asg2-sol

May 23, 2018

1 Introduction to Data Science - Homework 2 - Solutions

CENG 499

Adapted from University of Utah - Data Science Course

Due: Sunday, Mar 11, 11:59pm.

In this homework you will read in and analyze a movies dataset. First we'll do some basic analysis with vanilla Python, then we'll move on to doing more advanced analysis with Pandas.

1.1 Your Data

Fill out the following information:

First Name:
Last Name:
E-mail:
Student ID:

1.2 Part 1: Analyzing Data The Hard Way

In this part we'll do some manual analysis of a movies dataset.

1.2.1 Task 1.1: Read in the data

Parse the file `movies.csv` using the csv library. Lecture 6 and/or Homework 2 might be a good inspiration for this.

We recommend that you store the header in a separate array. Make sure that at least the "ratings" and the "votes" columns are cast to the appropriate data types for doing calculations with them.

Print the header, a row of the table, and the number of rows and columns.

```
In [2]: # import the csv library
        import csv

        # initialize the top-level array
        movies = []
        header = []

        cnt = 0
```

```
with open('movies.csv', newline='') as f:
    reader = csv.reader(f)
    for row in reader:    # iterate over the lines in the file
        if cnt == 0:      # first line
            header = row  # get the headers
        else:
            # append to movies list, change the following line
            movies.append(row)
        cnt += 1
        # print(len(movies))
header
```

```
Out[2]: ['',
         'title',
         'year',
         'length',
         'budget',
         'rating',
         'votes',
         'r1',
         'r2',
         'r3',
         'r4',
         'r5',
         'r6',
         'r7',
         'r8',
         'r9',
         'r10',
         'mpaa',
         'Action',
         'Animation',
         'Comedy',
         'Drama',
         'Documentary',
         'Romance',
         'Short']
```

1.2.2 Task 1.2: Calculate movie stats

In this task, you will calculate some statistics about movies. We suggest you implement your solutions for Tasks 1.2.1-1.2.3 in one code cell - you should be able to calculate this in a single iteration over the dataset.

Task 1.2.1 Calcualte average ratings Compute the average rating for the movies and print the output. Also print the number of movies. Your output could look like this:

```
Average rating: xxx.xxxx, Number of movies: xx
```

Task 1.2.2: Calculate average rating for major movies Compute the average rating for the movies that have more than 500 votes in your loaded dataset and print the output. (We'll call these movies with more than 500 votes *major movies* from now on).

Your output could look like this:

```
Average rating of movies with more than 500 votes: xxx.xxxx, Number of major movies: xx
```

Task 1.2.3: Find the highest rated major movie Find out which of the movies with more than 500 votes has the highest rating.

Your output could look like this:

```
Highest rating: xxx.xxxx, Title: MOVIE TITLE
```

Task 1.2.4: Interpret the data

- What's the size relationship of major movies to all movies?
- Are major movies usually better than the average movies?

```
In [19]: # Task 1.2.1 Calcualte average ratings
         sum = 0
         cnt = 0
         for row in movies:
             sum += float(row[5])
             cnt += 1
         print('Average rating: ' + str(round(sum/len(movies),4)) + ', Number of movies: ' + str

         # Task 1.2.2: Calculate average rating for major movies
         sum = 0
         cnt_major = 0
         for row in movies:
             try:
                 if int(row[6]) >= 500:
                     cnt_major += 1
                     sum += float(row[5])
             except ValueError:
                 continue
         print('Average rating of movies with more than 500 votes: ' + str(round(sum/cnt_major,4

         # Task 1.2.3: Find the highest rated major movie
         high_rate = 0
         high_movie = 0
         cnt = 0
         for row in movies:
             try:
                 if int(row[5]) > high_rate:
                     high_rate = int(row[5])
                     high_movie = cnt
             except ValueError:
```

```
            continue
        cnt += 1
    print('Highest rating: ' + str(high_rate) + ', Title: ')
```

```
Average rating: 5.9329, Number of movies: 58788
Average rating of movies with more than 500 votes: 6.3705, Number of major movies: 6462
Highest rating: 10, Title:
```

Your Interpretation:
About 10% of the movies are major movies.
Major movies have a slightly lower average rating than all movies.

1.3 Part 2: Pandas

In this part we will use the Pandas library for our analysis.

1.3.1 Task 2.1: Loading data

Read in the data again. This time you should create a Pandas DataFrame. Print the head of the dataset. * How many data rows did you load? How many columns? * Of which data types are the columns? * Do you have to do manual data type conversions?

```
In [22]: # Task 2.1: Loading data

         # import pansas and numpy
         import pandas as pd

         movies_pd = pd.read_csv("movies.csv")
         movies_pd.head(2)
```

```
Out[22]:    Unnamed: 0              title  year  length  budget  rating  votes   r1  \
         0           1                  $  1971     121     NaN     6.4    348  4.5
         1           2  $1000 a Touchdown  1939      71     NaN     6.0     20  0.0

              r2   r3  ...    r9   r10  mpaa  Action  Animation  Comedy  Drama  \
         0   4.5  4.5  ...   4.5   4.5   NaN       0          0       1      1
         1  14.5  4.5  ...   4.5  14.5   NaN       0          0       1      0

           Documentary  Romance  Short
         0           0        0      0
         1           0        0      0

         [2 rows x 25 columns]
```

```
In [23]: # print data types
         movies_pd.dtypes
```

```
Out[23]: Unnamed: 0       int64
         title           object
         year             int64
         length           int64
         budget         float64
         rating         float64
         votes            int64
         r1             float64
         r2             float64
         r3             float64
         r4             float64
         r5             float64
         r6             float64
         r7             float64
         r8             float64
         r9             float64
         r10            float64
         mpaa            object
         Action           int64
         Animation        int64
         Comedy           int64
         Drama            int64
         Documentary      int64
         Romance          int64
         Short            int64
         dtype: object
```

