Recommendations_with_IBM

December 16, 2021

1 Recommendations with IBM

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

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

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

1.1 Table of Contents

I. Section **??** II. Section **??** III. Section **??** IV. Section **??** V. Section **??** VI. Section **??**

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

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

%matplotlib inline

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

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

Out[1]:

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

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

In [2]:
```
# Show df_content to get an idea of the data
df_content.head()
```

Out[2]:
```
                                            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
```

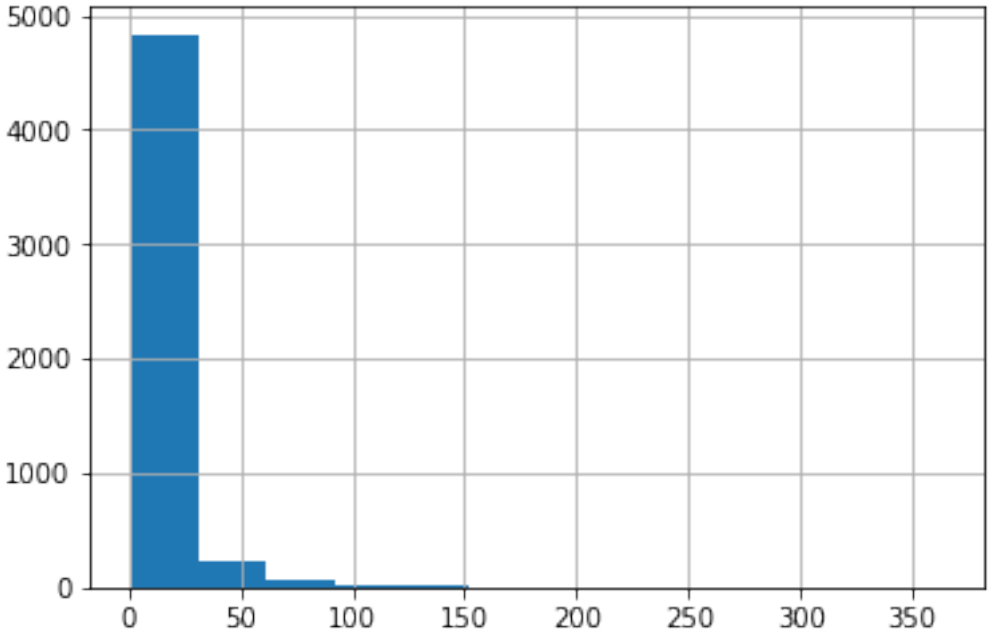
1.1.1 Part I : Exploratory Data Analysis

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 [3]:
```
interacts = df.groupby('email').count()['article_id']

interacts.hist(bins=12)
```

Out[3]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f307b167978>
```




```
In [4]: interacts.describe()
```

```
Out[4]: 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
```

```
In [5]: # Fill in the median and maximum number of user_article interactios below

        median_val = 3 # 50% of individuals interact with ____ number of articles or fewer.
        max_views_by_user = 364 # The maximum number of user-article interactions by any 1 user
```

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

```
In [6]: # Find and explore duplicate articles

        df_content.duplicated("article_id").sum()
```

```
Out[6]: 5
```

```
In [7]: # Remove any rows that have the same article_id - only keep the first

        df_content = df_content.drop_duplicates("article_id", keep="first")
```

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 [8]: df.article_id.nunique()

Out[8]: 714

In [9]: df_content.shape

Out[9]: (1051, 5)

In [10]: df.email.nunique()

Out[10]: 5148

In [11]: df.shape

Out[11]: (45993, 3)

In [12]: unique_articles = 714 # The number of unique articles that have at least one interactio
         total_articles = 1051 # The number of unique articles on the IBM platform
         unique_users = 5148 # The number of unique users
         user_article_interactions = 45993 # 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 [13]: df.groupby(["article_id"])["email"].count().sort_values(ascending=False).head()

Out[13]: article_id
         1429.0    937
         1330.0    927
         1431.0    671
         1427.0    643
         1364.0    627
         Name: email, dtype: int64

In [14]: most_viewed_article_id = '1429.0' # The most viewed article in the dataset as a string
         max_views = 937 # The most viewed article in the dataset was viewed how many times?

In [15]: ## 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[15]:

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

In [16]:
```
## If you stored all your results in the variable names above,
## you shouldn't need to change anything in this cell

sol_1_dict = {
    '`50% of individuals have _____ or fewer interactions.`': median_val,
    '`The total number of user-article interactions in the dataset is ______.`': user_a
    '`The maximum number of user-article interactions by any 1 user is ______.`': max_v
    '`The most viewed article in the dataset was viewed _____ times.`': max_views,
    '`The article_id of the most viewed article is ______.`': most_viewed_article_id,
    '`The number of unique articles that have at least 1 rating ______.`': unique_artic
    '`The number of unique users in the dataset is ______`': unique_users,
    '`The number of unique articles on the IBM platform`': total_articles
}

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

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

1.1.2 Part II: Rank-Based Recommendations

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

article can really only be based on how often an article was interacted with.

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

```
In [17]: 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

             title_count = df.groupby(by='title').count().sort_values(by='user_id', ascending=Fa

             top_articles = list(title_count.head(n).index)

             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

             id_count = df.groupby(by='article_id').count().sort_values(by='user_id', ascending=

             top_articles = list(id_count.head(n).index)

             return top_articles # Return the top article ids
```

```
In [18]: print(get_top_articles(10))
         print(get_top_article_ids(10))
```

```
['use deep learning for image classification', 'insights from new york car accident reports', 'v
[1429.0, 1330.0, 1431.0, 1427.0, 1364.0, 1314.0, 1293.0, 1170.0, 1162.0, 1304.0]
```

```
In [19]: # 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 [20]: # create the user-article matrix with 1's and 0's

def create_user_item_matrix(df):
    '''
    INPUT:
    df - pandas dataframe with article_id, title, user_id columns

    OUTPUT:
    user_item - user item matrix

    Description:
    Return a matrix with user ids as rows and article ids on the columns with 1 values
    an article and a 0 otherwise
    '''
    # Fill in the function here


    item = df.groupby(by=['user_id', 'article_id'])

    user_item = item.agg(lambda x: 1).unstack().fillna(0)
```

```
    return user_item # return the user_item matrix

user_item = create_user_item_matrix(df)
```

```
In [21]: ## Tests: You should just need to run this cell.  Don't change the code.
         assert user_item.shape[0] == 5149, "Oops!  The number of users in the user-article matr
         assert user_item.shape[1] == 714, "Oops!  The number of articles in the user-article ma
         assert user_item.sum(axis=1)[1] == 36, "Oops!  The number of articles seen by user 1 do
         print("You have passed our quick tests!  Please proceed!")
```

