Recommendations_with_IBM

December 26, 2021

1 Recommendations with IBM

In this notebook, you will be putting your recommendation skills to use on real data from the IBM Watson Studio platform.

You may either submit your notebook through the workspace here, or you may work from your local machine and submit through the next page. Either way assure that your code passes the project RUBRIC. **Please save regularly.**

By following the table of contents, you will build out a number of different methods for making recommendations that can be used for different situations.

1.1 Table of Contents

At the end of the notebook, you will find directions for how to submit your work. Let's get started by importing the necessary libraries and reading in the data.

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import project_tests as t
        import pickle

        %matplotlib inline

        df = pd.read_csv('data/user-item-interactions.csv')
        df_content = pd.read_csv('data/articles_community.csv')
        del df['Unnamed: 0']
        del df_content['Unnamed: 0']

        # Show df to get an idea of the data
        df.head()

Out[1]:    article_id                                          title  \
        0     1430.0  using pixiedust for fast, flexible, and easier...
        1     1314.0        healthcare python streaming application demo
        2     1429.0          use deep learning for image classification
        3     1338.0           ml optimization using cognitive assistant
```

```
           4       1276.0           deploy your python model as a restful api

                                                        email
           0   ef5f11f77ba020cd36e1105a00ab868bbdbf7fe7
           1   083cbdfa93c8444beaa4c5f5e0f5f9198e4f9e0b
           2   b96a4f2e92d8572034b1e9b28f9ac673765cd074
           3   06485706b34a5c9bf2a0ecdac41daf7e7654ceb7
           4   f01220c46fc92c6e6b161b1849de11faacd7ccb2
```

In [2]: `df.shape[0]`

Out[2]: 45993

In [3]: `df_content.shape[0]`

Out[3]: 1056

In [4]: ```
# Show df_content to get an idea of the data
df_content.head()
```

Out[4]:
```
                                                     doc_body \
         0 Skip navigation Sign in SearchLoading...\r\n\r...
         1 No Free Hunch Navigation * kaggle.com\r\n\r\n ...
         2   * Login\r\n * Sign Up\r\n\r\n * Learning Pat...
         3 DATALAYER: HIGH THROUGHPUT, LOW LATENCY AT SCA...
         4 Skip navigation Sign in SearchLoading...\r\n\r...

                                              doc_description \
         0 Detect bad readings in real time using Python ...
         1 See the forest, see the trees. Here lies the c...
         2 Heres this weeks news in Data Science and Bi...
         3 Learn how distributed DBs solve the problem of...
         4 This video demonstrates the power of IBM DataS...

                                                doc_full_name doc_status article_id
         0 Detect Malfunctioning IoT Sensors with Streami...   Live   0
         1 Communicating data science: A guide to present...   Live   1
         2   This Week in Data Science (April 18, 2017)   Live   2
         3 DataLayer Conference: Boost the performance of...   Live   3
         4   Analyze NY Restaurant data using Spark in DSX   Live   4
```

### 1.1.1 Part I : Exploratory Data Analysis

Use the dictionary and cells below to provide some insight into the descriptive statistics of the data.

1. What is the distribution of how many articles a user interacts with in the dataset? Provide a visual and descriptive statistics to assist with giving a look at the number of times each user interacts with an article.

**a. Exploratory analysis about "number of users interact with an article"**

- Number of times each user interacts with an article

```
In [5]: df.groupby(['article_id','email']).size()

Out[5]: article_id  email
        0.0         2841916b462a2b89d36f4f95ca2d1f42559a5788    1
                    384255292a8223e84f05ca1e1deaa450c993e148    3
                    451a9a4a4cb1cc4e5f38d04e8859cc3fb275cc66    1
                    74ca1ae8b034f7fad73a54d55fb1f58747f00493    1
                    8bd0afc488016810c287ac4ec844895d570b0af4    1
                    a60b7e945a8f2114d5dfbdd53182ad1d526534e2    1
                    ad06c765d31179e56f309438367ecb30e1059620    1
                    ca7d48adf2c7394ed5a8776de959fa8047e43d4b    1
                    db8ac9b2f552db35750239ada8bfcb59b3ae48c0    1
                    df722d3aac72766b93d4a65d8b4ac084a968d684    1
                    e667c9a1cd56368dfa2f4b974ab2d848585552d7    1
                    e6ed9e15addba353fe3c1f36d865a63fa254b9cc    1
        2.0         0246d11c827f90850ce7062e9554c9d5eeb30027    1
                    0286bfe26356436658cf4b29b232f0700f0bb9ce    2
                    12815feeacc6f27dff5b3441a54418d2d51001ef    1
                    12bb8a9740400ced27ae5a7d4c990ac3b7e3c77d    1
                    15a1660b6450e064200f1272d9b3d049cf8cf5f1    1
                    1d74fc07ef225ff993b9f80dfba85a6bd2bd55b8    1
                    249d60fc4edda28cd8fd76f549ecc43259e07038    1
                    26b8f921fac7a4d81f2749d64c10020491281545    1
                    2b6c0f514c2f2b04ad3c4583407dccd0810469ee    3
                    2f5c7feae533ce046f2cb16fb3a29fe00528ed66    1
                    3427a5a4065625363e28ac8e85a57a9436010e9c    3
                    387f29d1e6f4360fa1a2c9607edfa184520bd716    1
                    3e9be703aad3a99412af09cdefcdf28fe5ff2a32    1
                    40222b846f3cef9a645dfb34fc15f7c1c244e393    1
                    40a942b88fbc6b891eb335e12fbc589870098153    1
                    449235d86e9d80f808112130da55d08bfb41703d    1
                    4594c0796c9f32915d3f5c05c2cc5378ffcd40b4    1
                    497935037e41a94d2ae02488d098c7abda9a30bc    1
                                                               ..
        1440.0      a4ce0a47da79499778f8cf4c94fd5cd0fde39002    1
                    aaaf1a153d6b2f025b275c68324b6dd6d15f22ff    2
                    e685741240520687a02b033e21938ddf3acdab7f    2
        1441.0      3c3b60c3fb094373d6aa9420b2f8c08cd6a23354    1
                    3cdbb321cad01f39848fcde8288109a73ee7febb    2
                    436337957e43e0b1db33a58e87971319214d03a9    2
                    6cce7568da5452718e1a3702edffac34a8da74ec    1
                    8c37a3959fb30f349adff02cc545907dafd41b2a    1
                    d5843ed71361c87b364f578f20a48101289d60f9    1
        1442.0      0bd8ddb0c5cec4623a6bb663592747ad55477680    1
                    21026417853cd181c161ae20651318978bf177b0    1
                    5ad53e0336bcd4aef788745048a451f9c383bf81    1
```

3

```
                      66fd330648798030d5fe39b4ea5b8f61e618eb6e    1
          1443.0      32c368e390424c9326f736d32725e1e167abcbe4    2
                      3d840ffab77365a1b2e5ccd464bd439a347d6105    1
                      4ac1eb6a4d2dfbd4e18042c0908de9d3dd7c39b0    2
                      4accff186b3c4fb061b11ad6ae1920556bd68382    2
                      6cce7568da5452718e1a3702edffac34a8da74ec    1
                      8a7982ea3a9d4fe4fd7389e0a94a539ccc81a7bc    1
                      9676b33c28cfc8f8dea534ebfd26ec140f8e442f    5
                      c20e17b171ff2a34b4d684b4fe3af3be7e0700f3    2
                      d53071c9f495307a7a2b9d3619f7ec6e8721e1b7    1
                      d5843ed71361c87b364f578f20a48101289d60f9    1
                      ecee962fad63a7828319b3aa0a6557f94fc5691e    1
                      fa19d1470e496ac4a7bd0eeb07d17b2b3a2f9e30    3
          1444.0      6cce7568da5452718e1a3702edffac34a8da74ec    1
                      c45f9495a76bf95d2633444817f1be8205ad542d    1
                      d313c83ab3ed388ba16042a6cd33fce57d6a9e9a    1
                      d5843ed71361c87b364f578f20a48101289d60f9    1
                      fd824fc62b4753107e3db7704cd9e8a4a1c961f1    1
          Length: 33669, dtype: int64
```

- Number of users interact with an article

```
In [6]: article_interact_unique = df.groupby(['article_id'])['email'].nunique()
        article_interact_unique

Out[6]: article_id
        0.0       12
        2.0       44
        4.0       13
        8.0       82
        9.0       10
        12.0      99
        14.0      89
        15.0      26
        16.0      56
        18.0      68
        20.0     186
        25.0      15
        26.0      80
        28.0      39
        29.0      41
        30.0      17
        32.0      60
        33.0     109
        34.0      86
        36.0      18
        39.0      59
        40.0      64
```

```
43.0      299
48.0       11
50.0       69
51.0      107
53.0       93
54.0       20
57.0      128
58.0       11
          ...
1412.0     19
1414.0      4
1415.0     10
1416.0     73
1418.0     41
1419.0      6
1420.0     94
1421.0      3
1422.0    105
1423.0    102
1424.0    115
1425.0     57
1426.0     96
1427.0    308
1428.0     91
1429.0    397
1430.0    237
1431.0    320
1432.0    232
1433.0     86
1434.0     36
1435.0     75
1436.0    282
1437.0    127
1439.0     43
1440.0      8
1441.0      6
1442.0      4
1443.0     12
1444.0      5
Name: email, Length: 714, dtype: int64
```

- The distribution of "how many user interact with an article"

```
In [7]: x2 = pd.Series(article_interact_unique, name="number of users interact with an article -
        sns.distplot(x2)
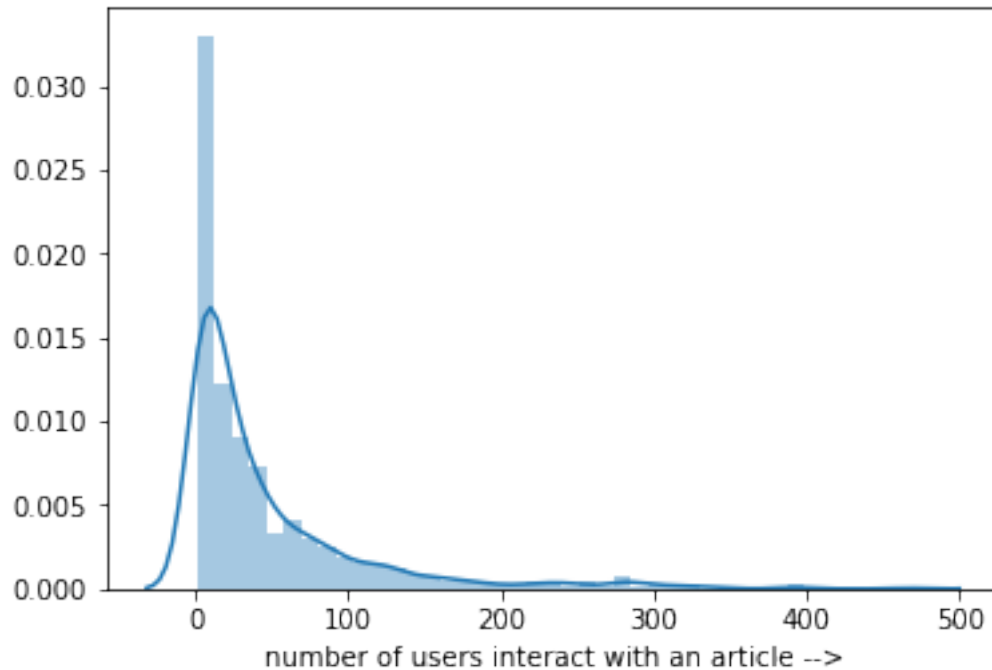
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x7f44c14d1e10>
```

b. Exploratory analysis about "number of articles a user interacts with"

- Number of times a user interacts with articles

```
In [8]: df.groupby(['email','article_id']).size()
```

```
Out[8]: email                                      article_id
        0000b6387a0366322d7fbfc6434af145adf7fed1   43.0          2
                                                   124.0         1
                                                   173.0         1
                                                   288.0         1
                                                   349.0         1
                                                   618.0         1
                                                   732.0         1
                                                   1162.0        1
                                                   1232.0        1
                                                   1314.0        1
                                                   1337.0        1
                                                   1354.0        1
        001055fc0bb67f71e8fa17002342b256a30254cd   124.0         1
                                                   254.0         1
                                                   390.0         1
                                                   1386.0        1
        00148e4911c7e04eeff8def7bbbdaf1c59c2c621   258.0         1
                                                   932.0         1
                                                   1386.0        1
```

```
001a852ecbd6cc12ab77a785efa137b2646505fe    232.0     1
                                            349.0     1
                                            593.0     1
                                            957.0     1
                                           1364.0     2
001fc95b90da5c3cb12c501d201a915e4f093290    379.0     1
                                           1364.0     1
0042719415c4fca7d30bd2d4e9d17c5fc570de13     20.0     1
                                           1060.0     1
00772abe2d0b269b2336fc27f0f4d7cb1d2b65d7    732.0     1
                                           1427.0     2
                                                     ..
ffe3d0543c9046d35c2ee3724ea9d774dff98a32    617.0     1
                                            701.0     1
                                            727.0     1
                                            782.0     1
                                            784.0     1
                                            878.0     1
                                            943.0     1
                                            986.0     1
                                           1047.0     1
                                           1162.0     1
                                           1165.0     1
                                           1314.0     2
                                           1360.0     2
                                           1386.0     1
                                           1422.0     3
                                           1425.0     1
                                           1427.0     1
fff9fc3ec67bd18ed57a34ed1e67410942c4cd81    116.0     1
                                            232.0     1
                                            268.0     2
                                            525.0     1
                                            684.0     3
                                            962.0     1
                                           1431.0     1
fffb93a166547448a0ff0232558118d59395fecd    329.0     1
                                            981.0     1
                                           1304.0     1
                                           1305.0     8
                                           1430.0     1
                                           1437.0     1
Length: 33669, dtype: int64
```

- Number of articles a user interacts with.

```
In [9]: user_interact_unique = df.groupby(['email'])['article_id'].nunique()
        user_interact_unique
```

7

```
Out[9]: email
        0000b6387a0366322d7fbfc6434af145adf7fed1       12
        001055fc0bb67f71e8fa17002342b256a30254cd        4
        00148e4911c7e04eeff8def7bbbdaf1c59c2c621        3
        001a852ecbd6cc12ab77a785efa137b2646505fe        5
        001fc95b90da5c3cb12c501d201a915e4f093290        2
        0042719415c4fca7d30bd2d4e9d17c5fc570de13        2
        00772abe2d0b269b2336fc27f0f4d7cb1d2b65d7        2
        008ba1d5b4ebf54babf516a2d5aa43e184865da5       10
        008ca24b82c41d513b3799d09ae276d37f92ce72        1
        008dfc7a327b5186244caec48e0ab61610a0c660       10
        009af4e0537378bf8e8caf0ad0e2994f954d822e        1
        00bda305223d05f6df5d77de41abd2a0c7d895fe        4
        00c2d5190e8c6b821b0e3848bf56f6e47e428994        3
        00ced21f957bbcee5edf7b107b2bd05628b04774        4
        00d9337ecd5f70fba1c4c7a78e21b3532e0112c4        1
        00e524e4f13137a6fac54f9c71d7769c6507ecde        8
        00f8341cbecd6af00ba8c78b3bb6ec49adf83248        2
        00f946b14100f0605fa25089437ee9486378872c        1
        01041260c97ab9221d923b0a2c525437f148d589        2
        0108ce3220657a9a89a85bdec959b0f2976dd51c        3
        011455e91a24c1fb815a4deac6b6eaf5ad16819e        9
        01198c58d684d79c9026abe355cfb532cb524dc5        1
        011ae4de07ffb332b0f51c155a35c23c80294962       29
        011fcfb582be9534e9a275336f7e7c3717100381        4
        0129dfcdb701b6e1d309934be6393004c6683a2d       12
        01327bbc4fd7bfe8ad62e599453d2876b928e725        3
        01455f0ab0a5a22a93d94ad35f6e78431aa90625        6
        014dedab269f1453c647598c92a3fa37b39eed97        2
        014e4fe6e6c5eb3fe5ca0b16c16fb4599df6375c        1
        01560f88312a91894d254e6406c25df19f0ad5e8        9
                                                       ..
        fe5396e3762c36767c9c915f7ed1731691d7e4b4        1
        fe5480ff15f0ac51eeb2314a192351f168d7aad7        1
        fe56a49b62752708ed2f6e30677c57881f7b78d1       10
        fe5885b80e91be887510a0b6dd04e011178d6364        3
        fe5f9d7528518e00b0a73c7a3994afc335496961        3
        fe66aa534c7824eca663b84b99a437a98a9b026e        2
        fe69c72c964a8346dbc7763309c4e07d818d360f        2
        fe88d1f683f308b32fb3d7554f007cc55cc48df5        1
        fe8c1cb974e39d8ea8c005044e927b3f0de8acd0        2
        fe90d98b0287090fe8e653bafba6ed3eff19331e        1
        fe9327be39fd457df70e83d3fc8cba9b8b3f95b1        1
        feaea388105a4ccc48795b191bbf0c26a23b1356        4
        fef0c6be3a2ed226e1fb8a811b0ee68a389f6f3c       11
        fef28e45f7217026b2684d1783a2e18b061bdffb        3
        fef3bc88def1aa787c99957ded7d5b2c0edc040e        3
        ff27ffd93e21154b8a9cf2722f2cc0f75dc39eff        1
```

8

```
ff288722b76eba5209cdbf9158c6dfbf229b9129      1
ff452614b91f4c9bd965150b1a82e7bf18f59334      2
ff4d3e1c359cfbb73bcae07fa1eb62c45da2b161      3
ff55d0c0b2a4f56aae87c2a21afb7070ab34383d      1
ff6e82c763fe2443643e48a03e239eb635f406dc     13
ff7a0f59ba022102ad22981141a7182c4d8273c3      5
ff833869969184d86f870f98405e7988eccc2309      9
ff979e07f9d906a32ba35a9b75fd9585f6306dbc     15
ffaefa3a1bc2d074d9a14c9924d4e67a46c35410      1
ffc6cfa435937ca0df967b44e9178439d04e3537      1
ffc96f8fbb35aac4cb0029332b0fc78e7766bb5d      2
ffe3d0543c9046d35c2ee3724ea9d774dff98a32     27
fff9fc3ec67bd18ed57a34ed1e67410942c4cd81      7
fffb93a166547448a0ff0232558118d59395fecd      6
Name: article_id, Length: 5148, dtype: int64
```

- Visualize the distribution of how many articles a user interacts with

```
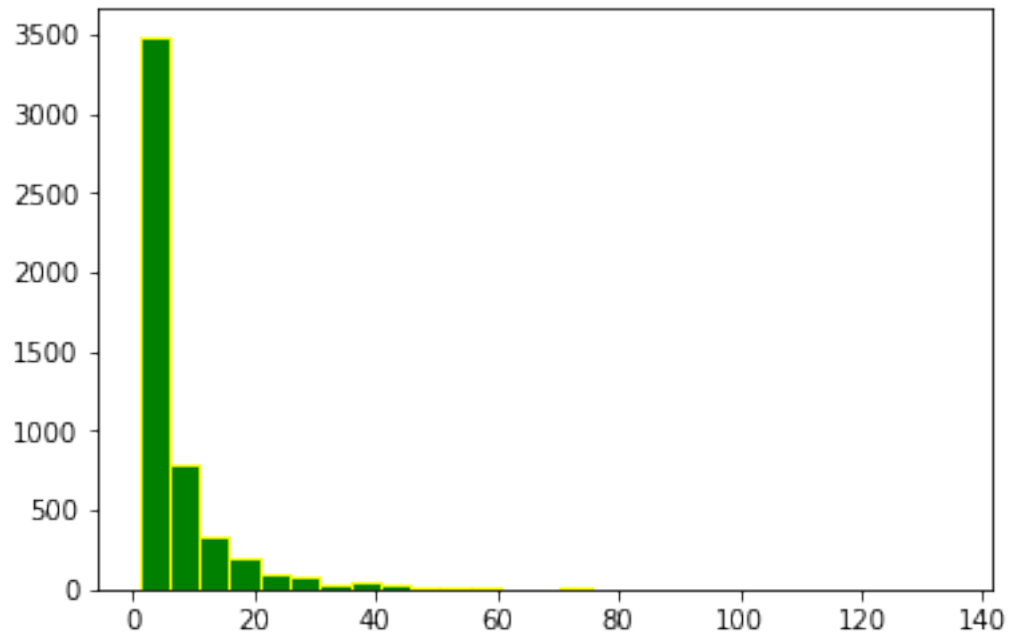In [10]: user_interact_unique.max()

Out[10]: 135

In [11]: # data visualization
         plt.hist(user_interact_unique, bins = 27, ec="yellow", fc="green") #bins =27 for bin_si

Out[11]: (array([  3.48900000e+03,    7.84000000e+02,    3.38000000e+02,
                  1.92000000e+02,    1.02000000e+02,    8.10000000e+01,
                  3.30000000e+01,    4.00000000e+01,    2.30000000e+01,
                  1.20000000e+01,    1.30000000e+01,    1.50000000e+01,
                  4.00000000e+00,    4.00000000e+00,    1.00000000e+01,
                  0.00000000e+00,    0.00000000e+00,    0.00000000e+00,
                  0.00000000e+00,    5.00000000e+00,    1.00000000e+00,
                  0.00000000e+00,    0.00000000e+00,    0.00000000e+00,
                  0.00000000e+00,    0.00000000e+00,    2.00000000e+00]),
          array([  1.        ,    5.96296296,   10.92592593,   15.88888889,
                  20.85185185,   25.81481481,   30.77777778,   35.74074074,
                  40.7037037 ,   45.66666667,   50.62962963,   55.59259259,
                  60.55555556,   65.51851852,   70.48148148,   75.44444444,
                  80.40740741,   85.37037037,   90.33333333,   95.2962963 ,
                 100.25925926,  105.22222222,  110.18518519,  115.14814815,
                 120.11111111,  125.07407407,  130.03703704,  135.        ]),
          <a list of 27 Patch objects>)
```

