Recommendations_with_IBM

November 1, 2024

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 [398]: import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          %matplotlib inline
          import project_tests as t
          import pickle
```

```
In [399]: df = pd.read_csv('data/user-item-interactions.csv')
          df_content = pd.read_csv('data/articles_community.csv')
          df.head()
```

```
Out[399]:    Unnamed: 0  article_id                                             title  \
          0           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

                                                email
          0  ef5f11f77ba020cd36e1105a00ab868bbdbf7fe7
          1  083cbdfa93c8444beaa4c5f5e0f5f9198e4f9e0b
          2  b96a4f2e92d8572034b1e9b28f9ac673765cd074
```

```
3  06485706b34a5c9bf2a0ecdac41daf7e7654ceb7
4  f01220c46fc92c6e6b161b1849de11faacd7ccb2
```

```
In [400]: df_content.head()
```

```
Out[400]:    Unnamed: 0                                           doc_body  \
          0           3  Skip navigation Sign in SearchLoading...\r\n\r...
          1           5  No Free Hunch Navigation * kaggle.com\r\n\r\n ...
          2           7   * Login\r\n * Sign Up\r\n\r\n * Learning Pat...
          3           8  DATALAYER: HIGH THROUGHPUT, LOW LATENCY AT SCA...
          4          12  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
```

```
In [401]: del df['Unnamed: 0']
          del df_content['Unnamed: 0']
```

```
In [402]: df.head()
```

```
Out[402]:    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 [403]: df_content.head()
```

```
Out[403]:                                             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.

```
In [404]: # cols and rows
          df.shape

Out[404]: (45993, 3)

In [405]: df.groupby('email')['article_id'].count().describe()

Out[405]: count    5148.000000
          mean        8.930847
          std        16.802267
          min         1.000000
          25%         1.000000
          50%         3.000000
          75%         9.000000
          max       364.000000
          Name: article_id, dtype: float64
```

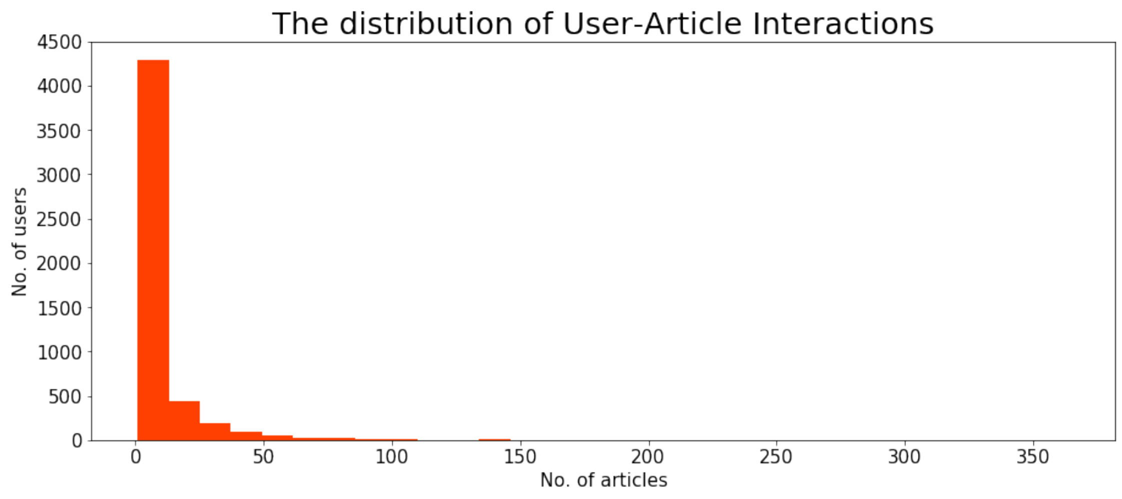
- Every user in the dataframe `df` engages with a atleast one article, and 25% of users have only a single interaction.
- The median number of interactions is three (50th percentile: 3), and 75% of users have nine interactions or fewer (75th percentile: 9).

- The arithmetic mean of the distribution is approximately nine (mean: 8.93), suggesting a positive skew. Users in the top quartile exhibit over nine interactions, with the most active user demonstrating a total of 364 interactions (maximum: 364).

```
In [406]: df.groupby('email')['article_id'].count().median()

Out[406]: 3.0

In [407]: df.groupby('email')['article_id'].count().max()

Out[407]: 364

In [408]: # Fill in the median and maximum number of user_article interactios below
          median_val = 3 # 50% of individuals interact with _3_ number of articles or fewer.
          max_views_by_user = 364 # The maximum number of user-article interactions by any 1 use

In [409]: user_interacts = df.groupby('email')['article_id'].count()
          user_interacts

Out[409]: email
          0000b6387a0366322d7fbfc6434af145adf7fed1    13
          001055fc0bb67f71e8fa17002342b256a30254cd     4
          00148e4911c7e04eeff8def7bbbdaf1c59c2c621     3
          001a852ecbd6cc12ab77a785efa137b2646505fe     6
          001fc95b90da5c3cb12c501d201a915e4f093290     2
          0042719415c4fca7d30bd2d4e9d17c5fc570de13     2
          00772abe2d0b269b2336fc27f0f4d7cb1d2b65d7     3
          008ba1d5b4ebf54babf516a2d5aa43e184865da5    10
          008ca24b82c41d513b3799d09ae276d37f92ce72     1
          008dfc7a327b5186244caec48e0ab61610a0c660    13
          009af4e0537378bf8e8caf0ad0e2994f954d822e     1
          00bda305223d05f6df5d77de41abd2a0c7d895fe     4
          00c2d5190e8c6b821b0e3848bf56f6e47e428994     3
          00ced21f957bbcee5edf7b107b2bd05628b04774     4
          00d9337ecd5f70fba1c4c7a78e21b3532e0112c4     3
          00e524e4f13137a6fac54f9c71d7769c6507ecde    11
          00f8341cbecd6af00ba8c78b3bb6ec49adf83248     3
          00f946b14100f0605fa25089437ee9486378872c     1
          01041260c97ab9221d923b0a2c525437f148d589     2
          0108ce3220657a9a89a85bdec959b0f2976dd51c     4
          011455e91a24c1fb815a4deac6b6eaf5ad16819e     9
          01198c58d684d79c9026abe355cfb532cb524dc5     1
          011ae4de07ffb332b0f51c155a35c23c80294962    35
          011fcfb582be9534e9a275336f7e7c3717100381    11
          0129dfcdb701b6e1d309934be6393004c6683a2d    15
          01327bbc4fd7bfe8ad62e599453d2876b928e725     3
          01455f0ab0a5a22a93d94ad35f6e78431aa90625     7
          014dedab269f1453c647598c92a3fa37b39eed97     2
          014e4fe6e6c5eb3fe5ca0b16c16fb4599df6375c     1
```

```
01560f88312a91894d254e6406c25df19f0ad5e8    11
                                            ..
fe5396e3762c36767c9c915f7ed1731691d7e4b4     1
fe5480ff15f0ac51eeb2314a192351f168d7aad7     1
fe56a49b62752708ed2f6e30677c57881f7b78d1    15
fe5885b80e91be887510a0b6dd04e011178d6364     3
fe5f9d7528518e00b0a73c7a3994afc335496961     3
fe66aa534c7824eca663b84b99a437a98a9b026e     2
fe69c72c964a8346dbc7763309c4e07d818d360f     4
fe88d1f683f308b32fb3d7554f007cc55cc48df5     1
fe8c1cb974e39d8ea8c005044e927b3f0de8acd0     3
fe90d98b0287090fe8e653bafba6ed3eff19331e     1
fe9327be39fd457df70e83d3fc8cba9b8b3f95b1     1
feaea388105a4ccc48795b191bbf0c26a23b1356     4
fef0c6be3a2ed226e1fb8a811b0ee68a389f6f3c    13
fef28e45f7217026b2684d1783a2e18b061bdffb     3
fef3bc88def1aa787c99957ded7d5b2c0edc040e     4
ff27ffd93e21154b8a9cf2722f2cc0f75dc39eff     1
ff288722b76eba5209cdbf9158c6dfbf229b9129     1
ff452614b91f4c9bd965150b1a82e7bf18f59334     2
ff4d3e1c359cfbb73bcae07fa1eb62c45da2b161     3
ff55d0c0b2a4f56aae87c2a21afb7070ab34383d     1
ff6e82c763fe2443643e48a03e239eb635f406dc    14
ff7a0f59ba022102ad22981141a7182c4d8273c3     7
ff833869969184d86f870f98405e7988eccc2309     9
ff979e07f9d906a32ba35a9b75fd9585f6306dbc    38
ffaefa3a1bc2d074d9a14c9924d4e67a46c35410     1
ffc6cfa435937ca0df967b44e9178439d04e3537     2
ffc96f8fbb35aac4cb0029332b0fc78e7766bb5d     4
ffe3d0543c9046d35c2ee3724ea9d774dff98a32    32
fff9fc3ec67bd18ed57a34ed1e67410942c4cd81    10
fffb93a166547448a0ff0232558118d59395fecd    13
Name: article_id, Length: 5148, dtype: int64
```

```
In [410]: # plot graph
          plt.figure(figsize=(15,6))
          user_interacts.plot(kind='hist', color='#ff4000', bins=30)
          plt.title('The distribution of User-Article Interactions', fontsize=25)
          plt.xlabel("No. of articles",fontsize=15)
          plt.ylabel("No. of users",fontsize=15)
          plt.xticks(fontsize=15)
          plt.yticks(fontsize=15)
          plt.show();
```