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

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

Use the tests to test your function.

```
In [22]: def find_similar_users(user_id, user_item=user_item):
             '''
             INPUT:
             user_id - (int) a user_id
             user_item - (pandas dataframe) matrix of users by articles:
                         1's when a user has interacted with an article, 0 otherwise

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

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

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

             # sort by similarity
             similarity = similarity.sort_values(ascending=False)

             # create list of just the ids
             most_similar_users = similarity.index.tolist()

             # remove 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 [23]: # 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 [24]: 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

             articles = df[df.article_id.isin(article_ids)]["title"]

             article_names = articles.drop_duplicates().tolist()

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


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

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

             Description:
```

```
    Provides a list of the article_ids and article titles that have been seen by a user
    '''
    # Your code here

    article_ids = [str(id) for id in list(user_item.loc[user_id][user_item.loc[user_id]

    article_names = get_article_names(article_ids)

    return article_ids, article_names # return the ids and names


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

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

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

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

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

    '''
    # Your code here

    user_ids = find_similar_users(user_id)

    recs = df[df['user_id'].isin(user_ids)]['article_id']

    recs = list(set(recs))

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

In [25]:
```
# Check Results
get_article_names(user_user_recs(1, 10)) # Return 10 recommendations for user 1
```

Out[25]:
```
['using pixiedust for fast, flexible, and easier data analysis and experimentation',
 'healthcare python streaming application demo',
 'use deep learning for image classification',
```

```
'ml optimization using cognitive assistant',
'deploy your python model as a restful api',
'visualize data with the matplotlib library',
'upload files to ibm data science experience using the command line',
'classify tumors with machine learning',
'configuring the apache spark sql context',
'got zip code data? prep it for analytics.  ibm watson data lab  medium',
'the unit commitment problem',
'apache spark lab, part 1: basic concepts',
'getting started with python',
'timeseries data analysis of iot events by using jupyter notebook',
'10 must attend data science, ml and ai conferences in 2018',
'housing (2015): united states demographic measures',
'access db2 warehouse on cloud and db2 with python',
'the nurse assignment problem',
'dsx: hybrid mode',
'predicting churn with the spss random tree algorithm',
'data science for real-time streaming analytics',
'analyze energy consumption in buildings',
'ibm watson facebook posts for 2015',
'web picks - dataminingapps',
'visualize car data with brunel',
'common excel tasks demonstrated in\xa0pandas',
'analyze open data sets with pandas dataframes',
'use xgboost, scikit-learn & ibm watson machine learning apis',
'intents & examples for ibm watson conversation',
'apache spark lab, part 3: machine learning',
'uci: sms spam collection',
'putting a human face on machine learning',
'gosales transactions for naive bayes model',
'access mysql with python',
'access mysql with r',
'ibm data science experience white paper - sparkr transforming r into a tool for big d
'gosales transactions for logistic regression model',
'welcome to pixiedust',
'leverage python, scikit, and text classification for behavioral profiling',
'dry bulb temperature, by country, station and year',
'world tourism data by the world tourism organization',
'an introduction to stock market data analysis with r (part 1)',
'insights from new york car accident reports',
'maximize oil company profits',
'airbnb data for analytics: austin listings',
'city population by sex, city and city type',
'using github for project control in dsx',
'use sql with data in hadoop python',
'discover hidden facebook usage insights',
'overfitting in machine learning: what it is and how to prevent it',
'use apache systemml and spark for machine learning',
```

```
'use the cloudant-spark connector in python notebook',
'analyze open data sets with spark & pixiedust',
'uci ml repository: chronic kidney disease data set',
'uci: heart disease - cleveland',
'income (2015): united states demographic measures',
'apache spark as the new engine of genomics',
'how to use version control (github) in rstudio within dsx?',
'pixieapp for outlier detection',
'excel files: loading from object storage  python',
'developing for the ibm streaming analytics service',
'building a business that combines human experts and data science',
'time series prediction using recurrent neural networks (lstms)',
'simple graphing with ipython and\xa0pandas',
'graph-based machine learning',
'airbnb data for analytics: washington d.c. listings',
'uci: white wine quality',
'model bike sharing data with spss',
'data visualization with r: scrum metrics',
'contraceptive prevalence (% women 15-49) by country',
'optimizing a marketing campaign: moving from predictions to actions',
'analyze precipitation data',
'airbnb data for analytics: amsterdam calendar',
'finding optimal locations of new store using decision optimization',
'analyze accident reports on amazon emr spark',
'quick guide to build a recommendation engine in python',
'percentage of internet users by country',
'neural language modeling from scratch (part 1)',
'learning statistics on youtube',
'accelerate your workflow with dsx',
'interest rates',
'analyzing data by using the sparkling.data library features',
'breast cancer wisconsin (diagnostic) data set',
'interactive time series with dygraphs',
'health insurance (2015): united states demographic measures',
'imitation learning in tensorflow (hopper from openai gym)',
'10 tips on using jupyter notebook',
'country statistics: infant mortality rate',
'use r dataframes & ibm watson natural language understanding',
'mobile-cellular telephone subscriptions per 100 inhabitants, worldwide',
'improved water source by country: % population with access',
'better together: spss and data science experience',
'502    forgetting the past to learn the future: long ...\nName: title, dtype: object'
'annual precipitation by country 1990-2009',
'520    using notebooks with pixiedust for fast, flexi...\nName: title, dtype: object'
'intentional homicide, number and rate per 100,000 population, by country',
'fortune 100 companies',
'jupyter notebook tutorial',
'high-tech exports as % of manufactured exports by country',
```

```
'a moving average trading strategy',
'education (2015): united states demographic measures',
'how to perform a logistic regression in r',
'python machine learning: scikit-learn tutorial',
'country statistics - europe - population and society',
'how smart catalogs can turn the big data flood into an ocean of opportunity',
'deep learning with tensorflow course by big data university',
'learn tensorflow and deep learning together and now!',
'apache spark lab, part 2: querying data',
'15 page tutorial for r',
'analyze traffic data from the city of san francisco',
'country statistics: unemployment rate',
'read and write data to and from amazon s3 buckets in rstudio',
'how the circle line rogue train was caught with data',
'use decision optimization to schedule league games',
'airbnb data for analytics: vienna reviews',
'country statistics: health expenditures',
'using brunel in ipython/jupyter notebooks',
'generative adversarial networks (gans)',
'what is hadoop?',
'use spark for r to load data and run sql queries',
'working with notebooks in dsx',
'probabilistic graphical models tutorial\u200a\u200apart 1  stats and bots',
'household consumption expenditure',
'the greatest public datasets for ai  startup grind',
'this week in data science (august 02, 2016)',
'use spark r to load and analyze data',
'learn about data science in world of watson',
"a beginner's guide to variational methods",
'occupation (2015): united states demographic measures',
'rapidly build machine learning flows with dsx',
'using machine learning to predict baseball injuries',
'use spark for scala to load data and run sql queries',
'simple linear regression? do it the bayesian way',
'deep learning trends and an example',
'analyze open data sets using pandas in a python notebook',
'airbnb data for analytics: barcelona reviews',
'machine learning for everyone',
'tidy up your jupyter notebooks with scripts',
'deep learning from scratch i: computational graphs',
'data visualization playbook: telling the data story',
'experience iot with coursera',
'use ibm data science experience to read and write data stored on amazon s3',
'this week in data science',
'what is text analytics?',
'analyzing streaming data from kafka topics',
'total population by country',
'working interactively with rstudio and notebooks in dsx',
```