9

using Seaborn for seeing better distribution

```
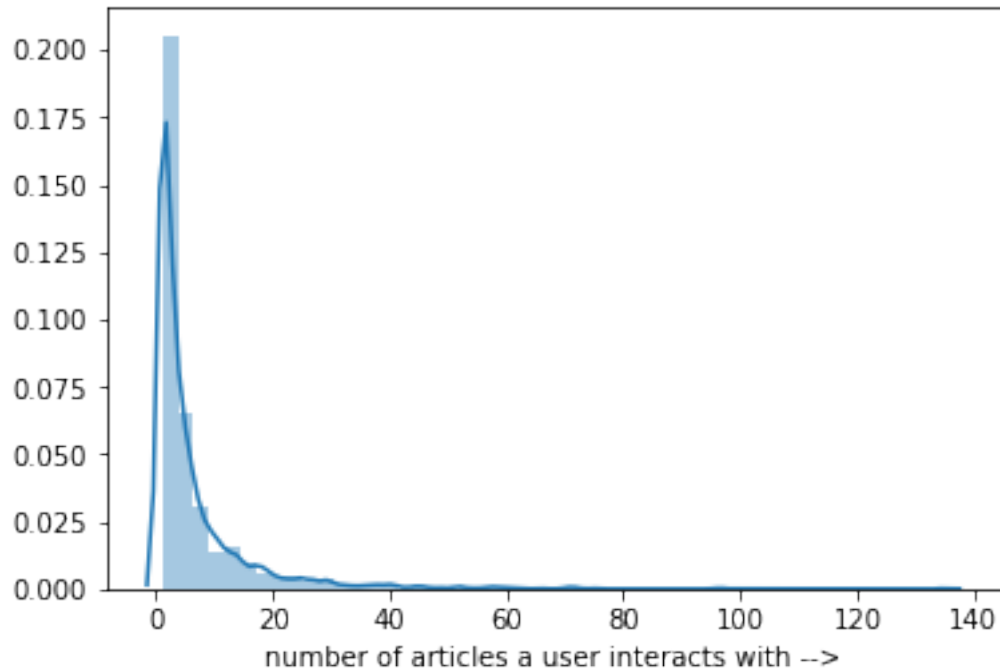In [12]: sns.__version__ # don't support histplot, only can support distplot

Out[12]: '0.8.1'

In [13]: x1 = pd.Series(user_interact_unique, name="number of articles a user interacts with -->
         sns.distplot(x1)

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7f44bf33b240>
```

- Cumulative distribution statistics of "how many articles a user interacts with" (use Mathplotlib because seaborn ver 0.8.1 doesnot support histplot)

```
In [14]: plt.hist(user_interact_unique, bins = 100, ec="yellow", fc="green", cumulative = True)

Out[14]: (array([ 2309.,  2833.,  3489.,  3735.,  3912.,  4159.,  4273.,  4353.,
                  4495.,  4564.,  4611.,  4693.,  4738.,  4776.,  4822.,  4844.,
                  4861.,  4905.,  4920.,  4940.,  4971.,  4986.,  4993.,  5008.,
                  5013.,  5019.,  5034.,  5043.,  5049.,  5067.,  5069.,  5071.,
                  5082.,  5087.,  5088.,  5093.,  5094.,  5096.,  5104.,  5106.,
                  5107.,  5112.,  5118.,  5120.,  5124.,  5125.,  5126.,  5126.,
                  5128.,  5128.,  5128.,  5130.,  5138.,  5138.,  5138.,  5140.,
                  5140.,  5140.,  5140.,  5140.,  5140.,  5140.,  5140.,  5140.,
                  5140.,  5140.,  5140.,  5140.,  5140.,  5140.,  5142.,  5144.,
                  5144.,  5145.,  5146.,  5146.,  5146.,  5146.,  5146.,  5146.,
                  5146.,  5146.,  5146.,  5146.,  5146.,  5146.,  5146.,  5146.,
                  5146.,  5146.,  5146.,  5146.,  5146.,  5146.,  5146.,  5146.,
                  5146.,  5146.,  5146.,  5148.]),
          array([   1.  ,    2.34,    3.68,    5.02,    6.36,    7.7 ,    9.04,
                   10.38,   11.72,   13.06,   14.4 ,   15.74,   17.08,   18.42,
                   19.76,   21.1 ,   22.44,   23.78,   25.12,   26.46,   27.8 ,
                   29.14,   30.48,   31.82,   33.16,   34.5 ,   35.84,   37.18,
                   38.52,   39.86,   41.2 ,   42.54,   43.88,   45.22,   46.56,
                   47.9 ,   49.24,   50.58,   51.92,   53.26,   54.6 ,   55.94,
                   57.28,   58.62,   59.96,   61.3 ,   62.64,   63.98,   65.32,
```

11

```
        66.66,    68.  ,   69.34,   70.68,   72.02,   73.36,   74.7 ,
        76.04,    77.38,   78.72,   80.06,   81.4 ,   82.74,   84.08,
        85.42,    86.76,   88.1 ,   89.44,   90.78,   92.12,   93.46,
        94.8 ,    96.14,   97.48,   98.82,  100.16,  101.5 ,  102.84,
       104.18,   105.52,  106.86,  108.2 ,  109.54,  110.88,  112.22,
       113.56,   114.9 ,  116.24,  117.58,  118.92,  120.26,  121.6 ,
       122.94,   124.28,  125.62,  126.96,  128.3 ,  129.64,  130.98,
       132.32,   133.66,  135.  ]),
 <a list of 100 Patch objects>)
```


- Calculate the descriptive parameters: the median value of "how many articles a user interacts with" distribution

```
In [15]: median_val = np.median(user_interact_unique)
         median_val
```

Out[15]: 3.0

- Calculate the descriptive parameters: the max value of "how many articles a user interacts with"

```
In [16]: df.groupby(['email']).size().max()
```

Out[16]: 364

- Fill in the descriptive parameters

```
In [17]: # Fill in the median and maximum number of user_article interactios below

         median_val = 3.0 # 50% of individuals interact with 3.0 number of articles or fewer.
         max_views_by_user = 364 # The maximum number of user-article interactions by any 1 user
```

2. Explore and remove duplicate articles from the **df_content** dataframe.

```
In [18]: # Check the number of items in df_content (articles)
         df_content.shape[0]

Out[18]: 1056

In [19]: # Find and explore duplicate articles
         df_content[df_content.duplicated(subset=['article_id'], keep='first') == True]

Out[19]:                                                    doc_body  \
         365   Follow Sign in / Sign up Home About Insight Da...
         692   Homepage Follow Sign in / Sign up Homepage * H...
         761   Homepage Follow Sign in Get started Homepage *...
         970   This video shows you how to construct queries ...
         971   Homepage Follow Sign in Get started * Home\r\n...


                                                doc_description  \
         365   During the seven-week Insight Data Engineering...
         692   One of the earliest documented catalogs was co...
         761   Todays world of data science leverages data f...
         970   This video shows you how to construct queries ...
         971   If you are like most data scientists, you are ...


                                            doc_full_name doc_status   article_id
         365                        Graph-based machine learning      Live           50
         692  How smart catalogs can turn the big data flood...      Live          221
         761  Using Apache Spark as a parallel processing fr...      Live          398
         970                            Use the Primary Index      Live          577
         971  Self-service data preparation with IBM Data Re...      Live          232

In [20]: # Remove any rows that have the same article_id - only keep the first
         df_content_drop = df_content.drop_duplicates(subset=['article_id'], keep='first') #drop

In [21]: df_content_drop.shape[0] #check the number of articles after drop the duplicated rows

Out[21]: 1051
```

3. Use the cells below to find:
   a. The number of unique articles that have an interaction with a user.
b. The number of unique articles in the dataset (whether they have any interactions or not). **c.** The number of unique users in the dataset. (excluding null values) **d.** The number of user-article interactions in the dataset.

```
In [22]: # The number of unique articles that have at least one interaction
         article_interact_unique.shape[0]

Out[22]: 714

In [23]: # The number of unique articles on the IBM platform
         df_content_drop.shape[0]

Out[23]: 1051

In [24]: # The number of unique users
         user_interact_unique.shape[0]

Out[24]: 5148

In [25]: # The number of user-article interactions
         df.shape[0]

Out[25]: 45993

In [26]: unique_articles = article_interact_unique.shape[0] # The number of unique articles that
         total_articles = df_content_drop.shape[0] # The number of unique articles on the IBM pl
         unique_users = user_interact_unique.shape[0] # The number of unique users
         user_article_interactions = df.shape[0] # The number of user-article interactions
```

4. Use the cells below to find the most viewed **article_id**, as well as how often it was viewed. After talking to the company leaders, the email_mapper function was deemed a reasonable way to map users to ids. There were a small number of null values, and it was found that all of these null values likely belonged to a single user (which is how they are stored using the function below).

```
In [27]: # The most viewed article in the dataset was viewed how many times?
         df.groupby(['article_id']).size().max()

Out[27]: 937

In [28]: article_interactions = df.groupby(['article_id']).size()
         article_interactions

Out[28]: article_id
         0.0        14
         2.0        58
         4.0        13
         8.0        85
         9.0        10
         12.0      157
         14.0       89
         15.0       26
         16.0       61
         18.0       78
         20.0      249
```

14

```
25.0       15
26.0       89
28.0       42
29.0       75
30.0       17
32.0       64
33.0      141
34.0       93
36.0       18
39.0       68
40.0       70
43.0      460
48.0       11
50.0       89
51.0      124
53.0      115
54.0       20
57.0      140
58.0       11
          ...
1412.0      25
1414.0       4
1415.0      11
1416.0     102
1418.0      43
1419.0       6
1420.0     113
1421.0       3
1422.0     163
1423.0     155
1424.0     131
1425.0      71
1426.0     138
1427.0     643
1428.0     120
1429.0     937
1430.0     336
1431.0     671
1432.0     340
1433.0     108
1434.0      42
1435.0     120
1436.0     481
1437.0     218
1439.0      59
1440.0      10
1441.0       8
1442.0       4
```

```
          1443.0     22
          1444.0      5
          Length: 714, dtype: int64

In [29]: article_interactions[article_interactions == article_interactions.max()]

Out[29]: article_id
         1429.0    937
         dtype: int64

In [30]: most_viewed_article_id = '1429.0' # The most viewed article in the dataset as a string
         max_views = 937 # The most viewed article in the dataset was viewed how many times?

In [31]: ## No need to change the code here - this will be helpful for later parts of the notebo
         # Run this cell to map the user email to a user_id column and remove the email column

         def email_mapper():
             coded_dict = dict()
             cter = 1
             email_encoded = []

             for val in df['email']:
                 if val not in coded_dict:
                     coded_dict[val] = cter
                     cter+=1

                 email_encoded.append(coded_dict[val])
             return email_encoded

         email_encoded = email_mapper()
         del df['email']
         df['user_id'] = email_encoded

         # show header
         df.head()

Out[31]:    article_id                                          title  user_id
         0      1430.0  using pixiedust for fast, flexible, and easier...        1
         1      1314.0            healthcare python streaming application demo        2
         2      1429.0             use deep learning for image classification        3
         3      1338.0             ml optimization using cognitive assistant        4
         4      1276.0             deploy your python model as a restful api        5

In [32]: ## If you stored all your results in the variable names above,
         ## you shouldn't need to change anything in this cell

         sol_1_dict = {
             '`50% of individuals have _____ or fewer interactions.`': median_val,
             '`The total number of user-article interactions in the dataset is _____.`': user_a
```

16

```
                '`The maximum number of user-article interactions by any 1 user is _____.`': max_v
                '`The most viewed article in the dataset was viewed _____ times.`': max_views,
                '`The article_id of the most viewed article is _____.`': most_viewed_article_id,
                '`The number of unique articles that have at least 1 rating _____.`': unique_artic
                '`The number of unique users in the dataset is _____`': unique_users,
                '`The number of unique articles on the IBM platform`': total_articles
        }

        # Test your dictionary against the solution
        t.sol_1_test(sol_1_dict)

It looks like you have everything right here! Nice job!
```

1.1.2   Part II: Rank-Based Recommendations

Unlike in the earlier lessons, we don't actually have ratings for whether a user liked an article or
not. We only know that a user has interacted with an article. In these cases, the popularity of an
article can really only be based on how often an article was interacted with.

   1. Fill in the function below to return the **n** top articles ordered with most interactions as the
top. Test your function using the tests below.

   • Draft the functions with n = 10

```
In [33]: df.groupby(['article_id']).size()

Out[33]: article_id
         0.0       14
         2.0       58
         4.0       13
         8.0       85
         9.0       10
         12.0     157
         14.0      89
         15.0      26
         16.0      61
         18.0      78
         20.0     249
         25.0      15
         26.0      89
         28.0      42
         29.0      75
         30.0      17
         32.0      64
         33.0     141
         34.0      93
         36.0      18
         39.0      68
         40.0      70
```

```
          43.0       460
          48.0        11
          50.0        89
          51.0       124
          53.0       115
          54.0        20
          57.0       140
          58.0        11
                     ...
          1412.0      25
          1414.0       4
          1415.0      11
          1416.0     102
          1418.0      43
          1419.0       6
          1420.0     113
          1421.0       3
          1422.0     163
          1423.0     155
          1424.0     131
          1425.0      71
          1426.0     138
          1427.0     643
          1428.0     120
          1429.0     937
          1430.0     336
          1431.0     671
          1432.0     340
          1433.0     108
          1434.0      42
          1435.0     120
          1436.0     481
          1437.0     218
          1439.0      59
          1440.0      10
          1441.0       8
          1442.0       4
          1443.0      22
          1444.0       5
          Length: 714, dtype: int64

In [34]: article_interactions_nlargest = df.groupby(['article_id']).size().nlargest(10)
         article_interactions_nlargest

Out[34]: article_id
         1429.0     937
         1330.0     927
         1431.0     671
```

```
          1427.0     643
          1364.0     627
          1314.0     614
          1293.0     572
          1170.0     565
          1162.0     512
          1304.0     483
          dtype: int64

In [35]: article_interactions_nlargest.index

Out[35]: Float64Index([1429.0, 1330.0, 1431.0, 1427.0, 1364.0, 1314.0, 1293.0, 1170.0,
                        1162.0, 1304.0],
                       dtype='float64', name='article_id')

In [36]: article_interactions_nlargest.index.map(str)

Out[36]: Index(['1429.0', '1330.0', '1431.0', '1427.0', '1364.0', '1314.0', '1293.0',
                '1170.0', '1162.0', '1304.0'],
               dtype='object', name='article_id')

In [37]: article_interactions_nlargest.index.map(str).tolist()

Out[37]: ['1429.0',
          '1330.0',
          '1431.0',
          '1427.0',
          '1364.0',
          '1314.0',
          '1293.0',
          '1170.0',
          '1162.0',
          '1304.0']

In [38]: top_articles_index = article_interactions_nlargest.index.tolist()
         top_articles_index

Out[38]: [1429.0,
          1330.0,
          1431.0,
          1427.0,
          1364.0,
          1314.0,
          1293.0,
          1170.0,
          1162.0,
          1304.0]

In [39]: df.head()
```

```
Out[39]:      article_id                                           title  user_id
        0       1430.0  using pixiedust for fast, flexible, and easier...       1
        1       1314.0       healthcare python streaming application demo       2
        2       1429.0           use deep learning for image classification       3
        3       1338.0           ml optimization using cognitive assistant       4
        4       1276.0           deploy your python model as a restful api       5

In [40]: df_drop = df[['article_id','title']]
        df_drop.head()

Out[40]:      article_id                                           title
        0       1430.0  using pixiedust for fast, flexible, and easier...
        1       1314.0       healthcare python streaming application demo
        2       1429.0           use deep learning for image classification
        3       1338.0           ml optimization using cognitive assistant
        4       1276.0           deploy your python model as a restful api

In [41]: df_drop = df_drop.drop_duplicates(subset=['article_id'], keep='first')
        df_drop.head()

Out[41]:      article_id                                           title
        0       1430.0  using pixiedust for fast, flexible, and easier...
        1       1314.0       healthcare python streaming application demo
        2       1429.0           use deep learning for image classification
        3       1338.0           ml optimization using cognitive assistant
        4       1276.0           deploy your python model as a restful api

In [42]: df_drop.shape[0]

Out[42]: 714

In [43]: df_drop[df_drop['article_id'].isin(top_articles_index)]

Out[43]:       article_id                                           title
        1        1314.0       healthcare python streaming application demo
        2        1429.0           use deep learning for image classification
        14       1170.0              apache spark lab, part 1: basic concepts
        29       1364.0  predicting churn with the spss random tree alg...
        31       1162.0          analyze energy consumption in buildings
        37       1431.0                     visualize car data with brunel
        42       1427.0  use xgboost, scikit-learn & ibm watson machine...
        56       1304.0  gosales transactions for logistic regression m...
        66       1330.0        insights from new york car accident reports
        154      1293.0  finding optimal locations of new store using d...

In [44]: top_articles_id = df.groupby(['article_id']).size().nlargest(10).index.map(str).tolist(
        top_articles_id
```

```
Out[44]: ['1429.0',
          '1330.0',
          '1431.0',
          '1427.0',
          '1364.0',
          '1314.0',
          '1293.0',
          '1170.0',
          '1162.0',
          '1304.0']
```

- Fill in the functions

```
In [45]: def get_top_articles(n, df=df):
             '''
             INPUT:
             n - (int) the number of top articles to return
             df - (pandas dataframe) df as defined at the top of the notebook

             OUTPUT:
             top_articles - (list) A list of the top 'n' article titles

             '''
             # Your code here
             article_interactions_nlargest = df.groupby(['article_id']).size().nlargest(n)
             top_articles_index = article_interactions_nlargest.index.tolist()
             df_drop = df[['article_id','title']]
             df_drop = df_drop.drop_duplicates(subset=['article_id'], keep='first')
             top_articles = df_drop[df_drop['article_id'].isin(top_articles_index)]['title']

             return top_articles # Return the top article titles from df (not df_content)

         def get_top_article_ids(n, df=df):
             '''
             INPUT:
             n - (int) the number of top articles to return
             df - (pandas dataframe) df as defined at the top of the notebook

             OUTPUT:
             top_articles - (list) A list of the top 'n' article titles

             '''
             # Your code here
             top_articles = df.groupby(['article_id']).size().nlargest(n).index.map(str).tolist(

             return top_articles # Return the top article ids

In [46]: print(get_top_articles(10))
         print(get_top_article_ids(10))
```

21

```
1              healthcare python streaming application demo
2                 use deep learning for image classification
14                   apache spark lab, part 1: basic concepts
29        predicting churn with the spss random tree alg...
31                  analyze energy consumption in buildings
37                          visualize car data with brunel
42        use xgboost, scikit-learn & ibm watson machine...
56        gosales transactions for logistic regression m...
66              insights from new york car accident reports
154      finding optimal locations of new store using d...
Name: title, dtype: object
['1429.0', '1330.0', '1431.0', '1427.0', '1364.0', '1314.0', '1293.0', '1170.0', '1162.0', '1304
```

```
In [47]: # Test your function by returning the top 5, 10, and 20 articles
         top_5 = get_top_articles(5)
         top_10 = get_top_articles(10)
         top_20 = get_top_articles(20)

         # Test each of your three lists from above
         t.sol_2_test(get_top_articles)
```

```
Your top_5 looks like the solution list! Nice job.
Your top_10 looks like the solution list! Nice job.
Your top_20 looks like the solution list! Nice job.
```

1.1.3 Part III: User-User Based Collaborative Filtering

1. Use the function below to reformat the **df** dataframe to be shaped with users as the rows and articles as the columns.

