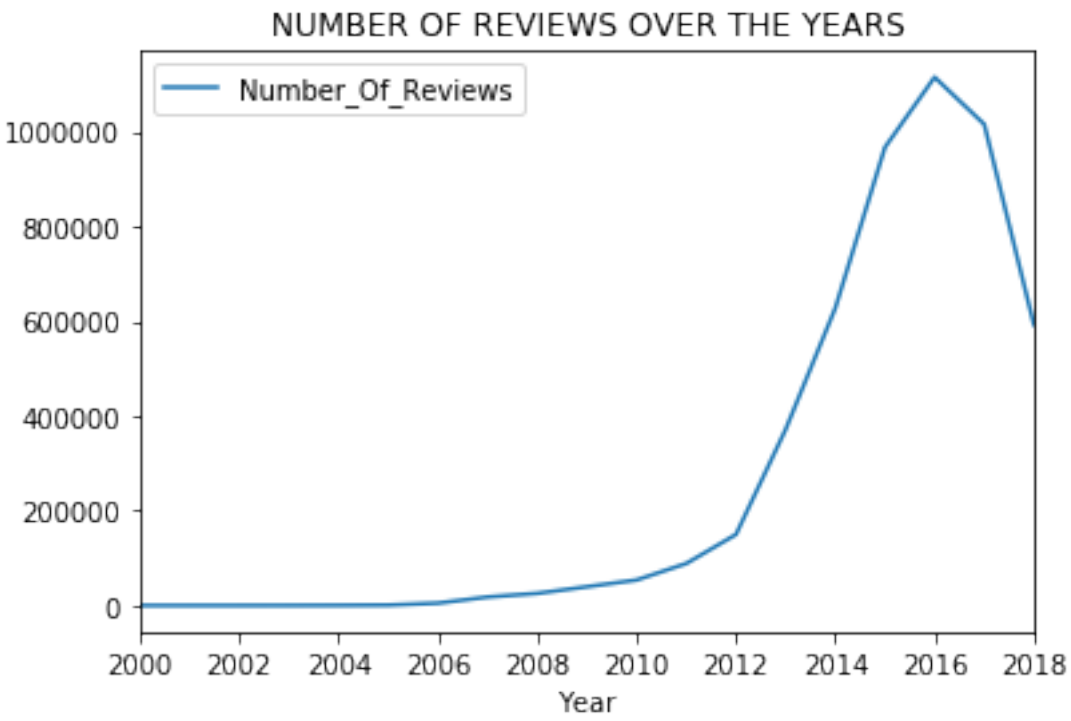
[46]:
```
#Displaying few rows of the output.
Yearly.head(19)
```

[46]:

	Year	Number_Of_Reviews
0	2000	3
1	2001	4
2	2002	14
3	2003	95
4	2004	297
5	2005	818
6	2006	4967
7	2007	17878
8	2008	25541
9	2009	39575
10	2010	53756
11	2011	88592
12	2012	149779
13	2013	371439
14	2014	627446
15	2015	966790
16	2016	1114859
17	2017	1015356
18	2018	592527

[42]:
```
#Line Plot for number of reviews over the years.
Yearly.plot(x="Year",y="Number_Of_Reviews",kind="line",title="NUMBER OF REVIEWS␣
↪OVER THE YEARS")
plt.show()
```

[42]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1638b0b2ec8>
```



3 NUMBER OF REVIEWS BY MONTH (2000-2016)

[73]:
```
#Grouping on Month and getting the count.

Monthly=dataset.groupby(['Month'])['Reviewer_ID'].count().reset_index()
Monthly.head()
```

[73]:

	Month	Reviewer_ID
0	1	501761
1	2	459221
2	3	504227
3	4	442276
4	5	433594

[61]:
```
import calendar
#Replacing digits of 'Month' column in 'Monthly' dataframe with words using␣
 ↪'Calendar' library
# Replacing digits of 'Month' column in 'Monthly' dataframe with words using␣
 ↪'Calendar' library
Monthly['Month'] = Monthly['Month'].apply(lambda x: calendar.month_name[x])
Monthly=Monthly.rename(columns={'Reviewer_ID':'Number_of_Reviews'})
```

[62]:
```
#Bar Chart Plot for number of reviews over the Month


Monthly.plot(x="Month",y="Number_of_Reviews",kind="bar",title="NUMBER OF␣
 ↪REVIEWS OVER THE MONTH")
plt.show()
```

[62]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x163a862de08>
```

[63]:
```
#DISTRIBUTION OF OVERALL RATING
Overall_Rating=dataset.groupby(['Rating'])['Reviewer_ID'].count().reset_index()
Overall_Rating=Overall_Rating.rename(columns={'Reviewer_ID':
 ↪'Number_of_Reviews'})
Overall_Rating
```

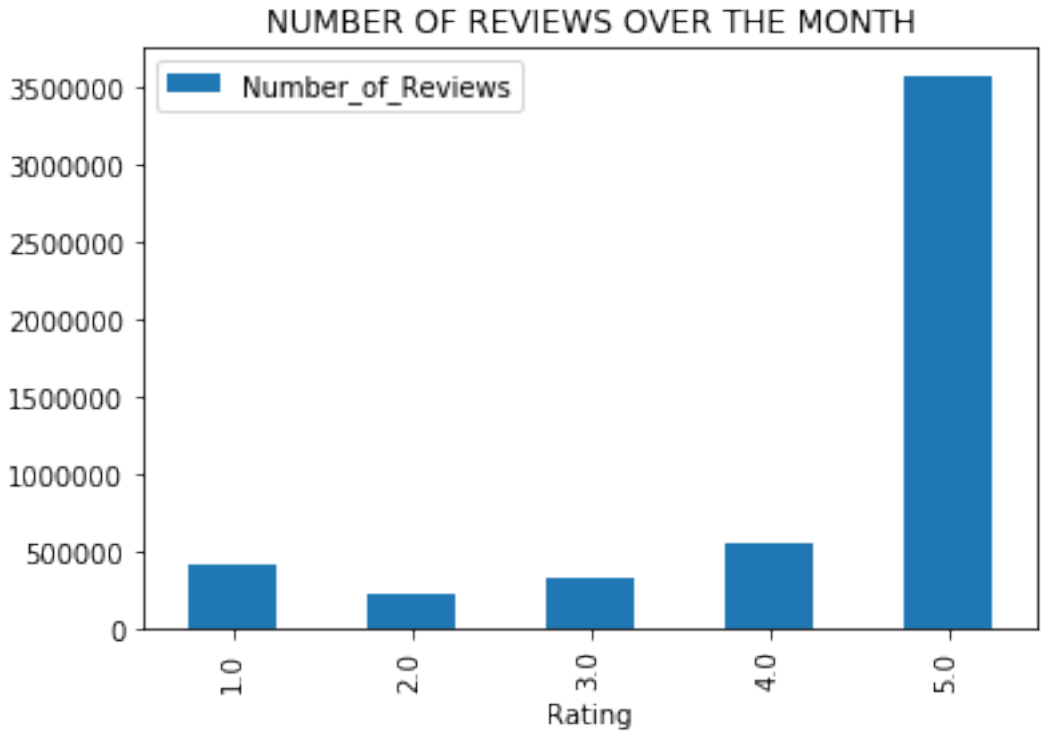
[63]:

	Rating	Number_of_Reviews
0	1.0	404938
1	2.0	219427
2	3.0	322033
3	4.0	552854

```
4      5.0              3570484
```

[64]:
```
#Bar Chart Plot for Distribution of Rating


Overall_Rating.plot(x="Rating",y="Number_of_Reviews",kind="bar",title="NUMBER␣
 ↪OF REVIEWS OVER THE MONTH")
plt.show()
```

[64]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x163a7bc2b08>
```



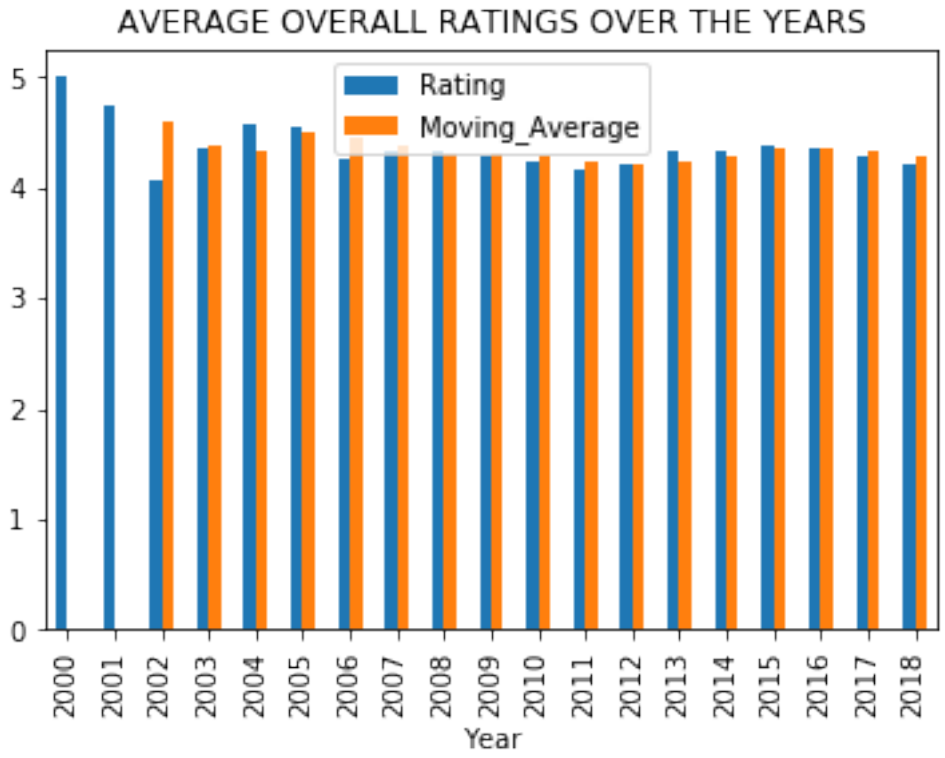
[65]:
```
#AVERAGE OVERALL RATINGS OVER THE YEARS
Yearly_Avg_Rating=dataset.groupby(['Year'])['Rating'].mean().reset_index()
Yearly_Avg_Rating['Moving_Average']=Yearly_Avg_Rating['Rating'].
 ↪rolling(window=3).mean()
Yearly_Avg_Rating.head()
```

[65]:

	Year	Rating	Moving_Average
0	2000	5.000000	NaN
1	2001	4.750000	NaN
2	2002	4.071429	4.607143
3	2003	4.347368	4.389599
4	2004	4.579125	4.332641

[66]:
```
#Bar Chart Plot for average overall ratings over the years