```
'transfer learning for flight delay prediction via variational autoencoders',
'the nurse assignment problem data',
'uci: car evaluation',
'adolescent fertility rate (births per 1,000 women ages 15-19), worldwide',
'tensorflow quick tips',
'airbnb data for analytics: amsterdam listings',
'an introduction to scientific python (and a bit of the maths behind it)  numpy',
'10 pieces of advice to beginner data scientists',
'uci: poker hand - testing data set',
'data model with streaming analytics and python',
'1357    what i learned implementing a classifier from ...\nName: title, dtype: object
'how to use db2 warehouse on cloud in data science experience notebooks',
'movie recommender system with spark machine learning',
'modeling energy usage in new york city',
'1448    i ranked every intro to data science course on...\nName: title, dtype: object
'airbnb data for analytics: berlin reviews',
'workflow in r',
'jupyter notebooks with scala, python, or r kernels',
'car performance data',
'fertility rate by country in total births per woman',
'brunel in jupyter',
'country statistics: stock of broad money',
'sparklyr  r interface for apache spark',
'deep learning with data science experience',
'learn basics about notebooks and apache spark',
'shaping data with ibm data refinery',
'enjoy python 3.5 in jupyter notebooks',
'the million dollar question: where is my data?',
'ibm watson machine learning: get started',
'challenges in deep learning',
'data science in the cloud',
'awesome deep learning papers',
'uci: forest fires',
'tidy data in python',
'from python nested lists to multidimensional numpy arrays',
'analyze db2 warehouse on cloud data in rstudio in dsx',
'neural networks for beginners: popular types and applications',
'rstudio ide  cheat sheet',
'world marriage data',
'flexdashboard: interactive dashboards for r',
'programmatic evaluation using watson conversation',
'forest area by country in sq km',
'uci: adult - predict income',
'developing ibm streams applications with the python api (version 1.6)',
'are your predictive models like broken clocks?',
'improving real-time object detection with yolo',
'persistent changes to spark config in dsx',
'access ibm analytics for apache spark from rstudio',
```

```
'mobile cellular subscriptions per 100 people by country',
'country population by gender 1985-2005',
'part-time employment rate, worldwide, by country and year',
'annual % population growth by country',
'country statistics: children under the age of 5 years underweight',
'cleaning the swamp: turn your data lake into a source of crystal-clear insight',
'leaflet: interactive web maps with r',
'an attempt to understand boosting algorithm(s)',
'wages',
'spark sql - rapid performance evolution',
'uci: poker hand - training data set',
'variational auto-encoder for "frey faces" using keras',
'easy json loading and social sharing in dsx notebooks',
'build a python app on the streaming analytics service',
'shiny 0.12: interactive plots with ggplot2',
'build a predictive analytic model',
'country statistics: telephones - mobile cellular',
'load data into rstudio for analysis in dsx',
'interactive web apps with shiny cheat sheet',
'times world university ranking analysis',
'uci: iris',
'self-service data preparation with ibm data refinery',
'real-time sentiment analysis of twitter hashtags with spark (+ pixiedust)',
'top 10 machine learning algorithms for beginners',
'collecting data science cheat sheets',
'agriculture, value added (% of gdp) by country',
'customer demographics and sales',
'score a predictive model built with ibm spss modeler, wml & dsx',
'sudoku',
'government consumption expenditure',
'super fast string matching in python',
'getting started with graphframes in apache spark',
'connect to db2 warehouse on cloud and db2 using scala',
"2875    hugo larochelle's neural network & deep learni...\nName: title, dtype: object
'10 powerful features on watson data platform, no coding necessary',
'make machine learning a reality for your enterprise',
'declarative machine learning',
'working with db2 warehouse on cloud in data science experience',
'united states demographic measures: zip code tabulation areas (zctas)',
'access postgresql with python',
'making data science a team sport',
'the pandas data analysis library',
'word2vec in data products',
'reducing overplotting in scatterplots',
'5 practical use cases of social network analytics: going beyond facebook and twitter'
'airbnb data for analytics: berlin calendar',
'notebooks: a power tool for data scientists',
'use the machine learning library',
```

```
'd3heatmap: interactive heat maps',
'births attended by skilled health staff (% of total) by country',
'the 3 kinds of context: machine learning and the art of the frame',
'introducing streams designer',
'country statistics: commercial bank prime lending rate',
'sector correlations shiny app',
'brunel interactive visualizations in jupyter notebooks',
'drowning in data sources: how data cataloging could fix your findability problems',
'brunel 2.0 preview',
'shiny: a data scientists best friend',
'new shiny cheat sheet and video tutorial',
'3992    using apache spark to predict attack vectors a...\nName: title, dtype: object
'this week in data science (may 30, 2017)',
'uci: wine recognition',
'use data assets in a project using ibm data catalog',
'model a golomb ruler',
'airbnb data for analytics: boston calendar',
'pixiedust: magic for your python notebook',
'blogging with brunel',
'create a connection and add it to a project using ibm data refinery',
'mapping points with folium',
'uci: red wine quality',
'country statistics: gdp - per capita (ppp)',
'what caused the challenger disaster?',
'airbnb data for analytics: portland listings',
'shiny 0.13.0',
'data visualization playbook: revisiting the basics',
'hyperparameter optimization: sven hafeneger',
'detect malfunctioning iot sensors with streaming analytics',
'a comparison of logistic regression and naive bayes ',
'country statistics: distribution of family income - gini index',
'uci: abalone',
'country statistics: life expectancy at birth',
'airbnb data for analytics: new york city calendar',
'manage object storage in dsx',
'airbnb data for analytics: boston listings',
'using machine learning to predict value of homes on airbnb',
'pixiedust gets its first community-driven feature in 1.0.4',
'pixiedust 1.0 is here!  ibm watson data lab',
'dt: an r interface to the datatables library',
'airbnb data for analytics: amsterdam reviews',
'which one to choose for your problem',
'machine learning exercises in python, part 1',
'electric power consumption (kwh per capita) by country',
'adoption of machine learning to software failure prediction',
'airbnb data for analytics: portland reviews',
'airbnb data for analytics: portland calendar',
'spark 1.4 for rstudio',
```