- Each **user** should only appear in each **row** once.

- Each **article** should only show up in one **column**.

- **If a user has interacted with an article, then place a 1 where the user-row meets for that article-column**. It does not matter how many times a user has interacted with the article, all entries where a user has interacted with an article should be a 1.

- **If a user has not interacted with an item, then place a zero where the user-row meets for that article-column**.

Use the tests to make sure the basic structure of your matrix matches what is expected by the solution.

a. Draft the function

```
In [48]: df.head()
```

```
Out[48]:    article_id                                              title  user_id
         0     1430.0  using pixiedust for fast, flexible, and easier...        1
         1     1314.0          healthcare python streaming application demo        2
         2     1429.0             use deep learning for image classification        3
         3     1338.0             ml optimization using cognitive assistant        4
         4     1276.0             deploy your python model as a restful api        5

In [49]: df['article_id'].shape[0]

Out[49]: 45993

In [50]: df_article_col = df.drop_duplicates(subset=['article_id'], keep='first')['article_id']
         df_article_col.head()

Out[50]: 0    1430.0
         1    1314.0
         2    1429.0
         3    1338.0
         4    1276.0
         Name: article_id, dtype: float64

In [51]: df_article_col.shape[0]

Out[51]: 714

In [52]: df_user_row = df.drop_duplicates(subset=['user_id'], keep='first')['user_id']
         df_user_row.head()

Out[52]: 0    1
         1    2
         2    3
         3    4
         4    5
         Name: user_id, dtype: int64

In [53]: df_user_row.shape[0]

Out[53]: 5149

In [54]: interactions = df.groupby(['user_id','article_id']).size()
         interactions.head()

Out[54]: user_id  article_id
         1        43.0          1
                  109.0         1
                  151.0         1
                  268.0         1
                  310.0         2
         dtype: int64
```

```
In [55]: df_interactions = df.groupby(['user_id','article_id']).size().index.to_frame()
         df_interactions.head()

Out[55]:                        user_id  article_id
         user_id article_id
         1       43.0                 1        43.0
                 109.0                1       109.0
                 151.0                1       151.0
                 268.0                1       268.0
                 310.0                1       310.0

In [56]: df_interactions['interaction'] = interactions
         df_interactions

Out[56]:                        user_id  article_id  interaction
         user_id article_id
         1       43.0                 1        43.0            1
                 109.0                1       109.0            1
                 151.0                1       151.0            1
                 268.0                1       268.0            1
                 310.0                1       310.0            2
                 329.0                1       329.0            1
                 346.0                1       346.0            1
                 390.0                1       390.0            1
                 494.0                1       494.0            1
                 525.0                1       525.0            1
                 585.0                1       585.0            2
                 626.0                1       626.0            1
                 668.0                1       668.0            2
                 732.0                1       732.0            1
                 768.0                1       768.0            1
                 910.0                1       910.0            1
                 968.0                1       968.0            1
                 981.0                1       981.0            1
                 1052.0               1      1052.0            2
                 1170.0               1      1170.0            2
                 1183.0               1      1183.0            2
                 1185.0               1      1185.0            2
                 1232.0               1      1232.0            1
                 1293.0               1      1293.0            1
                 1305.0               1      1305.0            1
                 1363.0               1      1363.0            2
                 1368.0               1      1368.0            1
                 1391.0               1      1391.0            1
                 1400.0               1      1400.0            1
                 1406.0               1      1406.0            2
         ...                        ...         ...          ...
         5143    485.0             5143       485.0            1
```

```
                     495.0       5143       495.0           1
                     588.0       5143       588.0           2
                    1324.0       5143      1324.0           1
                    1330.0       5143      1330.0           2
                    1343.0       5143      1343.0           1
                    1354.0       5143      1354.0           1
                    1360.0       5143      1360.0           1
                    1398.0       5143      1398.0           3
                    1400.0       5143      1400.0           3
                    1409.0       5143      1409.0           1
                    1430.0       5143      1430.0           2
                    1431.0       5143      1431.0           1
                    1436.0       5143      1436.0           1
            5144     270.0       5144       270.0           1
            5145      20.0       5145        20.0           1
                     138.0       5145       138.0           1
                     962.0       5145       962.0           2
                    1165.0       5145      1165.0           1
                    1305.0       5145      1305.0           1
            5146     142.0       5146       142.0           1
                    1125.0       5146      1125.0           1
                    1157.0       5146      1157.0           1
                    1282.0       5146      1282.0           1
                    1324.0       5146      1324.0           3
                    1394.0       5146      1394.0           1
                    1416.0       5146      1416.0           1
            5147     233.0       5147       233.0           1
            5148    1160.0       5148      1160.0           1
            5149      16.0       5149        16.0           1

            [33682 rows x 3 columns]

In [57]: user_article_matrix = df_interactions.pivot(*df_interactions.columns).fillna(0)
         user_article_matrix

Out[57]: article_id   0.0      2.0      4.0      8.0      9.0      12.0     14.0     15.0     \
         user_id
         1            0.0      0.0      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      0.0      0.0
         3            0.0      0.0      0.0      0.0      0.0      1.0      0.0      0.0
         4            0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
         5            0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
         6            0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
         7            0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
         8            0.0      0.0      0.0      0.0      0.0      0.0      1.0      0.0
         9            0.0      0.0      0.0      0.0      0.0      0.0      1.0      0.0
         10           0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
         11           0.0      0.0      0.0      0.0      0.0      2.0      0.0      0.0
```

12	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
13	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
22	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
23	0.0	3.0	0.0	0.0	0.0	7.0	1.0	0.0
24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
5120	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5121	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5122	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
5123	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5124	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5125	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5126	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5127	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5128	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5129	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5130	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5131	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5132	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5133	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5134	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5135	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5136	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5137	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5138	0.0	0.0	0.0	0.0	0.0	2.0	1.0	0.0
5139	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
5140	0.0	3.0	0.0	0.0	0.0	1.0	0.0	0.0
5141	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5142	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
5143	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5144	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5145	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5146	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5147	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
5148          0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
5149          0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0

article_id  16.0     18.0      ...    1434.0   1435.0   1436.0   1437.0   1439.0   \
user_id                        ...
1             0.0      0.0      ...      0.0      0.0      1.0      0.0      1.0
2             0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
3             0.0      0.0      ...      0.0      0.0      1.0      0.0      0.0
4             0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
5             0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
6             0.0      0.0      ...      0.0      0.0      1.0      0.0      0.0
7             0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
8             0.0      0.0      ...      0.0      0.0      4.0      0.0      0.0
9             1.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
10            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
11            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
12            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
13            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
14            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
15            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
16            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
17            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
18            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
19            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
20            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
21            0.0      0.0      ...      0.0      0.0      2.0      3.0      0.0
22            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
23            1.0      0.0      ...      0.0      0.0      6.0      0.0      1.0
24            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
25            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
26            0.0      0.0      ...      0.0      0.0      4.0      0.0      0.0
27            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
28            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
29            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
30            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
...           ...      ...      ...      ...      ...      ...      ...      ...
5120          0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
5121          0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
5122          0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
5123          0.0      0.0      ...      0.0      0.0      1.0      1.0      0.0
5124          0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
5125          0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
5126          0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
5127          0.0      0.0      ...      1.0      0.0      0.0      0.0      0.0
5128          0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
5129          0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
5130          0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
5131          0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
```

5132	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5133	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5134	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5135	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5136	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5137	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5138	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5139	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5140	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5141	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5142	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5143	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0
5144	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5145	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5146	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5147	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5148	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5149	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0

article_id	1440.0	1441.0	1442.0	1443.0	1444.0
user_id					
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	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0
24	1.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0

```
28              0.0     0.0     0.0     0.0     0.0
29              0.0     0.0     0.0     0.0     0.0
30              0.0     0.0     0.0     0.0     0.0
...             ...     ...     ...     ...     ...
5120            0.0     0.0     0.0     0.0     0.0
5121            0.0     0.0     0.0     0.0     0.0
5122            0.0     0.0     0.0     0.0     0.0
5123            0.0     0.0     0.0     0.0     0.0
5124            0.0     0.0     0.0     0.0     0.0
5125            0.0     0.0     0.0     0.0     0.0
5126            0.0     0.0     0.0     0.0     0.0
5127            0.0     0.0     0.0     0.0     0.0
5128            0.0     0.0     0.0     0.0     0.0
5129            0.0     0.0     0.0     0.0     0.0
5130            0.0     0.0     0.0     0.0     0.0
5131            0.0     0.0     0.0     0.0     0.0
5132            0.0     0.0     0.0     0.0     0.0
5133            0.0     0.0     0.0     0.0     0.0
5134            0.0     0.0     0.0     0.0     0.0
5135            0.0     0.0     0.0     0.0     0.0
5136            0.0     0.0     0.0     0.0     0.0
5137            0.0     0.0     0.0     0.0     0.0
5138            0.0     0.0     0.0     0.0     0.0
5139            0.0     0.0     0.0     0.0     0.0
5140            0.0     0.0     0.0     0.0     0.0
5141            0.0     0.0     0.0     0.0     0.0
5142            0.0     0.0     0.0     0.0     0.0
5143            0.0     0.0     0.0     0.0     0.0
5144            0.0     0.0     0.0     0.0     0.0
5145            0.0     0.0     0.0     0.0     0.0
5146            0.0     0.0     0.0     0.0     0.0
5147            0.0     0.0     0.0     0.0     0.0
5148            0.0     0.0     0.0     0.0     0.0
5149            0.0     0.0     0.0     0.0     0.0

[5149 rows x 714 columns]

In [58]: user_article_matrix [user_article_matrix > 1] = 1
         user_article_matrix

Out[58]: article_id  0.0     2.0     4.0     8.0     9.0     12.0    14.0    15.0    \
         user_id
         1           0.0     0.0     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     0.0     0.0
         3           0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0
         4           0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
         5           0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
         6           0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
```

7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
13	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
22	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
23	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0
24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
5120	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5121	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5122	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
5123	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5124	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5125	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5126	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5127	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5128	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5129	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5130	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5131	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5132	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5133	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5134	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5135	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5136	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5137	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5138	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0
5139	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
5140	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
5141	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5142	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

```
5143          0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
5144          0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
5145          0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
5146          0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
5147          0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
5148          0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
5149          0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0

article_id  16.0     18.0      ...    1434.0   1435.0   1436.0   1437.0   1439.0   \
user_id                        ...
1             0.0      0.0      ...      0.0      0.0      1.0      0.0      1.0
2             0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
3             0.0      0.0      ...      0.0      0.0      1.0      0.0      0.0
4             0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
5             0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
6             0.0      0.0      ...      0.0      0.0      1.0      0.0      0.0
7             0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
8             0.0      0.0      ...      0.0      0.0      1.0      0.0      0.0
9             1.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
10            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
11            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
12            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
13            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
14            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
15            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
16            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
17            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
18            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
19            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
20            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
21            0.0      0.0      ...      0.0      0.0      1.0      1.0      0.0
22            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
23            1.0      0.0      ...      0.0      0.0      1.0      0.0      1.0
24            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
25            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
26            0.0      0.0      ...      0.0      0.0      1.0      0.0      0.0
27            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
28            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
29            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
30            0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
...           ...      ...      ...      ...      ...      ...      ...      ...
5120          0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
5121          0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
5122          0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
5123          0.0      0.0      ...      0.0      0.0      1.0      1.0      0.0
5124          0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
5125          0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
5126          0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
```

5127	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0
5128	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5129	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5130	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5131	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5132	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5133	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5134	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5135	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5136	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5137	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5138	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5139	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5140	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5141	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5142	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5143	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0
5144	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5145	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5146	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5147	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5148	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5149	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0

article_id	1440.0	1441.0	1442.0	1443.0	1444.0
user_id					
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	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0

```
23              0.0     0.0     0.0     0.0     0.0
24              1.0     0.0     0.0     0.0     0.0
25              0.0     0.0     0.0     0.0     0.0
26              0.0     0.0     0.0     0.0     0.0
27              0.0     0.0     0.0     0.0     0.0
28              0.0     0.0     0.0     0.0     0.0
29              0.0     0.0     0.0     0.0     0.0
30              0.0     0.0     0.0     0.0     0.0
...             ...     ...     ...     ...     ...
5120            0.0     0.0     0.0     0.0     0.0
5121            0.0     0.0     0.0     0.0     0.0
5122            0.0     0.0     0.0     0.0     0.0
5123            0.0     0.0     0.0     0.0     0.0
5124            0.0     0.0     0.0     0.0     0.0
5125            0.0     0.0     0.0     0.0     0.0
5126            0.0     0.0     0.0     0.0     0.0
5127            0.0     0.0     0.0     0.0     0.0
5128            0.0     0.0     0.0     0.0     0.0
5129            0.0     0.0     0.0     0.0     0.0
5130            0.0     0.0     0.0     0.0     0.0
5131            0.0     0.0     0.0     0.0     0.0
5132            0.0     0.0     0.0     0.0     0.0
5133            0.0     0.0     0.0     0.0     0.0
5134            0.0     0.0     0.0     0.0     0.0
5135            0.0     0.0     0.0     0.0     0.0
5136            0.0     0.0     0.0     0.0     0.0
5137            0.0     0.0     0.0     0.0     0.0
5138            0.0     0.0     0.0     0.0     0.0
5139            0.0     0.0     0.0     0.0     0.0
5140            0.0     0.0     0.0     0.0     0.0
5141            0.0     0.0     0.0     0.0     0.0
5142            0.0     0.0     0.0     0.0     0.0
5143            0.0     0.0     0.0     0.0     0.0
5144            0.0     0.0     0.0     0.0     0.0
5145            0.0     0.0     0.0     0.0     0.0
5146            0.0     0.0     0.0     0.0     0.0
5147            0.0     0.0     0.0     0.0     0.0
5148            0.0     0.0     0.0     0.0     0.0
5149            0.0     0.0     0.0     0.0     0.0

[5149 rows x 714 columns]
```

b. Fill in the function

```
In [59]: # create the user-article matrix with 1's and 0's

         def create_user_item_matrix(df):
             '''
```

```
INPUT:
df - pandas dataframe with article_id, title, user_id columns

OUTPUT:
user_item - user item matrix

Description:
Return a matrix with user ids as rows and article ids on the columns with 1 values
an article and a 0 otherwise
'''
# Fill in the function here
interactions = df.groupby(['user_id','article_id']).size()
df_interactions = df.groupby(['user_id','article_id']).size().index.to_frame()
df_interactions['interaction'] = interactions
user_item = df_interactions.pivot(*df_interactions.columns).fillna(0)
user_item [user_item > 1] = 1

return user_item # return the user_item matrix

user_item = create_user_item_matrix(df)
```

c. Test the function

```
In [60]: ## Tests: You should just need to run this cell.  Don't change the code.
         assert user_item.shape[0] == 5149, "Oops!  The number of users in the user-article matr
         assert user_item.shape[1] == 714, "Oops!  The number of articles in the user-article ma
         assert user_item.sum(axis=1)[1] == 36, "Oops!  The number of articles seen by user 1 do
         print("You have passed our quick tests!  Please proceed!")

You have passed our quick tests!  Please proceed!
```

2. Complete the function below which should take a user_id and provide an ordered list of the most similar users to that user (from most similar to least similar). The returned result should not contain the provided user_id, as we know that each user is similar to him/herself. Because the results for each user here are binary, it (perhaps) makes sense to compute similarity as the dot product of two users.

Use the tests to test your function.

a. Draft the function

Computes the similarity of every pair of users based on the dot product

```
In [61]: # check the user_item matrix and its dimensions
         user_item
```

Out[61]: article_id	0.0	2.0	4.0	8.0	9.0	12.0	14.0	15.0	\
user_id									
1	0.0	0.0	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	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	

4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
13	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
22	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
23	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0
24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
5120	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5121	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5122	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
5123	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5124	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5125	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5126	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5127	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5128	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5129	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5130	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5131	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5132	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5133	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5134	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5135	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5136	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5137	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5138	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0
5139	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

```
5140           0.0      1.0      0.0      0.0      0.0      1.0      0.0      0.0
5141           0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
5142           0.0      0.0      0.0      0.0      0.0      0.0      1.0      0.0
5143           0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
5144           0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
5145           0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
5146           0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
5147           0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
5148           0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
5149           0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0

article_id  16.0     18.0     ...    1434.0   1435.0   1436.0   1437.0   1439.0  \
user_id                       ...
1              0.0      0.0    ...      0.0      0.0      1.0      0.0      1.0
2              0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
3              0.0      0.0    ...      0.0      0.0      1.0      0.0      0.0
4              0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
5              0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
6              0.0      0.0    ...      0.0      0.0      1.0      0.0      0.0
7              0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
8              0.0      0.0    ...      0.0      0.0      1.0      0.0      0.0
9              1.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
10             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
11             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
12             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
13             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
14             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
15             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
16             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
17             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
18             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
19             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
20             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
21             0.0      0.0    ...      0.0      0.0      1.0      1.0      0.0
22             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
23             1.0      0.0    ...      0.0      0.0      1.0      0.0      1.0
24             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
25             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
26             0.0      0.0    ...      0.0      0.0      1.0      0.0      0.0
27             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
28             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
29             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
30             0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
...            ...      ...    ...      ...      ...      ...      ...      ...
5120           0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
5121           0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
5122           0.0      0.0    ...      0.0      0.0      0.0      0.0      0.0
5123           0.0      0.0    ...      0.0      0.0      1.0      1.0      0.0
```

5124	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5125	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5126	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5127	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0
5128	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5129	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5130	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5131	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5132	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5133	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5134	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5135	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5136	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5137	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5138	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5139	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5140	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5141	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5142	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5143	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0
5144	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5145	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5146	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5147	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5148	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
5149	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0

article_id	1440.0	1441.0	1442.0	1443.0	1444.0
user_id					
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	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0

```
20              0.0     0.0     0.0     0.0     0.0
21              0.0     0.0     0.0     0.0     0.0
22              0.0     0.0     0.0     0.0     0.0
23              0.0     0.0     0.0     0.0     0.0
24              1.0     0.0     0.0     0.0     0.0
25              0.0     0.0     0.0     0.0     0.0
26              0.0     0.0     0.0     0.0     0.0
27              0.0     0.0     0.0     0.0     0.0
28              0.0     0.0     0.0     0.0     0.0
29              0.0     0.0     0.0     0.0     0.0
30              0.0     0.0     0.0     0.0     0.0
...             ...     ...     ...     ...     ...
5120            0.0     0.0     0.0     0.0     0.0
5121            0.0     0.0     0.0     0.0     0.0
5122            0.0     0.0     0.0     0.0     0.0
5123            0.0     0.0     0.0     0.0     0.0
5124            0.0     0.0     0.0     0.0     0.0
5125            0.0     0.0     0.0     0.0     0.0
5126            0.0     0.0     0.0     0.0     0.0
5127            0.0     0.0     0.0     0.0     0.0
5128            0.0     0.0     0.0     0.0     0.0
5129            0.0     0.0     0.0     0.0     0.0
5130            0.0     0.0     0.0     0.0     0.0
5131            0.0     0.0     0.0     0.0     0.0
5132            0.0     0.0     0.0     0.0     0.0
5133            0.0     0.0     0.0     0.0     0.0
5134            0.0     0.0     0.0     0.0     0.0
5135            0.0     0.0     0.0     0.0     0.0
5136            0.0     0.0     0.0     0.0     0.0
5137            0.0     0.0     0.0     0.0     0.0
5138            0.0     0.0     0.0     0.0     0.0
5139            0.0     0.0     0.0     0.0     0.0
5140            0.0     0.0     0.0     0.0     0.0
5141            0.0     0.0     0.0     0.0     0.0
5142            0.0     0.0     0.0     0.0     0.0
5143            0.0     0.0     0.0     0.0     0.0
5144            0.0     0.0     0.0     0.0     0.0
5145            0.0     0.0     0.0     0.0     0.0
5146            0.0     0.0     0.0     0.0     0.0
5147            0.0     0.0     0.0     0.0     0.0
5148            0.0     0.0     0.0     0.0     0.0
5149            0.0     0.0     0.0     0.0     0.0