Yearly_Avg_Rating.
 ↪plot(x="Year",y=["Rating","Moving_Average"],kind="bar",title="AVERAGE␣
 ↪OVERALL RATINGS OVER THE YEARS")
plt.show()
```

[66]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x163a7d3e6c8>
```



[67]:
```
#DISTRIBUTUTION OF LENGTH OF REVIEWS
Review_Length=dataset[['Reviewer_ID','Reviewer_Name','Review_Text']]
# Word count
Review_Length['Word_Length']=Review_Length['Review_Text'].apply(lambda x: len(x.
 ↪split()))
# character count
Review_Length['Character_Length']=Review_Length['Review_Text'].apply(lambda x:␣
 ↪len(x))
Review_Length.head()
```

[67]:
```
       Reviewer_ID       Reviewer_Name  \
0   ALP49FBWT4I7V                Lori
```

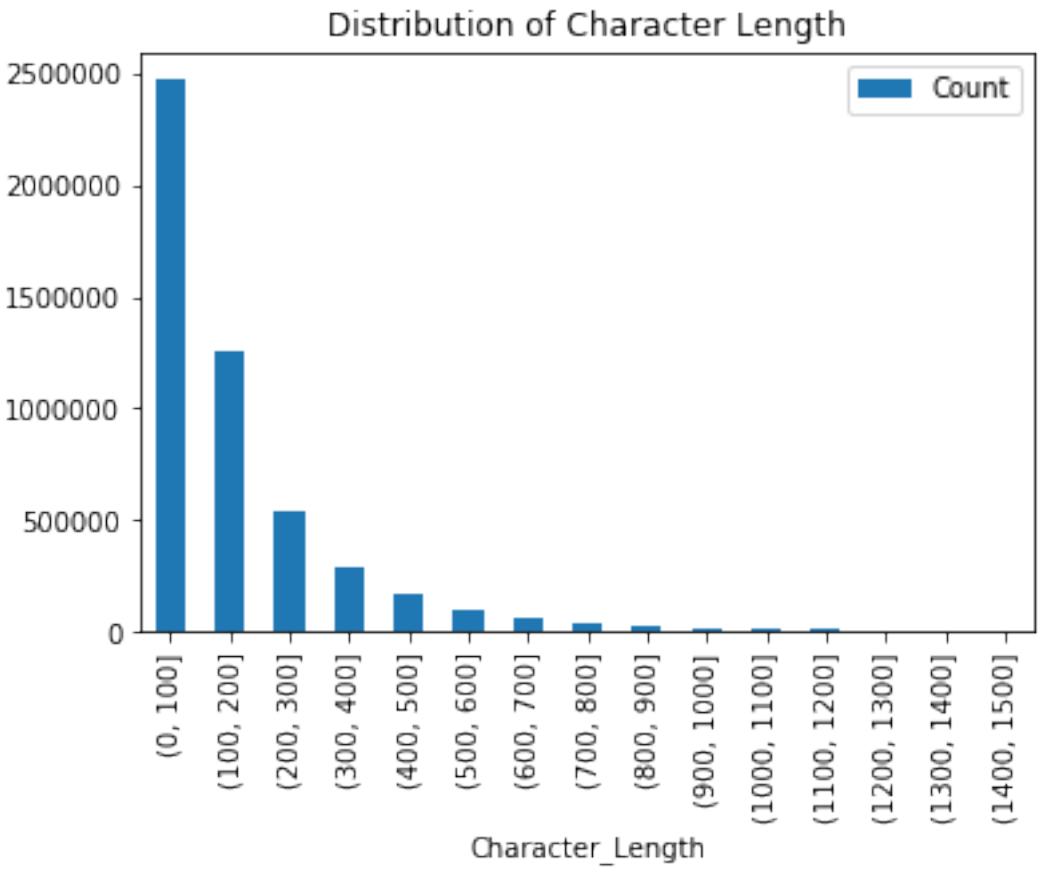
```
1  A1KPIZOCLB9FZ8          BK Shopper
2  A2W0FA06IYAYQE      daninethequeen
3  A2PTZTCH2QUYBC             Tammara
4  A2VNHGJ59N4Z90  LaQuinta Alexander

                                        Review_Text  Word_Length  \
0  Very pleased with my purchase. Looks exactly l…           21
1  Very nicely crafted but too small. Am going to…           21
2  still very pretty and well made…i am super p…           22
3  I got this for our wedding cake, and it was ev…           22
4  It was just what I want to put at the top of m…           24

   Character_Length
0               121
1               112
2               123
3               112
4                99
```

[69]:
```
#Creating an Interval of 100 for Charcters and Words Value.
Char_Review_Length=Review_Length.groupby(pd.cut(Review_Length.
 ↪Character_Length,np.arange(0, 1501, 100))).count()
Char_Review_Length=Char_Review_Length.rename(columns={'Character_Length':
 ↪'Count'})
result_Char_Review_Length=Char_Review_Length.reset_index()

Word_Review_Length=Review_Length.groupby(pd.cut(Review_Length.Word_Length,np.
 ↪arange(0, 801, 100))).count()
Word_Review_Length=Word_Review_Length.rename(columns={'Word_Length':'Count'})
result_Word_Review_Length=Word_Review_Length.reset_index()
result_Char_Review_Length[["Character_Length","Count"]].head()
```

[69]:
```
  Character_Length    Count
0         (0, 100]  2468138
1       (100, 200]  1259786
2       (200, 300]   545942
3       (300, 400]   289135
4       (400, 500]   166579
```

[70]:
```
result_Char_Review_Length.
 ↪plot(x="Character_Length",y="Count",kind="bar",title="Distribution of␣
 ↪Character Length")
plt.show()

#Bar Plot for distribution of Length of reviews on Amazon
```

[70]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1638b171d88>
```



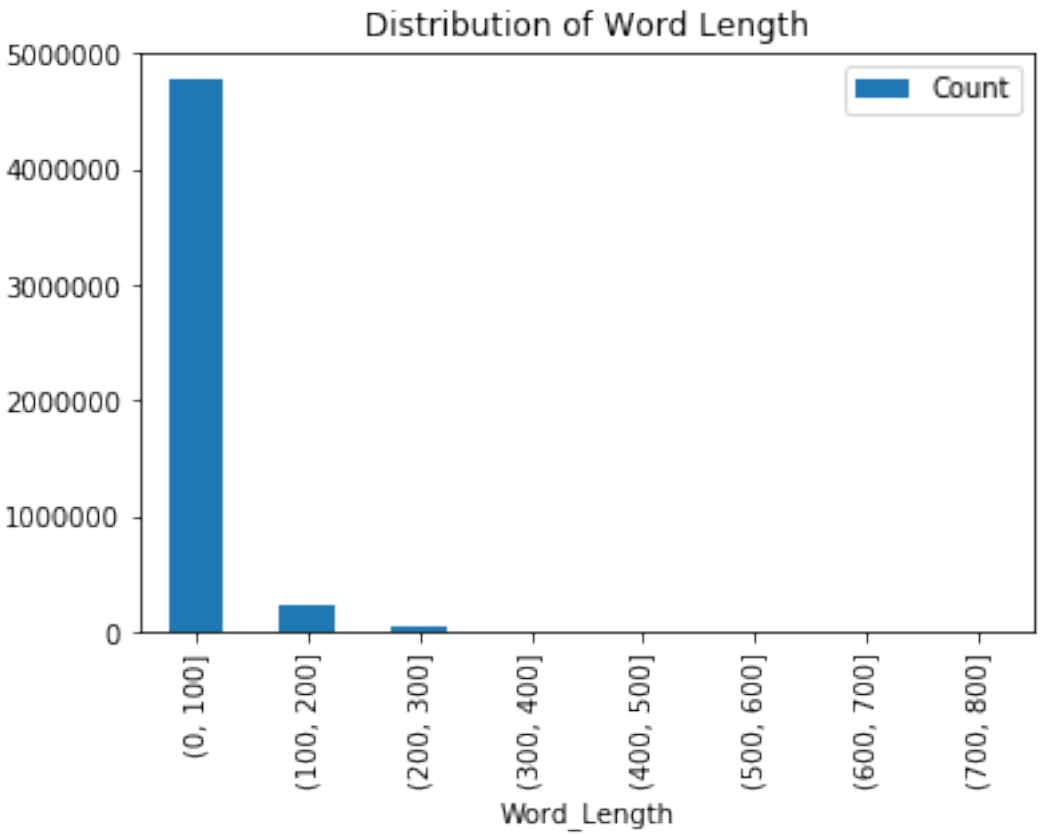
[71]:
```
result_Word_Review_Length[["Word_Length","Count"]].head()
```

[71]:

	Word_Length	Count
0	(0, 100]	4775774
1	(100, 200]	233478
2	(200, 300]	40619
3	(300, 400]	11759
4	(400, 500]	4168

[72]:
```
#Bar Plot for distribution of Length of reviews on Amazon
result_Word_Review_Length.
 ↪plot(x="Word_Length",y="Count",kind="bar",title="Distribution of Word␣
 ↪Length")
plt.show()
```

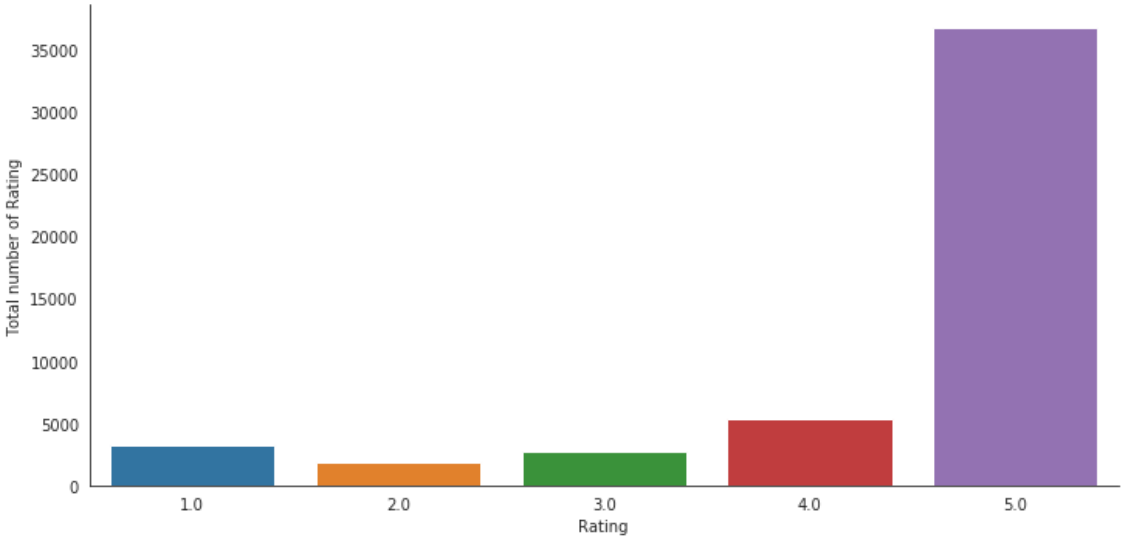
[72]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x163a84da2c8>
```



4 Ratings

[76]:
```
# Check the distribution of the rating
with sns.axes_style('white'):
    g = sns.factorplot("Rating", data=groceryandgourmet_df1, aspect=2.
 ↪0,kind='count')
    g.set_ylabels("Total number of Rating")
```

[76]: <seaborn.axisgrid.FacetGrid at 0x163a84ca2c8>



Users and products

[78]:
```
# Number of unique user id  in the data
print('Number of unique users in Raw data = ',␣
 ↪groceryandgourmet_df1['Reviewer_ID'].nunique())
# Number of unique product id  in the data
print('Number of unique product in Raw data = ', groceryandgourmet_df1['Asin'].
 ↪nunique())
```

```
Number of unique users in Raw data =  46505
Number of unique product in Raw data =  567
```

Taking the subset of dataset to make it less sparse/ denser.