```
'back to basics  jupyter notebooks',
'how to map usa rivers using ggplot2',
'airbnb data for analytics: venice listings',
'consumer prices',
'airbnb data for analytics: venice calendar',
'population below national poverty line, total, percentage',
'airbnb data for analytics: venice reviews',
'worldwide fuel oil consumption by household (in 1000 metric tons)',
'practical tutorial on random forest and parameter tuning in r',
'optimization for deep learning highlights in 2017',
'0 to life-changing app: scala first steps and an interview with jakob odersky',
'ibm data catalog is now generally available',
'using deep learning with keras to predict customer churn',
'spark-based machine learning tools for capturing word meanings',
'leverage scikit-learn models with core ml',
'house building with worker skills',
'gradient boosting explained',
'three reasons machine learning models go out of sync',
'working with sqlite databases using python and pandas',
'country statistics: crude oil - proved reserves',
'country statistics: crude oil - exports',
'how to write the first for loop in r',
'upload data and create data frames in jupyter notebooks',
"i'd rather predict basketball games than elections: elastic nba rankings",
'this week in data science (may 23, 2017)',
'i am not a data scientist  ibm watson data lab',
'predicting gentrification using longitudinal census data',
'this week in data science (march 28, 2017)',
'machine learning and the science of choosing',
'spark 2.1 and job monitoring available in dsx',
'deep forest: towards an alternative to deep neural networks',
'10 essential algorithms for machine learning engineers',
'whats new in the streaming analytics service on bluemix',
'neurally embedded emojis',
'data wrangling with dplyr and tidyr cheat sheet',
'why you should master r (even if it might eventually become obsolete)',
'best packages for data manipulation in r',
'this week in data science (april 18, 2017)',
'higher-order logistic regression for large datasets',
'roads paved as % of total roads by country',
'how to choose a project to practice data science',
'this week in data science (february 14, 2017)',
'data structures related to machine learning algorithms',
'united states demographic measures: population and age',
'data tidying in data science experience',
'trust in data science',
'a dynamic duo  inside machine learning  medium',
'this week in data science (april 25, 2017)',
```

```
'use spark for python to load data and run sql queries',
'ensemble learning to improve machine learning results',
'this week in data science (january 10, 2017)',
'apache spark 2.0: extend structured streaming for spark ml',
'using deep learning to reconstruct high-resolution audio',
'introduction to market basket analysis in\xa0python',
'7292    a dramatic tour through pythons data visualiz...\nName: title, dtype: object'
'when machine learning matters ů erik bernhardsson',
'the machine learning database',
'co2 emissions (metric tons per capita) by country',
'data science bowl 2017',
'the data science process',
"a kaggler's guide to model stacking in practice",
'statistics for hackers',
'10 data science podcasts you need to be listening to right now',
'employed population by occupation and age',
'this week in data science (april 4, 2017)',
'pearson correlation aggregation on sparksql',
'ml algorithm != learning machine',
'households by type of household, age and sex of head of household',
'hurricane how-to',
'this week in data science (january 24, 2017)',
'airbnb data for analytics: vienna listings',
'external debt stocks, total (dod, current us$) by country',
'what is machine learning?',
'automating web analytics through python',
'life expectancy at birth by country in total years',
'modern machine learning algorithms',
'markdown for jupyter notebooks cheatsheet',
'machine learning for the enterprise.',
'access postgresql with r',
'introduction to neural networks, advantages and applications',
'this week in data science (april 11, 2017)',
"let's have some fun with nfl data",
'pulling and displaying etf data',
'this week in data science (january 31, 2017)',
'interconnect with us',
'8170    data science expert interview: dez blanchfield...\nName: title, dtype: object
'environment statistics database - water',
'statistical bias types explained (with examples)',
'how to scale your analytics using r',
'7 types of job profiles that makes you a data scientist',
'airbnb data for analytics: antwerp listings test',
'r for data science',
'flightpredict ii: the sequel   ibm watson data lab',
'top 10 machine learning use cases: part 1',
'htmlwidgets: javascript data visualization for r',
'airbnb data for analytics: vancouver listings',
```

```
'what is smote in an imbalanced class setting (e.g. fraud detection)?',
'using machine learning to predict parking difficulty',
'improving the roi of big data and analytics through leveraging new sources of data',
'this week in data science (february 28, 2017)',
'this week in data science (may 2, 2017)',
'country statistics: reserves of foreign exchange and gold',
'airbnb data for analytics: sydney calendar',
'this week in data science (may 16, 2017)',
'some random weekend reading',
'airbnb data for analytics: antwerp reviews',
'artificial intelligence, ethically speaking  inside machine learning  medium',
'getting started with apache mahout',
'this week in data science (february 21, 2017)',
'airbnb data for analytics: toronto reviews',
'airbnb data for analytics: austin reviews',
'country statistics: internet users',
'clustering: a guide for the perplexed',
'using rstudio in ibm data science experience',
'recommendation system algorithms  stats and bots',
'use ibm data science experience to detect time series anomalies',
'recommender systems: approaches & algorithms',
'web picks by dataminingapps',
'enhanced color mapping',
'data science platforms are on the rise and ibm is leading the way',
'essentials of machine learning algorithms (with python and r codes)',
'web picks (week of 23 january 2017)',
'predicting the 2016 us presidential election',
'why relational databases and r?',
'publish notebooks to github in dsx',
'the art of side effects: curing apache spark streamings amnesia (part 1/2)',
'apache spark sql analyzer resolves order-by column',
'airbnb data for analytics: toronto calendar',
'dont overlook simpler techniques and algorithms',
'airbnb data for analytics: washington d.c. reviews',
'airbnb data for analytics: sydney reviews',
'seti data, publicly available, from ibm',
'airbnb data for analytics: antwerp calendar',
'brunel: imitation is a sincere form of flattery',
'this week in data science (march 7, 2017)',
'country statistics: electricity - from fossil fuels',
'bayesian regularization for #neuralnetworks  autonomous agents\u200a\u200a#ai',
'jupyter (ipython) notebooks features',
'apache spark 2.0: migrating applications',
'0 to life-changing app: new apache systemml api on spark shell',
'airbnb data for analytics: paris listings',
'the power of machine learning in spark',
'nips 2016  day 2 highlights',
'airbnb data for analytics: vienna calendar',
```

