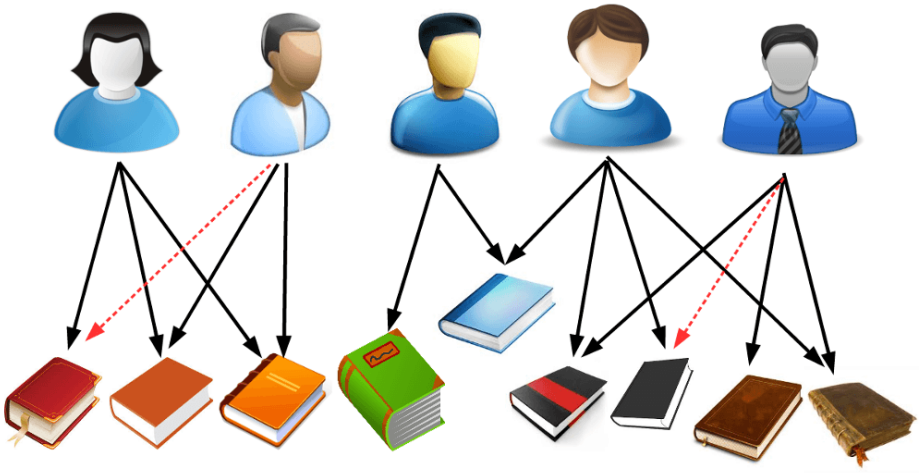
Milestone Report

"Capstone Project 2 : Book bundles recommendation from book readers ratings and book sales data"

This analysis aims to recommend book bundles to the readers from book ratings and book sales data. We all had an online experience where a website makes some personalized recommendations. Youtube tells you "...viewers also watch...", Amazon tells you "Customers Who Bought This Item Also Bought", and Udemy tells you "Students Who Viewed This Course Also Viewed". A recommender system is a simple algorithm whose aim is to provide the most relevant information to a user by discovering patterns in a dataset. The algorithm rates the items and shows the user the items that they would rate highly.



The recommendation system can be deployed as a web app where people can use to get recommendations based on reading history. Schools can use the system to increase students reading interest, adapts to student needs, moving at a slower

or faster pace to help students with different strengths and learning styles to reach their full potential. Online book stores can offer book bundles to their customers.

Data collection

The data can be collected by two means: explicitly and implicitly. I used explicit data that is provided intentionally by the users as rating.

I checked different datasets like Cai-Nicolas Ziegler' Book-Crossings dataset; Julian McAuley's Amazon product dataset, Open library book dataset, Worldcat book dataset, Goodreaders dataset. I used updated Goodreaders dataset in my project.

Before concatenating tables all datasets are diagnosed for the inconsistent column names, missing data, duplicate rows , untidy and unexpected data values. All datasets were explored with pandas methods such as .head(), .info(), and .describe(), and DataFrame attributes like .columns and .shape . Needed datasets are concatenated and a single dataframe is created and saved for later analysis.

The data consists of three tables: ratings, books info, and users info.

The books data set provides book details. It includes 10000 records and 24 fields: book id, goodreads book id, best book id, work id, books count, isbn, isbn13, authors, original publication year, original title, title, language code, average

rating, ratings count, work ratings count, work text reviews count, ratings 1, ratings 2, ratings 3, ratings 4, ratings 5, image url, and small image url.

The ratings data set provides a list of ratings that users have given to books. It includes 5976479 records and 3 fields: user id, book id, and rating. The users data dataset provides the user demographic information. It includes 912705 records and 2 fields: user id, and book id.

The datasets have missing data and duplicate rows. I filled the null ratings data with “0”. I checked sparsity level of the dataset to see the percentage of cells that are not populated and populated, respectively. Some changes may have to be made later on in the analysis process, but for now this is the first draft of the dataframe. This cleaned dataframe is saved for the further analyze.

Exploratory Data Analysis

As seen in below table over we are missing many books ISBN number. ISBN number is a unique number and we won’t be able to use it as reference and won’t be able to fill it with O or median value. Our data has negative publication years, missing data and very old dates. I investigate those books. Negative values filled with the median and left very old dates in database.

```
# See the column data types and non-missing values
books.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 23 columns):
book_id                     10000 non-null int64
goodreads_book_id           10000 non-null int64
best_book_id                10000 non-null int64
work_id                     10000 non-null int64
books_count                 10000 non-null int64
isbn                        9300 non-null object
isbn13                      9415 non-null float64
authors                     10000 non-null object
original_publication_year   9979 non-null float64
original_title              9415 non-null object
title                       10000 non-null object
language_code               8916 non-null object
average_rating              10000 non-null float64
ratings_count               10000 non-null int64
work_ratings_count          10000 non-null int64
work_text_reviews_count     10000 non-null int64
ratings_1                   10000 non-null int64
ratings_2                   10000 non-null int64
ratings_3                   10000 non-null int64
ratings_4                   10000 non-null int64
ratings_5                   10000 non-null int64
image_url                   10000 non-null object
small_image_url             10000 non-null object
dtypes: float64(3), int64(13), object(7)
memory usage: 1.8+ MB
```