[79]:
```
#Check the top 10 users based on Rating
most_rated=groceryandgourmet_df1.groupby('Reviewer_ID').size().
 ↪sort_values(ascending=False)[:10]
print('Top 10 users based on Rating: \n',most_rated)
```

```
Top 10 users based on Rating:
 Reviewer_ID
A2TY53RWDIO1L1    13
A1YUL9PCJR3JTY    12
A281NPSIMI1C2R     9
A2NYK9KWFMJV4Y     8
A35W3JQYPOM655     7
A3BODXA6INXB8K     7
A26A64X86VL1R4     7
A3FKGKUCI3DG9U     6
A3CK4LQOBHG1AE     6
```

```
A1X1CEGHTHMBL1     6
dtype: int64
```

[80]:
```
counts=groceryandgourmet_df1.Reviewer_ID.value_counts()
groceryandgourmet_df1_final=groceryandgourmet_df1[groceryandgourmet_df1.
 ↪Reviewer_ID.isin(counts[counts>=5].index)]
print('Number of users who have rated 5 or more items =',␣
 ↪len(groceryandgourmet_df1_final))
print('Number of unique users in the final data = ',␣
 ↪groceryandgourmet_df1_final['Reviewer_ID'].nunique())
print('Number of unique products in the final data = ',␣
 ↪groceryandgourmet_df1_final['Reviewer_ID'].nunique())
```

```
Number of users who have rated 5 or more items = 163
Number of unique users in the final data =  26
Number of unique products in the final data =  26
```

[81]:
```
#constructing the pivot table
final_Rating_matrix =pd.pivot_table(groceryandgourmet_df1_final,index =␣
 ↪['Reviewer_ID'], columns =['Asin'], values =['Rating']).fillna(0)
final_Rating_matrix.head(20)
```

[81]:

	Rating					\
Asin	9742356831	B00006FMLY	B00008RCN8	B0000A1OEF	B0000CDBQL	
Reviewer_ID						
A11PK2MO25K5QW	0.0	0.0	0.0	0.0	0.0	
A1BJVYTBOS2AGM	5.0	0.0	0.0	0.0	0.0	
A1DIMIK2OA38W2	0.0	0.0	0.0	0.0	0.0	
A1IU7S4HCK1XKO	0.0	0.0	0.0	0.0	0.0	
A1IW9LSLZFW9FK	0.0	0.0	0.0	0.0	0.0	
A1QP17D4X705C	0.0	0.0	0.0	0.0	0.0	
A1TJT6GXDGNN8Q	0.0	0.0	0.0	0.0	0.0	
A1X1CEGHTHMBL1	0.0	0.0	0.0	0.0	5.0	
A1YUL9PCJR3JTY	0.0	0.0	0.0	0.0	0.0	
A26A64X86VL1R4	0.0	0.0	0.0	0.0	0.0	
A281NPSIMI1C2R	0.0	0.0	0.0	0.0	0.0	
A2CL818RN52NWN	0.0	0.0	0.0	0.0	0.0	
A2NP9CGUSFP22E	0.0	0.0	0.0	0.0	0.0	
A2NYK9KWFMJV4Y	0.0	0.0	5.0	0.0	0.0	
A2TY53RWDIO1L1	0.0	0.0	0.0	0.0	0.0	
A2WVF9ZQO68DNO	0.0	0.0	0.0	0.0	0.0	
A2XKJ1KX6XUHYP	0.0	0.0	0.0	0.0	0.0	
A35W3JQYPOM655	0.0	0.0	0.0	0.0	0.0	
A37MH7ICH8OQOX	0.0	0.0	0.0	0.0	0.0	
A3BODXA6INXB8K	0.0	0.0	0.0	0.0	0.0	

… \

Asin	B0000CFH7B	B0000CFMU3	B0000CFMXV	B0000CH4FT	B0000CNU15	…
Reviewer_ID						…
A11PK2MO25K5QW	0.0	0.0	0.0	0.0	0.0	…
A1BJVYTBOS2AGM	0.0	0.0	0.0	0.0	0.0	…
A1DIMIK2OA38W2	0.0	0.0	0.0	0.0	0.0	…
A1IU7S4HCK1XK0	0.0	0.0	0.0	4.0	0.0	…
A1IW9LSLZFW9FK	0.0	0.0	0.0	0.0	5.0	…
A1QP17D4X705C	0.0	0.0	0.0	0.0	0.0	…
A1TJT6GXDGNN8Q	0.0	0.0	0.0	0.0	0.0	…
A1X1CEGHTHMBL1	0.0	0.0	0.0	0.0	0.0	…
A1YUL9PCJR3JTY	0.0	0.0	0.0	0.0	0.0	…
A26A64X86VL1R4	0.0	0.0	0.0	0.0	0.0	…
A281NPSIMI1C2R	0.0	0.0	0.0	0.0	0.0	…
A2CL818RN52NWN	0.0	0.0	0.0	0.0	0.0	…
A2NP9CGUSFP22E	0.0	0.0	0.0	0.0	0.0	…
A2NYK9KWFMJV4Y	0.0	0.0	0.0	0.0	5.0	…
A2TY53RWDIO1L1	0.0	0.0	0.0	0.0	0.0	…
A2WVF9ZQ068DN0	0.0	0.0	0.0	0.0	0.0	…
A2XKJ1KX6XUHYP	0.0	0.0	0.0	0.0	0.0	…
A35W3JQYP0M655	0.0	0.0	0.0	0.0	0.0	…
A37MH7ICH80QOX	0.0	0.0	0.0	0.0	0.0	…
A3B0DXA6INXB8K	0.0	0.0	0.0	0.0	0.0	…

\

Asin	B0001BH5YM	B0001CVIE4	B0001CVINK	B0001CXUDG	B0001CXUHW
Reviewer_ID					
A11PK2MO25K5QW	0.0	0.0	0.0	0.0	0.0
A1BJVYTBOS2AGM	0.0	0.0	0.0	0.0	0.0
A1DIMIK2OA38W2	0.0	0.0	0.0	0.0	0.0
A1IU7S4HCK1XK0	0.0	0.0	0.0	0.0	0.0
A1IW9LSLZFW9FK	0.0	0.0	0.0	0.0	5.0
A1QP17D4X705C	0.0	0.0	0.0	0.0	0.0
A1TJT6GXDGNN8Q	0.0	0.0	0.0	0.0	0.0
A1X1CEGHTHMBL1	0.0	0.0	0.0	0.0	0.0
A1YUL9PCJR3JTY	0.0	0.0	0.0	0.0	0.0
A26A64X86VL1R4	0.0	0.0	0.0	0.0	0.0
A281NPSIMI1C2R	0.0	0.0	0.0	0.0	5.0
A2CL818RN52NWN	0.0	0.0	0.0	0.0	0.0
A2NP9CGUSFP22E	5.0	0.0	0.0	0.0	0.0
A2NYK9KWFMJV4Y	0.0	5.0	5.0	0.0	0.0
A2TY53RWDIO1L1	0.0	0.0	0.0	0.0	0.0
A2WVF9ZQ068DN0	0.0	0.0	0.0	0.0	0.0
A2XKJ1KX6XUHYP	0.0	0.0	0.0	0.0	0.0
A35W3JQYP0M655	0.0	0.0	0.0	5.0	0.0
A37MH7ICH80QOX	0.0	0.0	0.0	0.0	0.0
A3B0DXA6INXB8K	0.0	0.0	0.0	0.0	0.0

Asin	B0001DMTPU	B0001EJ4CU	B0001ES9FI	B0001FQVCA	B0001GSPD2
Reviewer_ID					
A11PK2MO25K5QW	0.0	0.0	0.0	0.0	0.0
A1BJVYTBOS2AGM	0.0	0.0	0.0	0.0	0.0
A1DIMIK2OA38W2	0.0	0.0	0.0	0.0	0.0
A1IU7S4HCK1XK0	0.0	0.0	0.0	0.0	0.0
A1IW9LSLZFW9FK	0.0	4.0	0.0	0.0	0.0
A1QP17D4X705C	0.0	0.0	3.8	0.0	0.0
A1TJT6GXDGNN8Q	0.0	0.0	0.0	0.0	0.0
A1X1CEGHTHMBL1	0.0	0.0	0.0	0.0	0.0
A1YUL9PCJR3JTY	0.0	0.0	0.0	0.0	5.0
A26A64X86VL1R4	0.0	0.0	0.0	0.0	4.0
A281NPSIMI1C2R	0.0	0.0	0.0	0.0	0.0
A2CL818RN52NWN	0.0	0.0	0.0	0.0	0.0
A2NP9CGUSFP22E	5.0	5.0	0.0	0.0	0.0
A2NYK9KWFMJV4Y	0.0	0.0	0.0	0.0	0.0
A2TY53RWDIO1L1	0.0	0.0	0.0	0.0	0.0
A2WVF9ZQO68DN0	0.0	0.0	0.0	0.0	0.0
A2XKJ1KX6XUHYP	0.0	0.0	0.0	0.0	0.0
A35W3JQYP0M655	0.0	0.0	0.0	0.0	0.0
A37MH7ICH80QOX	0.0	0.0	0.0	3.0	0.0
A3B0DXA6INXB8K	0.0	0.0	0.0	0.0	0.0

