```
import numpy as np # linear algebra
import pandas as pd
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
import scipy.sparse
from scipy.sparse import csr_matrix
from scipy.sparse.linalg import svds
import warnings; warnings.simplefilter('ignore')
%matplotlib inline
```

```
electronics_data=pd.read_csv("ratings_Electronics(1).csv",names=['userId', 'productId','Rating','timestamp'])
```

```
electronics_data.head()
```

	userId	productId	Rating	timestamp
0	AKM1MP6P0OYPR	0132793040	5.0	1.365811e+09
1	A2CX7LUOHB2NDG	0321732944	5.0	1.341101e+09
2	A2NWSAGRHCP8N5	0439886341	1.0	1.367194e+09
3	A2WNBOD3WNDNKT	0439886341	3.0	1.374451e+09
4	A1GI0U4ZRJA8WN	0439886341	1.0	1.334707e+09

```
electronics_data.shape
```

```
(25850, 4)
```

```
electronics_data=electronics_data.iloc[:1048576,0:]
```

```
electronics_data.dtypes
```

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

```
electronics_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25850 entries, 0 to 25849
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   userId     25850 non-null  object 
 1   productId  25850 non-null  object 
 2   Rating     25849 non-null  float64
 3   timestamp  25849 non-null  float64
dtypes: float64(2), object(2)
memory usage: 807.9+ KB
```

```
electronics_data.describe()['Rating'].T
```

```
count    25849.000000
mean         3.974661
std          1.398456
min          1.000000
25%          3.000000
50%          5.000000
75%          5.000000
max          5.000000
Name: Rating, dtype: float64
```

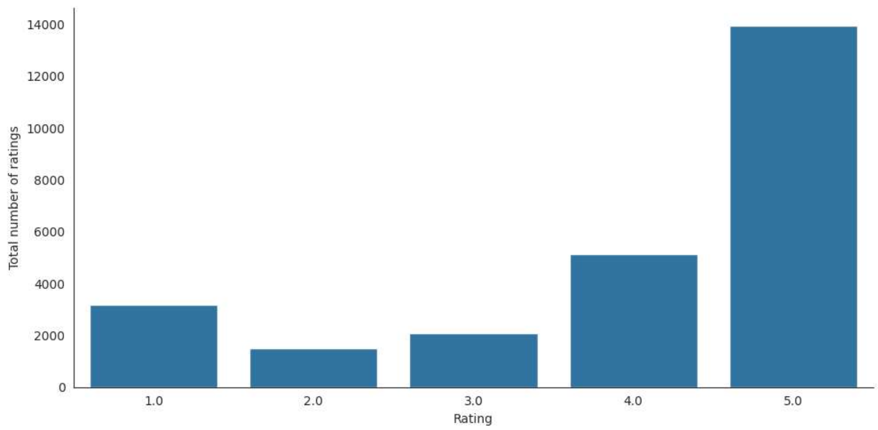
```
#Find the minimum and maximum ratings
print('Minimum rating is: %d' %(electronics_data.Rating.min()))
print('Maximum rating is: %d' %(electronics_data.Rating.max()))
```

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

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

```
Number of missing values across columns:
 userId       0
productId    0
Rating       1
timestamp    1
dtype: int64
```

```
# Check the distribution of the rating
with sns.axes_style('white'):
    g = sns.catplot(x="Rating", data=electronics_data, aspect=2.0, kind='count')
    g.set_ylabels("Total number of ratings")
```




```
pip install --upgrade seaborn
```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.1)
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl (294 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 294.9/294.9 kB 5.3 MB/s eta 0:00:00
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.23.5)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.10/dist-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.2
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (23.2
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.1
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2023.3.post1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4-
Installing collected packages: seaborn
  Attempting uninstall: seaborn
    Found existing installation: seaborn 0.13.1
    Uninstalling seaborn-0.13.1:
      Successfully uninstalled seaborn-0.13.1
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source
lida 0.0.10 requires fastapi, which is not installed.
lida 0.0.10 requires kaleido, which is not installed.
lida 0.0.10 requires python-multipart, which is not installed.
```