In [8]:
```
#missing values in 'original_publication_year'
books.original_publication_year.unique()
```

```
array([1891., 1897., 1963., 1844., 2013., 1862., 1961., 1876.,
       1962., 1955., 1991., 1600., 1965., 1939., 1908., 1850.,
       2014., 1606., 1860., 1942., 1978., 1815., 1877., 1986.,
       1866., 1922., 1987., 1851., 1982., 1843., 1976., 1994.,
       1915., 1956.,   nan, 1980., 1865., 1817., 1957., 1926.,
       1943., 1938., 1966., 1981., 2016., 1992., 1984., 1972.,
       1882., 1895., 1899., 1983., -750., 1900., 1975., 1971.,
       1603., 1929., 1968., 1838., 1903., 1886., 1940., -500.,
       1887., 1931., 1611., 1814., 1719., 1513., 1880., 1923.,
       1869., 1849., 1892., 1904., 1726., 1598.,  975., 1905.,
       1935., 1948., 1856., 1759., 1959., 1605., 1901., 1970.,
       1902., 1390., 1852., 1909., 1920., 1934., 1872., 1894.,
       1599., -380., 1593., 1831., 1812., 1623., 1854., -430.,
       1596., 1601., 1941., 1906., 1930., 1916., 1871., 1864.,
       1320., 1667., 1927., 1874., 1918., -300., 1835., 1883.,
       -560., 1914., 1308., 1881., 1933.,  -17., 1830., 1678.,
       1848., 1826., 1912., 1853., 1890., -441., 1928., 1896.,
       1879., 1819., 1889., 1592., 1855., 1845., 1924., 1944.,
       1898.,  800., 1910.,  180., -401., -1750.,  609., 1919.,
       -762., 1842., 1774., 1808., -400., 1861., 1846.,    8.,
```

We have 10000 books and 912705 users. The sparsity level of our Book dataset is 99.9 %.

In [27]:
```
print ("number of users: " + str(n_users))
print ("number of books: " + str(n_books))
```

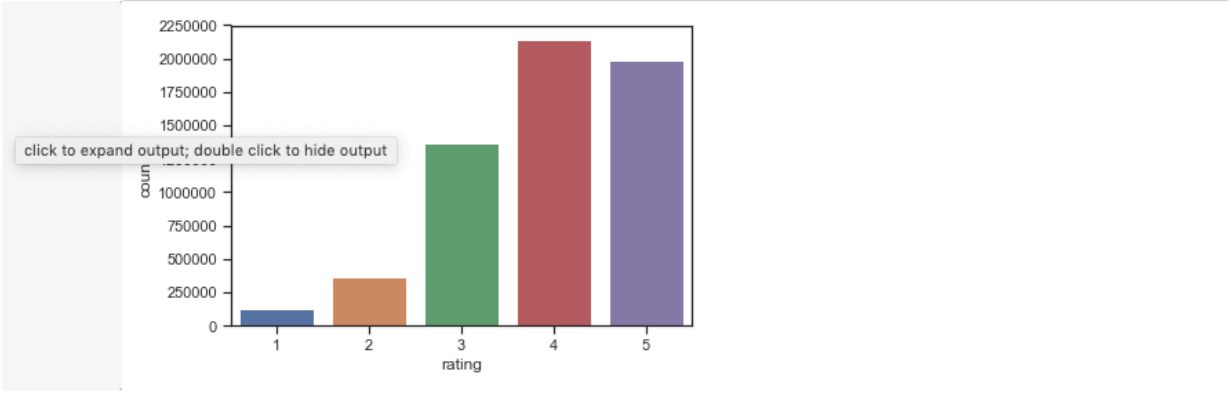
```
number of users: 912705
number of books: 10000
```

In [28]:
```
#Sparsity of dataset in %
sparsity=1.0-len(ratings_new)/float(n_users*n_books)
print ('The sparsity level of Book dataset is ' +  str(sparsity*100) + ' %')
```

```
The sparsity level of Book dataset is 99.93451905051468 %
```

Books are rated 1-5.

```
In [31]: #plotting count of rating
         sns.countplot(data=ratings_explicit , x='rating')
         plt.show()
```