[5149 rows x 714 columns]
```

In [62]: # need to transpose to match the matrix dimension when calculate the dot product
         user_item.transpose()

Out[62]: user_id   1   2   3   4   5   6   7   8   9   10   ... \

```
article_id                                                                            ...
0.0          0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
2.0          0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
4.0          0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
8.0          0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
9.0          0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
12.0         0.0    0.0    1.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
14.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    1.0    1.0    0.0    ...
15.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
16.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    1.0    0.0    ...
18.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
20.0         0.0    0.0    1.0    1.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
25.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
26.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
28.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    1.0    0.0    ...
29.0         0.0    0.0    1.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
30.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
32.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    1.0    0.0    ...
33.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
34.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    1.0    0.0    0.0    ...
36.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
39.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
40.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    1.0    0.0    ...
43.0         1.0    0.0    1.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
48.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
50.0         0.0    0.0    1.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
51.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
53.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
54.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
57.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    1.0    0.0    ...
58.0         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
...          ...    ...    ...    ...    ...    ...    ...    ...    ...    ...    ...
1412.0       0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
1414.0       0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
1415.0       0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
1416.0       0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
1418.0       0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
1419.0       0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
1420.0       0.0    0.0    0.0    1.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
1421.0       0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
1422.0       0.0    1.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    1.0    ...
1423.0       0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
1424.0       0.0    0.0    0.0    1.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
1425.0       0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
1426.0       0.0    0.0    0.0    1.0    0.0    0.0    0.0    0.0    0.0    0.0    ...
1427.0       1.0    1.0    0.0    1.0    0.0    1.0    0.0    0.0    0.0    1.0    ...
1428.0       0.0    0.0    0.0    0.0    0.0    0.0    0.0    1.0    0.0    0.0    ...
1429.0       1.0    0.0    1.0    0.0    0.0    1.0    1.0    0.0    0.0    1.0    ...
```

```
1430.0      1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   1.0   0.0   ...
1431.0      1.0   0.0   1.0   0.0   0.0   1.0   0.0   0.0   0.0   0.0   ...
1432.0      0.0   0.0   1.0   1.0   0.0   1.0   0.0   0.0   0.0   0.0   ...
1433.0      0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...
1434.0      0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...
1435.0      0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...
1436.0      1.0   0.0   1.0   0.0   0.0   1.0   0.0   1.0   0.0   0.0   ...
1437.0      0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...
1439.0      1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...
1440.0      0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...
1441.0      0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...
1442.0      0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...
1443.0      0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...
1444.0      0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...


user_id    5140  5141  5142  5143  5144  5145  5146  5147  5148  5149
article_id
0.0         0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
2.0         1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
4.0         0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
8.0         0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
9.0         0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
12.0        1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
14.0        0.0   0.0   1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
15.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
16.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   1.0
18.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
20.0        1.0   0.0   0.0   0.0   0.0   1.0   0.0   0.0   0.0   0.0
25.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
26.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
28.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
29.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
30.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
32.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
33.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
34.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
36.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
39.0        1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
40.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
43.0        1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
48.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
50.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
51.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
53.0        1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
54.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
57.0        1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
58.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
...         ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
```

```
1412.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1414.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1415.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1416.0        0.0   0.0   0.0   0.0   0.0   0.0   1.0   0.0   0.0   0.0
1418.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1419.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1420.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1421.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1422.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1423.0        1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1424.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1425.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1426.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1427.0        1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1428.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1429.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1430.0        1.0   0.0   0.0   1.0   0.0   0.0   0.0   0.0   0.0   0.0
1431.0        0.0   0.0   0.0   1.0   0.0   0.0   0.0   0.0   0.0   0.0
1432.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1433.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1434.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1435.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1436.0        0.0   0.0   0.0   1.0   0.0   0.0   0.0   0.0   0.0   0.0
1437.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1439.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1440.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1441.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1442.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1443.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1444.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0

[714 rows x 5149 columns]
```

In [63]: # set the owner's id and get the interaction data of this user
         selected_user_id = 3
         useri = user_item[user_item.index == selected_user_id]
         useri

Out[63]: article_id 0.0   2.0   4.0   8.0   9.0   12.0   14.0   15.0   \
         user_id
         3   0.0   0.0   0.0   0.0   0.0   1.0   0.0   0.0

         article_id 16.0   18.0   ...   1434.0 1435.0 1436.0 1437.0 1439.0 \
         user_id   ...
         3   0.0   0.0   ...   0.0   0.0   1.0   0.0   0.0

         article_id 1440.0 1441.0 1442.0 1443.0 1444.0
         user_id

```
3                0.0    0.0    0.0    0.0    0.0

[1 rows x 714 columns]
```

In [64]: # use the dot product to calculate the similarity of other users to the owner
         useri.dot(user_item.transpose())

```
Out[64]: user_id  1     2     3     4     5     6     7     8     9     10    ...   \
         user_id                                                             ...
         3        6.0   1.0   40.0  5.0   1.0   7.0   1.0   5.0   2.0   5.0   ...

         user_id  5140  5141  5142  5143  5144  5145  5146  5147  5148  5149
         user_id
         3        7.0   0.0   0.0   5.0   0.0   2.0   0.0   0.0   0.0   0.0

[1 rows x 5149 columns]
```

Find the way to sort by similarity

In [65]: # cannot sort the pivot matrix directly
         useri.dot(user_item.transpose()).sort_values(by = "user_id", axis = 1, ascending = True

```
Out[65]: user_id  1     2     3     4     5     6     7     8     9     10    ...   \
         user_id                                                             ...
         3        6.0   1.0   40.0  5.0   1.0   7.0   1.0   5.0   2.0   5.0   ...

         user_id  5140  5141  5142  5143  5144  5145  5146  5147  5148  5149
         user_id
         3        7.0   0.0   0.0   5.0   0.0   2.0   0.0   0.0   0.0   0.0

[1 rows x 5149 columns]
```

In [66]: # test to use stack, cannot use because of duplicated 'user_id'
         useri.dot(user_item.transpose()).stack()

```
Out[66]: user_id  user_id
         3        1           6.0
                  2           1.0
                  3           40.0
                  4           5.0
                  5           1.0
                  6           7.0
                  7           1.0
                  8           5.0
                  9           2.0
                  10          5.0
                  11          13.0
                  12          0.0
                  13          2.0
```

```
14          1.0
15          1.0
16          1.0
17          3.0
18          2.0
19          0.0
20          0.0
21         12.0
22          4.0
23         23.0
24          3.0
25          0.0
26          4.0
27          4.0
28         10.0
29          0.0
30          0.0
            ...
5120        0.0
5121        0.0
5122        0.0
5123        4.0
5124        3.0
5125        0.0
5126        0.0
5127        3.0
5128        0.0
5129        1.0
5130        1.0
5131        0.0
5132        0.0
5133        1.0
5134        3.0
5135        0.0
5136        0.0
5137        0.0
5138       13.0
5139        2.0
5140        7.0
5141        0.0
5142        0.0
5143        5.0
5144        0.0
5145        2.0
5146        0.0
5147        0.0
5148        0.0
5149        0.0
```

```
          Length: 5149, dtype: float64

In [67]: # need to to melt the pivot matrix to sort the similarity
         similarity = useri.dot(user_item.transpose()).melt().sort_values(by = 'value', ascendin
         similarity.head()

Out[67]:        user_id  value
         3352      3353   40.0
         2            3   40.0
         22          23   23.0
         3781      3782   23.0
         3763      3764   17.0
```

Create list of just the user ids

```
In [68]: # get the user_id
         most_similar_users = similarity['user_id']
         most_similar_users.head()

Out[68]: 3352    3353
         2          3
         22        23
         3781    3782
         3763    3764
         Name: user_id, dtype: int64
```

Remove the own user's id

```
In [69]: # the owner's id
         selected_user_id

Out[69]: 3

In [70]: # remove the owner's id
         most_similar_users = most_similar_users[most_similar_users!= selected_user_id]
         most_similar_users

Out[70]: 3352    3353
         22        23
         3781    3782
         3763    3764
         97        98
         4458    4459
         202      203
         48        49
         3696    3697
         3595    3596
         51        52
         911      912
```

```
3539     3540
203       204
39         40
4931     4932
241       242
5137     5138
3869     3870
3909     3910
3965     3966
2925     2926
130       131
10         11
3577     3578
4773     4774
764       765
124       125
213       214
4784     4785
          . . .
2484     2485
2485     2486
2454     2455
2453     2454
2447     2448
2442     2443
2405     2406
2407     2408
2408     2409
2410     2411
2411     2412
2412     2413
2418     2419
2419     2420
2420     2421
2423     2424
2425     2426
2426     2427
2427     2428
2428     2429
2431     2432
2432     2433
2433     2434
2435     2436
2436     2437
2437     2438
2438     2439
2440     2441
2441     2442
```

```
         5148    5149
         Name: user_id, Length: 5148, dtype: int64

In [71]: # test the top similar users
         most_similar_users.reset_index(drop=True, inplace=True)
         most_similar_users[:10]

Out[71]: 0     3353
         1       23
         2     3782
         3     3764
         4       98
         5     4459
         6      203
         7       49
         8     3697
         9     3596
         Name: user_id, dtype: int64

In [72]: print(type(most_similar_users))

<class 'pandas.core.series.Series'>


In [73]: most_similar_users.tolist()

Out[73]: [3353,
          23,
          3782,
          3764,
          98,
          4459,
          203,
          49,
          3697,
          3596,
          52,
          912,
          3540,
          204,
          40,
          4932,
          242,
          5138,
          3870,
          3910,
          3966,
          2926,
          131,
```

11,
3578,
4774,
765,
125,
214,
4785,
21,
619,
195,
3024,
754,
4755,
3141,
656,
4642,
290,
371,
28,
3784,
2161,
4201,
2982,
135,
334,
3485,
273,
186,
134,
4038,
288,
696,
4471,
4706,
3794,
4824,
211,
807,
46,
3801,
383,
3621,
3057,
4392,
58,
3172,
4293,
126,

113,
4883,
4892,
111,
184,
3740,
4161,
295,
72,
395,
3006,
67,
322,
4134,
5041,
3532,
3622,
1058,
3264,
755,
3483,
791,
223,
3169,
215,
3695,
750,
209,
199,
197,
187,
4209,
471,
4277,
170,
4449,
479,
1059,
4595,
3136,
3775,
60,
741,
3938,
5140,
3358,
648,
3058,

3967,
621,
4618,
3877,
591,
3856,
4934,
3079,
6,
64,
56,
722,
689,
538,
1068,
4904,
2989,
249,
1062,
90,
469,
4623,
4788,
5059,
3072,
4323,
4308,
235,
3589,
3636,
5057,
1355,
244,
3486,
1897,
2908,
5079,
280,
420,
261,
4272,
4484,
4543,
193,
4778,
4404,
4792,
3061,

196,
268,
1,
4660,
3783,
492,
693,
129,
3684,
3968,
688,
761,
3960,
38,
669,
651,
3933,
55,
3208,
4137,
665,
3878,
745,
726,
712,
488,
536,
4206,
4900,
3651,
3417,
4231,
860,
418,
3007,
1105,
256,
4491,
65,
69,
5080,
4494,
670,
4526,
409,
3147,
697,
4511,

262,
668,
347,
5078,
3238,
4,
3520,
3310,
8,
3829,
1384,
10,
330,
3100,
362,
3439,
3879,
1262,
1163,
640,
5143,
313,
312,
582,
375,
646,
3918,
1240,
3118,
647,
2981,
88,
85,
173,
3674,
4225,
3197,
4802,
790,
511,
220,
3474,
794,
2913,
4241,
4167,
3614,
4876,

456,
2903,
3611,
4145,
4142,
227,
145,
521,
169,
438,
545,
733,
542,
2975,
2290,
3163,
3741,
4842,
464,
4680,
1261,
1310,
500,
3414,
4872,
352,
4667,
3237,
3466,
1265,
3108,
900,
842,
1272,
3245,
4696,
1302,
896,
854,
644,
3441,
3240,
907,
3198,
715,
1251,
379,
770,

3291,
1139,
1040,
411,
753,
751,
535,
560,
1063,
700,
886,
723,
706,
1076,
512,
1026,
3495,
5023,
4720,
3420,
1215,
714,
649,
3296,
1198,
510,
3421,
970,
3176,
3500,
658,
1162,
3402,
600,
87,
4515,
4453,
4021,
239,
2974,
251,
3329,
4497,
2718,
2994,
3354,
294,
1588,

3765,
76,
1567,
2423,
5077,
287,
66,
3957,
2953,
2288,
225,
3336,
4200,
152,
3378,
3693,
5118,
4268,
3637,
3633,
5105,
5123,
3710,
4088,
4086,
3724,
114,
4378,
110,
4386,
3732,
3553,
155,
3812,
45,
1426,
3818,
3802,
3884,
1476,
3898,
319,
3535,
1353,
3082,
296,
22,
1386,

310,
3835,
48,
321,
26,
3943,
601,
3949,
1401,
3043,
27,
3831,
3864,
2202,
786,
4085,
4082,
3212,
3213,
4099,
4098,
785,
666,
4107,
4933,
833,
3206,
809,
808,
4148,
3442,
3691,
3824,
2144,
2636,
4877,
2790,
609,
3211,
797,
2186,
2194,
4119,
736,
4071,
3235,
3225,
707,

3919,
3986,
703,
3757,
699,
2487,
698,
3763,
3969,
660,
661,
683,
673,
3768,
3956,
4186,
713,
3754,
4039,
2270,
2284,
3888,
641,
2304,
4044,
2308,
3807,
2377,
2312,
4915,
4015,
4010,
721,
719,
3749,
3518,
895,
847,
3042,
4557,
1213,
1208,
4725,
1548,
1197,
3124,
3017,
1172,

1156,
3561,
1577,
3005,
4527,
4517,
1137,
3570,
3488,
4768,
1612,
1114,
2993,
1110,
4560,
3117,
2983,
1225,
1331,
4668,
1322,
1350,
1317,
3514,
1354,
1359,
1301,
1291,
4636,
1281,
1395,
4697,
1408,
1266,
4620,
1264,
3063,
1443,
1250,
1478,
3506,
1635,
1094,
3669,
932,
925,
4336,
3616,

923,
4834,
3618,
4319,
3632,
3634,
4285,
1345,
894,
4843,
4274,
3464,
866,
862,
2007,
858,
4204,
856,
3667,
851,
926,
938,
1084,
943,
1082,
1668,
1671,
4472,
3366,
3581,
1061,
2955,
2948,
1048,
3477,
4427,
1038,
4393,
1801,
1014,
1003,
992,
987,
980,
2919,
2907,
2901,
4577,

3833,
266,
35,
490,
32,
285,
271,
3325,
3326,
5086,
253,
252,
248,
508,
509,
208,
121,
3338,
4970,
181,
176,
5111,
3430,
168,
558,
5115,
3346,
4958,
3376,
304,
5070,
4992,
323,
3408,
5012,
424,
413,
410,
406,
17,
404,
3292,
3301,
5001,
5034,
3305,
3412,
365,

354,
5046,
24,
346,
3313,
5052,
5053,
5055,
4995,
324,
122,
445,
584,
107,
100,
5127,
598,
5124,
82,
86,
4943,
94,
575,
4949,
74,
5134,
63,
83,
5058,
335,
1375,
1394,
1385,
1389,
4648,
3523,
4632,
4026,
326,
1405,
3081,
1360,
3083,
4025,
1311,
3094,
2728,
3093,

351,
3092,
1325,
1329,
5048,
1337,
1338,
2731,
1340,
344,
4658,
4019,
2342,
4022,
4652,
1406,
120,
3069,
2328,
308,
5145,
4591,
3885,
3046,
4588,
305,
4041,
1487,
3809,
3538,
1499,
1500,
3808,
3542,
1509,
75,
102,
4571,
4599,
1455,
1452,
1420,
4626,
1303,
3067,
1413,
2319,
2316,

3065,
1418,
4619,
3056,
3527,
5066,
1427,
315,
1433,
3320,
1437,
1439,
92,
4685,
18,
359,
1095,
2416,
1112,
1115,
1118,
414,
2415,
1128,
1129,
2579,
1132,
1134,
1135,
3972,
4759,
1142,
1145,
3492,
3975,
1091,
1089,
5013,
4791,
4801,
4800,
442,
4795,
440,
4794,
1051,
2586,
1056,

3970,
1057,
4790,
3406,
434,
1067,
1070,
1071,
5011,
1150,
403,
5018,
2376,
3998,
370,
5039,
4700,
1270,
1271,
3396,
3107,
4001,
4707,
363,
1283,
1284,
2564,
3102,
3034,
360,
1300,
3400,
1245,
1169,
3876,
1170,
3499,
1183,
1187,
2653,
1191,
3404,
1202,
3505,
1243,
1218,
1219,
3300,

1223,
4716,
381,
1238,
1239,
2541,
299,
4564,
4248,
5106,
2488,
2440,
4291,
177,
4073,
4275,
5110,
4269,
171,
3641,
4266,
1984,
115,
1988,
5139,
4255,
4252,
4250,
3627,
4311,
2255,
2899,
4367,
4364,
4363,
4356,
4068,
1850,
4352,
3727,
2898,
1915,
3917,
200,
1882,
198,
3341,
188,

3625,
1914,
4076,
166,
4373,
2842,
5117,
140,
4087,
59,
4130,
2164,
4127,
3703,
2447,
3951,
132,
3705,
3715,
4110,
4108,
4097,
2205,
2209,
3713,
144,
4144,
4149,
160,
4234,
4230,
164,
3658,
51,
4215,
163,
2235,
3344,
57,
4194,
3671,
153,
54,
2097,
4170,
2116,
149,
2917,

2921,
3360,
2980,
3564,
1595,
4522,
4045,
1608,
4049,
3243,
4510,
1027,
4502,
1633,
1636,
2992,
4498,
4495,
2988,
1655,
4487,
2279,
4529,
1582,
284,
3554,
3889,
3714,
4553,
1541,
297,
292,
1549,
1550,
3803,
1574,
3556,
103,
5075,
1561,
3015,
2303,
1569,
1572,
4482,
4061,
213,
2978,

```
            2945,
            4420,
            4417,
            1762,
            1766,
            2941,
            2940,
            3595,
            4409,
            2935,
            1787,
            2932,
            2931,
            4390,
            4389,
            1807,
            4385,
            ...]

In [74]: def find_similar_users(user_id, user_item=user_item):
            '''
            INPUT:
            user_id - (int) a user_id
            user_item - (pandas dataframe) matrix of users by articles:
                        1's when a user has interacted with an article, 0 otherwise

            OUTPUT:
            similar_users - (list) an ordered list where the closest users (largest dot product
                        are listed first

            Description:
            Computes the similarity of every pair of users based on the dot product
            Returns an ordered

            '''
            # compute similarity of each user to the provided user
            useri = user_item[user_item.index == user_id]
            useri.dot(user_item.transpose())

            # sort by similarity
            similarity = useri.dot(user_item.transpose()).melt().sort_values(by = 'value', asce

            # create list of just the ids
            most_similar_users = similarity['user_id']

            # remove the own user's id
            most_similar_users = most_similar_users[most_similar_users!= user_id]
            most_similar_users.reset_index(drop=True, inplace=True)
```

```
            most_similar_users = most_similar_users.tolist()

            return most_similar_users # return a list of the users in order from most to least


In [75]: # Do a spot check of your function
         print("The 10 most similar users to user 1 are: {}".format(find_similar_users(1)[:10]))
         print("The 5 most similar users to user 3933 are: {}".format(find_similar_users(3933)[:
         print("The 3 most similar users to user 46 are: {}".format(find_similar_users(46)[:3]))

The 10 most similar users to user 1 are: [3933, 23, 3782, 203, 4459, 3870, 131, 4201, 46, 5041]
The 5 most similar users to user 3933 are: [1, 23, 3782, 203, 4459]
The 3 most similar users to user 46 are: [4201, 3782, 23]
```

3. Now that you have a function that provides the most similar users to each user, you will want to use these users to find articles you can recommend. Complete the functions below to return the articles you would recommend to each user.
  a. Draft the functions
  Get article names from article ids

```
In [76]: df.head()

Out[76]:    article_id                                            title  user_id
         0      1430.0  using pixiedust for fast, flexible, and easier...        1
         1      1314.0       healthcare python streaming application demo        2
         2      1429.0          use deep learning for image classification        3
         3      1338.0           ml optimization using cognitive assistant        4
         4      1276.0           deploy your python model as a restful api        5

In [77]: article_ids_list = ['1430.0', '1276.0', '1429.0']
         article_ids_list = np.array(article_ids_list, dtype=np.float64)
         article_ids_list

Out[77]: array([ 1430.,  1276.,  1429.])