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

```
In [411]: df_content.head()
```

```
Out[411]:                                             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
```

```
In [412]: df_content.shape
```

```
Out[412]: (1056, 5)
```

```
In [413]: df_content.article_id.duplicated().sum()
```

```
Out[413]: 5
```

```
In [414]: df_content[df_content.article_id.duplicated()]

Out[414]:                                               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 [415]: ids = df_content['article_id']

In [416]: # explore duplicate articles
          df_content[ids.isin(ids[ids.duplicated()])]

Out[416]:                                               doc_body  \
          50   Follow Sign in / Sign up Home About Insight Da...
          221  * United States\r\n\r\nIBMő * Site map\r\n\r\n...
          232  Homepage Follow Sign in Get started Homepage *...
          365  Follow Sign in / Sign up Home About Insight Da...
          399  Homepage Follow Sign in Get started * Home\r\n...
          578  This video shows you how to construct queries ...
          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  \
          50                        Community Detection at Scale
          221  When used to make sense of huge amounts of con...
          232  If you are like most data scientists, you are ...
          365  During the seven-week Insight Data Engineering...
          399  Todays world of data science leverages data f...
          578  This video shows you how to construct queries ...
          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
50                        Graph-based machine learning       Live          50
221  How smart catalogs can turn the big data flood...       Live         221
232  Self-service data preparation with IBM Data Re...       Live         232
365                       Graph-based machine learning       Live          50
399  Using Apache Spark as a parallel processing fr...       Live         398
578                              Use the Primary Index       Live         577
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 [417]: # Remove any rows that have the same article_id - only keep the first
          df_content.drop_duplicates(subset=['article_id'], keep='first', inplace=True)
          df_content
```

```
Out[417]:                                               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...
     5     Compose is all about immediacy. You want a new...
     6     UPGRADING YOUR POSTGRESQL TO 9.5Share on Twitt...
     7     Follow Sign in / Sign up 135 8 * Share\r\n * 1...
     8     * Host\r\n * Competitions\r\n * Datasets\r\n *...
     9     THE GRADIENT FLOW\r\nDATA / TECHNOLOGY / CULTU...
     10    OFFLINE-FIRST IOS APPS WITH SWIFT & PART 1: TH...
     11    Warehousing data from Cloudant to dashDB great...
     12    Skip to main content IBM developerWorks / Deve...
     13    Maureen McElaney Blocked Unblock Follow Follow...
     14    Raj Singh Blocked Unblock Follow Following Dev...
     15    * Home\r\n * Community\r\n * Projects\r\n * Bl...
     16    * Home\r\n * Research\r\n * Partnerships and C...
     17    Enterprise Pricing Articles Sign in Free 30-Da...
     18    Homepage Follow Sign in / Sign up * Home\r\n *...
     19    METRICS MAVEN: MODE D'EMPLOI - FINDING THE MOD...
     20    Homepage Follow Sign in / Sign up Homepage * H...
     21    Raj Singh Blocked Unblock Follow Following Dev...
     22    IMPORTING JSON DOCUMENTS WITH NOSQLIMPORT\r\nG...
     23    This video shows you how to build and query a ...
     24    THE CONVERSATIONAL INTERFACE IS THE NEW PARADI...
     25    Skip navigation Upload Sign in SearchLoading...
     26    GOOGLE RESEARCH BLOG The latest news from Rese...
     27    Skip navigation Upload Sign in SearchLoading...
```

```
28    ACCESS DENIED\r\nSadly, your client does not s...
29    Homepage Follow Sign in / Sign up Homepage * H...
...                                                ...
1026  Enterprise Pricing Articles Sign in Free 30-Da...
1027  Skip navigation Sign in SearchLoading...\r\n\r...
1028  Compose The Compose logo Articles Sign in Free...
1029  Follow Sign in / Sign up * Home\r\n * About In...
1030  Homepage Follow Sign in / Sign up Homepage * H...
1031  Develop in the cloud at the click of a button!...
1032  BLAZINGLY FAST GEOSPATIAL QUERIES WITH REDIS\r...
1033  Blog Home Dataquest.io Learn Data Science in Y...
1034  DATALAYER: MANAGING (OR NOT) THE DATA IN IMMUT...
1035  Skip to contentWin-Vector Blog\r\n\r\nThe Win-...
1036  This work is licensed under a Creative Commons...
1037                                               NaN
1038  The relational database has been the dominant ...
1039  Skip to main content IBM developerWorks / Deve...
1040  Skip to contentDinesh Nirmal's Blog\r\n\r\nA b...
1041  Compose The Compose logo Articles Sign in Free...
1042  Glynn Bird Blocked Unblock Follow Following De...
1043  MENU\r\nClose\r\nSubscribe SubscribeREDUCING O...
1044  Homepage IBM Watson Data Lab Follow Sign in / ...
1045  Although it is built around a JavaScript engin...
1046  Margriet Groenendijk Blocked Unblock Follow Fo...
1047  Homepage Follow Sign in / Sign up Homepage * H...
1048  Homepage Follow Sign in Get started * Home\r\n...
1049  * \r\n * \r\n * \r\n * \r\n * \r\n * \r\n * \r...
1050  lA SPEED GUIDE TO REDIS LUA SCRIPTING\r\nShare...
1051  PouchDB-find is a new API and syntax that allo...
1052  We compare discriminative and generative learn...
1053  Essays about data, building products and boots...
1054                                               NaN
1055  Homepage Follow Sign in / Sign up Homepage * H...

                                       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...
5              Using Compose's PostgreSQL data browser.
6     Upgrading your PostgreSQL deployment to versio...
7     For a company like Slack that strives to be as...
8     Kaggle is your home for data science. Learn ne...
9     [A version of this post appears on the OReill...
10    Apple's sample app, Food Tracker, taught you i...
11    Replicating data to a relational dashDB databa...
12    This recipe showcases how one can analyze the ...
```

```
13    Theres a reason youve been hearing a lot abo...
14    Who are those people lurking behind the statis...
15    Early methods to integrate machine learning us...
16    The performance of supervised predictive model...
17    We've always considered MySQL as a potential C...
18    It has never been easier to build AI or machin...
19    In our Metrics Maven series, Compose's data sc...
20    It is often useful to use RStudio for one piec...
21    Youre doing your data a disservice if you don...
22    Introducing nosqlimport, an npm module to help...
23    This video shows you how to build and query a ...
24    Botkit provides a simple framework to handle t...
25    Want to learn more about how we created the Da...
26    Much of driving is spent either stuck in traff...
27    This talk assumes you have a basic understandi...
28    In this paper, we propose gcForest, a decision...
29    Im very happy and proud to announce that IBM ...
...                                                ...
1026  Varun Singh, a software engineer at IBM's Wats...
1027  This video shows you how to create and adminis...
1028  With the latest 0.2.1 version of Transporter, ...
1029  Audio super-resolution aims to reconstruct a h...
1030  Since then, this metric has been ubiquitously ...
1031  Build a word game app and see how to manage an...
1032  Use Redis and and Python scripts to speed your...
1033  In this post, youll learn to query, update, a...
1034  Adron Hall of Thrashing Code and Home Depot, t...
1035  Describes the use of Laplace noise in machine ...
1036  A full guide to Elasticsearch, the real-time d...
1037  See how quick and easy it is to set up a dashD...
1038  The relational database has been the dominant ...
1039  Building your first data warehouse doesnt hav...
1040  In my last blog Business differentiation thro...
1041  MongoDB's aggregation pipeline makes finding d...
1042  Which write API endpoint is the right write ca...
1043        Nothing spoils a plot like (too much) data.
1044  Getting started with custom visualizations, si...
1045  Although it is built around a JavaScript engin...
1046  Last week I attended the GeoPython conference ...
1047  In this post, we will go through how to read a...
1048  As more devices become internet enabled, harne...
1049  Continuing my previous work on exploring Arlin...
1050  Lua is a compact language which can be embedde...
1051  PouchDB uses MapReduce as its default search m...
1052  We compare discriminative and generative learn...
1053  In order to demystify some of the magic behind...
1054  Learn how to use IBM dashDB as data store for ...
1055  Once you get used to developing in a Notebook ...
```

```
                                        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
5                 Browsing PostgreSQL Data with Compose       Live           5
6                      Upgrading your PostgreSQL to 9.5       Live           6
7                               Data Wrangling at Slack       Live           7
8                                Data Science Bowl 2017       Live           8
9     Using Apache Spark to predict attack vectors a...       Live           9
10    Offline-First iOS Apps with Swift & Cloudant S...       Live          10
11                         Warehousing GeoJSON documents       Live          11
12    Timeseries Data Analysis of IoT events by usin...       Live          12
13    Bridging the Gap Between Python and Scala Jupy...       Live          13
14    Got zip code data? Prep it for analytics.  IB...      Live          14
15    Apache Spark 2.0: Extend Structured Streaming...      Live          15
16    Higher-order Logistic Regression for Large Dat...       Live          16
17                         Compose for MySQL now for you       Live          17
18    The Greatest Public Datasets for AI  Startup ...      Live          18
19                          Finding the Mode in PostgreSQL       Live          19
20    Working interactively with RStudio and noteboo...       Live          20
21    Mapping for Data Science with PixieDust and Ma...       Live          21
22              Move CSVs into different JSON doc stores       Live          22
23    Tutorial: How to build and query a Cloudant ge...       Live          23
24     The Conversational Interface is the New Paradigm       Live          24
25                  Creating the Data Science Experience       Live          25
26    Using Machine Learning to predict parking diff...       Live          26
27             Getting The Best Performance With PySpark       Live          27
28    Deep Forest: Towards An Alternative to Deep Ne...       Live          28
29                          Experience IoT with Coursera       Live          29
...                                                 ...        ...         ...
1026           Redis and MongoDB in the biomedical domain       Live        1021
1027  Create and administer a data catalog using IBM...       Live        1022
1028  How to move data with Compose Transporter - Fr...       Live        1023
1029  Using Deep Learning to Reconstruct High-Resolu...       Live        1024
1030            Data tidying in Data Science Experience       Live        1025
1031  Build a simple word game app using Cloudant on...       Live        1026
1032       Blazingly Fast Geospatial Queries with Redis       Live        1027
1033  Working with SQLite Databases using Python and...       Live        1028
1034  DataLayer Conference: Managing (or not) the Da...       Live        1029
1035  Laplace noising versus simulated out of sample...       Live        1030
1036                                The Definitive Guide       Live        1031
1037                     Get started with dashDB on Bluemix       Live        1032
1038       The Many Flavors of NoSQL at That Conference       Live        1033
1039   Your First Data Warehouse Is Easy. Meet the ODS.       Live        1034
1040              Machine Learning for the Enterprise.       Live        1035
```

```
1041             Finding Duplicate Documents in MongoDB       Live        1036
1042  Piecemeal, Bulk, or Batch?  IBM Watson Data L...      Live        1037
1043              Reducing overplotting in scatterplots       Live        1038
1044  You Too Can Make Magic (in Jupyter Notebooks w...       Live        1039
1045  How I Stopped Worrying & Learned to Love the M...       Live        1040
1046  Mapping All the Things with Python  IBM Watso...      Live        1041
1047  Use IBM Data Science Experience to Read and Wr...       Live        1042
1048  Use IoT data in Streams Designer for billing a...       Live        1043
1049                        Mapping Points with Folium       Live        1044
1050               A Speed Guide To Redis Lua Scripting       Live        1045
1051            A look under the covers of PouchDB-find       Live        1046
1052  A comparison of logistic regression and naive ...       Live        1047
1053  What I Learned Implementing a Classifier from ...       Live        1048
1054                             Use dashDB with Spark       Live        1049
1055  Jupyter Notebooks with Scala, Python, or R Ker...       Live        1050

[1051 rows x 5 columns]
```

```
In [418]: # checking if this works for example user
          df_content.iloc[761]
```

```
Out[418]: doc_body           Elvis Dohmatob Home Blog Publications Photos C...
          doc_description    In this post, Ill demo variational auto-encod...
          doc_full_name      Variational auto-encoder for "Frey faces" usin...
          doc_status                                                      Live
          article_id                                                       761
          Name: 764, dtype: object
```