```
'airbnb data for analytics: trentino listings',
'making data cleaning simple with the sparkling.data library',
'airbnb data for analytics: vancouver reviews',
'how to ease the strain as your data volumes rise',
'country statistics: gross national saving',
'country statistics: telephones - fixed lines',
'data visualization playbook: the right level of detail',
"december '16 rstudio tips and tricks",
'r markdown reference guide',
'this week in data science (february 7, 2017)',
'ratio (% of population) at national poverty line by country',
'backpropagation  how neural networks learn complex behaviors',
'united states demographic measures: education',
'finding the user in data science',
'statistical bias types explained',
'airbnb data for analytics: trentino reviews',
'tidyverse practice: mapping large european cities',
'a classification problem',
'share the (pixiedust) magic  ibm watson data lab  medium',
'country statistics: central bank discount rate',
'data visualization with ggplot2 cheat sheet',
'airbnb data for analytics: sydney listings',
'data science experience demo: modeling energy usage in nyc',
'country statistics: roadways',
'tidyr 0.6.0',
'analyze starcraft ii replays with jupyter notebooks',
'twelve\xa0ways to color a map of africa using brunel',
'how open api economy accelerates the growth of big data and analytics',
'unstructured and structured data versus repetitive and non-repetitive',
'airbnb data for analytics: seattle reviews',
'country statistics: stock of domestic credit',
'run shiny applications in rstudio in dsx',
'airbnb data for analytics: dublin reviews',
"feature importance and why it's important",
'pseudo-labeling a simple semi-supervised learning method',
'using apply, sapply, lapply in r',
'gross national income per capita, atlas method (current us$) by country',
'airbnb data for analytics: barcelona listings',
'open sourcing 223gb of driving data  udacity inc',
'worldwide electricity demand and production 1990-2012',
'random forest interpretation  conditional feature contributions',
'how can data scientists collaborate to build better business',
'airbnb data for analytics: chicago listings',
'what is systemml? why is it relevant to you?',
'top 20 r machine learning and data science packages',
'data visualization: the importance of excluding unnecessary details',
'airbnb data for analytics: madrid listings',
'apache spark 2.0: impressive improvements to spark sql',
```

```
'one year as a data scientist at stack overflow',
'web picks (week of 4 september 2017)',
'united states demographic measures: income',
'this week in data science (november 01, 2016)',
'a visual explanation of the back propagation algorithm for neural networks',
'8 ways to turn data into value with apache spark machine learning',
'a glimpse inside the mind of a data scientist',
'airbnb data for analytics: nashville calendar',
'military expenditure as % of gdp by country',
'improving quality of life with spark-empowered machine learning',
'20405    how to tame the valley  hessian-free hacks fo...\nName: title, dtype: object
'the difference between ai, machine learning, and deep learning?',
'time series analysis using max/min and neuroscience',
'country statistics: population',
'country statistics: airports',
'airbnb data for analytics: toronto listings',
'airbnb data for analytics: antwerp listings',
'an interview with pythonista katharine jarmul',
'labor',
'country statistics: maternal mortality rate',
'the data processing inequality',
'total employment, by economic activity (thousands)',
'airbnb data for analytics: boston reviews',
'airbnb data for analytics: new york city reviews',
'top analytics tools in 2016',
'apache spark 2.0: machine learning. under the hood and over the rainbow.',
'what is spark?',
'energy use (kg of oil equivalent per capita) by country',
'apache systemml',
'beyond parallelize and collect',
'airbnb data for analytics: brussels reviews',
'how to get a job in deep learning',
'country statistics: crude oil - imports',
'building custom machine learning algorithms with apache systemml',
'laplace noising versus simulated out of sample methods (cross frames)',
'marital status of men and women',
'foundational methodology for data science',
'airbnb data for analytics: paris reviews',
'the new builders podcast ep 3: collaboration',
'creating the data science experience',
'intelligent applications - apache spark',
'recent trends in recommender systems',
'country populations 15 years of age and over, by educational attainment, age and sex'
'country statistics: current account balance',
'apache spark @scale: a 60 tb+ production use case',
'a survey of books about apache spark',
'using the maker palette in the ibm data science experience',
'airbnb data for analytics: san diego reviews',
```

```
'airbnb data for analytics: athens listings',
'a fast on-disk format for data frames for r and python, powered by apache arrow',
'country surface area (sq. km)',
'a day in the life of a data engineer',
'data science experience documentation',
'this week in data science (january 17, 2017)',
'airbnb data for analytics: new orleans listings',
'advancements in the spark community',
'airbnb data for analytics: vancouver calendar',
"h2o with ibm's data science experience (dsx)",
'this week in data science (july 26, 2016)',
'readr 1.0.0',
'airbnb data for analytics: oakland reviews',
'worldwide county and region - national accounts - gross national income 1948-2010',
'international liquidity',
'airbnb data for analytics: trentino calendar',
'primary school completion rate % of relevant age group by country',
'airbnb data for analytics: mallorca reviews',
'airbnb data for analytics: athens calendar',
'machine learning for the enterprise',
'do i need to learn r?',
'a new version of dt (0.2) on cran',
'airbnb data for analytics: san francisco listings',
'natural gas production, 1995 - 2012, worldwide',
'unmet need for family planning, spacing, percentage, worldwide, by country',
'cache table in apache spark sql',
'ggplot2 2.2.0 coming soon!',
'this week in data science (november 22, 2016)',
'airbnb data for analytics: paris calendar',
'xml2 1.0.0',
'this week in data science (december 27, 2016)',
'airbnb data for analytics: athens reviews',
'measles immunization % children 12-23 months by country',
'consumption of ozone-depleting cfcs in odp metric tons',
'airbnb data for analytics: montreal listings',
'airbnb data for analytics: new orleans reviews',
'dplyr 0.5.0',
'roads, paved (% of total roads), worldwide, 1990-2011',
'let data dictate the visualization',
'foreign direct investment, net inflows (bop, current us$) by country',
'big data is better data',
'package development with devtools  cheat sheet',
'can a.i. be taught to explain itself?',
'get social with your notebooks in dsx',
'run dsx notebooks on amazon emr',
'using bigdl in dsx for deep learning on spark',
'overlapping co-cluster recommendation algorithm (ocular)',
'using dsx notebooks to analyze github data',
```