```
lida 0.0.10 requires uvicorn, which is not installed.
Successfully installed seaborn-0.13.2
```

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

```
Total data
--------------------------------------------------

Total no of ratings : 25850
Total No of Users   : 24811
Total No of products  : 2216
```

```
#Dropping the Timestamp column

electronics_data.drop(['timestamp'], axis=1,inplace=True)
```

```
#Analysis of rating given by the user

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

no_of_rated_products_per_user.head()
```

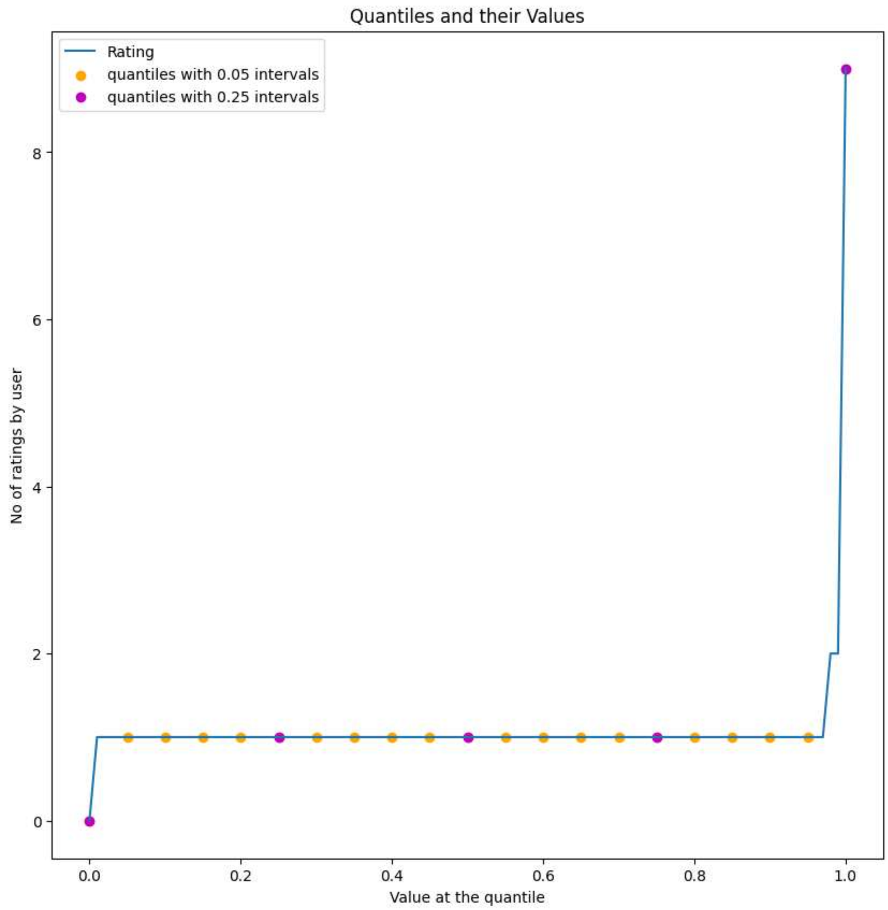
```
userId
A1ISUNUWG0K02V    9
A243HY69GIAHFI    9
A6ZPLVAUQ6695     8
A1ORUSHRRG0VWN    8
A3A15L96IYUO6V    8
Name: Rating, dtype: int64
```

```
no_of_rated_products_per_user.describe()
```

```
count    24811.000000
mean         1.041836
std          0.293458
min          0.000000
25%          1.000000
50%          1.000000
75%          1.000000
max          9.000000
Name: Rating, dtype: float64
```

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