[20 rows x 79 columns]

Rating analysis in final dataset

[82]:
```
counts=groceryandgourmet_df1.Reviewer_ID.value_counts()
groceryandgourmet_df1_final=groceryandgourmet_df1[groceryandgourmet_df1.
 ↪Reviewer_ID.isin(counts[counts>=5].index)]
print('Number of users who have rated 5 or more items =',␣
 ↪len(groceryandgourmet_df1_final))
print('Number of unique users in the final data = ',␣
 ↪groceryandgourmet_df1_final['Reviewer_ID'].nunique())
print('Number of unique products in the final data = ',␣
 ↪groceryandgourmet_df1_final['Reviewer_ID'].nunique())
```

```
Number of users who have rated 5 or more items = 163
Number of unique users in the final data =  26
Number of unique products in the final data =  26
```

[83]:
```
final_Rating_matrix =pd.pivot_table(groceryandgourmet_df1_final,index =␣
 ↪['Reviewer_ID'], columns =['Asin'], values =['Rating']).fillna(0)
```

Ratings analysis in final dataset

[84]:
```
#Calucating the density of the rating marix
given_num_of_ratings = np.count_nonzero(final_Rating_matrix)
print('given_num_of_ratings = ', given_num_of_ratings)
possible_num_of_ratings = final_Rating_matrix.shape[0] * final_Rating_matrix.
 ↪shape[1]
print('possible_num_of_ratings = ', possible_num_of_ratings)
density = (given_num_of_ratings/possible_num_of_ratings)
density *= 100
print ('density: {:4.2f}%'.format(density))
```

```
given_num_of_ratings =  111
possible_num_of_ratings =  2054
density: 5.40%
```

4. Train Test Split

[85]:
```
#Split the data randomnly into train and test datasets into 70:30 ratio
train_data, test_data = train_test_split(groceryandgourmet_df1_final, test_size␣
 ↪= 0.3, random_state=0)
train_data.head()
```

[85]:

	Reviewer_ID	Asin	Rating
10307	A3CK4LQOBHG1AE	B0000D9169	5.0
37269	A2WVF9ZQ068DN0	B000168QTU	4.0
32413	A1YUL9PCJR3JTY	B00016ATBS	4.0
20328	A1TJT6GXDGNN8Q	B0000GHNW2	4.0
36651	A2TY53RWDIO1L1	B000168QTU	5.0

[86]:
```
print('Shape of training data: ',train_data.shape)
print('Shape of testing data: ',test_data.shape)
```

```
Shape of training data:  (114, 3)
Shape of testing data:  (49, 3)
```

5 5. Building Popularity Recommder model

[87]:
```
#Count of user_id for each unique product as recommendation score
train_data_grouped = train_data.groupby('Asin').agg({'Reviewer_ID': 'count'}).
 ↪reset_index()
train_data_grouped.rename(columns = {'Reviewer_ID': 'score'},inplace=True)
train_data_grouped.head(40)
```

[87]:

	Asin	score
0	9742356831	1
1	B00006FMLY	1
2	B00008RCN8	1
3	B0000A1OEF	1

```
4   B0000CDBQL      1
5   B0000CFH7B      1
6   B0000CFMXV      1
7   B0000CNU15      2
8   B0000CNU1B      2
9   B0000CNU1X      1
10  B0000CNU28      1
11  B0000CNU5B      1
12  B0000D9169      6
13  B0000D916Y      4
14  B0000D9MYF      1
15  B0000D9N18      1
16  B0000DBN1H      1
17  B0000DBN1L      1
18  B0000DBN1O      2
19  B0000DBN2F      2
20  B0000DBN2J      2
21  B0000DHXGL      2
22  B0000DI145      5
23  B0000DID5R      1
24  B0000E5JIU      1
25  B0000ETAH7      1
26  B0000GHNUE      1
27  B0000GHNUY      1
28  B0000GHNW2      1
29  B0000VLH8S      1
30  B0000VLUOI      1
31  B0000WOGQQ      1
32  B00012NHAC      1
33  B00012OIOU      2
34  B00014D1LU      1
35  B00014JNIO      1
36  B00014VTNW      1
37  B000158YDY      1
38  B00015UC4I      1
39  B00015UC7K      1
```

[88]:
```
#Sort the products on recommendation score
train_data_sort = train_data_grouped.sort_values(['score', 'Asin'], ascending =␣
↪[0,1])

#Generate a recommendation rank based upon score
train_data_sort['rank'] = train_data_sort['score'].rank(ascending=0,␣
↪method='first')

#Get the top 5 recommendations
popularity_recommendations = train_data_sort.head(5)
```

```
popularity_recommendations
```

[88]:

	Asin	score	rank
40	B000168QTU	24	1.0
59	B0001ES9FI	7	2.0
12	B0000D9169	6	3.0
22	B0000DI145	5	4.0
13	B0000D916Y	4	5.0

[89]:
```
# Use popularity based recommender model to make predictions
def recommend(user_id):
    user_recommendations = popularity_recommendations

    #Add user_id column for which the recommendations are being generated
    user_recommendations['Reviewer_ID'] = user_id

    #Bring user_id column to the front
    cols = user_recommendations.columns.tolist()
    cols = cols[-1:] + cols[:-1]
    user_recommendations = user_recommendations[cols]

    return user_recommendations
```

[90]:
```
find_recom = [10,100,150]   # This list is user choice.
for i in find_recom:
    print("The list of recommendations for the userId: %d\n" %(i))
    print(recommend(i))
    print("\n")
```

```
The list of recommendations for the userId: 10

    Reviewer_ID        Asin  score  rank
40           10  B000168QTU     24   1.0
59           10  B0001ES9FI      7   2.0
12           10  B0000D9169      6   3.0
22           10  B0000DI145      5   4.0
13           10  B0000D916Y      4   5.0


The list of recommendations for the userId: 100

    Reviewer_ID        Asin  score  rank
40          100  B000168QTU     24   1.0
59          100  B0001ES9FI      7   2.0
12          100  B0000D9169      6   3.0
22          100  B0000DI145      5   4.0
13          100  B0000D916Y      4   5.0
```

```
The list of recommendations for the userId: 150

    Reviewer_ID        Asin  score  rank
40          150  B000168QTU     24   1.0
59          150  B0001ES9FI      7   2.0
12          150  B0000D9169      6   3.0
22          150  B0000DI145      5   4.0
13          150  B0000D916Y      4   5.0
```

6 6. Building Collaborative Filtering recommender model.¶

[91]:
```
groceryandgourmet_df_CF = pd.concat([train_data, test_data]).reset_index()
groceryandgourmet_df_CF.head()
```

[91]:
```
   index     Reviewer_ID        Asin  Rating
0  10307  A3CK4LQOBHG1AE  B0000D9169     5.0
1  37269  A2WVF9ZQ068DN0  B000168QTU     4.0
2  32413  A1YUL9PCJR3JTY  B00016ATBS     4.0
3  20328  A1TJT6GXDGNN8Q  B0000GHNW2     4.0
4  36651  A2TY53RWDIO1L1  B000168QTU     5.0
```

7 User Based Collaborative Filtering model

[92]:
```
# Matrix with row per 'user' and column per 'item'
pivot_df =pd.pivot_table(groceryandgourmet_df_CF,index = ['Reviewer_ID'],␣
 ↪columns =['Asin'], values =['Rating']).fillna(0)
pivot_df.head()
```

[92]:
```
                    Rating                                              \
Asin            9742356831 B00006FMLY B00008RCN8 B0000A1OEF B0000CDBQL
Reviewer_ID
A11PK2MO25K5QW         0.0        0.0        0.0        0.0        0.0
A1BJVYTBOS2AGM         5.0        0.0        0.0        0.0        0.0
A1DIMIK2OA38W2         0.0        0.0        0.0        0.0        0.0
A1IU7S4HCK1XK0         0.0        0.0        0.0        0.0        0.0
A1IW9LSLZFW9FK         0.0        0.0        0.0        0.0        0.0

                                                                    …  \
Asin            B0000CFH7B B0000CFMU3 B0000CFMXV B0000CH4FT B0000CNU15  …
Reviewer_ID                                                            …
A11PK2MO25K5QW         0.0        0.0        0.0        0.0        0.0  …
```

```
A1BJVYTBOS2AGM        0.0        0.0        0.0        0.0        0.0  …
A1DIMIK2OA38W2        0.0        0.0        0.0        0.0        0.0  …
A1IU7S4HCK1XKO        0.0        0.0        0.0        4.0        0.0  …
A1IW9LSLZFW9FK        0.0        0.0        0.0        0.0        5.0  …

                                                                      \
Asin           B0001BH5YM B0001CVIE4 B0001CVINK B0001CXUDG B0001CXUHW
Reviewer_ID
A11PK2MO25K5QW        0.0        0.0        0.0        0.0        0.0
A1BJVYTBOS2AGM        0.0        0.0        0.0        0.0        0.0
A1DIMIK2OA38W2        0.0        0.0        0.0        0.0        0.0
A1IU7S4HCK1XKO        0.0        0.0        0.0        0.0        0.0
A1IW9LSLZFW9FK        0.0        0.0        0.0        0.0        5.0


Asin           B0001DMTPU B0001EJ4CU B0001ES9FI B0001FQVCA B0001GSPD2
Reviewer_ID
A11PK2MO25K5QW        0.0        0.0        0.0        0.0        0.0
A1BJVYTBOS2AGM        0.0        0.0        0.0        0.0        0.0
A1DIMIK2OA38W2        0.0        0.0        0.0        0.0        0.0
A1IU7S4HCK1XKO        0.0        0.0        0.0        0.0        0.0
A1IW9LSLZFW9FK        0.0        4.0        0.0        0.0        0.0

[5 rows x 79 columns]
```

[93]:
```
print('Shape of the pivot table: ', pivot_df.shape)
```

```
Shape of the pivot table:  (26, 79)
```

[94]:
```
#define user index from 0 to 10
pivot_df['user_index'] = np.arange(0, pivot_df.shape[0], 1)
pivot_df.head()
```