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 [419]: print(df.shape)
          print(df_content.shape)
```

```
(45993, 3)
(1051, 5)
```

```
In [420]: # Counting unique articles that have at least one interaction
          df.article_id.nunique()
```

```
Out[420]: 714
```

```
In [421]: # Total articles
          df_content.article_id.nunique()
```

```
Out[421]: 1051
```

```
In [422]: # The number of unique users
          df.email.nunique()

Out[422]: 5148

In [423]: # The number of user-article interactions
          df.shape[0]

Out[423]: 45993

In [424]: unique_articles = df.article_id.nunique() # The number of unique articles that have at
          total_articles =  df_content.article_id.nunique() # The number of unique articles on t
          unique_users = df.email.nunique() # 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 [425]: # most viewed article_id
          df.article_id.value_counts().head(2)

Out[425]: 1429.0    937
          1330.0    927
          Name: article_id, dtype: int64

In [426]: most_viewed_article_id = str(df.article_id.value_counts().index[0]) # The most viewed
          max_views = df.article_id.value_counts().iloc[0] # The most viewed article in the data

In [428]: ## No need to change the code here - this will be helpful for later parts of the noteb
          # 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[428]:    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

             interactions  user_id
          0           1.0        1
          1           1.0        2
          2           1.0        3
          3           1.0        4
          4           1.0        5
```