In [78]: df_article_drop=df.drop_duplicates(subset=['article_id'], keep='first')[['article_id','
         df_article_drop.head()

Out[78]:    article_id                                            title
         0      1430.0  using pixiedust for fast, flexible, and easier...
         1      1314.0       healthcare python streaming application demo
         2      1429.0          use deep learning for image classification
         3      1338.0           ml optimization using cognitive assistant
         4      1276.0           deploy your python model as a restful api

In [79]: # isin doesn't keep the order of article_ids -> cannot use
         article_names = df_article_drop[df_article_drop['article_id'].isin(article_ids_list)==T
         article_names
```

```
Out[79]: 0       using pixiedust for fast, flexible, and easier...
         2               use deep learning for image classification
         4               deploy your python model as a restful api
         Name: title, dtype: object

In [80]: article_names = df_article_drop.set_index('article_id').loc[article_ids_list].reset_ind
         article_names

Out[80]: 0       using pixiedust for fast, flexible, and easier...
         1               deploy your python model as a restful api
         2               use deep learning for image classification
         Name: title, dtype: object
```

Get the articles seen by a user

```
In [81]: selected_user_id = 20
         user_article = user_item[user_item.index == selected_user_id]
         user_article

Out[81]: article_id  0.0      2.0      4.0      8.0      9.0      12.0     14.0     15.0     \
         user_id
         20               0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0

         article_id  16.0     18.0       ...     1434.0   1435.0   1436.0   1437.0   1439.0   \
         user_id                         ...
         20               0.0      0.0    ...       0.0      0.0      0.0      0.0      0.0

         article_id  1440.0   1441.0   1442.0   1443.0   1444.0
         user_id
         20               0.0      0.0      0.0      0.0      0.0

         [1 rows x 714 columns]

In [82]: user_article=user_article.melt()
         user_article.head()

Out[82]:    article_id  value
         0        0.0    0.0
         1        2.0    0.0
         2        4.0    0.0
         3        8.0    0.0
         4        9.0    0.0

In [83]: article_ids_list = user_article[user_article['value'] == 1.0]['article_id']
         #article_ids_list.reset_index(drop=True, inplace=True)
         article_ids_list.head()

Out[83]: 104     232.0
         347     844.0
         619    1320.0
         Name: article_id, dtype: float64
```

Recommend the articles the user hasn't seen before from the articles of close users

```
In [84]: #find the list of close users
         user_id = 1
         closeness_loop = find_similar_users(user_id, user_item)
         closeness_loop
```

```
Out[84]: [3933,
          23,
          3782,
          203,
          4459,
          3870,
          131,
          4201,
          46,
          5041,
          395,
          3697,
          49,
          322,
          242,
          3622,
          3910,
          98,
          754,
          2982,
          290,
          3540,
          4642,
          3764,
          912,
          268,
          40,
          3775,
          4932,
          4134,
          52,
          621,
          5138,
          1355,
          4785,
          3651,
          3637,
          256,
          273,
          371,
          204,
```

3784,
3621,
3596,
21,
135,
4038,
3532,
64,
186,
184,
3483,
214,
2926,
696,
249,
3141,
765,
619,
4755,
4778,
4892,
3136,
488,
334,
4293,
1897,
4934,
125,
4484,
445,
4206,
4774,
3684,
5013,
60,
38,
656,
2790,
187,
10,
4209,
665,
3856,
3632,
288,
5140,
4706,
5079,

54,
72,
67,
3740,
58,
3024,
2430,
409,
733,
4883,
3354,
3408,
193,
4277,
4225,
3485,
195,
3949,
3358,
4543,
379,
170,
211,
3693,
807,
223,
591,
3794,
3353,
235,
90,
3818,
3197,
4404,
295,
261,
263,
3006,
4471,
3572,
3879,
362,
28,
69,
3578,
2981,
324,
689,

2975,
87,
1040,
8,
3,
113,
3169,
111,
4595,
244,
330,
1059,
4824,
511,
2161,
4933,
1353,
319,
1330,
4082,
4088,
304,
312,
659,
3589,
598,
750,
280,
4037,
4167,
365,
3067,
3329,
1271,
639,
3118,
640,
4021,
251,
1244,
647,
215,
4130,
3264,
4901,
3057,
383,
3741,

4900,
3100,
3710,
641,
155,
4697,
3005,
4788,
456,
82,
557,
558,
88,
4495,
3172,
1567,
3898,
2908,
4502,
1062,
2903,
3801,
114,
4510,
4792,
4802,
3523,
3305,
926,
3636,
3829,
11,
510,
907,
22,
24,
26,
4453,
890,
45,
535,
471,
3500,
63,
65,
3376,
670,
829,

2423,
3967,
3966,
168,
4231,
4248,
5023,
4526,
1163,
1162,
3943,
4725,
3441,
3535,
418,
3237,
126,
4515,
4517,
373,
438,
1633,
3732,
4449,
3474,
521,
490,
492,
321,
2897,
512,
4241,
4323,
1705,
4282,
5034,
4620,
2144,
607,
668,
4588,
1426,
669,
3633,
1386,
3336,
536,
538,

3705,
347,
3466,
3614,
4186,
1446,
420,
4268,
3611,
343,
4356,
4272,
1401,
3061,
3058,
4255,
4254,
4274,
4511,
3292,
3654,
4522,
3430,
542,
649,
4497,
253,
5057,
745,
3960,
181,
3421,
3970,
3971,
763,
197,
3783,
213,
2305,
995,
5077,
3877,
57,
56,
3425,
55,
4843,
4002,

884,
5080,
4013,
3486,
2718,
860,
3417,
2471,
3802,
1068,
843,
122,
828,
824,
5124,
845,
4736,
134,
136,
1137,
140,
4732,
145,
4882,
794,
3518,
85,
164,
5078,
4904,
4668,
688,
707,
4648,
5069,
2586,
698,
277,
896,
697,
3812,
5143,
4071,
1301,
927,
9,
6,
681,

296,
3208,
5062,
4623,
305,
5059,
5070,
700,
4768,
41,
722,
1272,
1281,
37,
726,
2579,
240,
1671,
532,
4861,
987,
3196,
938,
4842,
4504,
885,
842,
4450,
2993,
565,
1566,
4969,
889,
1668,
1014,
866,
1018,
4795,
3007,
4855,
518,
3618,
555,
4491,
871,
895,
3191,
4496,

998,
970,
4786,
977,
4837,
854,
1038,
2989,
1574,
1058,
1513,
3610,
712,
4943,
4586,
1262,
1264,
3069,
631,
3072,
727,
1413,
1405,
4909,
721,
4663,
3108,
3564,
1437,
4647,
648,
650,
651,
3079,
4921,
3570,
3093,
662,
687,
1343,
4618,
4619,
1354,
1261,
741,
4506,
785,
833,

4763,
1076,
4749,
820,
4876,
798,
791,
790,
4716,
585,
4711,
3031,
674,
1499,
1251,
4707,
1198,
1213,
596,
3042,
4552,
4554,
761,
760,
757,
4685,
1240,
1247,
4947,
1673,
924,
260,
92,
354,
4145,
5115,
4142,
2447,
129,
4137,
346,
335,
4110,
332,
2128,
132,
326,
4100,

4098,
4097,
133,
5052,
4148,
121,
359,
107,
94,
4204,
3436,
95,
103,
4197,
3378,
3324,
2688,
2318,
3327,
375,
2845,
3720,
2029,
4173,
3434,
120,
5053,
3335,
3917,
4015,
2336,
2248,
2251,
4020,
245,
2335,
241,
3972,
230,
3968,
199,
225,
200,
2290,
2324,
2320,
2312,
205,

182,
4427,
310,
5103,
3918,
3749,
2410,
3338,
294,
3414,
2174,
2176,
3757,
3763,
2197,
2199,
160,
3363,
163,
169,
4047,
4045,
3695,
4161,
209,
4336,
3835,
3884,
74,
2953,
4311,
2510,
3310,
1747,
2949,
479,
2593,
2899,
475,
3876,
48,
44,
2934,
1787,
3807,
1775,
4367,
35,

487,
2913,
2898,
86,
4393,
4245,
3640,
4,
4390,
13,
5129,
404,
3404,
89,
1227,
1187,
2661,
3951,
975,
2637,
2370,
2377,
3195,
2,
2667,
2376,
4717,
1229,
3790,
1230,
1165,
2374,
925,
974,
2646,
2621,
2725,
3543,
1214,
3819,
3833,
1219,
931,
939,
2728,
950,
4695,
953,

1208,
930,
3198,
3147,
1200,
1199,
955,
1223,
2342,
4700,
2350,
1225,
2355,
1194,
957,
3947,
2488,
1154,
2465,
3159,
3163,
4800,
2507,
1003,
3912,
3883,
3885,
2450,
2451,
2682,
3888,
2458,
1025,
1026,
2384,
3889,
1065,
3520,
1063,
3890,
1061,
1031,
2475,
4783,
2479,
2482,
2484,
2485,

1035,
3878,
996,
1098,
1101,
2387,
2710,
3938,
2561,
2406,
3872,
1139,
2487,
4809,
2556,
2552,
984,
2521,
1132,
3155,
1129,
1127,
2516,
988,
4741,
3528,
1114,
4746,
4806,
3190,
2514,
4747,
1105,
3913,
2415,
2974,
1234,
3691,
4246,
4527,
2879,
1527,
4250,
4275,
1543,
1879,
2901,
4296,

4300,
4302,
4503,
3674,
4307,
2912,
4316,
1523,
1512,
4321,
3600,
1449,
2007,
3711,
3052,
3051,
4559,
4557,
1476,
4199,
4553,
1994,
3046,
4545,
1988,
4230,
1498,
4233,
4319,
2919,
2850,
1752,
4456,
2992,
4378,
4379,
4444,
1655,
3629,
4384,
1661,
4386,
4437,
1665,
4392,
4401,
4414,
1685,

3634,
3650,
4371,
4494,
2994,
4329,
2924,
3664,
919,
1588,
4485,
2932,
3009,
1595,
1780,
4363,
4364,
2941,
1766,
1620,
1627,
1764,
3713,
4577,
3986,
1284,
4658,
4034,
4652,
4039,
1302,
3565,
4049,
3566,
1313,
2768,
2208,
2205,
1320,
2202,
4057,
1329,
2180,
4660,
3559,
1340,
3765,
2304,

3991,
3554,
2742,
2288,
4674,
4673,
2284,
4004,
4016,
1265,
2258,
4667,
2257,
1274,
4025,
3768,
4068,
4073,
3065,
3745,
4608,
4107,
2800,
3737,
4127,
4128,
2813,
2815,
3733,
3581,
1409,
1411,
2819,
2826,
2830,
1427,
4170,
1389,
3746,
4076,
1384,
1347,
2164,
1350,
3088,
2163,
1352,
2788,

4624,
4086,
1359,
1361,
3087,
4091,
4096,
1370,
3082,
3081,
1807,
4638,
3824,
233,
706,
709,
4915,
711,
4914,
227,
714,
226,
3482,
4910,
220,
219,
723,
5083,
730,
4907,
734,
3255,
239,
636,
246,
285,
646,
3341,
3428,
276,
661,
272,
271,
666,
667,
673,
266,
264,

262,
3426,
254,
248,
736,
189,
751,
3420,
149,
3413,
146,
3238,
5108,
3368,
3488,
3235,
809,
810,
4877,
3373,
4872,
112,
110,
822,
104,
150,
780,
151,
159,
752,
753,
755,
176,
759,
173,
166,
770,
152,
156,
153,
771,
773,
4891,
778,
3240,
638,
635,
...]

b. Fill in the functions

```
In [85]: def get_article_names(article_ids, df=df):
             '''
             INPUT:
             article_ids - (list) a list of article ids
             df - (pandas dataframe) df as defined at the top of the notebook

             OUTPUT:
             article_names - (list) a list of article names associated with the list of article
                             (this is identified by the title column)
             '''
             # Your code here
             article_ids = np.array(article_ids, dtype=np.float64)
             df_article_drop=df.drop_duplicates(subset=['article_id'], keep='first')[['article_i
             #article_names = df_article_drop[df_article_drop['article_id'].isin(article_ids)==1
             article_names = df_article_drop.set_index('article_id').loc[article_ids].reset_inde
             #article_names.reset_index(drop=True, inplace=True)
             article_names = article_names.tolist()

             return article_names # Return the article names associated with list of article ids


         def get_user_articles(user_id, user_item=user_item):
             '''
             INPUT:
             user_id - (int) a user id
             user_item - (pandas dataframe) matrix of users by articles:
                         1's when a user has interacted with an article, 0 otherwise

             OUTPUT:
             article_ids - (list) a list of the article ids seen by the user
             article_names - (list) a list of article names associated with the list of article
                             (this is identified by the doc_full_name column in df_content)

             Description:
             Provides a list of the article_ids and article titles that have been seen by a user
             '''
             # Your code here
             user_article_list = user_item[user_item.index == user_id]
             user_article_list = user_article_list.melt()
             article_ids = user_article_list[user_article_list['value'] == 1.0]['article_id']
             #article_ids.reset_index(drop=True, inplace=True)
             article_ids = article_ids.tolist()

             article_ids = [str(x) for x in article_ids]

             article_names = get_article_names(article_ids, df=df)
```

91

```python
            return article_ids, article_names # return the ids and names


        def user_user_recs(user_id, m=10):
            '''
            INPUT:
            user_id - (int) a user id
            m - (int) the number of recommendations you want for the user

            OUTPUT:
            recs - (list) a list of recommendations for the user

            Description:
            Loops through the users based on closeness to the input user_id
            For each user - finds articles the user hasn't seen before and provides them as rec
            Does this until m recommendations are found

            Notes:
            Users who are the same closeness are chosen arbitrarily as the 'next' user

            For the user where the number of recommended articles starts below m
            and ends exceeding m, the last items are chosen arbitrarily

            '''
            # Your code here
            closeness_loop = find_similar_users(user_id, user_item)
            r = 0
            recs = []
            #read_article_ids, read_article_names = get_user_articles(user_id, user_item)
            read_article_ids, read_article_names = get_user_articles(user_id)
            for i in closeness_loop:
                #rec_article_ids , rec_article_names = get_user_articles(i, user_item)
                rec_article_ids , rec_article_names = get_user_articles(i)
                for j in rec_article_ids:
                    if (j not in read_article_ids):
                        r = r + 1
                        recs.append(j)
                        if r == m:
                            break
                if r == m:
                    break
            return recs # return your recommendations for this user_id

In [86]: # Check Results
         article_ids = [1430.0, 1429.0, 1276.0]
         #get_article_names(article_ids, df)
         get_article_names(article_ids)
```

92

```
Out[86]: ['using pixiedust for fast, flexible, and easier data analysis and experimentation',
          'use deep learning for image classification',
          'deploy your python model as a restful api']

In [87]: # Check Results
         #get_user_articles(1, user_item)
         get_user_articles(1)

Out[87]: (['43.0',
           '109.0',
           '151.0',
           '268.0',
           '310.0',
           '329.0',
           '346.0',
           '390.0',
           '494.0',
           '525.0',
           '585.0',
           '626.0',
           '668.0',
           '732.0',
           '768.0',
           '910.0',
           '968.0',
           '981.0',
           '1052.0',
           '1170.0',
           '1183.0',
           '1185.0',
           '1232.0',
           '1293.0',
           '1305.0',
           '1363.0',
           '1368.0',
           '1391.0',
           '1400.0',
           '1406.0',
           '1427.0',
           '1429.0',
           '1430.0',
           '1431.0',
           '1436.0',
           '1439.0'],
          ['deep learning with tensorflow course by big data university',
           'tensorflow quick tips',
           'jupyter notebook tutorial',
           'sector correlations shiny app',
```

```
                'time series prediction using recurrent neural networks (lstms)',
                'introduction to market basket analysis in\xa0python',
                'fighting gerrymandering: using data science to draw fairer congressional districts',
                'introducing ibm watson studio ',
                'python for loops explained (python for data science basics #5)',
                'new shiny cheat sheet and video tutorial',
                'tidyverse practice: mapping large european cities',
                'analyze db2 warehouse on cloud data in rstudio in dsx',
                'shiny: a data scientists best friend',
                'rapidly build machine learning flows with dsx',
                'python if statements explained (python for data science basics #4)',
                'working with ibm cloud object storage in python',
                'shiny 0.13.0',
                'super fast string matching in python',
                'access db2 warehouse on cloud and db2 with python',
                'apache spark lab, part 1: basic concepts',
                'categorize urban density',
                'classify tumors with machine learning',
                'country statistics: life expectancy at birth',
                'finding optimal locations of new store using decision optimization',
                'gosales transactions for naive bayes model',
                'predict loan applicant behavior with tensorflow neural networking',
                'putting a human face on machine learning',
                'sudoku',
                'uci ml repository: chronic kidney disease data set',
                'uci: iris',
                'use xgboost, scikit-learn & ibm watson machine learning apis',
                'use deep learning for image classification',
                'using pixiedust for fast, flexible, and easier data analysis and experimentation',
                'visualize car data with brunel',
                'welcome to pixiedust',
                'working with ibm cloud object storage in r'])
```

In [88]: # Check Results
         get_article_names(user_user_recs(1, 10)) # Return 10 recommendations for user 1

Out[88]: ['this week in data science (april 18, 2017)',
         'timeseries data analysis of iot events by using jupyter notebook',
         'got zip code data? prep it for analytics. ibm watson data lab medium',
         'higher-order logistic regression for large datasets',
         'using machine learning to predict parking difficulty',
         'deep forest: towards an alternative to deep neural networks',
         'experience iot with coursera',
         'using brunel in ipython/jupyter notebooks',
         'graph-based machine learning',
         'the 3 kinds of context: machine learning and the art of the frame']

In [89]: get_article_names(['1024.0','1176.0', '1305.0', '1314.0', '1422.0', '1427.0'])

```
Out[89]: ['using deep learning to reconstruct high-resolution audio',
          'build a python app on the streaming analytics service',
          'gosales transactions for naive bayes model',
          'healthcare python streaming application demo',
          'use r dataframes & ibm watson natural language understanding',
          'use xgboost, scikit-learn & ibm watson machine learning apis']

In [90]: get_article_names(['1314.0'])

Out[90]: ['healthcare python streaming application demo']

In [91]: get_article_names(['1320.0', '232.0', '844.0'])

Out[91]: ['housing (2015): united states demographic measures',
          'self-service data preparation with ibm data refinery',
          'use the cloudant-spark connector in python notebook']

In [92]: get_user_articles(2)[0]

Out[92]: ['1024.0', '1176.0', '1305.0', '1314.0', '1422.0', '1427.0']

In [93]: get_user_articles(20)[0]

Out[93]: ['232.0', '844.0', '1320.0']

In [94]: # Test your functions here - No need to change this code - just run this cell
         assert set(get_article_names(['1024.0', '1176.0', '1305.0', '1314.0', '1422.0', '1427.0
         assert set(get_article_names(['1320.0', '232.0', '844.0'])) == set(['housing (2015): un
         assert set(get_user_articles(20)[0]) == set(['1320.0', '232.0', '844.0'])
         assert set(get_user_articles(20)[1]) == set(['housing (2015): united states demographic
         assert set(get_user_articles(2)[0]) == set(['1024.0', '1176.0', '1305.0', '1314.0', '14
         assert set(get_user_articles(2)[1]) == set(['using deep learning to reconstruct high-re
         print("If this is all you see, you passed all of our tests!  Nice job!")
```

If this is all you see, you passed all of our tests! Nice job!

4. Now we are going to improve the consistency of the **user_user_recs** function from above.

- Instead of arbitrarily choosing when we obtain users who are all the same closeness to a given user - choose the users that have the most total article interactions before choosing those with fewer article interactions.

- Instead of arbitrarily choosing articles from the user where the number of recommended articles starts below m and ends exceeding m, choose articles with the articles with the most total interactions before choosing those with fewer total interactions. This ranking should be what would be obtained from the **top_articles** function you wrote earlier.

a. Draft the functions

- The neighbor data frame including the neighbor's similarity and the number of articles viewed by the neighbor

Calculate the neighbor's similarity