[94]:
```
                    Rating                                            \
Asin           9742356831 B00006FMLY B00008RCN8 B0000A1OEF B0000CDBQL
Reviewer_ID
A11PK2MO25K5QW        0.0        0.0        0.0        0.0        0.0
A1BJVYTBOS2AGM        5.0        0.0        0.0        0.0        0.0
A1DIMIK2OA38W2        0.0        0.0        0.0        0.0        0.0
A1IU7S4HCK1XKO        0.0        0.0        0.0        0.0        0.0
A1IW9LSLZFW9FK        0.0        0.0        0.0        0.0        0.0

                                                                  …  \
Asin           B0000CFH7B B0000CFMU3 B0000CFMXV B0000CH4FT B0000CNU15  …
Reviewer_ID                                                           …
A11PK2MO25K5QW        0.0        0.0        0.0        0.0        0.0  …
A1BJVYTBOS2AGM        0.0        0.0        0.0        0.0        0.0  …
```

```
A1DIMIK2OA38W2        0.0        0.0        0.0        0.0        0.0  …
A1IU7S4HCK1XKO        0.0        0.0        0.0        4.0        0.0  …
A1IW9LSLZFW9FK        0.0        0.0        0.0        0.0        5.0  …

                                                                      \
Asin           B0001CVIE4 B0001CVINK B0001CXUDG B0001CXUHW B0001DMTPU
Reviewer_ID
A11PK2MO25K5QW        0.0        0.0        0.0        0.0        0.0
A1BJVYTBOS2AGM        0.0        0.0        0.0        0.0        0.0
A1DIMIK2OA38W2        0.0        0.0        0.0        0.0        0.0
A1IU7S4HCK1XKO        0.0        0.0        0.0        0.0        0.0
A1IW9LSLZFW9FK        0.0        0.0        0.0        5.0        0.0

                                                            user_index
Asin           B0001EJ4CU B0001ES9FI B0001FQVCA B0001GSPD2
Reviewer_ID
A11PK2MO25K5QW        0.0        0.0        0.0        0.0          0
A1BJVYTBOS2AGM        0.0        0.0        0.0        0.0          1
A1DIMIK2OA38W2        0.0        0.0        0.0        0.0          2
A1IU7S4HCK1XKO        0.0        0.0        0.0        0.0          3
A1IW9LSLZFW9FK        4.0        0.0        0.0        0.0          4

[5 rows x 80 columns]
```

[95]:
```
pivot_df.set_index(['user_index'], inplace=True)
# Actual ratings given by users
pivot_df.head()
```

[95]:
```
                Rating                                                  \
Asin       9742356831 B00006FMLY B00008RCN8 B0000A1OEF B0000CDBQL B0000CFH7B
user_index
0                 0.0        0.0        0.0        0.0        0.0        0.0
1                 5.0        0.0        0.0        0.0        0.0        0.0
2                 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
4                 0.0        0.0        0.0        0.0        0.0        0.0

                                                      …              \
Asin       B0000CFMU3 B0000CFMXV B0000CH4FT B0000CNU15  … B0001BH5YM
user_index                                            …
0                 0.0        0.0        0.0        0.0  …        0.0
1                 0.0        0.0        0.0        0.0  …        0.0
2                 0.0        0.0        0.0        0.0  …        0.0
3                 0.0        0.0        4.0        0.0  …        0.0
4                 0.0        0.0        0.0        5.0  …        0.0

                                                                    \
```

```
Asin       B0001CVIE4 B0001CVINK B0001CXUDG B0001CXUHW B0001DMTPU B0001EJ4CU
user_index
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
2                 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
4                 0.0        0.0        0.0        5.0        0.0        4.0


Asin       B0001ES9FI B0001FQVCA B0001GSPD2
user_index
0                 0.0        0.0        0.0
1                 0.0        0.0        0.0
2                 0.0        0.0        0.0
3                 0.0        0.0        0.0
4                 0.0        0.0        0.0

[5 rows x 79 columns]
```

8 Singular Value Decomposition

[96]:
```
# Singular Value Decomposition
U, sigma, Vt = svds(pivot_df, k = 10)
```

[97]:
```
print('Left singular matrix: \n',U)
```

```
Left singular matrix:
 [[ 7.07767178e-15 -2.33568279e-01  2.34849366e-16  1.05733585e-01
  -7.66216008e-02  3.09710312e-02  3.27836584e-17  1.24377398e-02
   1.67481599e-02 -1.86675752e-01]
 [ 6.96475213e-15  2.06111736e-01 -3.05205983e-16 -2.55539646e-02
   1.97770313e-02  6.12104604e-01  2.52000140e-15  3.58859969e-04
   1.58981913e-04 -1.26376615e-04]
 [ 9.15933995e-15 -2.56200949e-01 -1.34461494e-16 -4.24516827e-02
  -4.27949351e-02  3.20377620e-03 -7.01394981e-17  1.04962773e-02
   1.63955858e-02 -2.25982403e-01]
 [ 0.00000000e+00 -1.77527694e-02 -2.33969707e-16 -6.73052231e-01
   6.47960522e-01 -2.39453344e-02  2.41949547e-15  2.45522326e-02
   2.59080929e-02 -2.16466117e-01]
 [-1.78190795e-14  4.35792873e-01 -7.34937392e-16 -2.06435873e-01
  -2.45730472e-01 -1.59251172e-03 -6.54287818e-16 -2.72192179e-01
  -3.78638948e-01 -6.92466800e-02]
 [-1.71895290e-17 -5.17179933e-19  5.31033113e-17  3.80764859e-17
  -2.69120943e-17 -8.88873712e-18  4.02947501e-17 -1.19740121e-17
  -6.26500448e-18 -6.00220468e-18]
 [-5.57072404e-17  5.02963109e-17  5.39622624e-17 -1.93661450e-16
```

```
 -2.59734725e-18  2.52862597e-17 -2.20332225e-16  4.07522578e-17
  1.11219689e-16  6.80055432e-17]
[ 1.07969189e-14 -3.05881413e-01  5.24874352e-17  1.36898606e-01
 -9.84073095e-02  8.80812316e-03 -2.66820259e-16  1.46364377e-02
  1.91464734e-02 -1.97942952e-01]
[-1.69309011e-14  4.68259271e-01  7.08797287e-16  4.97530131e-01
  2.46354401e-01 -7.60575277e-03  1.37421941e-16  5.25250202e-02
  5.75030074e-02 -4.98945834e-01]
[-2.33146835e-15  4.61597962e-02  3.98209965e-16  1.64976596e-01
  3.04033769e-01 -4.40131266e-03  7.39356533e-16  4.49448083e-02
  4.82349766e-02 -3.72757656e-01]
[ 1.99840144e-15 -1.16732961e-02 -7.13641798e-16 -3.54447735e-01
 -5.09268147e-01 -5.82484942e-03 -1.17091043e-15 -6.36047782e-02
 -5.02626978e-02 -5.73577609e-01]
[ 1.98852236e-16 -3.43157534e-16 -1.00000000e+00  1.25749309e-15
  8.46368019e-16 -7.01496003e-17 -6.90710715e-16 -4.50850591e-16
  2.70497070e-16 -2.09297890e-17]
[-3.59434704e-15  7.54814622e-02  3.43585505e-16  5.08591081e-02
  8.35859583e-02  1.35760186e-03 -3.55510700e-16 -6.43665555e-01
 -2.98578673e-01 -1.55075897e-02]
[ 5.21804822e-15 -1.29253582e-01  1.05504552e-17  8.29610256e-02
  1.05502203e-01  8.87594069e-04  6.18169461e-16  4.66466448e-01
 -8.52663313e-01 -1.25725381e-02]
[ 6.07847106e-15 -1.71595426e-01  2.70765464e-16  8.61631558e-02
 -6.29272428e-02  5.89695009e-03 -2.66712469e-16  1.05596030e-02
  1.44341106e-02 -1.66586746e-01]
[ 7.35522754e-15 -2.02783130e-01  6.85063404e-17 -7.89773767e-02
 -2.31566579e-02  1.34918114e-03  1.90737374e-16  8.10729398e-03
  1.38032272e-02 -2.15888631e-01]
[-3.35668993e-15 -1.77648701e-01  1.18860093e-16  1.88642398e-02
 -1.08765857e-02  2.77883015e-01  1.08571206e-15  1.73665937e-03
  1.72679666e-03 -9.02104004e-03]
[ 2.91739085e-17 -1.05224680e-16  7.70319919e-16 -1.03502341e-15
  2.77370529e-15  3.64223041e-15 -1.00000000e+00  8.37338286e-16
  1.06615318e-17  7.94130125e-18]
[-1.87247136e-15 -5.60234384e-02  1.00846548e-16  3.50693081e-03
 -1.88269882e-03  3.90585372e-02 -5.74284139e-17  1.71247887e-04
  1.35859495e-04 -3.55587002e-04]
[ 7.88205438e-01  2.55034364e-14  6.37834932e-17  5.12383253e-17
  1.59371760e-16  3.97376096e-17 -5.26637737e-17 -2.38030508e-17
 -1.40505051e-16  1.32073442e-16]
[ 6.15412209e-01  1.98427491e-14  2.69372709e-16  9.00192896e-17
 -1.63429781e-16  1.75178543e-17  7.06167135e-17 -7.67753066e-17
  9.55028006e-17  1.74962153e-16]
[ 6.43929354e-15 -1.82847585e-01 -1.44022800e-16  9.18131988e-02
 -6.70536194e-02  6.28363534e-03 -1.53614062e-16  1.12520359e-02
  1.53806097e-02 -1.77510468e-01]
[-3.67775495e-17  1.63126759e-17 -1.08344152e-17  9.78598061e-17
```

```
  2.42817147e-17 -7.44371941e-18  6.95562989e-17  8.72563634e-18
 -6.70323516e-17 -4.63277608e-17]
 [ 1.68268177e-14 -4.06238478e-01  4.26695247e-16  1.98803377e-01
  2.49194239e-01  2.25176602e-03 -1.81893331e-16 -5.32729732e-01
 -1.73387302e-01 -3.64551112e-03]
 [-5.25664581e-15 -7.65935087e-02  2.03602791e-17 -1.30747462e-02
  1.32674184e-02  7.38082326e-01  3.32530452e-15  8.67178186e-04
  5.47544231e-04 -1.07518350e-03]
 [-4.58155270e-16 -8.94409656e-16 -2.64870847e-17 -2.06941387e-17
  1.02685189e-16 -1.04666462e-16 -1.30368004e-16  1.21536281e-16
  1.42336879e-16  1.27144238e-17]]
```

[98]:
```
print('Sigma: \n',sigma)
```

```
Sigma:
 [10.67889603 10.75684482 11.18033989 11.73845428 11.90677312 12.45573239
 13.22875656 13.61660814 14.52933773 18.37646086]
```