```
'airbnb data for analytics: santa cruz county reviews',
'51822    using apache spark as a parallel processing fr...\nName: title, dtype: objec
'web picks (december 2017)',
'3 scenarios for machine learning on multicloud',
'discover, catalog and govern data with ibm data catalog',
'greenhouse gas emissions worldwide',
'visualising data the node.js way',
'governance overview for ibm data catalog',
'mycheatsheets.com',
'perform sentiment analysis with lstms, using tensorflow',
'this week in data science (october 18, 2016)',
'poverty (2015): united states demographic measures',
'visual information theory ',
'build a logistic regression model with wml & dsx',
'country statistics: electricity - consumption',
'how ibm builds an effective data science team',
'join and enrich data from multiple sources',
'deep learning achievements over the past year ',
'airbnb data for analytics: austin calendar',
'visualize the 1854 london cholera outbreak',
'process events from the watson iot platform in a streams python application',
'aspiring data scientists! start to learn statistics with these 6 books!',
'airbnb data for analytics: barcelona calendar',
'country statistics: imports',
'54174    detect potentially malfunctioning sensors in r...\nName: title, dtype: objec
'calculate moving averages on real time data with streams designer',
'airbnb data for analytics: london reviews',
'transform anything into a vector',
'breaking the 80/20 rule: how data catalogs transform data scientists productivity',
'generalization in deep learning',
'ingest data from message hub in a streams flow',
'9 mistakes to avoid when starting your career in data science',
'predicting flight cancellations using weather data, part 3',
'probabilistic graphical models tutorial\u200a\u200apart 2  stats and bots',
'develop a scala spark model on chicago building violations',
'10 data science, machine learning and ai podcasts you must listen to',
'country statistics: budget surplus or deficit',
'small steps to tensorflow',
'migrating to python 3 with pleasure',
'collect your own fitbit data with python',
'load db2 warehouse on cloud data with apache spark in dsx',
'how to solve 90% of nlp problems',
'56594    lifelong (machine) learning: how automation ca...\nName: title, dtype: objec
'python if statements explained (python for data science basics #4)',
'fighting gerrymandering: using data science to draw fairer congressional districts',
'work with data connections in dsx',
'analyze ny restaurant data using spark in dsx',
'working with ibm cloud object storage in python',
```

```
'watson machine learning for developers',
'apache spark: upgrade and speed-up your analytics',
'making sense of the bias / variance trade-off in (deep) reinforcement learning',
'categorize urban density',
'ibm data catalog overview',
'70 amazing free data sources you should know',
'working with ibm cloud object storage in r',
'predict chronic kidney disease using spss modeler flows',
'building your first machine learning system ',
'country statistics: natural gas - consumption',
'python for loops explained (python for data science basics #5)',
"for ai to get creative, it must learn the rules--then how to break 'em",
'data science of variable selection',
'why even a moths brain is smarter than an ai',
'deep learning, structure and innate priors',
'a guide to convolution arithmetic for deep learning',
'a guide to receptive field arithmetic for convolutional neural networks',
'the two phases of gradient descent in deep learning',
'the random forest algorithm ',
'get started with streams designer by following this roadmap',
'time series anomaly detection algorithms  stats and bots',
'announcing dsx environments in beta!',
'effectively using\xa0matplotlib',
'missing data conundrum: exploration and imputation techniques',
'calls by customers of a telco company',
'customers of a telco including services used',
'empirical bayes for multiple sample sizes',
'country statistics: industrial production growth rate',
'the t-distribution: a key statistical concept discovered by a beer brewery',
'talent vs luck: the role of randomness in success and failure',
'working with on-premises databases  step by step',
'airbnb data for analytics: london listings',
'airbnb data for analytics: san diego listings',
'environment statistics database - waste',
'understanding empirical bayes estimation (using baseball statistics)',
'bayesian nonparametric models  stats and bots',
'cifar-100 - python version',
'fashion-mnist',
'country statistics: refined petroleum products - production',
'ibm cloud sql query',
'cifar-10 - python version',
'watson assistant workspace analysis with user logs',
'airbnb data for analytics: chicago calendar',
'use iot data in streams designer for billing and alerts',
'breast cancer detection with xgboost, wml and scikit',
'introducing ibm watson studio ',
'best practices for custom models in watson visual recognition',
'apple, ibm add machine learning to partnership with watson-core ml coupling',
```

```
 'analyze facebook data using ibm watson and watson studio',
 'find airbnb deals in portland with machine learning using r',
 'from scikit-learn model to cloud with wml client',
 'watson speech-to-text services  tl;dr need not apply',
 'from local spark mllib model to cloud with watson machine learning',
 'social media insights with watson developer cloud & watson studio',
 'from spark ml model to online scoring with scala',
 'analyze data, build a dashboard with spark and pixiedust',
 'predict loan applicant behavior with tensorflow neural networking',
 'annual % inflation by country',
 'continuous learning on watson',
 'using shell scripts to control data flows created in watson applications',
 '66855    migration from ibm bluemix data connect api (a...\nName: title, dtype: objec
 'whats new in data refinery?',
 'stacking multiple custom models in watson visual recognition',
 'style transfer experiments with watson machine learning',
 'country statistics: merchant marine',
 'geographic coordinates of world locations',
 'webinar: april 11 - thinking inside the box: you can do that inside a data frame?!',
 'dimensionality reduction algorithms',
 'build deep learning architectures with neural network modeler',
 'a tensorflow regression model to predict house values',
 'use pmml to predict iris species',
 'country statistics: railways',
 'airbnb data for analytics: chicago reviews',
 'refugees',
 'refugees, worldwide, 2003 - 2013',
 '68879    dont throw more data at the problem! heres h...\nName: title, dtype: object'
 'you could be looking at it all wrong',
 'airbnb data for analytics: seattle calendar',
 'web picks (week of 2 october 2017)',
 'data science expert interview: holden karau',
 'create a project for watson machine learning in dsx',
 'country statistics: market value of publicly traded shares',
 'airbnb data for analytics: washington d.c. calendar',
 'build a naive-bayes model with wml & dsx',
 'load and analyze public data sets in dsx',
 'the new builders ep. 13: all the data thats fit to analyze',
 'create a project in dsx']
```

In [26]:
```
# Test your functions here - No need to change this code - just run this cell
assert set(get_article_names(['1024.0', '1176.0', '1305.0', '1314.0', '1422.0', '1427.0
assert set(get_article_names(['1320.0', '232.0', '844.0'])) == set(['housing (2015): un
assert set(get_user_articles(20)[0]) == set(['1320.0', '232.0', '844.0'])
assert set(get_user_articles(20)[1]) == set(['housing (2015): united states demographic
assert set(get_user_articles(2)[0]) == set(['1024.0', '1176.0', '1305.0', '1314.0', '14
assert set(get_user_articles(2)[1]) == set(['using deep learning to reconstruct high-re
print("If this is all you see, you passed all of our tests!  Nice job!")
```

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

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

- Instead of arbitrarily choosing when we obtain users who are all the same closeness to a given user - choose the users that have the most total article interactions before choosing those with fewer article interactions.
- Instead of arbitrarily choosing articles from the user where the number of recommended articles starts below m and ends exceeding m, choose articles with the articles with the most total interactions before choosing those with fewer total interactions. This ranking should be what would be obtained from the **top_articles** function you wrote earlier.