```
In [429]: ## 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_
              '`The maximum number of user-article interactions by any 1 user is ______.`': max_
              '`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_arti
              '`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.

```
In [430]: df['title'].value_counts().index.tolist()
```

```
Out[430]: ['use deep learning for image classification',
           'insights from new york car accident reports',
           'visualize car data with brunel',
           'use xgboost, scikit-learn & ibm watson machine learning apis',
           'predicting churn with the spss random tree algorithm',
'healthcare python streaming application demo',
'finding optimal locations of new store using decision optimization',
'apache spark lab, part 1: basic concepts',
'analyze energy consumption in buildings',
'gosales transactions for logistic regression model',
'welcome to pixiedust',
'customer demographics and sales',
'total population by country',
'deep learning with tensorflow course by big data university',
'model bike sharing data with spss',
'the nurse assignment problem',
'classify tumors with machine learning',
'analyze accident reports on amazon emr spark',
'movie recommender system with spark machine learning',
'putting a human face on machine learning',
'gosales transactions for naive bayes model',
'ml optimization using cognitive assistant',
'learn basics about notebooks and apache spark',
'analyze precipitation data',
'apache spark lab, part 3: machine learning',
'jupyter notebook tutorial',
'deploy your python model as a restful api',
'visualize data with the matplotlib library',
'using pixiedust for fast, flexible, and easier data analysis and experimentation',
'access db2 warehouse on cloud and db2 with python',
'python machine learning: scikit-learn tutorial',
'maximize oil company profits',
'analyze open data sets with spark & pixiedust',
'uci ml repository: chronic kidney disease data set',
'introducing ibm watson studio ',
'analyze open data sets with pandas dataframes',
'working interactively with rstudio and notebooks in dsx',
'rapidly build machine learning flows with dsx',
'the pandas data analysis library',
'the machine learning database',
'learn tensorflow and deep learning together and now!',
'real-time sentiment analysis of twitter hashtags with spark (+ pixiedust)',
'access mysql with r',
'what caused the challenger disaster?',
'access mysql with python',
'pixieapp for outlier detection',
'apache spark lab, part 2: querying data',
'breast cancer wisconsin (diagnostic) data set',
'access ibm analytics for apache spark from rstudio',
'intents & examples for ibm watson conversation',
'times world university ranking analysis',
'data model with streaming analytics and python',
'tensorflow quick tips',
'deep learning with data science experience',
'fortune 100 companies',
'analyzing data by using the sparkling.data library features',
'employed population by occupation and age',
'sudoku',
'520    using notebooks with pixiedust for fast, flexi...\nName: title, dtype: object
'small steps to tensorflow',
'programmatic evaluation using watson conversation',
'ibm watson facebook posts for 2015',
'the nurse assignment problem data',
'data tidying in data science experience',
'getting started with python',
'build a python app on the streaming analytics service',
'i am not a data scientist  ibm watson data lab',
'categorize urban density',
'use r dataframes & ibm watson natural language understanding',
'using deep learning with keras to predict customer churn',
'practical tutorial on random forest and parameter tuning in r',
'housing (2015): united states demographic measures',
'timeseries data analysis of iot events by using jupyter notebook',
'use sql with data in hadoop python',
'building your first machine learning system ',
'how to map usa rivers using ggplot2',
'using machine learning to predict value of homes on airbnb',
'machine learning exercises in python, part 1',
'sector correlations shiny app',
'an introduction to stock market data analysis with r (part 1)',
'income (2015): united states demographic measures',
'connect to db2 warehouse on cloud and db2 using scala',
'developing for the ibm streaming analytics service',
'using brunel in ipython/jupyter notebooks',
'transfer learning for flight delay prediction via variational autoencoders',
'flightpredict ii: the sequel   ibm watson data lab',
'use spark for scala to load data and run sql queries',
'a comparison of logistic regression and naive bayes ',
'uci: heart disease - cleveland',
'deep learning from scratch i: computational graphs',
'using github for project control in dsx',
'use spark for python to load data and run sql queries',
'super fast string matching in python',
'a dynamic duo  inside machine learning  medium',
'upload files to ibm data science experience using the command line',
'statistics for hackers',
'discover hidden facebook usage insights',
'working with ibm cloud object storage in python',
'modern machine learning algorithms',
'perform sentiment analysis with lstms, using tensorflow',
'spark 2.1 and job monitoring available in dsx',
'analyze traffic data from the city of san francisco',
'overlapping co-cluster recommendation algorithm (ocular)',
'the unit commitment problem',
'pixiedust 1.0 is here!  ibm watson data lab',
'use decision optimization to schedule league games',
'watson assistant workspace analysis with user logs',
'1448    i ranked every intro to data science course on...\nName: title, dtype: objec
'car performance data',
'introducing streams designer',
'data science for real-time streaming analytics',
'use apache systemml and spark for machine learning',
'how smart catalogs can turn the big data flood into an ocean of opportunity',
'this week in data science (may 30, 2017)',
'modeling energy usage in new york city',
'7292    a dramatic tour through pythons data visualiz...\nName: title, dtype: object
'watson machine learning for developers',
'uci: sms spam collection',
'analyzing streaming data from kafka topics',
'visualize the 1854 london cholera outbreak',
'use the machine learning library',
'1357    what i learned implementing a classifier from ...\nName: title, dtype: objec
'flexdashboard: interactive dashboards for r',
'predict chronic kidney disease using spss modeler flows',
'workflow in r',
'probabilistic graphical models tutorial\u200a\u200apart 1  stats and bots',
'develop a scala spark model on chicago building violations',
'access postgresql with python',
'united states demographic measures: population and age',
'how to perform a logistic regression in r',
'machine learning and the science of choosing',
'brunel interactive visualizations in jupyter notebooks',
'use the cloudant-spark connector in python notebook',
"i'd rather predict basketball games than elections: elastic nba rankings",
'10 essential algorithms for machine learning engineers',
'house building with worker skills',
'top 10 machine learning use cases: part 1',
'upload data and create data frames in jupyter notebooks',
'10 must attend data science, ml and ai conferences in 2018',
'markdown for jupyter notebooks cheatsheet',
'leverage python, scikit, and text classification for behavioral profiling',
'analyze open data sets using pandas in a python notebook',
'graph-based machine learning',
'got zip code data? prep it for analytics.  ibm watson data lab  medium',
'how to choose a project to practice data science',
'using machine learning to predict parking difficulty',
'jupyter notebooks with scala, python, or r kernels',
'ibm data science experience white paper - sparkr transforming r into a tool for big
'new shiny cheat sheet and video tutorial',
'predicting flight cancellations using weather data, part 3',
'data science bowl 2017',
'using bigdl in dsx for deep learning on spark',
'challenges in deep learning',
'time series prediction using recurrent neural networks (lstms)',
'70 amazing free data sources you should know',
'using rstudio in ibm data science experience',
'automating web analytics through python',
'using dsx notebooks to analyze github data',
'why you should master r (even if it might eventually become obsolete)',
'dsx: hybrid mode',
'adolescent fertility rate (births per 1,000 women ages 15-19), worldwide',
'the greatest public datasets for ai  startup grind',
'common excel tasks demonstrated in\xa0pandas',
'how to scale your analytics using r',
'calls by customers of a telco company',
'use spark r to load and analyze data',
'simple graphing with ipython and\xa0pandas',
'best packages for data manipulation in r',
'experience iot with coursera',
'the data science process',
'54174    detect potentially malfunctioning sensors in r...\nName: title, dtype: obje
'using deep learning to reconstruct high-resolution audio',
'data structures related to machine learning algorithms',
'easy json loading and social sharing in dsx notebooks',
'city population by sex, city and city type',
'use spark for r to load data and run sql queries',
'ensemble learning to improve machine learning results',
'process events from the watson iot platform in a streams python application',
'the 3 kinds of context: machine learning and the art of the frame',
'uci: car evaluation',
'self-service data preparation with ibm data refinery',
'tidy up your jupyter notebooks with scripts',
'top 10 machine learning algorithms for beginners',
'pulling and displaying etf data',
'quick guide to build a recommendation engine in python',
'sparklyr  r interface for apache spark',
'15 page tutorial for r',
'airbnb data for analytics: amsterdam calendar',
'7 types of job profiles that makes you a data scientist',
'introduction to market basket analysis in\xa0python',
'understanding empirical bayes estimation (using baseball statistics)',
'making data science a team sport',
'pixiedust gets its first community-driven feature in 1.0.4',
"a kaggler's guide to model stacking in practice",
'pixiedust: magic for your python notebook',
'neural language modeling from scratch (part 1)',
"a beginner's guide to variational methods",
'5 practical use cases of social network analytics: going beyond facebook and twitter
'from scikit-learn model to cloud with wml client',
'higher-order logistic regression for large datasets',
'neurally embedded emojis',
'whats new in the streaming analytics service on bluemix',
'data science in the cloud',
'deep learning trends and an example',
'how to solve 90% of nlp problems',
'working with ibm cloud object storage in r',
'apple, ibm add machine learning to partnership with watson-core ml coupling',
'excel files: loading from object storage  python',
'this week in data science (april 18, 2017)',
'ibm cloud sql query',
'access postgresql with r',
'uci: iris',
'an introduction to scientific python (and a bit of the maths behind it)  numpy',
'this week in data science (may 2, 2017)',
'airbnb data for analytics: amsterdam listings',
'this week in data science (may 16, 2017)',
'annual precipitation by country 1990-2009',
'this week in data science (april 4, 2017)',
'use ibm data science experience to detect time series anomalies',
'variational auto-encoder for "frey faces" using keras',
'getting started with apache mahout',
'predicting gentrification using longitudinal census data',
'airbnb data for analytics: washington d.c. reviews',
'breast cancer detection with xgboost, wml and scikit',
'trust in data science',
'an attempt to understand boosting algorithm(s)',
'uci: red wine quality',
"feature importance and why it's important",
'declarative machine learning',
'this week in data science (february 14, 2017)',
'mapping points with folium',
'using machine learning to predict baseball injuries',
'uci: adult - predict income',
'hurricane how-to',
'working with sqlite databases using python and pandas',
'some random weekend reading',
'data visualization with r: scrum metrics',
'uci: forest fires',
'a moving average trading strategy',
'this week in data science (april 11, 2017)',
'this week in data science (may 23, 2017)',
'python for loops explained (python for data science basics #5)',
'introduction to neural networks, advantages and applications',
'spark 1.4 for rstudio',
'health insurance (2015): united states demographic measures',
'brunel 2.0 preview',