[99]:
```
# Construct diagonal array in SVD
sigma = np.diag(sigma)
print('Diagonal matrix: \n',sigma)
```

```
Diagonal matrix:
 [[10.67889603  0.          0.          0.          0.          0.
   0.          0.          0.          0.        ]
 [ 0.         10.75684482  0.          0.          0.          0.
   0.          0.          0.          0.        ]
 [ 0.          0.         11.18033989  0.          0.          0.
   0.          0.          0.          0.        ]
 [ 0.          0.          0.         11.73845428  0.          0.
   0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.         11.90677312  0.
   0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.          0.         12.45573239
   0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.          0.          0.
  13.22875656  0.          0.          0.        ]
 [ 0.          0.          0.          0.          0.          0.
   0.         13.61660814  0.          0.        ]
 [ 0.          0.          0.          0.          0.          0.
   0.          0.         14.52933773  0.        ]
 [ 0.          0.          0.          0.          0.          0.
   0.          0.          0.         18.37646086]]
```

[100]:
```
print('Right singular matrix: \n',Vt)
```

```
Right singular matrix:
 [[ 3.26098883e-15  7.87853804e-15  2.44315901e-15 -1.28708605e-16
```

```
  5.05525987e-15 -2.14514341e-16 -1.71611473e-16 -2.14514341e-16
  0.00000000e+00 -5.89996911e-15  6.19561734e-15  0.00000000e+00
 -8.34312812e-15  2.44315901e-15  6.57192300e-01  6.57192300e-01
  9.31052403e-17  9.31052403e-17  9.31052403e-17  9.31052403e-17
  0.00000000e+00  3.69048184e-01  9.35677920e-16 -5.40613909e-15
 -5.84798700e-15 -5.40613909e-15 -7.43344125e-15  2.44315901e-15
 -1.19104510e-15 -8.76715794e-16  5.82719416e-15  7.48542336e-16
  2.44315901e-15  2.44315901e-15 -1.04331459e-17 -1.04331459e-17
 -2.08662919e-17 -1.82580054e-17  5.05525987e-15  9.31052403e-17
 -2.14514341e-16  9.35677920e-16  4.28852380e-15  1.70809954e-15
 -1.00260488e-14 -1.57164651e-15 -1.57164651e-15  0.00000000e+00
  7.87853804e-15 -2.46123092e-15 -2.46123092e-15  1.36596088e-17
  1.38489829e-14 -7.92727127e-15 -6.34181701e-15 -7.92727127e-15
  1.36596088e-17  1.36596088e-17  1.36596088e-17  7.87853804e-15
 -1.19104510e-15 -4.03287742e-15 -2.46123092e-15  3.26098883e-15
  7.99757910e-16  3.26098883e-15  3.26098883e-15  1.36596088e-17
  1.36596088e-17 -1.68292070e-15  2.44315901e-15  2.44315901e-15
  1.36596088e-17 -7.40745020e-15  6.19561734e-15 -8.35742320e-15
 -2.26477013e-17 -5.26029477e-16 -8.80057066e-15]
[ 9.58049221e-02 -1.88827898e-01 -6.00796905e-02 -2.49443867e-16
 -1.42179895e-01 -4.15739779e-16 -3.32591823e-16 -4.15739779e-16
 -6.60147828e-03  1.42485690e-01 -1.53742581e-01 -8.25184786e-03
  2.02565381e-01 -6.00796905e-02  2.10778283e-14  2.10778283e-14
 -1.59506593e-16 -1.59506593e-16 -1.59506593e-16 -1.59506593e-16
 -8.25184786e-03  1.18545154e-14 -5.42598517e-03  1.68699152e-01
  1.43798016e-01  1.68699152e-01  1.95581149e-01 -6.00796905e-02
 -4.25557774e-02 -2.60408323e-02 -1.51954047e-01 -4.34078814e-03
 -6.00796905e-02 -6.00796905e-02  9.35149883e-18  9.35149883e-18
  1.87029977e-17  1.63651230e-17 -1.42179895e-01 -1.59506593e-16
 -4.15739779e-16 -5.42598517e-03 -1.19087406e-01 -1.36398660e-01
  2.37650698e-01 -8.25747252e-02 -8.25747252e-02 -8.25184786e-03
 -1.88827898e-01 -3.56022189e-02 -3.56022189e-02 -4.89105689e-17
 -3.93372619e-01  2.17656422e-01  1.74125138e-01  2.17656422e-01
 -4.89105689e-17 -4.89105689e-17 -4.89105689e-17 -1.88827898e-01
 -4.25557774e-02 -1.18176944e-01 -3.56022189e-02  9.58049221e-02
  6.02027032e-02  9.58049221e-02  9.58049221e-02 -4.89105689e-17
 -4.89105689e-17  3.50853171e-02 -6.00796905e-02 -6.00796905e-02
 -4.89105689e-17  1.97139395e-01 -1.53742581e-01  1.97137621e-01
  7.09646388e-18 -1.56244994e-02  2.34821231e-01]
[-1.36492265e-16  1.90823916e-16  4.71830702e-18 -7.10723062e-18
  2.34730946e-17 -1.18453844e-17 -9.47630750e-18 -1.18453844e-17
 -8.37075472e-17 -3.23955687e-16  3.44480025e-16 -1.04634434e-16
 -3.28673994e-16  4.71830702e-18  1.48991983e-16  1.48991983e-16
 -4.47213595e-01 -4.47213595e-01 -4.47213595e-01 -4.47213595e-01
 -1.04634434e-16  2.85248453e-17 -3.19150314e-16 -6.55632875e-17
  2.63640746e-16 -6.55632875e-17  4.31671937e-16  4.71830702e-18
  5.57311173e-17  4.50999474e-17 -3.64121015e-16 -2.55320251e-16
  4.71830702e-18  4.71830702e-18  9.65306295e-18  9.65306295e-18
```

```
  1.93061259e-17  1.68928602e-17  2.34730946e-17 -4.47213595e-01
 -1.18453844e-17 -3.19150314e-16 -6.01330080e-17  1.15915773e-16
 -1.75017884e-16  5.31558494e-17  5.31558494e-17 -1.04634434e-16
  1.90823916e-16  9.10539362e-18  9.10539362e-18  3.44497541e-16
  2.49600214e-16  3.16983783e-16  2.53587027e-16  3.16983783e-16
  3.44497541e-16  3.44497541e-16  3.44497541e-16  1.90823916e-16
  5.57311173e-17  6.22612430e-17  9.10539362e-18 -1.36492265e-16
 -1.27386872e-16 -1.36492265e-16 -1.36492265e-16  3.44497541e-16
  3.44497541e-16  1.53656109e-16  4.71830702e-18  4.71830702e-18
  3.44497541e-16 -6.47824308e-16  3.44480025e-16 -1.09283086e-16
  1.33973914e-17  2.70599684e-17  4.59451712e-16]
[-1.08847230e-02  8.46803897e-02  3.53372870e-02 -5.28880673e-18
  5.83120241e-02 -8.81467788e-18 -7.05174231e-18 -8.81467788e-18
 -2.29349526e-01 -5.25941678e-02  1.06343850e-01 -2.86686908e-01
 -8.79314549e-02  3.53372870e-02  6.01687460e-17  6.01687460e-17
  5.35629758e-16  5.35629758e-16  5.35629758e-16  5.35629758e-16
 -2.86686908e-01  2.18249882e-17 -1.50977176e-01  1.85613746e-02
 -8.92611376e-02  1.85613746e-02  2.39810408e-01  3.53372870e-02
  3.10082512e-03  1.49377879e-03 -4.47317089e-01 -1.20781741e-01
  3.53372870e-02  3.53372870e-02 -3.29960735e-17 -3.29960735e-17
 -6.59921471e-17 -5.77431287e-17  5.83120241e-02  5.35629758e-16
 -8.81467788e-18 -1.50977176e-01 -1.80823138e-02  4.08509543e-02
 -6.62679944e-02  8.03523163e-03  8.03523163e-03 -2.86686908e-01
  8.46803897e-02 -5.56919416e-03 -5.56919416e-03 -4.40868698e-16
  2.15549007e-01  2.11923188e-01  1.69538550e-01  2.11923188e-01
 -4.40868698e-16 -4.40868698e-16 -4.40868698e-16  8.46803897e-02
  3.10082512e-03  2.46603748e-03 -5.56919416e-03 -1.08847230e-02
 -1.64539171e-02 -1.08847230e-02 -1.08847230e-02 -4.40868698e-16
 -4.40868698e-16  2.16634605e-02  3.53372870e-02  3.53372870e-02
 -4.40868698e-16 -2.38908631e-01  1.06343850e-01 -4.86817034e-02
  5.23423017e-17  8.96267274e-04  2.68140674e-01]
[ 8.30495008e-03  1.04643902e-01  4.43034405e-02  2.58722968e-17
 -4.13240886e-02  4.31204946e-17  3.44963957e-17  4.31204946e-17
  2.17677960e-01 -5.88859248e-02  1.39744074e-01  2.72097451e-01
 -1.03189365e-01  4.43034405e-02 -1.70408095e-18 -1.70408095e-18
  3.55414524e-16  3.55414524e-16  3.55414524e-16  3.55414524e-16
  2.72097451e-01  6.69248326e-17 -2.13856493e-01 -1.31095396e-01
  4.61840887e-01 -1.31095396e-01  2.10433711e-01  4.43034405e-02
 -1.70407881e-03 -7.90599939e-04  9.07481786e-02 -1.71085194e-01
  4.43034405e-02  4.43034405e-02 -4.36280632e-19 -4.36280632e-19
 -8.72561264e-19 -7.63491106e-19 -4.13240886e-02  3.55414524e-16
  4.31204946e-17 -2.13856493e-01 -1.79708367e-02 -2.84809458e-02
 -6.80891928e-02 -4.56739437e-03 -4.56739437e-03  2.72097451e-01
  1.04643902e-01  5.57137448e-03  5.57137448e-03  1.16475944e-15
 -1.59678447e-01  1.03451371e-01  8.27610968e-02  1.03451371e-01
  1.16475944e-15  1.16475944e-15  1.16475944e-15  1.04643902e-01
 -1.70407881e-03  1.00398011e-03  5.57137448e-03  8.30495008e-03
  1.38763246e-02  8.30495008e-03  8.30495008e-03  1.16475944e-15
```

```
  1.16475944e-15  3.51001726e-02  4.43034405e-02  4.43034405e-02
  1.16475944e-15 -3.17045858e-01  1.39744074e-01 -4.74513197e-02
  1.19984415e-18 -4.74359964e-04  2.05589462e-01]
[ 2.45712008e-01  9.03907514e-04  3.56299429e-04 -2.52092271e-17
  3.53577088e-03 -4.20153785e-17 -3.36123028e-17 -4.20153785e-17
 -7.68973953e-03 -2.82969169e-04  1.44887822e-03 -9.61217441e-03
 -6.39268598e-04  3.56299429e-04  2.29835798e-17  2.29835798e-17
 -2.81595647e-17 -2.81595647e-17 -2.81595647e-17 -2.81595647e-17
 -9.61217441e-03  1.59515348e-17 -2.33822036e-03 -4.78071111e-03
 -1.32108244e-02 -4.78071111e-03 -4.20927271e-03  3.56299429e-04
  3.79885892e-02  1.56789404e-02 -1.14521962e-02 -1.87057629e-03
  3.56299429e-04  3.56299429e-04  4.06018032e-18  4.06018032e-18
  8.12036063e-18  7.10531555e-18  3.53577088e-03 -2.81595647e-17
 -4.20153785e-17 -2.33822036e-03  1.28606496e-03  7.68748990e-02
 -9.42978895e-05  1.11548244e-01  1.11548244e-01 -9.61217441e-03
  9.03907514e-04  2.96282187e-01  2.96282187e-01  1.46206995e-15
  1.56534560e-02 -3.05311343e-03 -2.44249075e-03 -3.05311343e-03
  1.46206995e-15  1.46206995e-15  1.46206995e-15  9.03907514e-04
  3.79885892e-02  4.07830430e-01  2.96282187e-01  2.45712008e-01
  5.41994195e-01  2.45712008e-01  2.45712008e-01  1.46206995e-15
  1.46206995e-15  5.44970709e-04  3.56299429e-04  3.56299429e-04
  1.46206995e-15 -2.97748896e-03  1.44887822e-03  3.35558301e-05
 -5.58032656e-18  9.40736425e-03 -4.46653900e-03]
[ 9.52471000e-16 -6.87492172e-17  2.33646095e-16 -2.95646844e-17
 -1.00848579e-16 -4.92744740e-17 -3.94195792e-17 -4.92744740e-17
  7.31586665e-16 -1.36514555e-17 -2.03119631e-16  9.14483331e-16
 -2.47297550e-16  2.33646095e-16  6.78557343e-18  6.78557343e-18
 -2.61064112e-16 -2.61064112e-16 -2.61064112e-16 -2.61064112e-16
  9.14483331e-16 -1.99050355e-17 -4.42562545e-16 -4.01010056e-16
  1.22509820e-15 -4.01010056e-16  3.21002991e-16  2.33646095e-16
  6.03662174e-17 -2.17059002e-17  6.25468405e-16 -3.54050036e-16
  2.33646095e-16  2.33646095e-16 -3.33111014e-17 -3.33111014e-17
 -6.66222028e-17 -5.82944274e-17 -1.00848579e-16 -2.61064112e-16
 -4.92744740e-17 -4.42562545e-16 -2.65102384e-17  2.56129199e-16
 -3.81667965e-16  4.10360588e-16  4.10360588e-16  9.14483331e-16
 -6.87492172e-17  1.25684697e-15  1.25684697e-15 -3.77964473e-01
 -3.41571595e-16  5.19406114e-17  4.15524891e-17  5.19406114e-17
 -3.77964473e-01 -3.77964473e-01 -3.77964473e-01 -6.87492172e-17
  6.03662174e-17  1.66720756e-15  1.25684697e-15  9.52471000e-16
  2.20931797e-15  9.52471000e-16  9.52471000e-16 -3.77964473e-01
 -3.77964473e-01 -1.34370414e-16  2.33646095e-16  2.33646095e-16
 -3.77964473e-01 -6.89860095e-16 -2.03119631e-16 -3.32208454e-16
  3.68130053e-17 -1.30235401e-17  2.75501013e-16]
[ 1.31772893e-04 -1.95617633e-01  1.71285846e-01  2.67767744e-17
  5.37448001e-03  4.46279573e-17  3.57023658e-17  4.46279573e-17
  7.21243714e-03  7.13372474e-02 -4.31970750e-01  9.01554642e-03
 -9.99485982e-02  1.71285846e-01 -3.69322361e-17 -3.69322361e-17
 -1.65551724e-16 -1.65551724e-16 -1.65551724e-16 -1.65551724e-16
```

```
  9.01554642e-03 -8.74044791e-18 -2.33555881e-02 -7.92589527e-03
  3.70914885e-02 -7.92589527e-03  3.19333653e-02  1.71285846e-01
  1.90421784e-04  6.28819915e-05 -3.04445155e-03 -1.86844705e-02
  1.71285846e-01  1.71285846e-01  5.98566947e-18  5.98566947e-18
  1.19713389e-17  1.04749216e-17  5.37448001e-03 -1.65551724e-16
  4.46279573e-17 -2.33555881e-02  3.85421875e-03  4.03631631e-03
 -3.36301715e-01  6.37698961e-04  6.37698961e-04  9.01554642e-03
 -1.95617633e-01  3.18426651e-04  3.18426651e-04  3.07469480e-16
  3.06429129e-02  1.92871160e-02  1.54296928e-02  1.92871160e-02
  3.07469480e-16  3.07469480e-16  3.07469480e-16 -1.95617633e-01
  1.90421784e-04  9.56125612e-04  3.18426651e-04  1.31772893e-04
  4.50199544e-04  1.31772893e-04  1.31772893e-04  3.07469480e-16
  3.07469480e-16 -2.36353117e-01  1.71285846e-01  1.71285846e-01
  3.07469480e-16 -1.23304186e-01 -4.31970750e-01 -3.16311996e-01
 -2.65718989e-19  3.77291949e-05  3.24900540e-02]
[ 5.47106537e-05 -5.96679992e-02 -2.93428141e-01  2.93895459e-17
  6.58890093e-03  4.89825765e-17  3.91860612e-17  4.89825765e-17
  7.13262871e-03 -4.23729658e-01 -1.62418268e-01  8.91578589e-03
 -1.30301517e-01 -2.93428141e-01 -1.54866834e-17 -1.54866834e-17
  9.30865106e-17  9.30865106e-17  9.30865106e-17  9.30865106e-17
  8.91578589e-03 -4.83521871e-17 -1.72969679e-02 -1.46609981e-03
  3.73881026e-02 -1.46609981e-03  3.24300337e-02 -2.93428141e-01
  1.65602464e-04  4.67535057e-05  5.44392294e-03 -1.38375744e-02
 -2.93428141e-01 -2.93428141e-01  1.53096709e-17  1.53096709e-17
  3.06193418e-17  2.67919241e-17  6.58890093e-03  9.30865106e-17
  4.89825765e-17 -1.72969679e-02  5.64223440e-03  4.96739982e-03
 -2.33051786e-01  5.94244794e-04  5.94244794e-04  8.91578589e-03
 -5.96679992e-02  1.88427113e-04  1.88427113e-04  3.66896688e-18
  4.46940145e-02  1.97885852e-02  1.58308681e-02  1.97885852e-02
  3.66896688e-18  3.66896688e-18  3.66896688e-18 -5.96679992e-02
  1.65602464e-04  7.82671906e-04  1.88427113e-04  5.47106537e-05
  2.43137766e-04  5.47106537e-05  5.47106537e-05  3.66896688e-18
  3.66896688e-18 -1.02750269e-01 -2.93428141e-01 -2.93428141e-01
  3.66896688e-18 -1.47598485e-01 -1.62418268e-01 -2.06991483e-01
 -2.37837616e-17  2.80521034e-05  3.30679177e-02]
[-3.43854609e-05 -9.91896957e-04 -3.42082682e-03  2.07565927e-18
 -5.38577459e-02  3.45943212e-18  2.76754570e-18  3.45943212e-18
 -4.71181298e-02 -2.22619629e-02 -5.21131381e-03 -5.88976623e-02
 -1.88411361e-02 -3.42082682e-03  8.35404589e-17  8.35404589e-17
 -5.69472794e-18 -5.69472794e-18 -5.69472794e-18 -5.69472794e-18
 -5.88976623e-02  3.59354946e-17 -1.56063132e-01 -2.64668557e-01
 -2.41774322e-01 -2.64668557e-01 -2.10028016e-01 -3.42082682e-03
 -5.87652603e-04 -9.67506759e-05 -3.39414413e-01 -1.24850506e-01
 -3.42082682e-03 -3.42082682e-03  7.40137545e-18  7.40137545e-18
  1.48027509e-17  1.29524070e-17 -5.38577459e-02 -5.69472794e-18
  3.45943212e-18 -1.56063132e-01 -6.14869220e-02 -4.21063737e-02
 -2.30605530e-02 -2.45450963e-03 -2.45450963e-03 -5.88976623e-02
 -9.91896957e-04 -2.92543680e-04 -2.92543680e-04  2.16072651e-18
```

```
 -6.52402832e-01 -1.35756781e-01 -1.08605425e-01 -1.35756781e-01
  2.16072651e-18  2.16072651e-18  2.16072651e-18 -9.91896957e-04
 -5.87652603e-04 -2.74705331e-03 -2.92543680e-04 -3.43854609e-05
 -3.26929141e-04 -3.43854609e-05 -3.43854609e-05  2.16072651e-18
  2.16072651e-18 -4.21941685e-03 -3.42082682e-03 -3.42082682e-03
  2.16072651e-18 -1.74904269e-01 -5.21131381e-03 -1.92923257e-02
 -1.33421572e-17 -5.80504055e-05 -2.16894854e-01]]
```