```
In [27]: 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 provided
                            num_interactions - the number of articles viewed by the user - if a

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

             '''
             # Your code here

             neighbors_df = pd.DataFrame(columns=['neighbor_id', 'similarity', 'num_interactions

             for user in user_item.index:

                 if user == user_id:
                     continue

                 neighbors_df.loc[user] = [user, np.dot(user_item.loc[user_id, :], user_item.loc
                                           df[df['user_id']==user]['article_id'].count()]

             neighbors_df.sort_values(by=['similarity', 'num_interactions'], ascending=False, in


             return neighbors_df # Return the dataframe specified in the doc_string
```

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

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

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

    recs = []

    neighbors_df = get_top_sorted_users(user_id)

    the_user_articles, the_article_names = get_user_articles(user_id)

    for user in neighbors_df['neighbor_id']:

        article_ids, article_names = get_user_articles(user)

        for id in article_ids:

            if id not in the_user_articles:
                recs.append(id)

            if len(recs) >= m:
                break

        if len(recs) >= m:
                break
```

```
            if len(recs) < m:

                for id in [str(id) for id in get_top_article_ids(100)]:

                    if id not in the_user_articles:
                        recs.append(id)

                    if len(recs) >= m:
                            break

            rec_names = get_article_names(recs)

            return recs, rec_names
```

```
In [28]: # Quick spot check - don't change this code - just use it to test your functions
         rec_ids, rec_names = user_user_recs_part2(20, 10)
         print("The top 10 recommendations for user 20 are the following article ids:")
         print(rec_ids)
         print()
         print("The top 10 recommendations for user 20 are the following article names:")
         print(rec_names)
```

```
The top 10 recommendations for user 20 are the following article ids:
['12.0', '109.0', '125.0', '142.0', '164.0', '205.0', '302.0', '336.0', '362.0', '465.0']

The top 10 recommendations for user 20 are the following article names:
['timeseries data analysis of iot events by using jupyter notebook', 'dsx: hybrid mode', 'accele
```

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 [29]: get_top_sorted_users(1).iloc[0]
```

```
Out[29]: neighbor_id         3933.0
         similarity            35.0
         num_interactions      45.0
         Name: 3933, dtype: float64
```

```
In [30]: get_top_sorted_users(131).iloc[9]
```

```
Out[30]: neighbor_id         242.0
         similarity           25.0
         num_interactions    148.0
         Name: 242, dtype: float64
```

```
In [31]: ### Tests with a dictionary of results

         user1_most_sim = 3933 # Find the user that is most similar to user 1
         user131_10th_sim = 242 # Find the 10th most similar user to user 131
```

```
In [32]: ## 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.

Provide your response here.

Because the user hasn't viewed any articles before and we don't have any information on the user, get top article ids would be a better way for us to generate recommendations.

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 [33]: new_user = '0.0'

         # What would your recommendations be for this new user '0.0'?  As a new user, they have
         # Provide a list of the top 10 article ids you would give to

         new_user_recs = [str(id) for id in get_top_article_ids(10)] # Your recommendations here
```

```
In [34]: 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 [35]: 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 [36]: # 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 [37]: # Load the matrix here
         user_item_matrix = pd.read_pickle('user_item_matrix.p')

In [38]: # quick look at the matrix
         user_item_matrix.head()

Out[38]: article_id  0.0  100.0  1000.0  1004.0  1006.0  1008.0  101.0  1014.0  1015.0  \
         user_id
         1           0.0    0.0     0.0     0.0     0.0     0.0    0.0     0.0     0.0
```

```
2           0.0    0.0     0.0     0.0     0.0     0.0    0.0     0.0     0.0
3           0.0    0.0     0.0     0.0     0.0     0.0    0.0     0.0     0.0
4           0.0    0.0     0.0     0.0     0.0     0.0    0.0     0.0     0.0
5           0.0    0.0     0.0     0.0     0.0     0.0    0.0     0.0     0.0

article_id  1016.0  ...    977.0  98.0  981.0  984.0  985.0  986.0  990.0  \
user_id             ...
1              0.0  ...      0.0   0.0    1.0    0.0    0.0    0.0    0.0
2              0.0  ...      0.0   0.0    0.0    0.0    0.0    0.0    0.0
3              0.0  ...      1.0   0.0    0.0    0.0    0.0    0.0    0.0
4              0.0  ...      0.0   0.0    0.0    0.0    0.0    0.0    0.0
5              0.0  ...      0.0   0.0    0.0    0.0    0.0    0.0    0.0

article_id  993.0  996.0  997.0
user_id
1             0.0    0.0    0.0
2             0.0    0.0    0.0
3             0.0    0.0    0.0
4             0.0    0.0    0.0
5             0.0    0.0    0.0

[5 rows x 714 columns]
```

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

```
In [39]: # 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
```

Provide your response here.

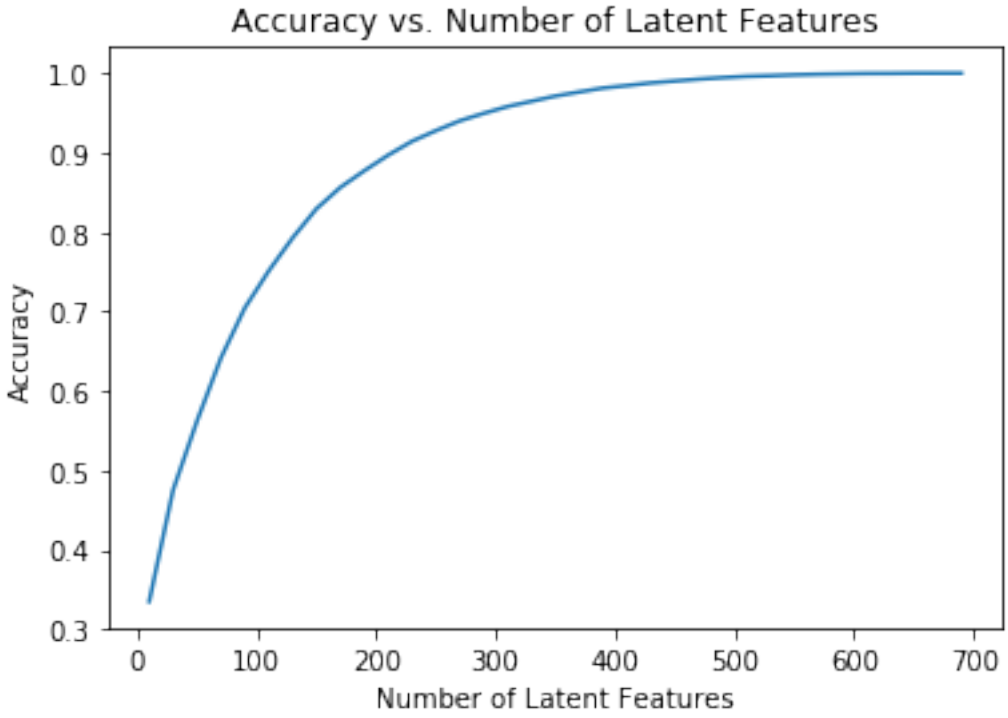
When there are no null values, numpy's SVD works. Because the matix was full of null values in the lesson, numpy SVD failed. Our matrix only has two values in this case, 1 if the user interacted and 0 otherwise, therefore there are no null values, and we can utilise numpy's SVD.

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

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

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

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

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

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



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 [41]:
```
df_train = df.head(40000)
df_test = df.tail(5993)