'tidy data in python',
'recommendation system algorithms  stats and bots',
'56594    lifelong (machine) learning: how automation ca...\nName: title, dtype: obje
'spark-based machine learning tools for capturing word meanings',
'use ibm data science experience to read and write data stored on amazon s3',
'502    forgetting the past to learn the future: long ...\nName: title, dtype: object
'twelve\xa0ways to color a map of africa using brunel',
'ml algorithm != learning machine',
'how to use version control (github) in rstudio within dsx?',
"december '16 rstudio tips and tricks",
'aspiring data scientists! start to learn statistics with these 6 books!',
'what is text analytics?',
'collecting data science cheat sheets',
'model a golomb ruler',
'simple linear regression? do it the bayesian way',
'united states demographic measures: zip code tabulation areas (zctas)',
'collect your own fitbit data with python',
'airbnb data for analytics: washington d.c. listings',
'wages',
'uci: white wine quality',
'a tensorflow regression model to predict house values',
'deep forest: towards an alternative to deep neural networks',
'neural networks for beginners: popular types and applications',
'using apply, sapply, lapply in r',
'the t-distribution: a key statistical concept discovered by a beer brewery',
'8170    data science expert interview: dez blanchfield...\nName: title, dtype: objec
'improving real-time object detection with yolo',
'what is smote in an imbalanced class setting (e.g. fraud detection)?',
'this week in data science (march 7, 2017)',
'awesome deep learning papers',
'10 data science podcasts you need to be listening to right now',
'occupation (2015): united states demographic measures',
'improving the roi of big data and analytics through leveraging new sources of data',
'ingest data from message hub in a streams flow',
'this week in data science (january 24, 2017)',
'generative adversarial networks (gans)',
'from spark ml model to online scoring with scala',
'fashion-mnist',
'get social with your notebooks in dsx',
'cifar-100 - python version',
'working with db2 warehouse on cloud in data science experience',
'this week in data science (january 31, 2017)',
'9 mistakes to avoid when starting your career in data science',
'd3heatmap: interactive heat maps',
'overfitting in machine learning: what it is and how to prevent it',
'notebooks: a power tool for data scientists',
'analyze facebook data using ibm watson and watson studio',
'10 powerful features on watson data platform, no coding necessary',
'life expectancy at birth by country in total years',
'this week in data science (february 7, 2017)',
'this week in data science (february 21, 2017)',
'better together: spss and data science experience',
'apache spark as the new engine of genomics',
'0 to life-changing app: new apache systemml api on spark shell',
'leaflet: interactive web maps with r',
'enjoy python 3.5 in jupyter notebooks',
'pearson correlation aggregation on sparksql',
'accelerate your workflow with dsx',
'generalization in deep learning',
'probabilistic graphical models tutorial\u200a\u200apart 2  stats and bots',
'interactive web apps with shiny cheat sheet',
'web picks (week of 23 january 2017)',
'how to use db2 warehouse on cloud in data science experience notebooks',
'optimizing a marketing campaign: moving from predictions to actions',
'getting started with graphframes in apache spark',
'this week in data science (march 28, 2017)',
'contraceptive prevalence (% women 15-49) by country',
'web picks (december 2017)',
'from python nested lists to multidimensional numpy arrays',
'how to write the first for loop in r',
'word2vec in data products',
'country statistics: unemployment rate',
'ibm data catalog is now generally available',
'births attended by skilled health staff (% of total) by country',
'10 tips on using jupyter notebook',
'the power of machine learning in spark',
'a visual explanation of the back propagation algorithm for neural networks',
'dry bulb temperature, by country, station and year',
'environment statistics database - water',
'country statistics: health expenditures',
'unstructured and structured data versus repetitive and non-repetitive',
'rstudio ide  cheat sheet',
'interconnect with us',
'a classification problem',
'analyze starcraft ii replays with jupyter notebooks',
'effectively using\xa0matplotlib',
'intentional homicide, number and rate per 100,000 population, by country',
'how to ease the strain as your data volumes rise',
'imitation learning in tensorflow (hopper from openai gym)',
'dimensionality reduction algorithms',
'data visualization playbook: revisiting the basics',
'shiny 0.12: interactive plots with ggplot2',
'annual % population growth by country',
'this week in data science (february 28, 2017)',
'when machine learning matters û erik bernhardsson',
'python if statements explained (python for data science basics #4)',
'cleaning the swamp: turn your data lake into a source of crystal-clear insight',
'time series anomaly detection algorithms  stats and bots',
'shiny 0.13.0',
'this week in data science (january 10, 2017)',
'airbnb data for analytics: vienna reviews',
'airbnb data for analytics: sydney calendar',
'tidyverse practice: mapping large european cities',
'social media insights with watson developer cloud & watson studio',
'apache spark 2.0: extend structured streaming for spark ml',
'0 to life-changing app: scala first steps and an interview with jakob odersky',
'making sense of the bias / variance trade-off in (deep) reinforcement learning',
'education (2015): united states demographic measures',
'10 data science, machine learning and ai podcasts you must listen to',
'fighting gerrymandering: using data science to draw fairer congressional districts',
'uci: wine recognition',
'why even a moths brain is smarter than an ai',
'visualising data the node.js way',
'apache spark 2.0: machine learning. under the hood and over the rainbow.',
'top 20 r machine learning and data science packages',
'mobile-cellular telephone subscriptions per 100 inhabitants, worldwide',
'total employment, by economic activity (thousands)',
'statistical bias types explained (with examples)',
'the random forest algorithm ',
'transform anything into a vector',
'customers of a telco including services used',
'analyze data, build a dashboard with spark and pixiedust',
'make machine learning a reality for your enterprise',
'this week in data science (april 25, 2017)',
'airbnb data for analytics: venice calendar',
'brunel in jupyter',
'breaking the 80/20 rule: how data catalogs transform data scientists productivity',
'country statistics: commercial bank prime lending rate',
'the million dollar question: where is my data?',
'web picks (week of 4 september 2017)',
'country statistics: gdp - per capita (ppp)',
'data science platforms are on the rise and ibm is leading the way',
'predict loan applicant behavior with tensorflow neural networking',
'agriculture, value added (% of gdp) by country',
'68879    dont throw more data at the problem! heres h...\nName: title, dtype: object
'worldwide electricity demand and production 1990-2012',
'announcing dsx environments in beta!',
'discover, catalog and govern data with ibm data catalog',
'uci: poker hand - testing data set',
'airbnb data for analytics: amsterdam reviews',
'what is hadoop?',
'random forest interpretation  conditional feature contributions',
'data visualization: the importance of excluding unnecessary details',
'51822    using apache spark as a parallel processing fr...\nName: title, dtype: obje
'three reasons machine learning models go out of sync',
'visual information theory ',
'data wrangling with dplyr and tidyr cheat sheet',
'artificial intelligence, ethically speaking  inside machine learning  medium',
'cifar-10 - python version',
'htmlwidgets: javascript data visualization for r',
'8 ways to turn data into value with apache spark machine learning',
'developing ibm streams applications with the python api (version 1.6)',
'web picks - dataminingapps',
'building custom machine learning algorithms with apache systemml',
'can a.i. be taught to explain itself?',
'3 scenarios for machine learning on multicloud',
'configuring the apache spark sql context',
'deep learning achievements over the past year ',
'get started with streams designer by following this roadmap',
'gradient boosting explained',
'statistical bias types explained',
'how ibm builds an effective data science team',
'leverage scikit-learn models with core ml',
'part-time employment rate, worldwide, by country and year',
'manage object storage in dsx',
'data visualization playbook: the right level of detail',
'mycheatsheets.com',
'hyperparameter optimization: sven hafeneger',
'machine learning for the enterprise.',
'interactive time series with dygraphs',
'how can data scientists collaborate to build better business',
'data visualization playbook: telling the data story',
'how the circle line rogue train was caught with data',
'score a predictive model built with ibm spss modeler, wml & dsx',
'this week in data science (july 26, 2016)',
'a guide to receptive field arithmetic for convolutional neural networks',
'blogging with brunel',
'join and enrich data from multiple sources',
'electric power consumption (kwh per capita) by country',
'share the (pixiedust) magic  ibm watson data lab  medium',
'how open api economy accelerates the growth of big data and analytics',
'drowning in data sources: how data cataloging could fix your findability problems',
"2875    hugo larochelle's neural network & deep learni...\nName: title, dtype: objec
'finding the user in data science',
'calculate moving averages on real time data with streams designer',
'poverty (2015): united states demographic measures',
'readr 1.0.0',
'learning statistics on youtube',
'country population by gender 1985-2005',
'bayesian nonparametric models  stats and bots',
'airbnb data for analytics: venice reviews',
'recommender systems: approaches & algorithms',
'bayesian regularization for #neuralnetworks  autonomous agents\u200a\u200a#ai',
'optimization for deep learning highlights in 2017',
'uci: poker hand - training data set',
'best practices for custom models in watson visual recognition',
'what is machine learning?',
'find airbnb deals in portland with machine learning using r',
'creating the data science experience',
'open sourcing 223gb of driving data  udacity inc',
'migrating to python 3 with pleasure',
'using shell scripts to control data flows created in watson applications',
'making data cleaning simple with the sparkling.data library',
'work with data connections in dsx',
'airbnb data for analytics: vienna listings',
'airbnb data for analytics: vancouver reviews',
'data visualization with ggplot2 cheat sheet',
'detect malfunctioning iot sensors with streaming analytics',
'essentials of machine learning algorithms (with python and r codes)',
"let's have some fun with nfl data",
'r for data science',
'deep learning, structure and innate priors',
'20405    how to tame the valley  hessian-free hacks fo...\nName: title, dtype: objec
'percentage of internet users by country',
'spark sql - rapid performance evolution',
'analyze ny restaurant data using spark in dsx',
'external debt stocks, total (dod, current us$) by country',
'adoption of machine learning to software failure prediction',
'data science of variable selection',
'build a logistic regression model with wml & dsx',
'predicting the 2016 us presidential election',
'country statistics: life expectancy at birth',
'are your predictive models like broken clocks?',
'top analytics tools in 2016',
'airbnb data for analytics: toronto reviews',
'shiny: a data scientists best friend',
'the difference between ai, machine learning, and deep learning?',
'population below national poverty line, total, percentage',
'airbnb data for analytics: trentino listings',
"for ai to get creative, it must learn the rules--then how to break 'em",
'web picks by dataminingapps',
'time series analysis using max/min and neuroscience',
'country statistics: telephones - mobile cellular',
'roads paved as % of total roads by country',
'what is spark?',
'apache spark sql analyzer resolves order-by column',
'clustering: a guide for the perplexed',
'working with notebooks in dsx',
'airbnb data for analytics: toronto calendar',