I combined ratings and book datasets and drop columns that i won't use in my recommendation. New dataset has 5975161 rows and 11 columns.

```
In [33]: if not combine_book_rating[combine_book_rating.duplicated(['user_id', 'title'])].empty:
             initial_rows = combine_book_rating.shape[0]

             print('Initial dataframe shape {0}'.format(combine_book_rating.shape))
             combine_book_rating = combine_book_rating.drop_duplicates(['user_id', 'title'])
             current_rows = combine_book_rating.shape[0]
             print('New dataframe shape {0}'.format(combine_book_rating.shape))
             print('Removed {0} rows'.format(initial_rows - current_rows))
```

```
Initial dataframe shape (5976479, 11)
New dataframe shape (5975161, 11)
Removed 1318 rows
```

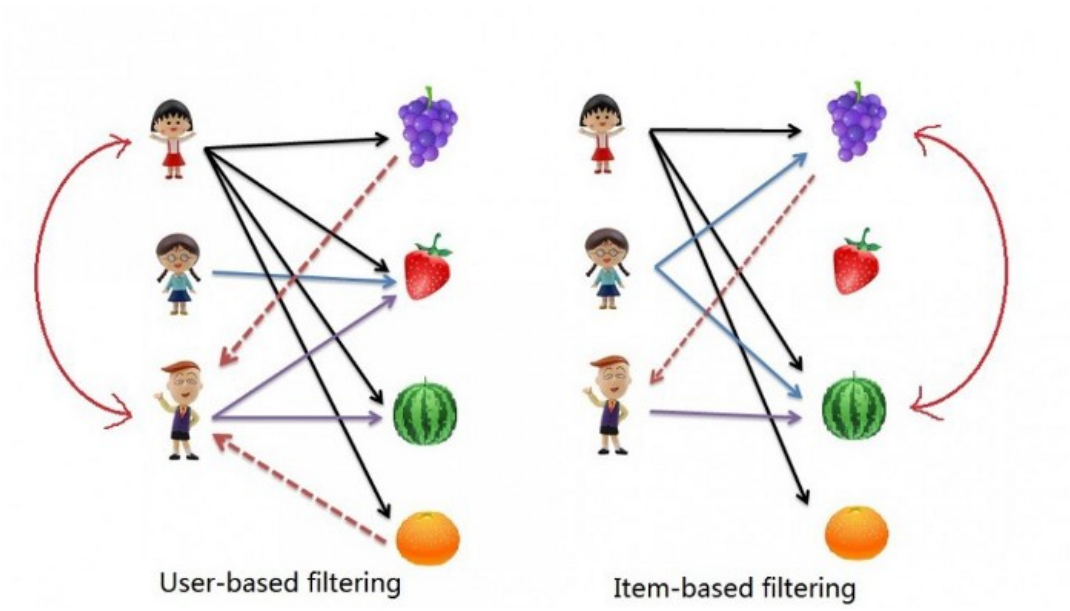
The mean of average rating is 4.02 and 1% of books have 4.54 ratings, 2% have 4.52 ratings.

```
n [35]: print(combine_book_rating['average_rating'].quantile(np.arange(.9, 1, .01)))
```

```
0.900    4.330
0.910    4.340
0.920    4.350
0.930    4.370
0.940    4.390
0.950    4.420
0.960    4.440
0.970    4.460
0.980    4.520
0.990    4.540
Name: average_rating, dtype: float64
```

Recommendation system

I used nearest neighbor model of collaborative filtering which is based on assumption that people like things similar to other things they like, and things that are liked by other people with similar taste.



As seen below the system is calculating the distance between the book and similar books. It recommend nearest 5 books.

```
Recommendations for Brain Rules: 12 Principles for Surviving and Thriving at Work, Home, and School:

1: A Whole New Mind: Why Right-Brainers Will Rule the Future, with distance of 0.8454638172875433:

2: How We Decide, with distance of 0.8603980043240806:

3: Drive: The Surprising Truth About What Motivates Us, with distance of 0.8629423447204392:

4: Talent is Overrated: What Really Separates World-Class Performers from Everybody Else, with distance of 0.86510736
96099224:

5: The Upside of Irrationality: The Unexpected Benefits of Defying Logic at Work and at Home, with distance of 0.8722
994549069018:

Perfect! This books are definitely should be recommended one after another
```

References

1. “Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering” , R. He, J. McAuley, WWW, 2016

2. “Image-based recommendations on styles and substitutes” J. McAuley, C. Targett, J. Shi, A. van den Hengel, SIGIR, 2015

3. http://jmcauley.ucsd.edu/data/amazon/

4. Book database: http://www2.informatik.uni-freiburg.de/~cziegler/BX/

5. Open library: https://openlibrary.org/dev/docs/restful_api

6. Worldcat: https://www.oclc.org/developer/develop/web-services/worldcat-search-api.en.html

7. Goodreaders dataset: https://www.goodreads.com/api

8. image source from https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f