```
plt.figure(figsize=(10,10))
plt.title("Quantiles and their Values")
quantiles.plot()
# quantiles with 0.05 difference
plt.scatter(x=quantiles.index[::5], y=quantiles.values[::5], c='orange', label="quantiles with 0.05 intervals")
# quantiles with 0.25 difference
plt.scatter(x=quantiles.index[::25], y=quantiles.values[::25], c='m', label = "quantiles with 0.25 intervals")
plt.ylabel('No of ratings by user')
plt.xlabel('Value at the quantile')
plt.legend(loc='best')
plt.show()
```




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

```
 No of rated product more than 50 per user : 0
```

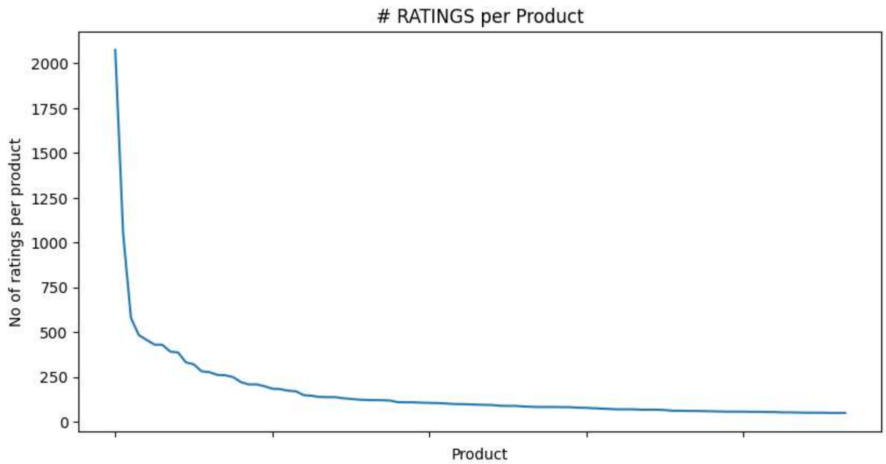
Popularity Based Recommendation

```
new_df=electronics_data.groupby("productId").filter(lambda x:x['Rating'].count() >=50)
```

```
no_of_ratings_per_product = new_df.groupby(by='productId')['Rating'].count().sort_values(ascending=False)

fig = plt.figure(figsize=plt.figaspect(.5))
ax = plt.gca()
plt.plot(no_of_ratings_per_product.values)
plt.title('# RATINGS per Product')
plt.xlabel('Product')
plt.ylabel('No of ratings per product')
ax.set_xticklabels([])

plt.show()
```




```
#Average rating of the product

new_df.groupby('productId')['Rating'].mean().head()
```

```
productId
0972683275    4.470980
1400501466    3.560000
1400501520    4.243902
1400501776    3.884892
1400532620    3.684211
Name: Rating, dtype: float64
```

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

```
productId
B00000J4EY    4.735294
B00000JDF6    4.708738
B00000J1EP    4.651515
9985511476    4.645161
B00000JFMK    4.617647
Name: Rating, dtype: float64
```

```
#Total no of rating for product

new_df.groupby('productId')['Rating'].count().sort_values(ascending=False).head()
```

```
productId
B00001P4ZH    2075
0972683275    1051
B00001P4XA     579
1400532655     484
B00000K2YR     457
Name: Rating, dtype: int64
```

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

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