'united states demographic measures: income',
'web picks (week of 2 october 2017)',
'data science experience documentation',
'apache spark @scale: a 60 tb+ production use case',
'pseudo-labeling a simple semi-supervised learning method',
'advancements in the spark community',
'airbnb data for analytics: vienna calendar',
'the two phases of gradient descent in deep learning',
'3992    using apache spark to predict attack vectors a...\nName: title, dtype: objec
'this week in data science (november 01, 2016)',
'analyze db2 warehouse on cloud data in rstudio in dsx',
'empirical bayes for multiple sample sizes',
'build a predictive analytic model',
'talent vs luck: the role of randomness in success and failure',
'read and write data to and from amazon s3 buckets in rstudio',
'forest area by country in sq km',
'backpropagation  how neural networks learn complex behaviors',
'xml2 1.0.0',
'country statistics - europe - population and society',
'a guide to convolution arithmetic for deep learning',
'an interview with pythonista katharine jarmul',
'watson speech-to-text services  tl;dr need not apply',
'apache spark 2.0: migrating applications',
'what is systemml? why is it relevant to you?',
'ibm watson machine learning: get started',
'ratio (% of population) at national poverty line by country',
'data science experience demo: modeling energy usage in nyc',
'world marriage data',
'brunel: imitation is a sincere form of flattery',
'create a project for watson machine learning in dsx',
'airbnb data for analytics: portland listings',
'data science expert interview: holden karau',
'apache systemml',
'airbnb data for analytics: barcelona reviews',
'recent trends in recommender systems',
'intelligent applications - apache spark',
'airbnb data for analytics: berlin reviews',
'use data assets in a project using ibm data catalog',
'airbnb data for analytics: antwerp reviews',
'airbnb data for analytics: toronto listings',
'reducing overplotting in scatterplots',
'airbnb data for analytics: trentino reviews',
'government consumption expenditure',
'country statistics: airports',
'world tourism data by the world tourism organization',
'airbnb data for analytics: paris calendar',
'r markdown reference guide',
'airbnb data for analytics: new york city calendar',
'country statistics: telephones - fixed lines',
'country statistics: gross national saving',
'shaping data with ibm data refinery',
'airbnb data for analytics: vancouver listings',
'laplace noising versus simulated out of sample methods (cross frames)',
'airbnb data for analytics: antwerp calendar',
'airbnb data for analytics: madrid listings',
'interest rates',
'dplyr 0.5.0',
'you could be looking at it all wrong',
'whats new in data refinery?',
'airbnb data for analytics: boston listings',
'the art of side effects: curing apache spark streamings amnesia (part 1/2)',
'airbnb data for analytics: portland reviews',
'airbnb data for analytics: venice listings',
'fertility rate by country in total births per woman',
'country statistics: roadways',
'persistent changes to spark config in dsx',
'improving quality of life with spark-empowered machine learning',
'jupyter (ipython) notebooks features',
'seti data, publicly available, from ibm',
'mobile cellular subscriptions per 100 people by country',
'how to get a job in deep learning',
'uci: abalone',
'foundational methodology for data science',
'a fast on-disk format for data frames for r and python, powered by apache arrow',
'do i need to learn r?',
'country statistics: stock of domestic credit',
'a survey of books about apache spark',
'co2 emissions (metric tons per capita) by country',
'consumer prices',
'household consumption expenditure',
'country statistics: stock of broad money',
'airbnb data for analytics: austin listings',
'airbnb data for analytics: antwerp listings',
'airbnb data for analytics: portland calendar',
'airbnb data for analytics: sydney listings',
'airbnb data for analytics: sydney reviews',
'households by type of household, age and sex of head of household',
'this week in data science (august 02, 2016)',
'this week in data science',
'tidyr 0.6.0',
'load data into rstudio for analysis in dsx',
'unmet need for family planning, spacing, percentage, worldwide, by country',
'machine learning for everyone',
'worldwide fuel oil consumption by household (in 1000 metric tons)',
'airbnb data for analytics: chicago listings',
'country statistics: reserves of foreign exchange and gold',
'airbnb data for analytics: barcelona listings',
'airbnb data for analytics: athens calendar',
'country statistics: crude oil - exports',
'country statistics: infant mortality rate',
'dt: an r interface to the datatables library',
'enhanced color mapping',
'publish notebooks to github in dsx',
'run shiny applications in rstudio in dsx',
'airbnb data for analytics: boston reviews',
'10 pieces of advice to beginner data scientists',
'why relational databases and r?',
'learn about data science in world of watson',
'greenhouse gas emissions worldwide',
'dont overlook simpler techniques and algorithms',
'airbnb data for analytics: vancouver calendar',
'airbnb data for analytics: chicago calendar',
'apache spark: upgrade and speed-up your analytics',
'country populations 15 years of age and over, by educational attainment, age and sex
'machine learning for the enterprise',
'country statistics: railways',
'which one to choose for your problem',
'a day in the life of a data engineer',
'gross national income per capita, atlas method (current us$) by country',
'refugees',
'cache table in apache spark sql',
'build a naive-bayes model with wml & dsx',
'use iot data in streams designer for billing and alerts',
'united states demographic measures: education',
'from local spark mllib model to cloud with watson machine learning',
'country statistics: maternal mortality rate',
'apache spark 2.0: impressive improvements to spark sql',
'airbnb data for analytics: new york city reviews',
'high-tech exports as % of manufactured exports by country',
'missing data conundrum: exploration and imputation techniques',
'worldwide county and region - national accounts - gross national income 1948-2010',
'airbnb data for analytics: athens reviews',
'airbnb data for analytics: austin reviews',
'back to basics  jupyter notebooks',
'foreign direct investment, net inflows (bop, current us$) by country',
'environment statistics database - waste',
'country statistics: distribution of family income - gini index',
'run dsx notebooks on amazon emr',
'airbnb data for analytics: nashville calendar',
'airbnb data for analytics: san diego reviews',
'airbnb data for analytics: montreal listings',
'66855    migration from ibm bluemix data connect api (a...\nName: title, dtype: obje
'beyond parallelize and collect',
'use pmml to predict iris species',
'natural gas production, 1995 - 2012, worldwide',
'airbnb data for analytics: antwerp listings test',
'airbnb data for analytics: boston calendar',
'this week in data science (january 17, 2017)',
'airbnb data for analytics: athens listings',
'one year as a data scientist at stack overflow',
'military expenditure as % of gdp by country',
'airbnb data for analytics: seattle reviews',
'country statistics: refined petroleum products - production',
'style transfer experiments with watson machine learning',
'airbnb data for analytics: austin calendar',
'country statistics: budget surplus or deficit',
'airbnb data for analytics: paris reviews',
'airbnb data for analytics: new orleans listings',
'airbnb data for analytics: trentino calendar',
'country statistics: current account balance',
'geographic coordinates of world locations',
'working with on-premises databases  step by step',
'labor',
'stacking multiple custom models in watson visual recognition',
'improved water source by country: % population with access',
'airbnb data for analytics: dublin reviews',
'airbnb data for analytics: chicago reviews',
'consumption of ozone-depleting cfcs in odp metric tons',
'marital status of men and women',
'country statistics: central bank discount rate',
'big data is better data',
'country statistics: population',
'annual % inflation by country',
'the new builders ep. 13: all the data thats fit to analyze',
'airbnb data for analytics: mallorca reviews',
'country statistics: electricity - consumption',
"h2o with ibm's data science experience (dsx)",
'country statistics: merchant marine',
'create a project in dsx',
'country statistics: internet users',
'airbnb data for analytics: paris listings',
'load db2 warehouse on cloud data with apache spark in dsx',
'continuous learning on watson',
'airbnb data for analytics: san francisco listings',
'energy use (kg of oil equivalent per capita) by country',
'using the maker palette in the ibm data science experience',
'international liquidity',
'building a business that combines human experts and data science',
'roads, paved (% of total roads), worldwide, 1990-2011',
'airbnb data for analytics: santa cruz county reviews',
'ggplot2 2.2.0 coming soon!',
'primary school completion rate % of relevant age group by country',
          'this week in data science (december 27, 2016)',
          'a new version of dt (0.2) on cran',
          'airbnb data for analytics: berlin calendar',
          'country statistics: imports',
          'a glimpse inside the mind of a data scientist',
          'load and analyze public data sets in dsx',
          'the new builders podcast ep 3: collaboration',
          'this week in data science (october 18, 2016)',
          'airbnb data for analytics: london listings',
          'country statistics: children under the age of 5 years underweight',
          'governance overview for ibm data catalog',
          'airbnb data for analytics: brussels reviews',
          'country statistics: industrial production growth rate',
          'let data dictate the visualization',
          'airbnb data for analytics: seattle calendar',
          'package development with devtools  cheat sheet',
          'refugees, worldwide, 2003 - 2013',
          'country statistics: electricity - from fossil fuels',
          'airbnb data for analytics: washington d.c. calendar',
          'country statistics: natural gas - consumption',
          'ibm data catalog overview',
          'measles immunization % children 12-23 months by country',
          'country surface area (sq. km)',
          'the data processing inequality',
          'country statistics: crude oil - imports',
          'airbnb data for analytics: new orleans reviews',
          'country statistics: crude oil - proved reserves',
          'webinar: april 11 - thinking inside the box: you can do that inside a data frame?!',
          'airbnb data for analytics: oakland reviews',
          'country statistics: market value of publicly traded shares',
          'airbnb data for analytics: barcelona calendar',
          'create a connection and add it to a project using ibm data refinery',
          'build deep learning architectures with neural network modeler',
          'airbnb data for analytics: san diego listings',
          'this week in data science (november 22, 2016)',
          'airbnb data for analytics: london reviews',
          'nips 2016  day 2 highlights']
```

```
In [431]: df['title'].value_counts().index.tolist()[:4] # First 4 articles
```

```
Out[431]: ['use deep learning for image classification',
           'insights from new york car accident reports',
           'visualize car data with brunel',
           'use xgboost, scikit-learn & ibm watson machine learning apis']
```

```
In [432]: 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

        top_articles = list(df.groupby('title')['user_id'].count().sort_values(ascending=F

        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 = list((df.groupby('article_id')['user_id'].count().sort_values(ascen


        return top_articles # Return the top article ids
```

```
In [433]: print(get_top_articles(10))
```

```
['use deep learning for image classification', 'insights from new york car accident reports', 'v
```

```
In [434]: print(get_top_article_ids(10))
```

```
['1429.0', '1330.0', '1431.0', '1427.0', '1364.0', '1314.0', '1293.0', '1170.0', '1162.0', '1304
```

```
In [435]: # 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.