[101]:
```
#Predicted ratings
all_user_predicted_ratings = np.dot(np.dot(U, sigma), Vt)
# Convert predicted ratings to dataframe
preds_df = pd.DataFrame(all_user_predicted_ratings, columns = pivot_df.columns)
preds_df.head()
```

[101]:

	Rating						\
Asin	9742356831	B00006FMLY	B00008RCN8	B0000A1OEF	B0000CDBQL	B0000CFH7B	
0	-0.166851	0.440157	0.123866	5.913905e-16	0.655929	9.856509e-16	
1	2.090992	-0.413612	-0.130487	-7.373727e-16	-0.315327	-1.228954e-15	
2	-0.252858	0.386857	0.094192	6.781005e-16	0.609971	1.130168e-15	
3	0.058690	0.090191	0.034405	3.083044e-16	-0.534906	5.138406e-16	
4	0.445568	-0.342028	0.486866	-1.495282e-15	-0.674604	-2.492137e-15	

					…	\
Asin	B0000CFMU3	B0000CFMXV	B0000CH4FT	B0000CNU15	…	B0001BH5YM
0	7.885207e-16	9.856509e-16	-0.305035	-0.384313	…	-0.143633
1	-9.831636e-16	-1.228954e-15	0.046952	0.315081	…	0.082328
2	9.041340e-16	1.130168e-15	0.219657	-0.334771	…	-0.166086
3	4.110725e-16	5.138406e-16	3.687489	-0.113011	…	-0.008126
4	-1.993710e-15	-2.492137e-15	-0.117932	3.062708	…	1.455907