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

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

    '''
    # Your code here

    user_item_train = create_user_item_matrix(df_train)

    user_item_test = 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 [42]:
```
# Replace the values in the dictionary below
a = 662
b = 574
c = 20
d = 0


sol_4_dict = {
    'How many users can we make predictions for in the test set?': c,
    'How many users in the test set are we not able to make predictions for because of
    'How many articles can we make predictions for in the test set?': b,
    '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 movies are in the training data, but there are only
```

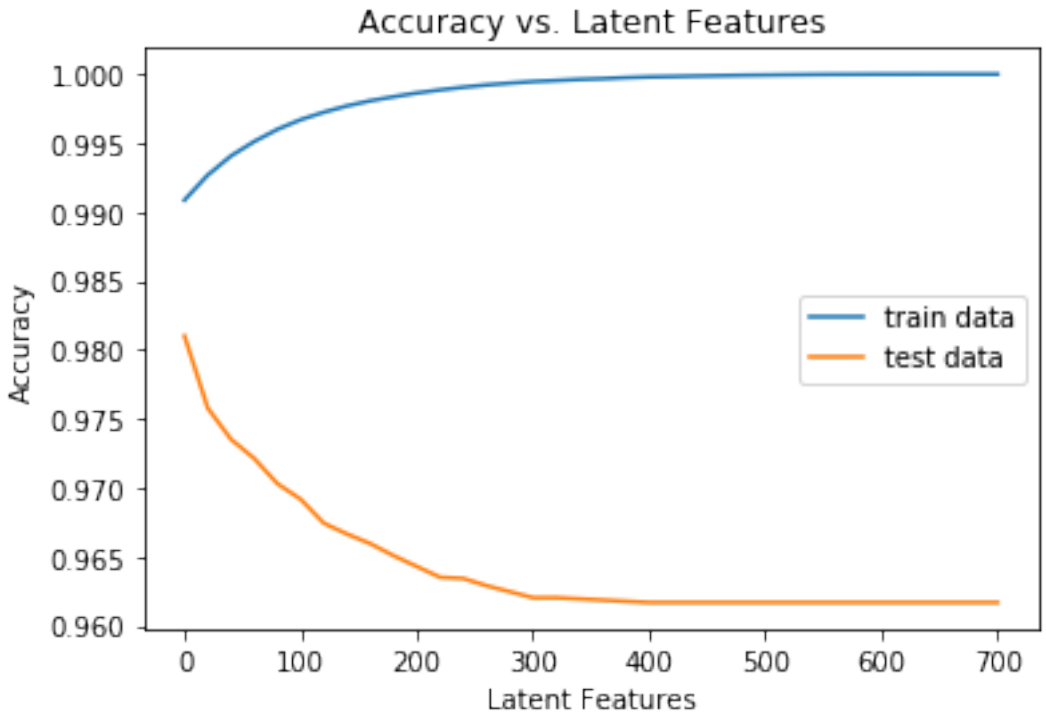
5. Now use the **user_item_train** dataset from above to find U, S, and V transpose using SVD. Then find the subset of rows in the **user_item_test** dataset that you can predict using this matrix decomposition with different numbers of latent features to see how many features makes sense to keep based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

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

```
In [43]: # fit SVD on the user_item_train matrix
         u_train, s_train, vt_train = np.linalg.svd(user_item_train) # fit svd similar to above

In [74]: u_train.shape, s_train.shape, vt_train.shape

Out[74]: ((4487, 4487), (714,), (714, 714))

In [75]: train_common_ids = user_item_train.index.isin(test_idx)

         train_common_cols = user_item_train.columns.isin(test_arts)

In [76]: u_test = u_train[train_common_ids, :]

         vt_test= vt_train[:, train_common_cols]

In [77]: train_idx = user_item_train.index

         common_ids = list(set(train_idx)&set(test_idx))

         common_cols = user_item_train.columns.intersection(test_arts)

In [78]: user_item_test = user_item_test.loc[common_ids]

In [45]: # Use these cells to see how well you can use the training
         # decomposition to predict on test data

In [79]: sum_train_errs=[]

         sum_test_errs=[]

         num_latent_feat=np.arange(0,714,20)

         for k in num_latent_feat:

             u_train_lat, s_train_lat, vt_train_lat = u_train[:, :k], np.diag(s_train[:k]), vt_t

             u_test_lat, vt_test_lat = u_test[:, :k], vt_test[:k,:]
```

```
    user_item_train_preds = np.around(np.dot(np.dot(u_train_lat, s_train_lat), vt_train

    user_item_test_preds = np.around(np.dot(np.dot(u_test_lat, s_train_lat), vt_test_la

    diffs_train = np.subtract(user_item_train, user_item_train_preds)

    diffs_test = np.subtract(user_item_test.loc[common_ids, :], user_item_test_preds)

    train_err = np.sum(np.sum(np.abs(diffs_train)))

    sum_train_errs.append(train_err)

    test_err = np.sum(np.sum(np.abs(diffs_test)))

    sum_test_errs.append(test_err)
```

In [85]:
```
plt.plot(num_latent_feat, 1 - (np.array(sum_train_errs)/(user_item_train.shape[0]*user_
plt.plot(num_latent_feat, 1 - (np.array(sum_test_errs)/(user_item_test.shape[0]*user_it
plt.legend()
plt.xlabel('Latent Features');
plt.ylabel('Accuracy');
plt.title('Accuracy vs. Latent Features');
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?

Your response here.

The higher the accuracy of our training data, the more latent features we add. However, the accuracy of the predictions in the test dataset suffers as a result of this. Overfitting a training dataset limits its ability to predict data other than its own. Furthermore, the data set is unbalanced. Only 20 of the trainset's over 4000 users are also in the test dataset. Using the technique presented, it appears that roughly 100 latent features would be a decent compromise between a good fit of the training data and not being overfitted to produce appropriate suggestions. Additional content-based recommendations could be used in another method. For example, using the information from df content, categories may be defined for each article. These categories could be merged with rank-based recommendations and used as a filter for collaborative filtering between users.

A/B testing of the model appears to be appropriate for evaluating the predictions in practise. A portion of users may receive offers, while others may continue to use the app as is. In order to correctly segregate the groups, A/B testing should be based on userIDs. Time spent using the app, following up on advice, or the total amount of money made with the individual user could all be useful metrics to track.

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

```
Out[86]: 0
```

```
In [ ]:
```