```
In [436]: df.head()
```

```
Out[436]:    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

             interactions  user_id
          0           1.0        1
          1           1.0        2
          2           1.0        3
          3           1.0        4
          4           1.0        5
```

```
In [437]: # 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
              '''
              user_item = df.groupby(['user_id', 'article_id'])['article_id'].count().unstack()
```

```
            user_item = user_item.replace(np.nan,0).astype(int)

            user_item[user_item > 0] = 1

            return user_item # return the user_item matrix

        user_item = create_user_item_matrix(df)
```

In [438]:
```
user_item.head()
```

Out[438]:
```
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
2                0       0       0       0       0       0       0       0
3                0       0       0       0       0       1       0       0
4                0       0       0       0       0       0       0       0
5                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       1       0       1
2                0       0   ...         0       0       0       0       0
3                0       0   ...         0       0       1       0       0
4                0       0   ...         0       0       0       0       0
5                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
2                0       0       0       0       0
3                0       0       0       0       0
4                0       0       0       0       0
5                0       0       0       0       0

[5 rows x 714 columns]
```

In [439]:
```
user_item.shape
```

Out[439]:
```
(5149, 714)
```

In [440]:
```
## 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 mat
assert user_item.shape[1] == 714, "Oops!  The number of articles in the user-article m
assert user_item.sum(axis=1)[1] == 36, "Oops!  The number of articles seen by user 1 d
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.

```
In [441]: 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 produc
                              are listed first

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

              '''
              # computing similarity of each user to the provided user
              similarity = user_item.dot(np.transpose(user_item))

              # sorting by similarity
              similar_users = similarity[user_id].sort_values(ascending = False)

              # creating list of just the ids
              most_similar_users = similar_users.index.tolist()

              # removing the own user's id
              most_similar_users.remove(user_id)

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

```
In [442]: # 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, 131, 3870, 46, 4201, 5041]
The 5 most similar users to user 3933 are: [1, 23, 3782, 4459, 203]
The 3 most similar users to user 46 are: [4201, 23, 3782]
```

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.

```
In [443]: 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_names = list(set(df[df['article_id'].isin(article_ids)]['title']))

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


          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 use
              '''
              # Your code here
              article_ids = user_item.loc[user_id][user_item.loc[user_id] == 1].index.astype('st

              article_names = []

              for idx in article_ids:
                  article_names.append(df[df['article_id']==float(idx)].max()['title'])


              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 re
        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
        recs = np.array([])

        user_checked_articles = get_user_articles(user_id)[0]
        similar_users = find_similar_users(user_id)

        for usrs in similar_users:

            others_checked_articles = get_user_articles(usrs)[0]
            new_recs = np.setdiff1d(others_checked_articles, user_checked_articles, assume
            recs = np.unique(np.concatenate([new_recs, recs], axis = 0))

            if len(recs) > m-1:
                break

        recs = list(recs)[:m]

        return recs # return your recommendations for this user_id
```

```
In [444]: # Check Results
          get_article_names(user_user_recs(331, 10)) # Return 10 recommendations for user 331
```

```
Out[444]: ['categorize urban density',
           'access db2 warehouse on cloud and db2 with python',
           'tensorflow quick tips',
           'putting a human face on machine learning',
           'country statistics: life expectancy at birth',
           'apache spark lab, part 1: basic concepts',
           'finding optimal locations of new store using decision optimization',
 'predict loan applicant behavior with tensorflow neural networking',
 'gosales transactions for naive bayes model',
 'classify tumors with machine learning']
```

```
In [445]: # 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.
          assert set(get_article_names(['1320.0', '232.0', '844.0'])) == set(['housing (2015): u
          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 demographi
          assert set(get_user_articles(2)[0]) == set(['1024.0', '1176.0', '1305.0', '1314.0', '1
          assert set(get_user_articles(2)[1]) == set(['using deep learning to reconstruct high-r
          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.

```
In [446]: df.head()
```

```
Out[446]:    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

             interactions  user_id
          0           1.0        1
          1           1.0        2
          2           1.0        3
          3           1.0        4
          4           1.0        5
```

```
In [447]: 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
                    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_interactions = df.groupby('user_id')['article_id'].count()

    neighbors_data = []

    for uid in range(1, user_item.shape[0]):
        if uid == user_id:
            continue

        similarity = sum(user_item.loc[user_id] * user_item.loc[uid])

        neighbors_data.append({
            'neighbor_id': uid,
            'similarity': similarity,
            'num_interactions': user_interactions.loc[uid]
        })

    neighbors_df = pd.DataFrame(neighbors_data)
    neighbors_df.sort_values(['similarity', 'num_interactions'],
                             ascending=[False, False],
                             inplace=True)


    return neighbors_df # Return the dataframe specified in the doc_string

def user_user_recs_improved(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

neighbors_df = get_top_sorted_users(user_id)

user_articles_names, user_article_ids = get_user_articles(user_id)

recs = []
explored_articles = set(user_article_ids)

article_interactions = df.groupby('article_id')['user_id'].count().sort_values(asc

for _, neighbor in neighbors_df.iterrows():
    neighbor_id = neighbor['neighbor_id']

    neighbor_article_names, neighbor_article_ids = get_user_articles(neighbor_id)

    top_global_articles = list(article_interactions.index)

    for article_id in top_global_articles:
        if (article_id not in explored_articles and article_id in neighbor_article
            recs.append(article_id)
            explored_articles.add(article_id)

    if len(recs) >= m:
        break

if len(recs) < m:
    for article_id in top_global_articles:
        if article_id not in explored_articles and len(recs) < m:
            recs.append(article_id)

rec_names = get_article_names(recs)

return recs, rec_names
```

```
In [448]: # Quick spot check - don't change this code - just use it to test your functions
          rec_ids, rec_names = user_user_recs_improved(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:
[1429.0, 1330.0, 1431.0, 1427.0, 1364.0, 1314.0, 1293.0, 1170.0, 1162.0, 1304.0]

The top 10 recommendations for user 20 are the following article names:
['healthcare python streaming application demo', 'visualize car data with brunel', 'analyze ener
```

```
In [449]: rec_names
```

```
Out[449]: ['healthcare python streaming application demo',
           'visualize car data with brunel',
           'analyze energy consumption in buildings',
           'gosales transactions for logistic regression model',
           'use xgboost, scikit-learn & ibm watson machine learning apis',
           'predicting churn with the spss random tree algorithm',
           'insights from new york car accident reports',
           'use deep learning for image classification',
           'apache spark lab, part 1: basic concepts',
           'finding optimal locations of new store using decision optimization']
```

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 [450]: ### Tests with a dictionary of results
          user1_most_sim = get_top_sorted_users(1).iloc[0].neighbor_id # Find the user that is m
          user131_10th_sim = get_top_sorted_users(131).iloc[9].neighbor_id# Find the 10th most s
```

```
In [451]: ## 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.

It would make sense to use the get_top_articles function and Rank Based Recommendations when recommending content to a new user. Since we don't know anything about the user or their interactions, we can't determine which other users they are most like, thus we would just suggest the most popular articles. We could combine three different recommendation techniques—Rank, Content, and Collaborative—once we have more user data.

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 [452]: 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 = get_top_article_ids(10, df)

In [453]: 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 [454]: 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 [455]: # 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 [456]: # Load the matrix here
          user_item_matrix = pd.read_pickle('user_item_matrix.p')
```

```
In [457]: # quick look at the matrix
          user_item_matrix.head()
```

```
Out[457]: 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]
```

```
In [458]: user_item_matrix.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5149 entries, 1 to 5149
Columns: 714 entries, 0.0 to 997.0
dtypes: float64(714)
memory usage: 28.1 MB
```

```
In [459]: user_item_matrix.isna().sum().sum() #  no null vals

Out[459]: 0
```

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.

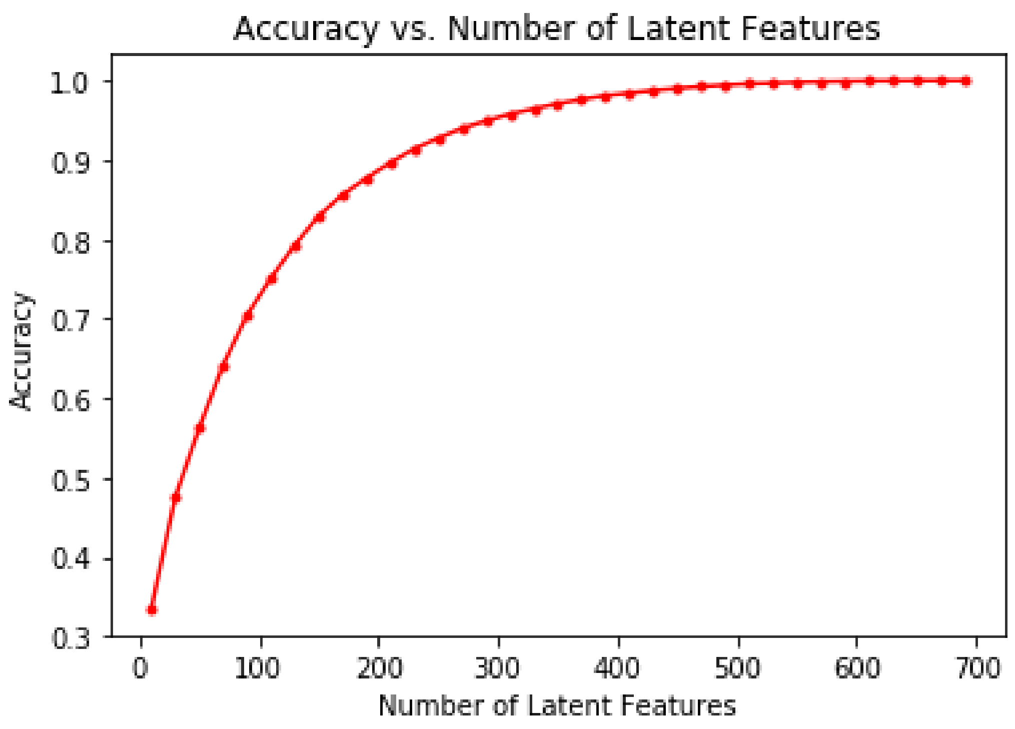
```
In [460]: # Perform SVD on the User-Item Matrix Here
          u, s, vt = np.linalg.svd(user_item_matrix)# use the built in to get the three matrices

In [461]: s.shape, u.shape, vt.shape

Out[461]: ((714,), (5149, 5149), (714, 714))
```

We have no missing values in this matrix therefore we can perform SVD. In the classroom, our matrix had missing values which meant that we had to use FunkSVD..

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.