```
ratings_mean_count.head()
```

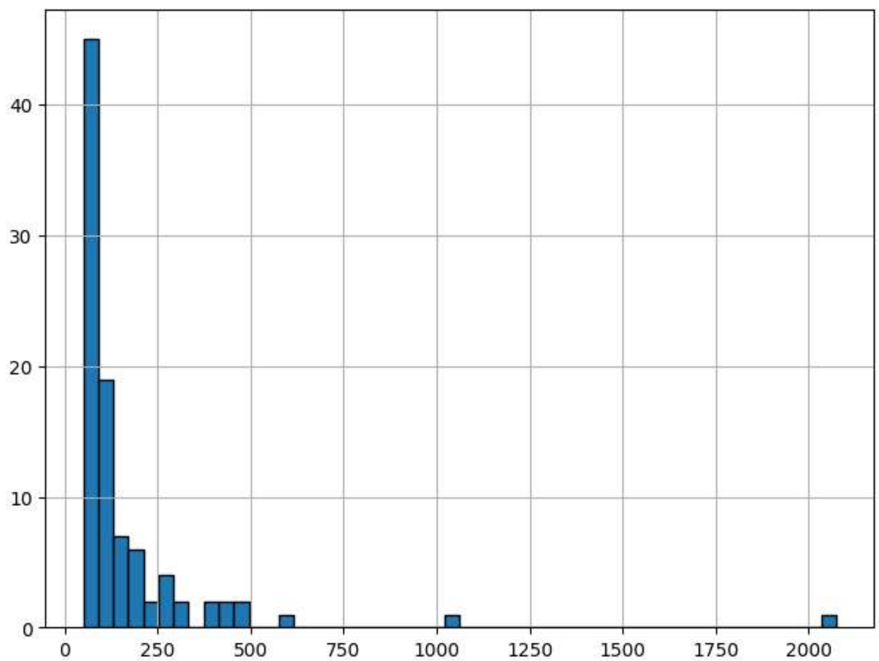
productId	Rating	rating_counts
0972683275	4.470980	1051
1400501466	3.560000	250
1400501520	4.243902	82
1400501776	3.884892	139
1400532620	3.684211	171

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

```
2075
```

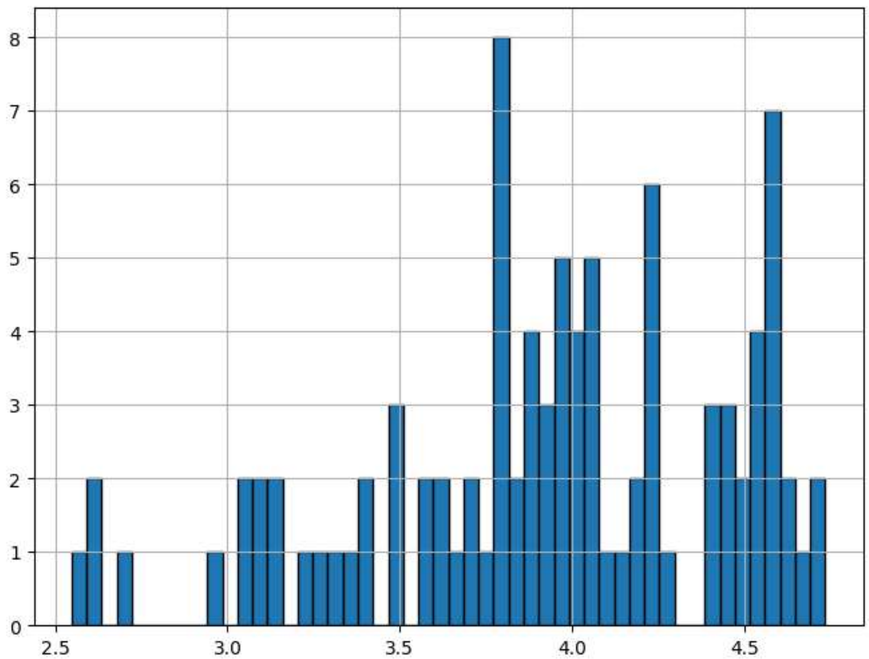
```
plt.figure(figsize=(8,6))
plt.rcParams['patch.force_edgecolor'] = True
ratings_mean_count['rating_counts'].hist(bins=50)
```