```
In [95]: selected_user_id = 3
         useri = user_item[user_item.index == selected_user_id]
         useri

Out[95]: article_id  0.0     2.0     4.0     8.0     9.0     12.0    14.0    15.0    \
         user_id
         3                   0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0

         article_id  16.0    18.0      ...     1434.0  1435.0  1436.0  1437.0  1439.0  \
         user_id                       ...
         3                   0.0     0.0   ...     0.0     0.0     1.0     0.0     0.0

         article_id  1440.0  1441.0  1442.0  1443.0  1444.0
         user_id
         3                   0.0     0.0     0.0     0.0     0.0

         [1 rows x 714 columns]
```

```
In [96]: neighbors = useri.dot(user_item.transpose()).melt().sort_values(by = 'user_id', ascendi
         neighbors.columns = ['user_id', 'similarity']
         neighbors.head()

Out[96]:    user_id  similarity
         0        1         6.0
         1        2         1.0
         2        3        40.0
         3        4         5.0
         4        5         1.0
```

Calculate the number of articles viewed by the neighbor

```
In [97]: df.groupby(['user_id']).size()

Out[97]: user_id
         1           47
         2            6
         3           82
         4           45
         5            5
         6           19
         7            4
         8           82
         9           32
         10          22
```

```
11        35
12        13
13        21
14        28
15        17
16         3
17        35
18         3
19         8
20         3
21       137
22        37
23       364
24        30
25        10
26        27
27        34
28        42
29         1
30         5
         . . .
5120       1
5121       1
5122       1
5123      13
5124      22
5125       1
5126       2
5127      29
5128       4
5129      29
5130       1
5131       1
5132       2
5133       3
5134       9
5135       3
5136       2
5137       2
5138      95
5139      13
5140     101
5141       1
5142       2
5143      25
5144       1
5145       6
5146       9
```

```
      5147       1
      5148       1
      5149       1
      Length: 5149, dtype: int64

In [100]: df.groupby(['user_id']).size().tolist()

Out[100]: [47,
           6,
           82,
           45,
           5,
           19,
           4,
           82,
           32,
           22,
           35,
           13,
           21,
           28,
           17,
           3,
           35,
           3,
           8,
           3,
           137,
           37,
           364,
           30,
           10,
           27,
           34,
           42,
           1,
           5,
           12,
           13,
           8,
           7,
           19,
           7,
           29,
           68,
           2,
           78,
           14,
```

9,
9,
15,
73,
63,
1,
12,
147,
8,
11,
132,
5,
24,
33,
38,
25,
142,
7,
103,
25,
2,
31,
57,
32,
36,
58,
5,
35,
5,
14,
49,
6,
9,
7,
8,
4,
4,
7,
5,
8,
25,
10,
2,
25,
14,
69,
57,
8,

34,
5,
13,
1,
27,
12,
3,
3,
170,
1,
13,
1,
10,
35,
7,
8,
2,
19,
3,
7,
25,
35,
6,
68,
20,
31,
2,
3,
15,
6,
21,
33,
22,
2,
1,
59,
39,
1,
1,
47,
2,
145,
25,
20,
30,
82,
5,
7,

3,
8,
26,
1,
2,
4,
10,
39,
7,
8,
6,
26,
7,
11,
25,
13,
12,
33,
8,
7,
4,
7,
29,
3,
1,
16,
16,
3,
7,
4,
52,
55,
116,
8,
7,
53,
1,
2,
40,
2,
4,
1,
10,
28,
7,
11,
104,
8,

79,
63,
18,
3,
14,
6,
16,
15,
21,
72,
15,
43,
14,
50,
13,
1,
1,
160,
97,
26,
3,
26,
25,
21,
9,
52,
8,
35,
62,
29,
2,
20,
1,
8,
32,
6,
6,
79,
1,
12,
10,
15,
6,
7,
31,
6,
7,
9,

1,
51,
1,
9,
5,
20,
35,
26,
148,
10,
23,
10,
13,
2,
8,
94,
3,
49,
25,
6,
2,
3,
60,
1,
2,
4,
21,
24,
41,
26,
3,
2,
22,
20,
45,
2,
3,
21,
5,
84,
6,
2,
4,
9,
1,
1,
29,
13,

1,
6,
4,
13,
2,
33,
91,
4,
80,
7,
15,
2,
17,
91,
35,
16,
6,
24,
2,
6,
4,
6,
47,
28,
2,
2,
9,
1,
22,
1,
38,
52,
1,
16,
1,
2,
4,
46,
5,
59,
85,
25,
39,
3,
10,
3,
14,
2,

76,
1,
15,
8,
49,
13,
3,
5,
2,
6,
6,
3,
1,
15,
11,
6,
41,
32,
3,
1,
4,
13,
17,
1,
17,
39,
4,
8,
2,
17,
16,
6,
55,
17,
4,
22,
2,
2,
2,
2,
11,
95,
2,
14,
1,
29,
2,
5,

1,
53,
11,
12,
4,
46,
2,
5,
1,
10,
1,
4,
2,
12,
1,
4,
1,
69,
5,
1,
1,
7,
1,
1,
20,
27,
38,
8,
10,
4,
2,
50,
12,
12,
1,
22,
8,
6,
3,
5,
60,
16,
53,
17,
7,
4,
7,
4,

5,
4,
5,
2,
4,
5,
12,
4,
19,
2,
1,
5,
29,
12,
6,
6,
10,
8,
7,
41,
2,
1,
5,
1,
11,
2,
1,
12,
13,
7,
23,
11,
1,
3,
9,
3,
3,
2,
14,
1,
4,
2,
1,
11,
24,
26,
13,
5,

7,
5,
4,
4,
11,
42,
10,
3,
6,
17,
14,
2,
4,
8,
47,
4,
35,
3,
28,
1,
12,
4,
1,
6,
14,
10,
18,
10,
9,
2,
1,
1,
1,
1,
12,
28,
25,
31,
39,
13,
2,
23,
17,
2,
35,
2,
2,
37,

2,
5,
1,
13,
1,
10,
1,
3,
11,
1,
8,
12,
14,
45,
37,
2,
38,
10,
4,
2,
27,
1,
1,
15,
3,
5,
11,
4,
2,
6,
3,
1,
6,
35,
1,
23,
25,
1,
13,
17,
2,
13,
3,
10,
1,
6,
20,
2,

11,
1,
4,
4,
1,
23,
5,
1,
4,
1,
2,
2,
24,
10,
9,
11,
4,
7,
2,
1,
3,
84,
7,
2,
1,
2,
19,
1,
18,
1,
21,
13,
1,
1,
1,
1,
5,
24,
1,
10,
1,
1,
17,
14,
6,
1,
2,
2,

7,
84,
2,
52,
1,
2,
11,
2,
9,
3,
1,
5,
3,
8,
1,
1,
2,
14,
8,
14,
24,
38,
35,
29,
11,
3,
38,
1,
36,
38,
26,
29,
8,
98,
19,
5,
4,
1,
48,
4,
18,
16,
5,
26,
21,
16,
7,
77,

13,
13,
59,
67,
41,
5,
4,
21,
16,
4,
15,
2,
4,
6,
8,
18,
12,
10,
8,
8,
12,
24,
52,
34,
3,
3,
10,
55,
29,
4,
79,
26,
18,
14,
44,
1,
8,
8,
2,
7,
18,
23,
2,
8,
8,
6,
36,
19,

6,
45,
3,
1,
6,
8,
8,
21,
35,
31,
7,
2,
64,
15,
5,
6,
11,
3,
9,
22,
15,
14,
7,
5,
16,
8,
5,
14,
2,
2,
7,
40,
16,
1,
6,
3,
64,
18,
5,
9,
60,
16,
7,
11,
11,
12,
35,
20,

14,
27,
3,
68,
5,
1,
2,
3,
9,
10,
3,
20,
23,
2,
3,
4,
11,
2,
5,
3,
3,
31,
10,
57,
28,
8,
6,
11,
18,
31,
5,
1,
34,
1,
9,
19,
21,
4,
1,
7,
3,
3,
5,
2,
3,
50,
15,
14,

17,
8,
13,
5,
23,
3,
10,
5,
2,
6,
36,
7,
34,
18,
4,
21,
5,
1,
16,
53,
4,
3,
11,
19,
3,
3,
7,
4,
1,
2,
11,
6,
19,
15,
16,
31,
4,
16,
2,
4,
7,
22,
2,
1,
27,
8,
8,
3,

8,
3,
42,
1,
18,
2,
2,
7,
8,
1,
4,
2,
11,
18,
2,
4,
6,
9,
8,
1,
10,
5,
7,
8,
5,
2,
7,
13,
13,
18,
1,
15,
11,
14,
5,
4,
8,
21,
28,
11,
5,
2,
20,
3,
14,
3,
2,
1,

8,
27,
4,
4,
8,
2,
102,
13,
14,
3,
3,
9,
14,
4,
9,
1,
11,
19,
23,
14,
48,
35,
3,
2,
3,
3,
11,
10,
2,
3,
12,
8,
21,
18,
7,
4,
7,
19,
6,
5,
16,
3,
15,
5,
19,
2,
6,
10,

```
7,
14,
3,
18,
10,
14,
8,
6,
22,
1,
4,
11,
8,
6,
2,
16,
30,
11,
13,
7,
8,
9,
6,
13,
9,
2,
9,
11,
7,
3,
7,
12,
3,
19,
30,
1,
5,
2,
14,
2,
3,
12,
8,
9,
8,
6,
4,
...]
```

```
In [98]: neighbors['num_interactions'] = df.groupby(['user_id']).size().tolist()
         neighbors.head()

Out[98]:    user_id  similarity  num_interactions
         0        1         6.0                47
         1        2         1.0                 6
         2        3        40.0                82
         3        4         5.0                45
         4        5         1.0                 5

In [99]: neighbors = neighbors.sort_values(by = ['similarity','num_interactions'], ascending = F
         neighbors.head()

Out[99]:        user_id  similarity  num_interactions
         2            3        40.0                82
         3352      3353        40.0                80
         22          23        23.0               364
         3781      3782        23.0               363
         97          98        17.0               170
```

- Recommendations for the user

```
In [100]: rec_article_ids_test = ['1430.0','1429.0']
          rec_article_ids_test

Out[100]: ['1430.0', '1429.0']

In [101]: df.head()

Out[101]:    article_id                                            title  user_id
          0      1430.0  using pixiedust for fast, flexible, and easier...        1
          1      1314.0        healthcare python streaming application demo        2
          2      1429.0           use deep learning for image classification        3
          3      1338.0           ml optimization using cognitive assistant        4
          4      1276.0           deploy your python model as a restful api        5

In [102]: rec_article_ids_test = np.array(rec_article_ids_test, dtype=np.float64)
          count_interactions = df.set_index('article_id').loc[rec_article_ids_test].reset_index(
          count_interactions.head()

Out[102]:    article_id                                            title  user_id
          0      1430.0  using pixiedust for fast, flexible, and easier...        1
          1      1430.0  using pixiedust for fast, flexible, and easier...       15
          2      1430.0  using pixiedust for fast, flexible, and easier...       33
          3      1430.0  using pixiedust for fast, flexible, and easier...       41
          4      1430.0  using pixiedust for fast, flexible, and easier...       21

In [103]: count_interactions = count_interactions.groupby('article_id').size().sort_values(ascen
          count_interactions
```

```
Out[103]: article_id
          1429.0    937
          1430.0    336
          dtype: int64

In [104]: list_articles_sorted = count_interactions.index
          list_articles_sorted = [str(x) for x in list_articles_sorted]
          list_articles_sorted

Out[104]: ['1429.0', '1430.0']

In [105]: # The function to sort the articles descending by the number of interactions
          def get_sorted_articles(article_ids, df=df):
              '''
              INPUT:
              article_ids - list of articles
              df - (pandas dataframe) df as defined at the top of the notebook

              OUTPUT:
              articles_sorted - list of articles that are sorted from the most interactions to t

              '''
              article_ids = np.array(article_ids, dtype=np.float64)
              count_interactions = df.set_index('article_id').loc[article_ids].reset_index(inpla
              count_interactions = count_interactions.groupby('article_id').size().sort_values(a
              articles_sorted = count_interactions.index
              articles_sorted = [str(x) for x in articles_sorted]

              return articles_sorted

In [106]: # Test
          get_sorted_articles(rec_article_ids_test, df)

Out[106]: ['1429.0', '1430.0']
```

b. Fill in the functions

```
In [107]: def get_top_sorted_users(user_id, df=df, user_item=user_item):
              '''
              INPUT:
              user_id - (int)
              df - (pandas dataframe) df as defined at the top of the notebook
              user_item - (pandas dataframe) matrix of users by articles:
                      1's when a user has interacted with an article, 0 otherwise


              OUTPUT:
              neighbors_df - (pandas dataframe) a dataframe with:
                          neighbor_id - is a neighbor user_id
```

120

```python
                    similarity - measure of the similarity of each user to the provide
                    num_interactions - the number of articles viewed by the user - if

        Other Details - sort the neighbors_df by the similarity and then by number of inte
                    highest of each is higher in the dataframe

        '''
        # Your code here
        user_article_i = user_item[user_item.index == user_id]
        neighbors_df = user_article_i.dot(user_item.transpose()).melt().sort_values(by = '
        neighbors_df.columns = ['user_id', 'similarity']
        neighbors_df['num_interactions'] = df.groupby(['user_id']).size().tolist()
        neighbors_df = neighbors_df[neighbors_df['user_id'] != user_id]
        neighbors_df = neighbors_df.sort_values(by = ['similarity','num_interactions'], as

        return neighbors_df # Return the dataframe specified in the doc_string


def user_user_recs_part2(user_id, m=10):
    '''
    INPUT:
    user_id - (int) a user id
    m - (int) the number of recommendations you want for the user

    OUTPUT:
    recs - (list) a list of recommendations for the user by article id
    rec_names - (list) a list of recommendations for the user by article title

    Description:
    Loops through the users based on closeness to the input user_id
    For each user - finds articles the user hasn't seen before and provides them as re
    Does this until m recommendations are found

    Notes:
    * Choose the users that have the most total article interactions
    before choosing those with fewer article interactions.

    * Choose articles with the articles with the most total interactions
    before choosing those with fewer total interactions.

    '''
    # Your code here
    closeness_list = get_top_sorted_users(user_id, df, user_item)['user_id'].tolist()
    r = 0
    recs = []
    read_article_ids, read_article_names = get_user_articles(user_id)

    for i in closeness_list:
```

121

```
                 if i == user_id:
                     continue
                 rec_article_ids , rec_article_names = get_user_articles(i)
                 rec_article_ids = get_sorted_articles(rec_article_ids, df)
                 for j in rec_article_ids:
                     if (j not in read_article_ids):
                         r = r + 1
                         recs.append(j)
                         if r == m:
                             break
                 if r == m:
                     break

          rec_names = get_article_names(recs, df=df)
          return recs, rec_names
```

In [108]: # Test
          get_top_sorted_users(3, df, user_item)

Out[108]:   user_id similarity num_interactions
          3352   3353   40.0   80
          22   23   23.0   364
          3781   3782   23.0   363
          97   98   17.0   170
          3763   3764   17.0   169
          202   203   16.0   160
          4458   4459   16.0   158
          48   49   16.0   147
          3696   3697   16.0   145
          51   52   15.0   132
          3595   3596   15.0   131
          911   912   14.0   102
          3539   3540   14.0   101
          241   242   13.0   148
          3909   3910   13.0   147
          130   131   13.0   145
          3869   3870   13.0   144
          203   204   13.0   97
          5137   5138   13.0   95
          2925   2926   13.0   83
          39   40   13.0   78
          4931   4932   13.0   76
          10   11   13.0   35
          3965   3966   13.0   33
          20   21   12.0   137
          4784   4785   12.0   136
          618   619   12.0   84
          194   195   12.0   72
```

```
        3577    3578         12.0                      70
         764     765         12.0                      68
         ...     ...          ...                     ...
        5055    5056          0.0                       1
        5059    5060          0.0                       1
        5064    5065          0.0                       1
        5067    5068          0.0                       1
        5070    5071          0.0                       1
        5072    5073          0.0                       1
        5075    5076          0.0                       1
        5083    5084          0.0                       1
        5084    5085          0.0                       1
        5086    5087          0.0                       1
        5090    5091          0.0                       1
        5091    5092          0.0                       1
        5097    5098          0.0                       1
        5099    5100          0.0                       1
        5100    5101          0.0                       1
        5103    5104          0.0                       1
        5106    5107          0.0                       1
        5112    5113          0.0                       1
        5115    5116          0.0                       1
        5118    5119          0.0                       1
        5119    5120          0.0                       1
        5120    5121          0.0                       1
        5121    5122          0.0                       1
        5124    5125          0.0                       1
        5130    5131          0.0                       1
        5140    5141          0.0                       1
        5143    5144          0.0                       1
        5146    5147          0.0                       1
        5147    5148          0.0                       1
        5148    5149          0.0                       1

        [5148 rows x 3 columns]
```

In [109]: *# Quick spot check - don't change this code - just use it to test your functions*
```python
rec_ids, rec_names = user_user_recs_part2(20, 10)
print("The top 10 recommendations for user 20 are the following article ids:")
print(rec_ids)
print()
print("The top 10 recommendations for user 20 are the following article names:")
print(rec_names)
```

```
The top 10 recommendations for user 20 are the following article ids:
['1330.0', '1427.0', '1364.0', '1170.0', '1162.0', '1304.0', '1351.0', '1160.0', '1354.0', '1368

The top 10 recommendations for user 20 are the following article names:
```

```
['insights from new york car accident reports', 'use xgboost, scikit-learn & ibm watson machine
```

5. Use your functions from above to correctly fill in the solutions to the dictionary below. Then test your dictionary against the solution. Provide the code you need to answer each following the comments below.

```
In [110]: get_top_sorted_users(1, df, user_item).head(1)

Out[110]:       user_id  similarity  num_interactions
          3932     3933        35.0                45

In [111]: get_top_sorted_users(131, df, user_item).head(10)

Out[111]:       user_id  similarity  num_interactions
          3869     3870        74.0               144
          3781     3782        39.0               363
          22         23        38.0               364
          202       203        33.0               160
          4458     4459        33.0               158
          97         98        29.0               170
          3763     3764        29.0               169
          48         49        29.0               147
          3696     3697        29.0               145
          241       242        25.0               148

In [112]: ### Tests with a dictionary of results

          user1_most_sim = get_top_sorted_users(1, df, user_item)['user_id'].tolist()[0] # Find
          user131_10th_sim = get_top_sorted_users(131, df, user_item)['user_id'].tolist()[9] # F

In [113]: ## Dictionary Test Here
          sol_5_dict = {
              'The user that is most similar to user 1.': user1_most_sim,
              'The user that is the 10th most similar to user 131': user131_10th_sim
          }

          t.sol_5_test(sol_5_dict)

This all looks good!  Nice job!
```

6. If we were given a new user, which of the above functions would you be able to use to make recommendations? Explain. Can you think of a better way we might make recommendations? Use the cell below to explain a better method for new users.