							\
Asin	B0001CVIE4	B0001CVINK	B0001CXUDG	B0001CXUHW	B0001DMTPU	B0001EJ4CU	
0	0.123866	0.123866	-1.041273e-15	0.039471	0.296524	-0.550176	
1	-0.130487	-0.130487	-1.153288e-15	0.410848	-0.331284	0.438782	
2	0.094192	0.094192	2.058707e-16	0.410736	0.220771	-0.509256	
3	0.034405	0.034405	4.075098e-17	0.003680	0.082065	-0.126058	
4	0.486866	0.486866	-4.889515e-16	3.922332	1.113879	3.516581	

Asin	B0001ES9FI	B0001FQVCA	B0001GSPD2
0	8.382461e-17	0.044642	0.311132
1	-4.225545e-17	0.036702	0.455290
2	3.228686e-18	0.043483	0.027530
3	-3.599373e-16	-0.010309	0.310258
4	5.183496e-17	-0.074435	-0.176758

[5 rows x 79 columns]

[102]:
```
# Recommend the items with the highest predicted ratings

def recommend_items(userID, pivot_df, preds_df, num_recommendations):
    # index starts at 0
    user_idx = userID-1
    # Get and sort the user's ratings
    sorted_user_ratings = pivot_df.iloc[user_idx].sort_values(ascending=False)
    #sorted_user_ratings
    sorted_user_predictions = preds_df.iloc[user_idx].
 ↪sort_values(ascending=False)
    #sorted_user_predictions
    temp = pd.concat([sorted_user_ratings, sorted_user_predictions], axis=1)
    temp.index.name = 'Recommended Items'
    temp.columns = ['user_ratings', 'user_predictions']
    temp = temp.loc[temp.user_ratings == 0]
    temp = temp.sort_values('user_predictions', ascending=False)
    print('\nBelow are the recommended items for user(user_id = {}):\n'.
 ↪format(userID))
    print(temp.head(num_recommendations))
```

[103]:
```
userID = 4
num_recommendations = 5
recommend_items(userID, pivot_df, preds_df, num_recommendations)
```

```
Below are the recommended items for user(user_id = 4):

                user_ratings  user_predictions
       Asin
Rating B0000DBN2J          0.0          0.551133
       B0001GSPD2          0.0          0.310258
       B0000W0GQQ          0.0          0.274572
       B0000DBN1H          0.0          0.151096
       B0000VLU0I          0.0          0.151096
```

[104]:
```
userID = 6
num_recommendations = 5
recommend_items(userID, pivot_df, preds_df, num_recommendations)
```

```
Below are the recommended items for user(user_id = 6):

                user_ratings  user_predictions
       Asin
Rating B000168QTU          0.0      2.108587e-16
       B0000CNU1B          0.0      8.923719e-17
       B0001DMTPU          0.0      8.923719e-17
```

```
       B0000DBN2F           0.0      8.051316e-17
       B0000DBN1L           0.0      8.051316e-17
```

[105]:
```
userID = 8
num_recommendations = 5
recommend_items(userID, pivot_df, preds_df, num_recommendations)
```

```
Below are the recommended items for user(user_id = 8):

                  user_ratings  user_predictions
       Asin
Rating B0000DI145          0.0          0.909086
       B00012NHAC          0.0          0.711595
       B0000W0GQQ          0.0          0.609971
       B0000DBN1L          0.0          0.588577
       B0000DBN2F          0.0          0.588577
```

9 7. Evaluation of Collabrative recommendation model

[106]:
```
# Actual ratings given by the users
final_Rating_matrix.head()
```

[106]:
```
                    Rating                                                \
Asin           9742356831 B00006FMLY B00008RCN8 B0000A1OEF B0000CDBQL
Reviewer_ID
A11PK2MO25K5QW        0.0        0.0        0.0        0.0        0.0
A1BJVYTBOS2AGM        5.0        0.0        0.0        0.0        0.0
A1DIMIK2OA38W2        0.0        0.0        0.0        0.0        0.0
A1IU7S4HCK1XK0        0.0        0.0        0.0        0.0        0.0
A1IW9LSLZFW9FK        0.0        0.0        0.0        0.0        0.0

                                                                      …  \
Asin           B0000CFH7B B0000CFMU3 B0000CFMXV B0000CH4FT B0000CNU15  …
Reviewer_ID                                                           …
A11PK2MO25K5QW        0.0        0.0        0.0        0.0        0.0  …
A1BJVYTBOS2AGM        0.0        0.0        0.0        0.0        0.0  …
A1DIMIK2OA38W2        0.0        0.0        0.0        0.0        0.0  …
A1IU7S4HCK1XK0        0.0        0.0        0.0        4.0        0.0  …
A1IW9LSLZFW9FK        0.0        0.0        0.0        0.0        5.0  …

                                                                      \
Asin           B0001BH5YM B0001CVIE4 B0001CVINK B0001CXUDG B0001CXUHW
Reviewer_ID
A11PK2MO25K5QW        0.0        0.0        0.0        0.0        0.0
A1BJVYTBOS2AGM        0.0        0.0        0.0        0.0        0.0
A1DIMIK2OA38W2        0.0        0.0        0.0        0.0        0.0
```

```
A1IU7S4HCK1XK0        0.0        0.0        0.0        0.0        0.0
A1IW9LSLZFW9FK        0.0        0.0        0.0        0.0        5.0


Asin           B0001DMTPU B0001EJ4CU B0001ES9FI B0001FQVCA B0001GSPD2
Reviewer_ID
A11PK2MO25K5QW        0.0        0.0        0.0        0.0        0.0
A1BJVYTBOS2AGM        0.0        0.0        0.0        0.0        0.0
A1DIMIK2OA38W2        0.0        0.0        0.0        0.0        0.0
A1IU7S4HCK1XK0        0.0        0.0        0.0        0.0        0.0
A1IW9LSLZFW9FK        0.0        4.0        0.0        0.0        0.0

[5 rows x 79 columns]
```

[107]:
```
# Average ACTUAL rating for each item
final_Rating_matrix.mean().head()
```

[107]:
```
         Asin
Rating  9742356831    0.192308
        B00006FMLY    0.192308
        B00008RCN8    0.192308
        B0000A1OEF    0.115385
        B0000CDBQL    0.192308
dtype: float64
```

[108]:
```
# Predicted ratings
preds_df.head()
```

[108]:
```
          Rating                                                          \
Asin 9742356831 B00006FMLY B00008RCN8    B0000A1OEF B0000CDBQL    B0000CFH7B
0     -0.166851   0.440157   0.123866  5.913905e-16   0.655929  9.856509e-16
1      2.090992  -0.413612  -0.130487 -7.373727e-16  -0.315327 -1.228954e-15
2     -0.252858   0.386857   0.094192  6.781005e-16   0.609971  1.130168e-15
3      0.058690   0.090191   0.034405  3.083044e-16  -0.534906  5.138406e-16
4      0.445568  -0.342028   0.486866 -1.495282e-15  -0.674604 -2.492137e-15

                                                        …              \
Asin    B0000CFMU3    B0000CFMXV B0000CH4FT B0000CNU15  … B0001BH5YM
0     7.885207e-16  9.856509e-16  -0.305035  -0.384313  …  -0.143633
1    -9.831636e-16 -1.228954e-15   0.046952   0.315081  …   0.082328
2     9.041340e-16  1.130168e-15   0.219657  -0.334771  …  -0.166086
3     4.110725e-16  5.138406e-16   3.687489  -0.113011  …  -0.008126
4    -1.993710e-15 -2.492137e-15  -0.117932   3.062708  …   1.455907

                                                                      \
Asin B0001CVIE4 B0001CVINK    B0001CXUDG B0001CXUHW B0001DMTPU B0001EJ4CU
0      0.123866   0.123866 -1.041273e-15   0.039471   0.296524  -0.550176
```

```
1     -0.130487  -0.130487 -1.153288e-15   0.410848  -0.331284   0.438782
2      0.094192   0.094192  2.058707e-16   0.410736   0.220771  -0.509256
3      0.034405   0.034405  4.075098e-17   0.003680   0.082065  -0.126058
4      0.486866   0.486866 -4.889515e-16   3.922332   1.113879   3.516581


Asin    B0001ES9FI B0001FQVCA B0001GSPD2
0     8.382461e-17   0.044642   0.311132
1    -4.225545e-17   0.036702   0.455290
2     3.228686e-18   0.043483   0.027530
3    -3.599373e-16  -0.010309   0.310258
4     5.183496e-17  -0.074435  -0.176758

[5 rows x 79 columns]
```

[109]:
```
# Average PREDICTED rating for each item
preds_df.mean().head().apply(lambda x: format(x, 'f'))
```

[109]:
```
         Asin
Rating  9742356831     0.160598
        B00006FMLY     0.245947
        B00008RCN8     0.216918
        B0000A1OEF    -0.000000
        B0000CDBQL     0.156622
dtype: object
```

[110]:
```
rmse_df = pd.concat([final_Rating_matrix.mean(), preds_df.mean()], axis=1)
rmse_df.columns = ['Avg_actual_ratings', 'Avg_predicted_ratings']
print(rmse_df.shape)
rmse_df['item_index'] = np.arange(0, rmse_df.shape[0], 1)
rmse_df.head()
```

```
(79, 2)
```

[110]:
```
                   Avg_actual_ratings  Avg_predicted_ratings  item_index
        Asin
Rating 9742356831            0.192308           1.605984e-01           0
       B00006FMLY            0.192308           2.459472e-01           1
       B00008RCN8            0.192308           2.169181e-01           2
       B0000A1OEF            0.115385          -8.750389e-18           3
       B0000CDBQL            0.192308           1.566216e-01           4
```

[111]:
```
RMSE = round((((rmse_df.Avg_actual_ratings - rmse_df.Avg_predicted_ratings) **␣
 ↪2).mean() ** 0.5), 5)
print('\nRMSE SVD Model = {} \n'.format(RMSE))
```

```
RMSE SVD Model = 0.0859
```

10 8. Getting top - K (K = 5) recommendations.

[112]:
```
# Enter 'userID' and 'num_recommendations' for the user #
userID = 9
num_recommendations = 5
recommend_items(userID, pivot_df, preds_df, num_recommendations)
```

```
Below are the recommended items for user(user_id = 9):

                  user_ratings  user_predictions
       Asin
Rating 9742356831          0.0          0.420538
       B0001990A6          0.0          0.420538
       B00017Y986          0.0          0.420538
       B00019OV8M          0.0          0.420538
       B00012OI0U          0.0          0.386509
```