```
<Axes: >
```



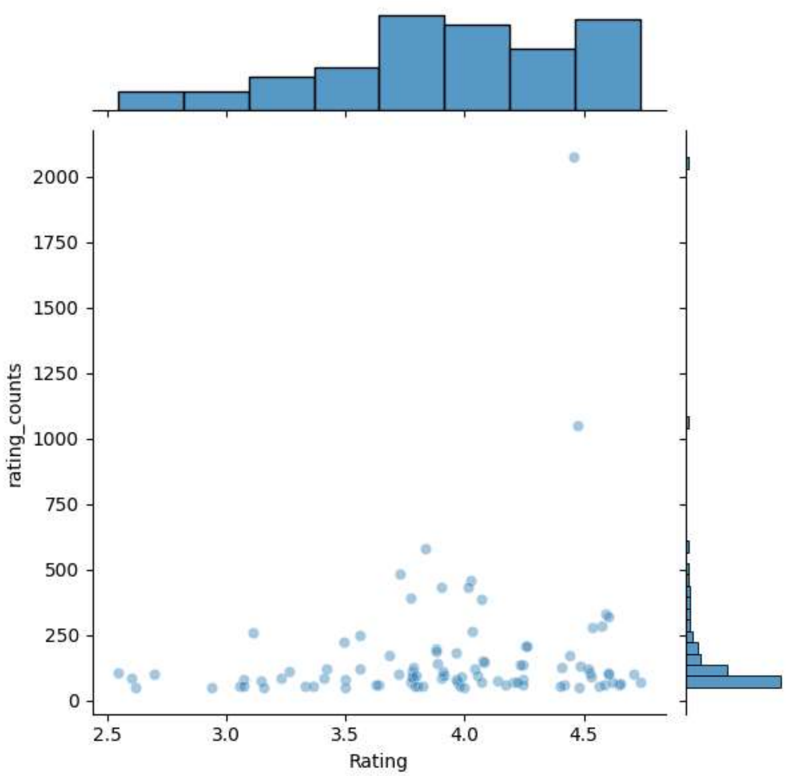

```
plt.figure(figsize=(8,6))
plt.rcParams['patch.force_edgecolor'] = True
ratings_mean_count['Rating'].hist(bins=50)
```

```
<Axes: >
```



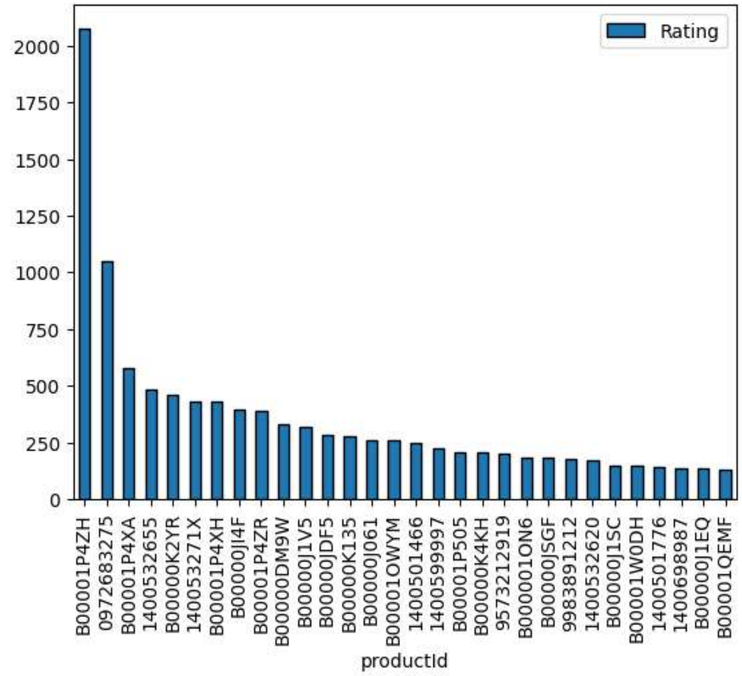

```
plt.figure(figsize=(8,6))
plt.rcParams['patch.force_edgecolor'] = True
sns.jointplot(x='Rating', y='rating_counts', data=ratings_mean_count, alpha=0.4)
```