If we were given a new user, we cannot use user_user_recs_part2(new_user, m) because with a new user we cannot calculate the similarity of existing users to the new user. We should recommend the new user just the articles with the highest interactions and the function get_top_article_ids(m, df) can be used

7. Using your existing functions, provide the top 10 recommended articles you would provide for the a new user below. You can test your function against our thoughts to make sure we are all on the same page with how we might make a recommendation.

```
In [114]: new_user = 0
          user_user_recs_part2(new_user, 10)

Out[114]: ([], [])
```

The above result is ok because with a new user we cannot calculate the similarity of existing users to the new user.

```
In [115]: get_top_sorted_users(new_user, df, user_item).head()

Out[115]:       user_id  similarity  num_interactions
          22        NaN         NaN                364
          3781      NaN         NaN                363
          97        NaN         NaN                170
          3763      NaN         NaN                169
          202       NaN         NaN                160
```

Thus we should recommend the new user just the articles with the highest interactions.

```
In [116]: articles_new_user = df.groupby('article_id').size().sort_values(ascending = False).hea
          articles_new_user

Out[116]: article_id
          1429.0    937
          1330.0    927
          1431.0    671
          1427.0    643
          1364.0    627
          1314.0    614
          1293.0    572
          1170.0    565
          1162.0    512
          1304.0    483
          dtype: int64

In [117]: articles_new_user = articles_new_user.index
          articles_new_user = [str(x) for x in articles_new_user]
          articles_new_user

Out[117]: ['1429.0',
           '1330.0',
           '1431.0',
           '1427.0',
           '1364.0',
           '1314.0',
           '1293.0',
           '1170.0',
           '1162.0',
           '1304.0']
```

Or an existing function in Part II can be used to get this article ids list

```
In [118]: get_top_article_ids(10, df)

Out[118]: ['1429.0',
           '1330.0',
           '1431.0',
           '1427.0',
           '1364.0',
           '1314.0',
           '1293.0',
           '1170.0',
           '1162.0',
           '1304.0']
```

Fill in the result

```
In [119]: new_user = '0.0'

          # What would your recommendations be for this new user '0.0'?  As a new user, they hav
          # Provide a list of the top 10 article ids you would give to
          new_user_recs = ['1429.0',
           '1330.0',
           '1431.0',
           '1427.0',
           '1364.0',
           '1314.0',
           '1293.0',
           '1170.0',
           '1162.0',
           '1304.0'] # Copy from above or we can put get_top_article_ids(10, df) here

In [120]: assert set(new_user_recs) == set(['1314.0','1429.0','1293.0','1427.0','1162.0','1364.0

          print("That's right!  Nice job!")

That's right!  Nice job!
```

1.1.4   Part IV: Content Based Recommendations (EXTRA - NOT REQUIRED)

Another method we might use to make recommendations is to perform a ranking of the highest ranked articles associated with some term. You might consider content to be the **doc_body**, **doc_description**, or **doc_full_name**. There isn't one way to create a content based recommendation, especially considering that each of these columns hold content related information.

1. Use the function body below to create a content based recommender. Since there isn't one right answer for this recommendation tactic, no test functions are provided. Feel free to change the function inputs if you decide you want to try a method that requires more input values. The input values are currently set with one idea in mind that you may use to make content based

recommendations. One additional idea is that you might want to choose the most popular recommendations that meet your 'content criteria', but again, there is a lot of flexibility in how you might make these recommendations.

1.1.5 This part is NOT REQUIRED to pass this project. However, you may choose to take this on as an extra way to show off your skills.

```
In [ ]: def make_content_recs():
            '''
            INPUT:

            OUTPUT:

            '''
```

2. Now that you have put together your content-based recommendation system, use the cell below to write a summary explaining how your content based recommender works. Do you see any possible improvements that could be made to your function? Is there anything novel about your content based recommender?

1.1.6 This part is NOT REQUIRED to pass this project. However, you may choose to take this on as an extra way to show off your skills.

Write an explanation of your content based recommendation system here.

3. Use your content-recommendation system to make recommendations for the below scenarios based on the comments. Again no tests are provided here, because there isn't one right answer that could be used to find these content based recommendations.

1.1.7 This part is NOT REQUIRED to pass this project. However, you may choose to take this on as an extra way to show off your skills.

```
In [ ]: # make recommendations for a brand new user


        # make a recommendations for a user who only has interacted with article id '1427.0'
```

1.1.8 Part V: Matrix Factorization

In this part of the notebook, you will build use matrix factorization to make article recommendations to the users on the IBM Watson Studio platform.

1. You should have already created a **user_item** matrix above in **question 1** of **Part III** above. This first question here will just require that you run the cells to get things set up for the rest of **Part V** of the notebook.

```
In [121]: # Load the matrix here
          user_item_matrix = pd.read_pickle('user_item_matrix.p')

In [122]: # quick look at the matrix
          user_item_matrix.head()
```

```
Out[122]: article_id  0.0  100.0  1000.0  1004.0  1006.0  1008.0  101.0  1014.0  1015.0  \
          user_id
          1           0.0    0.0     0.0     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    0.0     0.0     0.0
          3           0.0    0.0     0.0     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    0.0     0.0     0.0
          5           0.0    0.0     0.0     0.0     0.0     0.0    0.0     0.0     0.0

          article_id  1016.0  ...   977.0  98.0  981.0  984.0  985.0  986.0  990.0  \
          user_id             ...
          1              0.0  ...     0.0   0.0    1.0    0.0    0.0    0.0    0.0
          2              0.0  ...     0.0   0.0    0.0    0.0    0.0    0.0    0.0
          3              0.0  ...     1.0   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    0.0    0.0
          5              0.0  ...     0.0   0.0    0.0    0.0    0.0    0.0    0.0

          article_id  993.0  996.0  997.0
          user_id
          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             0.0    0.0    0.0

          [5 rows x 714 columns]
```

2. In this situation, you can use Singular Value Decomposition from numpy on the user-item matrix. Use the cell to perform SVD, and explain why this is different than in the lesson.

```
In [123]: # Perform SVD on the User-Item Matrix Here

          u, s, vt = np.linalg.svd(user_item_matrix, full_matrices=True, compute_uv=True) # use

          u.shape, vt.shape, s.shape

Out[123]: ((5149, 5149), (714, 714), (714,))
```

So we can use SVD to do decomposition for the user_item_matrix.

In this result: - the u matrix is a square matrix with the number of rows and columns equaling the number of users, - the v transpose matrix is also a square matrix with the number of rows and columns equaling the number of articles (items), - and the sigma matrix is about the latent features.

In the lesson there existed just 4 latent features but here there exist a much bigger number of latent features but the SVD can help to do the singular value decomposition, then to compute the prediction and evaluate the accuracy of the prediction as in the next section (section 3).

3. Now for the tricky part, how do we choose the number of latent features to use? Running the below cell, you can see that as the number of latent features increases, we obtain a lower error rate on making predictions for the 1 and 0 values in the user-item matrix. Run the cell below to get an idea of how the accuracy improves as we increase the number of latent features.

128

```
In [124]: num_latent_feats = np.arange(10,700+10,20)
          sum_errs = []

          for k in num_latent_feats:
              # restructure with k latent features
              s_new, u_new, vt_new = np.diag(s[:k]), u[:, :k], vt[:k, :]

              # take dot product
              user_item_est = np.around(np.dot(np.dot(u_new, s_new), vt_new))

              # compute error for each prediction to actual value
              diffs = np.subtract(user_item_matrix, user_item_est)

              # total errors and keep track of them
              err = np.sum(np.sum(np.abs(diffs)))
              sum_errs.append(err)


          plt.plot(num_latent_feats, 1 - np.array(sum_errs)/df.shape[0]);
          plt.xlabel('Number of Latent Features');
          plt.ylabel('Accuracy');
          plt.title('Accuracy vs. Number of Latent Features');
```
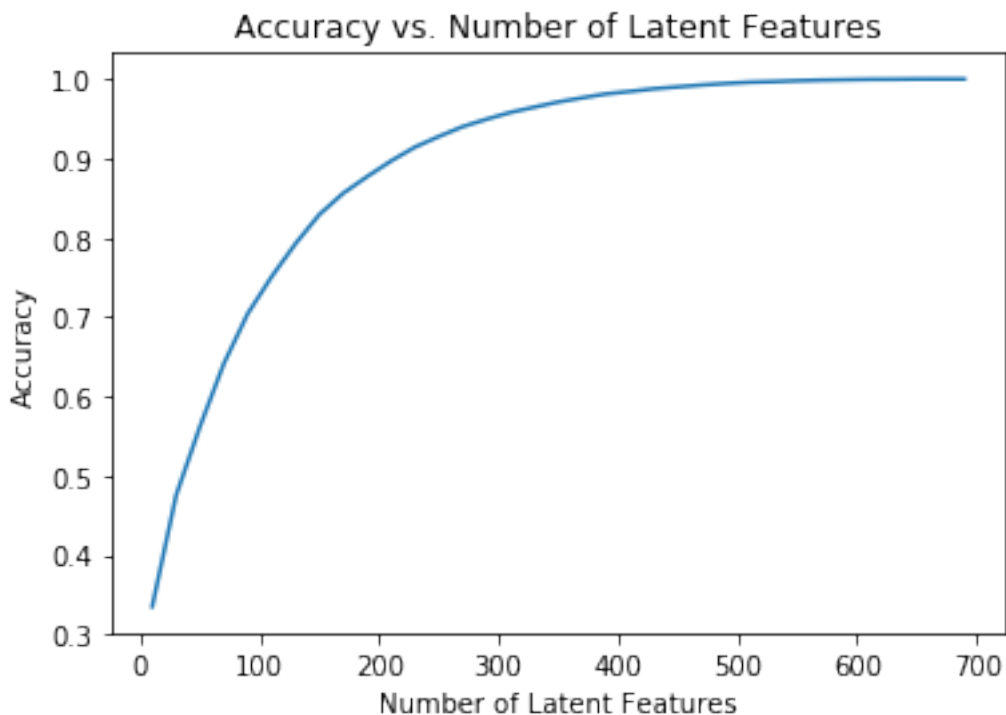


```
In [125]: user_item_est.shape[0]
```

```
Out[125]: 5149

In [126]: user_item_matrix.shape[0]

Out[126]: 5149

In [127]: user_item_est.shape[1]

Out[127]: 714

In [128]: user_item_matrix.shape[1]

Out[128]: 714
```

4. From the above, we can't really be sure how many features to use, because simply having a better way to predict the 1's and 0's of the matrix doesn't exactly give us an indication of if we are able to make good recommendations. Instead, we might split our dataset into a training and test set of data, as shown in the cell below.

Use the code from question 3 to understand the impact on accuracy of the training and test sets of data with different numbers of latent features. Using the split below:

- How many users can we make predictions for in the test set?

- How many users are we not able to make predictions for because of the cold start problem?
- How many articles can we make predictions for in the test set?

- How many articles are we not able to make predictions for because of the cold start problem?

```
In [129]: df_train = df.head(40000)
          df_test = df.tail(5993)

In [130]: df_train.head()

Out[130]:    article_id                                           title  user_id
          0      1430.0  using pixiedust for fast, flexible, and easier...        1
          1      1314.0         healthcare python streaming application demo        2
          2      1429.0            use deep learning for image classification        3
          3      1338.0           ml optimization using cognitive assistant        4
          4      1276.0            deploy your python model as a restful api        5

In [132]: user_item_train = create_user_item_matrix(df_train)
          user_item_train.head()

Out[132]: article_id  0.0     2.0     4.0     8.0     9.0     12.0    14.0    15.0    \
          user_id
          1               0.0     0.0     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     0.0     0.0
          3               0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0
          4               0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          5               0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
```

```
          article_id  16.0    18.0       ...   1434.0  1435.0  1436.0  1437.0  1439.0  \
          user_id                         ...
          1              0.0     0.0       ...      0.0     0.0     1.0     0.0     1.0
          2              0.0     0.0       ...      0.0     0.0     0.0     0.0     0.0
          3              0.0     0.0       ...      0.0     0.0     1.0     0.0     0.0
          4              0.0     0.0       ...      0.0     0.0     0.0     0.0     0.0
          5              0.0     0.0       ...      0.0     0.0     0.0     0.0     0.0


          article_id  1440.0  1441.0  1442.0  1443.0  1444.0
          user_id
          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     0.0     0.0     0.0
          4              0.0     0.0     0.0     0.0     0.0
          5              0.0     0.0     0.0     0.0     0.0


          [5 rows x 714 columns]

In [165]: user_item_test_all = create_user_item_matrix(df_test)
          user_item_test_all.head()

Out[165]: article_id  0.0     2.0     4.0     8.0     9.0     12.0    14.0    15.0    \
          user_id
          2917           0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          3024           0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0
          3093           0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          3193           0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          3527           0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0


          article_id  16.0    18.0       ...   1432.0  1433.0  1434.0  1435.0  1436.0  \
          user_id                         ...
          2917           0.0     0.0       ...      0.0     0.0     0.0     0.0     0.0
          3024           0.0     0.0       ...      0.0     0.0     0.0     0.0     0.0
          3093           0.0     0.0       ...      0.0     0.0     0.0     0.0     1.0
          3193           0.0     0.0       ...      0.0     0.0     0.0     0.0     0.0
          3527           0.0     0.0       ...      0.0     0.0     0.0     0.0     0.0


          article_id  1437.0  1439.0  1440.0  1441.0  1443.0
          user_id
          2917           0.0     0.0     0.0     0.0     0.0
          3024           0.0     0.0     0.0     0.0     0.0
          3093           0.0     0.0     0.0     0.0     0.0
          3193           0.0     0.0     0.0     0.0     0.0
          3527           0.0     0.0     0.0     0.0     0.0


          [5 rows x 574 columns]

In [145]: user_item_train.shape
```

```
Out[145]: (4487, 714)

In [166]: user_item_test_all.shape

Out[166]: (682, 574)

In [168]: test_article_ids_all = user_item_test_all.columns.tolist()
          test_article_ids_all = [str(x) for x in test_article_ids_all]
          test_article_ids_all

Out[168]: ['0.0',
           '2.0',
           '4.0',
           '8.0',
           '9.0',
           '12.0',
           '14.0',
           '15.0',
           '16.0',
           '18.0',
           '20.0',
           '25.0',
           '26.0',
           '28.0',
           '29.0',
           '30.0',
           '32.0',
           '33.0',
           '34.0',
           '36.0',
           '39.0',
           '40.0',
           '43.0',
           '48.0',
           '50.0',
           '51.0',
           '53.0',
           '54.0',
           '57.0',
           '58.0',
           '59.0',
           '60.0',
           '62.0',
           '64.0',
           '65.0',
           '68.0',
           '74.0',
           '76.0',
           '77.0',
```

```
'78.0',
'80.0',
'81.0',
'82.0',
'87.0',
'89.0',
'92.0',
'96.0',
'98.0',
'100.0',
'101.0',
'102.0',
'103.0',
'107.0',
'108.0',
'109.0',
'110.0',
'111.0',
'112.0',
'113.0',
'115.0',
'116.0',
'117.0',
'120.0',
'122.0',
'124.0',
'125.0',
'130.0',
'131.0',
'132.0',
'134.0',
'136.0',
'138.0',
'142.0',
'143.0',
'145.0',
'146.0',
'151.0',
'152.0',
'153.0',
'157.0',
'158.0',
'162.0',
'164.0',
'173.0',
'176.0',
'181.0',
'183.0',
```

```
'184.0',
'188.0',
'189.0',
'191.0',
'193.0',
'194.0',
'195.0',
'202.0',
'205.0',
'210.0',
'213.0',
'215.0',
'221.0',
'223.0',
'224.0',
'225.0',
'230.0',
'232.0',
'233.0',
'234.0',
'236.0',
'237.0',
'239.0',
'240.0',
'241.0',
'244.0',
'250.0',
'251.0',
'252.0',
'253.0',
'254.0',
'256.0',
'258.0',
'260.0',
'263.0',
'266.0',
'268.0',
'270.0',
'273.0',
'277.0',
'278.0',
'283.0',
'284.0',
'288.0',
'291.0',
'295.0',
'297.0',
'298.0',
```

```
'299.0',
'302.0',
'303.0',
'310.0',
'311.0',
'313.0',
'314.0',
'315.0',
'316.0',
'323.0',
'324.0',
'329.0',
'336.0',
'337.0',
'339.0',
'346.0',
'347.0',
'348.0',
'349.0',
'350.0',
'351.0',
'352.0',
'353.0',
'355.0',
'356.0',
'359.0',
'362.0',
'367.0',
'369.0',
'373.0',
'376.0',
'379.0',
'381.0',
'382.0',
'383.0',
'384.0',
'389.0',
'390.0',
'395.0',
'398.0',
'399.0',
'404.0',
'409.0',
'411.0',
'412.0',
'415.0',
'420.0',
'422.0',
```

```
'426.0',
'427.0',
'437.0',
'440.0',
'444.0',
'448.0',
'455.0',
'460.0',
'462.0',
'463.0',
'464.0',
'465.0',
'468.0',
'470.0',
'473.0',
'475.0',
'477.0',
'480.0',
'482.0',
'485.0',
'486.0',
'491.0',
'494.0',
'495.0',
'499.0',
'500.0',
'508.0',
'510.0',
'515.0',
'521.0',
'522.0',
'524.0',
'525.0',
'528.0',
'542.0',
'544.0',
'547.0',
'553.0',
'559.0',
'563.0',
'564.0',
'566.0',
'567.0',
'569.0',
'585.0',
'588.0',
'593.0',
'599.0',
```

```
'600.0',
'606.0',
'607.0',
'609.0',
'616.0',
'617.0',
'618.0',
'622.0',
'631.0',
'632.0',
'634.0',
'641.0',
'645.0',
'647.0',
'651.0',
'655.0',
'656.0',
'658.0',
'659.0',
'665.0',
'667.0',
'669.0',
'670.0',
'673.0',
'678.0',
'680.0',
'681.0',
'682.0',
'684.0',
'692.0',
'693.0',
'695.0',
'701.0',
'705.0',
'708.0',
'714.0',
'715.0',
'720.0',
'721.0',
'722.0',
'723.0',
'725.0',
'727.0',
'729.0',
'730.0',
'732.0',
'735.0',
'740.0',
```

'744.0',
'749.0',
'751.0',
'757.0',
'759.0',
'761.0',
'763.0',
'764.0',
'766.0',
'768.0',
'784.0',
'785.0',
'788.0',
'793.0',
'795.0',
'809.0',
'812.0',
'813.0',
'821.0',
'825.0',
'833.0',
'843.0',
'844.0',
'846.0',
'853.0',
'855.0',
'857.0',
'858.0',
'861.0',
'862.0',
'864.0',
'865.0',
'868.0',
'870.0',
'871.0',
'876.0',
'878.0',
'880.0',
'881.0',
'882.0',
'884.0',
'891.0',
'892.0',
'898.0',
'903.0',
'905.0',
'906.0',
'910.0',

```
'911.0',
'919.0',
'926.0',
'928.0',
'930.0',
'932.0',
'933.0',
'935.0',
'936.0',
'939.0',
'941.0',
'943.0',
'949.0',
'951.0',
'952.0',
'955.0',
'957.0',
'958.0',
'959.0',
'962.0',
'965.0',
'967.0',
'968.0',
'969.0',
'972.0',
'973.0',
'975.0',
'977.0',
'981.0',
'986.0',
'990.0',
'993.0',
'996.0',
'997.0',
'1000.0',
'1006.0',
'1008.0',
'1014.0',
'1016.0',
'1017.0',
'1018.0',
'1024.0',
'1025.0',
'1028.0',
'1030.0',
'1035.0',
'1038.0',
'1042.0',
```

'1044.0',
'1047.0',
'1048.0',
'1050.0',
'1051.0',
'1052.0',
'1053.0',
'1054.0',
'1055.0',
'1056.0',
'1057.0',
'1058.0',
'1059.0',
'1060.0',
'1065.0',
'1068.0',
'1070.0',
'1071.0',
'1073.0',
'1074.0',
'1077.0',
'1079.0',
'1080.0',
'1083.0',
'1120.0',
'1121.0',
'1123.0',
'1124.0',
'1125.0',
'1137.0',
'1139.0',
'1140.0',
'1141.0',
'1142.0',
'1143.0',
'1144.0',
'1145.0',
'1146.0',
'1147.0',
'1148.0',
'1149.0',
'1150.0',
'1151.0',
'1152.0',
'1153.0',
'1154.0',
'1155.0',
'1157.0',

```
'1158.0',
'1159.0',
'1160.0',
'1161.0',
'1162.0',
'1163.0',
'1164.0',
'1165.0',
'1166.0',
'1169.0',
'1170.0',
'1171.0',
'1172.0',
'1173.0',
'1174.0',
'1175.0',
'1176.0',
'1177.0',
'1178.0',
'1179.0',
'1180.0',
'1181.0',
'1183.0',
'1184.0',
'1185.0',
'1186.0',
'1187.0',
'1189.0',
'1190.0',
'1191.0',
'1192.0',
'1196.0',
'1198.0',
'1219.0',
'1225.0',
'1232.0',
'1254.0',
'1261.0',
'1263.0',
'1267.0',
'1271.0',
'1273.0',
'1274.0',
'1276.0',
'1277.0',
'1278.0',
'1279.0',
'1280.0',
```

```
'1282.0',
'1283.0',
'1285.0',
'1286.0',
'1289.0',
'1290.0',
'1291.0',
'1292.0',
'1293.0',
'1296.0',
'1297.0',
'1298.0',
'1299.0',
'1304.0',
'1305.0',
'1306.0',
'1307.0',
'1313.0',
'1314.0',
'1315.0',
'1317.0',
'1318.0',
'1319.0',
'1320.0',
'1321.0',
'1324.0',
'1326.0',
'1328.0',
'1329.0',
'1330.0',
'1331.0',
'1332.0',
'1333.0',
'1336.0',
'1337.0',
'1338.0',
'1343.0',
'1349.0',
'1350.0',
'1351.0',
'1354.0',
'1356.0',
'1357.0',
'1358.0',
'1359.0',
'1360.0',
'1361.0',
'1362.0',
```

```
'1364.0',
'1365.0',
'1366.0',
'1367.0',
'1368.0',
'1377.0',
'1378.0',
'1386.0',
'1387.0',
'1391.0',
'1393.0',
'1394.0',
'1395.0',
'1396.0',
'1397.0',
'1398.0',
'1400.0',
'1401.0',
'1402.0',
'1403.0',
'1404.0',
'1405.0',
'1406.0',
'1407.0',
'1408.0',
'1409.0',
'1410.0',
'1411.0',
'1412.0',
'1414.0',
'1415.0',
'1416.0',
'1418.0',
'1419.0',
'1420.0',
'1422.0',
'1423.0',
'1424.0',
'1425.0',
'1426.0',
'1427.0',
'1428.0',
'1429.0',
'1430.0',
'1431.0',
'1432.0',
'1433.0',
'1434.0',
```

```
              '1435.0',
              '1436.0',
              '1437.0',
              '1439.0',
              '1440.0',
              '1441.0',
              '1443.0']
```