```
In [462]: num_latent_feats = np.arange(10,700+10,20)
          sum_errs = []

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

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

              diffs = np.subtract(user_item_matrix, user_item_est)
```

```
        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], color='r', marker='.');
    plt.xlabel('Number of Latent Features');
    plt.ylabel('Accuracy');
    plt.title('Accuracy vs. Number of Latent Features');
```



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 [463]: 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

    '''
    user_item_train = create_user_item_matrix(df_train)
    user_item_test = create_user_item_matrix(df_test)

    test_idx = user_item_test.index
    test_arts = user_item_test.columns

    return user_item_train, user_item_test, test_idx, test_arts

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

```
In [464]: test_idx # 682 users in testset
```

```
Out[464]: 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 [465]: train_idx = user_item_train.index # 4487 users in trainingset
          train_idx
```

```
Out[465]: Int64Index([   1,    2,    3,    4,    5,    6,    7,    8,    9,   10,
             ...
             4478, 4479, 4480, 4481, 4482, 4483, 4484, 4485, 4486, 4487],
            dtype='int64', name='user_id', length=4487)
```

```
In [466]: test_idx.difference(train_idx) # Total 682 users in testset but 20 among them are in t
```

```
Out[466]: Int64Index([4488, 4489, 4490, 4491, 4492, 4493, 4494, 4495, 4496, 4497,
             ...
             5140, 5141, 5142, 5143, 5144, 5145, 5146, 5147, 5148, 5149],
            dtype='int64', name='user_id', length=662)
```

```
In [467]: test_arts #574 movies in testset
```

```
Out[467]: Float64Index([   0.0,    2.0,    4.0,    8.0,    9.0,   12.0,   14.0,   15.0,
                          16.0,   18.0,
                        ...
                        1432.0, 1433.0, 1434.0, 1435.0, 1436.0, 1437.0, 1439.0, 1440.0,
                        1441.0, 1443.0],
                       dtype='float64', name='article_id', length=574)
```

```
In [468]: train_arts = user_item_train.columns #714 movies in trainset
          train_arts
```

```
Out[468]: Float64Index([   0.0,    2.0,    4.0,    8.0,    9.0,   12.0,   14.0,   15.0,
                          16.0,   18.0,
                        ...
                        1434.0, 1435.0, 1436.0, 1437.0, 1439.0, 1440.0, 1441.0, 1442.0,
                        1443.0, 1444.0],
                       dtype='float64', name='article_id', length=714)
```

```
In [469]: # 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, # letter here,
              'How many users in the test set are we not able to make predictions for because of
              'How many articles can we make predictions for in the test set?': b,# letter here,
              'How many articles in the test set are we not able to make predictions for because
          }

          t.sol_4_test(sol_4_dict)
```

```
Awesome job!  That's right!  All of the test articles are in the training data, but there are on
```

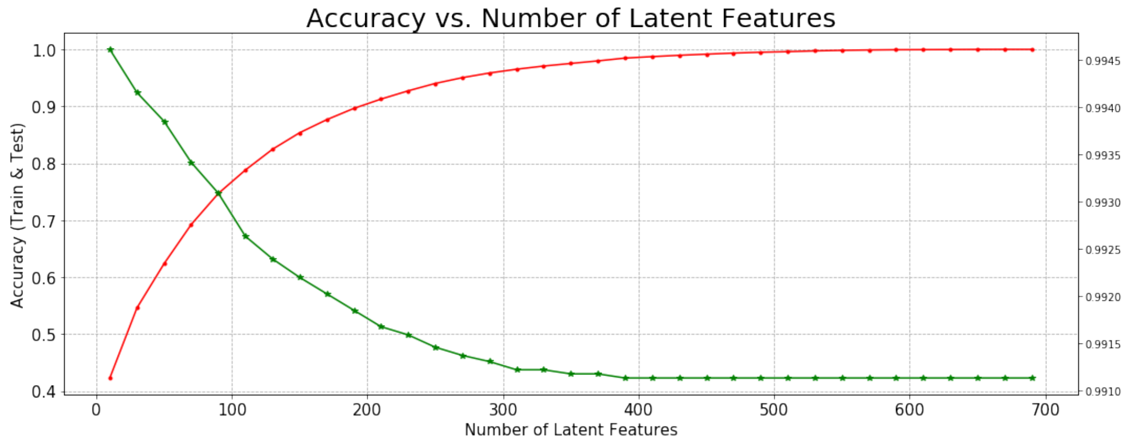
5. Now use the **user_item_train** dataset from above to find U, S, and V transpose using SVD. Then find the subset of rows in the **user_item_test** dataset that you can predict using this matrix decomposition with different numbers of latent features to see how many features makes sense to keep based 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.

```
In [470]: # fit SVD on the user_item_train matrix
          u_train, s_train, vt_train = np.linalg.svd(user_item_train) # fit svd
```

```
In [471]: # Use these cells to see how well you can use the training
          s_train.shape, u_train.shape, vt_train.shape
```

```
Out[471]: ((714,), (4487, 4487), (714, 714))

In [472]: num_latent_feats = np.arange(10,700+10,20)

          row_index = user_item_train.index.isin(test_idx)
          col_index = user_item_train.columns.isin(test_arts)

          u_test = u_train[row_index, :]
          vt_test = vt_train[:, col_index]

          users_prediction = np.intersect1d(list(user_item_train.index),list(user_item_test.inde

          sum_errs_train = []
          sum_errs_test = []

          f1_train = []
          f1_test = []

          for f in num_latent_feats:
              s_train_new, u_train_new, vt_train_new = np.diag(s_train[:f]), u_train[:, :f], vt_
              u_test_new, vt_test_new = u_test[:, :f], vt_test[:f, :]

              user_item_train_exp = np.around(np.dot(np.dot(u_train_new, s_train_new), vt_train_
              user_item_test_exp = np.around(np.dot(np.dot(u_test_new, s_train_new), vt_test_new

              train_diffs = np.subtract(user_item_train, user_item_train_exp)
              test_diffs = np.subtract(user_item_test.loc[users_prediction,:], user_item_test_ex

              err_train = np.sum(np.sum(np.abs(train_diffs)))
              err_test = np.sum(np.sum(np.abs(test_diffs)))

              sum_errs_train.append(err_train)
              sum_errs_test.append(err_test)

In [473]: # plotting the training and test accuracies
          fig, ax1 = plt.subplots(figsize=(15,6))

          ax1.set_xlabel('Number of Latent Features', fontsize=15)
          ax1.set_ylabel('Accuracy (Train & Test)', fontsize=15)
          ax1.plot(num_latent_feats, 1 - np.array(sum_errs_train)/df.shape[0], color='r',marker=
          ax1.tick_params(axis='y')
          ax1.set_title('Accuracy vs. Number of Latent Features', fontsize=25)
          plt.xticks(fontsize=15)
          plt.yticks(fontsize=15)
          ax1.grid(linestyle='--')

          ax2 = ax1.twinx()
          ax2.plot(num_latent_feats, 1 - np.array(sum_errs_test)/df.shape[0], color='g', marker=
fig.tight_layout()
plt.show()
```



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 of the test data diminishes with an increase in latent characteristics, whereas the accuracy of the training data improves. The number of latent features should be maintained relatively minimal because this is most likely the result of the data becoming overfitted as latent features increase. It's crucial to remember that when we apply SVD in this case, we can only really propose the 20 users in the training and test datasets. Additionally, our matrix is extremely sparse, which is probably why the test data accuracy is so high at >99%. If more people were included in both the test and training data, it would be intriguing to examine the outcomes. This would be a nice number of latent features to include since, as we can see, there is a cross over point at about 80 features where the accuracy for test data starts to decline. Beyond that, our accuracy for training grows but testing declines. To address the cold start issue, we may run an A/B test for new users to see how well our recommendation engine performs in real-world scenarios. For instance, we might utilise our recommendation engine to suggest articles to one set of users and then only suggest the most popular items to the other group. To determine whether our recommendation engine increases clicks, we would then compare the click-through rates. We may determine that this is effective and need to be implemented if we observed a notable increase in clicks when utilising our recommendation engine..

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

Out[475]: 0

In [ ]:
```