```
<seaborn.axisgrid.JointGrid at 0x7ee48018de40>
<Figure size 800x600 with 0 Axes>
```




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

```
<Axes: xlabel='productId'>
```




```
!pip install scikit-surprise
```

```
Collecting scikit-surprise
  Downloading scikit-surprise-1.1.3.tar.gz (771 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 772.0/772.0 kB 6.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: joblib>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-surprise) (1.3.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-surprise) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-surprise) (1.11.4)
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (setup.py) ... done
  Created wheel for scikit-surprise: filename=scikit_surprise-1.1.3-cp310-cp310-linux_x86_64.whl size=3162668 sha256=f754d0af1ae4365c57
  Stored in directory: /root/.cache/pip/wheels/a5/ca/a8/4e28def53797fdc4363ca4af740db15a9c2f1595ebc51fb445
Successfully built scikit-surprise
Installing collected packages: scikit-surprise
Successfully installed scikit-surprise-1.1.3
```

```
pip list
```

```
types-pytz                  2023.3.1.1
types-setuptools            69.0.0.20240125
typing_extensions           4.5.0
tzlocal                     5.2
uc-micro-py                 1.0.2
uritemplate                 4.1.1
urllib3                     2.0.7
vega-datasets               0.9.0
wadllib                     1.3.6
wasabi                      1.1.2
wcwidth                     0.2.13
webcolors                   1.13
webencodings                0.5.1
websocket-client            1.7.0
Werkzeug                    3.0.1
wheel                       0.42.0
widgetsnbextension          3.6.6
wordcloud                   1.9.3
wrapt                       1.14.1
xarray                      2023.7.0
xarray-einstats             0.7.0
xgboost                     2.0.3
xlrd                        2.0.1
xxhash                      3.4.1
xyzservices                 2023.10.1
yarl                        1.9.4
yellowbrick                 1.5
yfinance                    0.2.36
zict                        3.0.0
zipp                        3.17.0
```

Collaberative filtering (Item-Item recommedation)

```
from surprise import KNNWithMeans
from surprise import Dataset
from surprise import accuracy
from surprise import Reader, Dataset
from surprise.model_selection import train_test_split
```

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

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

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

```
Estimating biases using als...
Computing the pearson_baseline similarity matrix...
Done computing similarity matrix.
<surprise.prediction_algorithms.knns.KNNWithMeans at 0x7ee47db1cb20>
```

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