In [169]: `test_user_ids_all = user_item_test_all.index.tolist()`
`test_user_ids_all[:5]`

Out[169]: `[2917, 3024, 3093, 3193, 3527]`

In [148]: `train_user_ids = user_item_train.index.tolist()`
`len(train_user_ids)`

Out[148]: 4487

In [170]: `len(test_user_ids_all)`

Out[170]: 682

In [171]: `list(set(test_user_ids_all) & set(train_user_ids))`

Out[171]: [3968,
          3777,
          4002,
          3684,
          4293,
          2917,
          4487,
          4231,
          3527,
          4204,
          3532,
          3024,
          4274,
          3801,
          3093,
          3989,
          3990,
          3193,
          3740,
          3998]

In [172]: `test_user_ids_intersection = list(set(test_user_ids_all) & set(train_user_ids))`
`len(test_user_ids_intersection)`

Out[172]: 20

```
In [173]: user_item_test_all.index

Out[173]: Int64Index([2917, 3024, 3093, 3193, 3527, 3532, 3684, 3740, 3777, 3801,
                       ...
                       5140, 5141, 5142, 5143, 5144, 5145, 5146, 5147, 5148, 5149],
                      dtype='int64', name='user_id', length=682)

In [174]: user_item_test_all.index.isin(test_user_ids_intersection)

Out[174]: array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
                  True,  True,  True,  True,  True,  True,  True,  True,  True,
                  True,  True, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False, False,
```

```
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False, False, False,
            False, False, False, False, False, False, False], dtype=bool)

In [175]: user_item_test_intersection = user_item_test_all[user_item_test_all.index.isin(test_us
          user_item_test_intersection

Out[175]: article_id  0.0     2.0     4.0     8.0     9.0     12.0    14.0    15.0    \
          user_id
          2917           0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          3024           0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0
          3093           0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          3193           0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          3527           0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
```

146

```
3532       0.0      0.0      0.0      0.0      0.0      1.0      0.0      0.0
3684       0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
3740       0.0      0.0      0.0      0.0      0.0      1.0      0.0      0.0
3777       0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
3801       0.0      0.0      0.0      0.0      0.0      1.0      0.0      0.0
3968       0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
3989       0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
3990       0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
3998       0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
4002       0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
4204       0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
4231       0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
4274       0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
4293       0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
4487       0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0

article_id 16.0     18.0     ...      1432.0   1433.0   1434.0   1435.0   1436.0   \
user_id                       ...
2917       0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
3024       0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
3093       0.0      0.0      ...      0.0      0.0      0.0      0.0      1.0
3193       0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
3527       0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
3532       0.0      0.0      ...      0.0      0.0      0.0      0.0      1.0
3684       0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
3740       0.0      0.0      ...      0.0      0.0      0.0      0.0      1.0
3777       0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
3801       0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
3968       0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
3989       0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
3990       0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
3998       0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
4002       0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
4204       0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
4231       0.0      0.0      ...      0.0      0.0      0.0      0.0      1.0
4274       0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0
4293       0.0      0.0      ...      0.0      0.0      0.0      0.0      1.0
4487       0.0      0.0      ...      0.0      0.0      0.0      0.0      0.0

article_id 1437.0   1439.0   1440.0   1441.0   1443.0
user_id
2917       0.0      0.0      0.0      0.0      0.0
3024       0.0      0.0      0.0      0.0      0.0
3093       0.0      0.0      0.0      0.0      0.0
3193       0.0      0.0      0.0      0.0      0.0
3527       0.0      0.0      0.0      0.0      0.0
3532       0.0      0.0      0.0      0.0      0.0
3684       0.0      0.0      0.0      0.0      0.0
```

```
3740            0.0      0.0      0.0      0.0      0.0
3777            0.0      0.0      0.0      0.0      0.0
3801            0.0      0.0      0.0      0.0      0.0
3968            0.0      0.0      0.0      0.0      0.0
3989            0.0      0.0      0.0      0.0      0.0
3990            0.0      0.0      0.0      0.0      0.0
3998            0.0      0.0      0.0      0.0      0.0
4002            0.0      0.0      0.0      0.0      0.0
4204            0.0      0.0      0.0      0.0      0.0
4231            0.0      0.0      0.0      0.0      0.0
4274            0.0      0.0      0.0      0.0      0.0
4293            0.0      0.0      0.0      0.0      0.0
4487            0.0      0.0      0.0      0.0      0.0

[20 rows x 574 columns]
```

```python
In [176]: df_train = df.head(40000)
          df_test = df.tail(5993)

          def create_test_and_train_user_item(df_train, df_test):
              '''
              INPUT:
              df_train - training dataframe
              df_test - test dataframe

              OUTPUT:
              user_item_train - a user-item matrix of the training dataframe
                                (unique users for each row and unique articles for each column)
              user_item_test - a user-item matrix of the testing dataframe
                               (unique users for each row and unique articles for each column)
              test_idx - all of the test user ids
              test_arts - all of the test article ids


              '''
              # Your code here
              user_item_train = create_user_item_matrix(df_train)
              user_item_test_all = create_user_item_matrix(df_test)
              test_idx_all = user_item_test_all.index.tolist()
              train_idx = user_item_train.index.tolist()

              test_idx = list(set(test_idx_all) & set(train_idx))
              user_item_test = user_item_test_all[user_item_test_all.index.isin(test_idx)]

              test_arts = user_item_test.columns.tolist()
              test_arts = [str(x) for x in test_arts]

              return user_item_train, user_item_test, test_idx, test_arts
```

148

```
          user_item_train, user_item_test, test_idx, test_arts = create_test_and_train_user_item
```

In [137]: user_item_train.shape[0]

Out[137]: 4487

In [138]: user_item_train.shape[1]

Out[138]: 714

In [177]: user_item_test.head()

Out[177]: article_id 0.0   2.0   4.0   8.0   9.0   12.0   14.0   15.0   \
          user_id
          2917   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
          3024   0.0   0.0   0.0   0.0   0.0   1.0   0.0   0.0
          3093   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
          3193   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
          3527   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0

          article_id 16.0   18.0   ...   1432.0 1433.0 1434.0 1435.0 1436.0 \
          user_id   ...
          2917   0.0   0.0   ...   0.0   0.0   0.0   0.0   0.0
          3024   0.0   0.0   ...   0.0   0.0   0.0   0.0   0.0
          3093   0.0   0.0   ...   0.0   0.0   0.0   0.0   1.0
          3193   0.0   0.0   ...   0.0   0.0   0.0   0.0   0.0
          3527   0.0   0.0   ...   0.0   0.0   0.0   0.0   0.0

          article_id 1437.0 1439.0 1440.0 1441.0 1443.0
          user_id
          2917   0.0   0.0   0.0   0.0   0.0
          3024   0.0   0.0   0.0   0.0   0.0
          3093   0.0   0.0   0.0   0.0   0.0
          3193   0.0   0.0   0.0   0.0   0.0
          3527   0.0   0.0   0.0   0.0   0.0

          [5 rows x 574 columns]

In [178]: len(test_idx)

Out[178]: 20

In [179]: user_item_test.shape[0]

Out[179]: 20

In [192]: user_item_test
```

```
Out[192]: article_id  0.0     2.0     4.0     8.0     9.0     12.0    14.0    15.0    \
          user_id
          2917          0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          3024          0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0
          3093          0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          3193          0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          3527          0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          3532          0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0
          3684          0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          3740          0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0
          3777          0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          3801          0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0
          3968          0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          3989          0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          3990          0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          3998          0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          4002          0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          4204          0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          4231          0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          4274          0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          4293          0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
          4487          0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0

          article_id  16.0    18.0    ...     1432.0  1433.0  1434.0  1435.0  1436.0  \
          user_id                     ...
          2917          0.0     0.0     ...      0.0     0.0     0.0     0.0     0.0
          3024          0.0     0.0     ...      0.0     0.0     0.0     0.0     0.0
          3093          0.0     0.0     ...      0.0     0.0     0.0     0.0     1.0
          3193          0.0     0.0     ...      0.0     0.0     0.0     0.0     0.0
          3527          0.0     0.0     ...      0.0     0.0     0.0     0.0     0.0
          3532          0.0     0.0     ...      0.0     0.0     0.0     0.0     1.0
          3684          0.0     0.0     ...      0.0     0.0     0.0     0.0     0.0
          3740          0.0     0.0     ...      0.0     0.0     0.0     0.0     1.0
          3777          0.0     0.0     ...      0.0     0.0     0.0     0.0     0.0
          3801          0.0     0.0     ...      0.0     0.0     0.0     0.0     0.0
          3968          0.0     0.0     ...      0.0     0.0     0.0     0.0     0.0
          3989          0.0     0.0     ...      0.0     0.0     0.0     0.0     0.0
          3990          0.0     0.0     ...      0.0     0.0     0.0     0.0     0.0
          3998          0.0     0.0     ...      0.0     0.0     0.0     0.0     0.0
          4002          0.0     0.0     ...      0.0     0.0     0.0     0.0     0.0
          4204          0.0     0.0     ...      0.0     0.0     0.0     0.0     0.0
          4231          0.0     0.0     ...      0.0     0.0     0.0     0.0     1.0
          4274          0.0     0.0     ...      0.0     0.0     0.0     0.0     0.0
          4293          0.0     0.0     ...      0.0     0.0     0.0     0.0     1.0
          4487          0.0     0.0     ...      0.0     0.0     0.0     0.0     0.0

          article_id  1437.0  1439.0  1440.0  1441.0  1443.0
          user_id
```

```
2917              0.0       0.0       0.0       0.0       0.0
3024              0.0       0.0       0.0       0.0       0.0
3093              0.0       0.0       0.0       0.0       0.0
3193              0.0       0.0       0.0       0.0       0.0
3527              0.0       0.0       0.0       0.0       0.0
3532              0.0       0.0       0.0       0.0       0.0
3684              0.0       0.0       0.0       0.0       0.0
3740              0.0       0.0       0.0       0.0       0.0
3777              0.0       0.0       0.0       0.0       0.0
3801              0.0       0.0       0.0       0.0       0.0
3968              0.0       0.0       0.0       0.0       0.0
3989              0.0       0.0       0.0       0.0       0.0
3990              0.0       0.0       0.0       0.0       0.0
3998              0.0       0.0       0.0       0.0       0.0
4002              0.0       0.0       0.0       0.0       0.0
4204              0.0       0.0       0.0       0.0       0.0
4231              0.0       0.0       0.0       0.0       0.0
4274              0.0       0.0       0.0       0.0       0.0
4293              0.0       0.0       0.0       0.0       0.0
4487              0.0       0.0       0.0       0.0       0.0

[20 rows x 574 columns]
```

In [182]: user_item_test_all.shape[0]

Out[182]: 682

In [183]: user_item_test_all.shape[0] - len(test_idx)

Out[183]: 662

In [180]: len(test_arts)

Out[180]: 574

In [184]: user_item_test_all.shape[1]

Out[184]: 574

In [185]: user_item_test_all.shape[1] - len(test_arts)

Out[185]: 0

```
In [187]: # Replace the values in the dictionary below
          a = 662
          b = 574
          c = 20
          d = 0
```

```
sol_4_dict = {
    'How many users can we make predictions for in the test set?': c,
    'How many users in the test set are we not able to make predictions for because of
    'How many movies can we make predictions for in the test set?': b,
    'How many movies in the test set are we not able to make predictions for because o
}

t.sol_4_test(sol_4_dict)
```

Awesome job! That's right! All of the test movies are in the training data, but there are only

5. Now use the **user_item_train** dataset from above to find U, S, and V transpose using SVD. Then find the subset of rows in the **user_item_test** dataset that you can predict using this matrix decomposition with different numbers of latent features to see how many features makes sense to keep based on the accuracy on the test data. This will require combining what was done in questions 2 - 4.

Use the cells below to explore how well SVD works towards making predictions for recommendations on the test data.

- **Decompose the training set user-item matrix**

```
In [188]: # fit SVD on the user_item_train matrix
          u_train, s_train, vt_train = np.linalg.svd(user_item_train, full_matrices=True, comput

          s_train.shape, u_train.shape, vt_train.shape
```

```
Out[188]: ((714,), (4487, 4487), (714, 714))
```

- **Use this decomposition to predict values for the user-item test dataset and check the accuracy of the predictions from the train set and compared with the test set**

```
In [200]: # Use these cells to see how well you can use the training decomposition to predict on
          num_latent_feats_2 = np.arange(10,700+10,20)
          sum_errs_2 = []

          for k_2 in num_latent_feats_2:
              # restructure with k latent features
              s_new_2, u_new_2, vt_new_2 = np.diag(s_train[:k_2]), u_train[:, :k_2], vt_train[:k

              # take dot product
              user_item_est_2 = np.around(np.dot(np.dot(u_new_2, s_new_2), vt_new_2))

              # compute error for each prediction to actual value
              diffs_2 = np.subtract(user_item_train, user_item_est_2)

              # total errors and keep track of them
              err_2 = np.sum(np.sum(np.abs(diffs_2)))
              sum_errs_2.append(err_2)
```

```python
u_test, s_test, vt_test = np.linalg.svd(user_item_test, full_matrices=True, compute_uv

num_latent_feats_3 = np.arange(10,20+10,20)
sum_errs_3 = []

for k_3 in num_latent_feats_3:
    # restructure with k latent features
    s_new_3, u_new_3, vt_new_3 = np.diag(s_test[:k_3]), u_test[:, :k_3], vt_test[:k_3,

    # take dot product
    user_item_est_3 = np.around(np.dot(np.dot(u_new_3, s_new_3), vt_new_3))

    # compute error for each prediction to actual value
    diffs_3 = np.subtract(user_item_test, user_item_est_3)

    # total errors and keep track of them
    err_3 = np.sum(np.sum(np.abs(diffs_3)))
    sum_errs_3.append(err_3)

num_latent_feats_4 = num_latent_feats_2
#sum_errs_4 = np.append(np.array(sum_errs_3), [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]) + np.arr
sum_errs_4 = np.array(sum_errs_3) + np.array(sum_errs_2)


plt.plot(num_latent_feats_2, 1 - np.array(sum_errs_2)/df.shape[0], label = 'train');
plt.plot(num_latent_feats_3, 1 - np.array(sum_errs_3)/df.shape[0], label = 'test');
plt.plot(num_latent_feats_4, 1 - np.array(sum_errs_4)/df.shape[0], label = 'total');
plt.legend();
plt.xlabel('Number of Latent Features');
plt.ylabel('Accuracy');
plt.title('Accuracy vs. Number of Latent Features');
```

Accuracy vs. Number of Latent Features
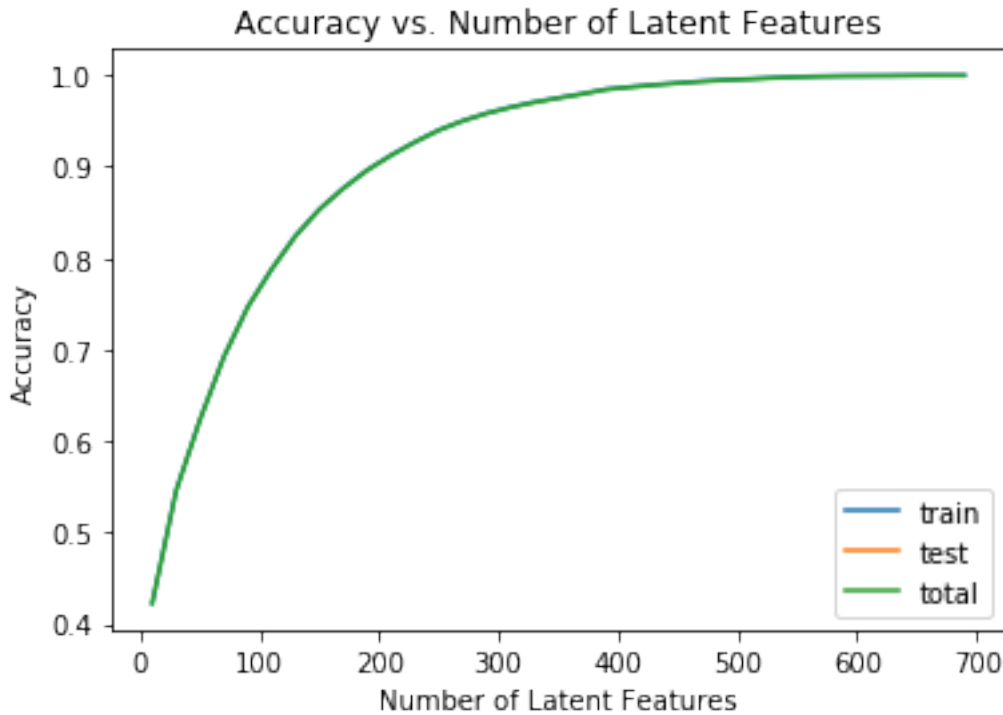
```
In [188]: user_item_est.shape[0]

Out[188]: 4487

In [189]: user_item_test.shape[0]

Out[189]: 682

In [190]: user_item_est.shape[1]

Out[190]: 714

In [191]: user_item_test.shape[1]

Out[191]: 574
```

6. Use the cell below to comment on the results you found in the previous question. Given the circumstances of your results, discuss what you might do to determine if the recommendations you make with any of the above recommendation systems are an improvement to how users currently find articles?

The accuracy is a good metric to use here in this case. It can provide us with a fair assessment of the model's performance when we can evaluate . But the current assessment framework is not robust enough to make conclusive results about the model as comparing the accuracy in section 3. and 5. And consider about the number of train and test users, we cannot make predictions for all users in the test set, it's not sufficient. It's understandable as we see the

similarity we base on is very simple, the users are considered similar to each other just by only 1 metric that's the history of article the users interacted with.

I see we can use an online evaluation technique like A/B testing here to improve. I will separate the user groups base on device types, cookies, IP addresses... that show up more information such as location, OS, device, time... And the experiment should be run for the time base at least a month, then more metrics of user similarity will be tracked during this experiment such as location, timing behavior... Those new metrics will help to improve the prediction results more conclusive.

Extras Using your workbook, you could now save your recommendations for each user, develop a class to make new predictions and update your results, and make a flask app to deploy your results. These tasks are beyond what is required for this project. However, from what you learned in the lessons, you certainly capable of taking these tasks on to improve upon your work here!

1.2   Conclusion

Congratulations! You have reached the end of the Recommendations with IBM project!

Tip: Once you are satisfied with your work here, check over your report to make sure that it is satisfies all the areas of the rubric. You should also probably remove all of the "Tips" like this one so that the presentation is as polished as possible.

1.3   Directions to Submit

Before you submit your project, you need to create a .html or .pdf version of this notebook in the workspace here. To do that, run the code cell below. If it worked correctly, you should get a return code of 0, and you should see the generated .html file in the workspace directory (click on the orange Jupyter icon in the upper left).

Alternatively, you can download this report as .html via the **File** > **Download as** submenu, and then manually upload it into the workspace directory by clicking on the orange Jupyter icon in the upper left, then using the Upload button.

Once you've done this, you can submit your project by clicking on the "Submit Project" button in the lower right here. This will create and submit a zip file with this .ipynb doc and the .html or .pdf version you created. Congratulations!

```
In [ ]: from subprocess import call
        call(['python', '-m', 'nbconvert', 'Recommendations_with_IBM.ipynb'])
```