```
test_pred
```

```
 Prediction(uid='A22WMZMHRAR43Z', iid='B00000J1QK', r_ui=4.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
 Prediction(uid='A1Y07T8GMP0XF6', iid='B00001W0DD', r_ui=4.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
 Prediction(uid='A25G6ATFVAW9V0', iid='B00000JPPI', r_ui=1.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
 Prediction(uid='A2KPR6PUZS26Q8', iid='140053271X', r_ui=5.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
 Prediction(uid='A80DXXEKX12RI', iid='B00001P4XA', r_ui=4.0, est=3.8312655086848637, details={'actual_k': 0, 'was_impossible':
False}),
 Prediction(uid='A28N5MLYX9Q8ZN', iid='0972683275', r_ui=5.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
 Prediction(uid='A3MAKX14WHMXXK', iid='0972683275', r_ui=4.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
 Prediction(uid='AOOZR8IN9NYTA', iid='B00001P4ZH', r_ui=5.0, est=4.062176165803109, details={'was_impossible': True, 'reason': 'User
and/or item is unknown.'}),
 Prediction(uid='AM178A0W4L6TT', iid='0972683275', r_ui=4.0, est=4.062176165803109, details={'was_impossible': True, 'reason': 'User
and/or item is unknown.'}),
 Prediction(uid='A3JX1D26WFEXOS', iid='B00000JDF6', r_ui=5.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
 Prediction(uid='A17J1191W4F8MI', iid='B00000J1SC', r_ui=3.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
 Prediction(uid='A3JJMT5DZG8LC1', iid='B00001OWYM', r_ui=1.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
 Prediction(uid='A1B3LQWALW1IXI', iid='B00001P4ZH', r_ui=4.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
 Prediction(uid='A1ROFLLG5TMH97', iid='B00001P4XA', r_ui=5.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
 Prediction(uid='A3IRASTCORS6X4', iid='1400599997', r_ui=4.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
 Prediction(uid='A2K4HQA1K0CVGP', iid='0972683275', r_ui=4.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
 Prediction(uid='A3FZ9BSETDQZ7W', iid='140053271X', r_ui=4.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
 Prediction(uid='ANEUPVXP9GKJQ', iid='1400599997', r_ui=5.0, est=4.062176165803109, details={'was_impossible': True, 'reason': 'User
and/or item is unknown.'}),
 Prediction(uid='A2TADAUDP0OA6', iid='B00001P505', r_ui=5.0, est=4.062176165803109, details={'was_impossible': True, 'reason': 'User
and/or item is unknown.'}),
 Prediction(uid='A1CLVDQRMC4NHE', iid='B00001P4ZH', r_ui=5.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
 Prediction(uid='A1XGVNH4Y6G5OC', iid='B00000JI4F', r_ui=5.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
 Prediction(uid='A3A18QLK7CNIVX', iid='B00001P4ZH', r_ui=4.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
 Prediction(uid='A3945ISJV9TI1P', iid='B00000JFIF', r_ui=5.0, est=4.062176165803109, details={'was_impossible': True, 'reason':
'User and/or item is unknown.'}),
```

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

```
Item-based Model : Test Set
RMSE: 1.3413
1.341251982207542
```

Model-based collaborative filtering system

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

productId	0972683275	1400501466	1400501520	1400501776	1400532620	1400532655	140053271X	1400532736	1400599997	14
userId										
A01852072Z7B68UHLI5UG	0	0	0	0	0	0	0	0	0	
A0266076X6KPZ6CCHGVS	0	0	0	0	0	0	0	0	0	
A0293130VTX2ZXA70JQS	5	0	0	0	0	0	0	0	0	
A030530627MK66BD8V4LN	4	0	0	0	0	0	0	0	0	
A0571176384K8RBNKGF8O	0	0	0	0	0	0	0	0	0	

5 rows × 76 columns

```
ratings_matrix.shape
```

```
(9832, 76)
```

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

userId	A01852072Z7B68UHLI5UG	A0266076X6KPZ6CCHGVS	A0293130VTX2ZXA70JQS	A030530627MK66BD8V4LN	A0571176384K8RBNKGF8O	A0590501PZ
productId						
0972683275	0	0	5	4	0	
1400501466	0	0	0	0	0	
1400501520	0	0	0	0	0	
1400501776	0	0	0	0	0	
1400532620	0	0	0	0	0	

5 rows × 9832 columns

```
X.shape
```

```
(76, 9832)
```

```
X1 = X
```

```
#Decomposing the Matrix
from sklearn.decomposition import TruncatedSVD
SVD = TruncatedSVD(n_components=10)
decomposed_matrix = SVD.fit_transform(X)
decomposed_matrix.shape
```

```
(76, 10)
```

```
#Correlation Matrix

correlation_matrix = np.corrcoef(decomposed_matrix)
correlation_matrix.shape
```

```
(76, 76)
```

```
X.index[75]
```

```
'B00000K135'
```

```
i = "B00000K135"

product_names = list(X.index)
product_ID = product_names.index(i)
product_ID
```

```
75
```

```
correlation_product_ID = correlation_matrix[product_ID]
correlation_product_ID.shape
```

```
(76,)
```

```
Recommend = list(X.index[correlation_product_ID > 0.65])
```