Your Interpretation: Looks line no, no need for data conversion.

1.3.2 Task 2.2: Calculate the average rating

Compute the average rating for all movies and print the output in a formatted way.

Your output could look like this:

```
Average rating: xxx.xxxx
```

```
In [24]: # Task 2.2: Calculate the average rating
         print('Average rating: ' + str("%8.4f" % movies_pd['rating'].mean()))
```

```
Average rating:   5.9329
```

1.3.3 Task 2.3: Compare the runtime

Measure the runtime of the mean calculation using Pandas and compare it to the computation time for calculating the mean using a for loop (you can copy the relevant parts from part one).

You can use time.clock() to set timestamps before and after the execution of the code you want to measure, then you simply substract end time from start time.

Print your results in a human readable way and add a metric to the output. Calculate the factor of the difference and print it. Note that the exact times and the factors will vary when you re-run this and especially between machines.

E.g.:
time using own code: xxx.xxx s
time using Pandas: xxx.xxx s
difference factor: xxx

```
In [28]: # Task 2.3: Compare the runtime

         import time

         start = time.clock()

         sum = 0
         for row in movies:
             try:
                 sum += float(row[5])
             except ValueError:
                 continue
         print('Average rating: ' + str(round(sum/len(movies),4)) + ', Number of movies: ' + str
         time_loop = time.clock() - start
         print("time using own code",time_loop,"s")

         start = time.clock()
         print('Average rating: ' + str("%8.4f" % movies_pd['rating'].mean()))
         time_pandas = time.clock() - start
         print("time using Pandas",time_pandas,"s")

         print("difference factor: ",str("%0.4f" % (time_pandas/time_loop)))
```

```
Average rating: 5.9329, Number of movies: 58788
time using own code 0.034253999999999785 s
Average rating:   5.9329
time using Pandas 0.000588000000000477 s
difference factor:  0.0172
```

1.3.4 Task 2.4: Filter out rows

The whole movies dataset has about 60k entries. Use pandas to filter your dataframe to contain only the major movies with more than 500 votes.

Count and print the number of movies with more than 500 votes.

E.g.: xxx.xxx movies have more than 500 votes.

```
In [29]: # Task 2.4: Filter out rows
         majorMovies = movies_pd[movies_pd.votes>500]
         print(len(majorMovies), "movies have more than 500 votes")
```

```
6458 movies have more than 500 votes
```

1.3.5 Task 2.5: Calculate the average rating for major movies

Compute the average rating for the major movies. Your output could look like this:

```
Average rating of movies with more than 500 votes: xxx.xxxx
```

```
In [51]: # Task 2.5: Calculate the average rating for major movies
         avg_rate_major_movies = movies_pd[movies_pd.votes > 500].rating.mean()
         print('Average rating of movies with more than 500 votes:', str("%8.4f" % avg_rate_majo
```

```
Average rating of movies with more than 500 votes:   6.3706
```

1.3.6 Task 2.6: Find the highest rated major movie

Find the highest rated major movie in the dataframe. Hint: idxmax() could be a helpful function.
 Print the title and the rating.
 Your output could look like this:

```
Highest rated movie: TITLE, rating: x.x
```

```
In [31]: # Task 2.6: Find the highest rated major movie
         maxrate = majorMovies.rating.max()
         maxMovies = majorMovies[majorMovies.rating == maxrate]
         print("Highest rated movies:", maxMovies.title.values, " rating:", maxrate)
```

```
Highest rated movies: ['Godfather, The' 'Shawshank Redemption, The']  rating: 9.1
```

1.3.7 Task 2.7: Filter out rows and count specific features

From the major movies you filtered out in a previous task, find out * How many are comedies. * How many are dramas * How many are both, comedies and dramas
 Create new dataframed for each of these subsets.
 Hint: you can combine broadcasting statements with a boolean AND: `&`.
 Print the results, e.g.:

```
xxx.xxx major movies are comedies.
xxx.xxx major movies are dramas.
xxx.xxx major movies are both, comedies and dramas.
```

```
In [107]: # Task 2.7: Filter out rows and count specific features
          movComedy = majorMovies[majorMovies.Comedy==1]
          movDrama = majorMovies[majorMovies.Drama==1]
          movBoth = majorMovies[(majorMovies.Comedy==1) & (majorMovies.Drama==1)]
          print("%6d" % len(movComedy),"major movies are comedies.")
          print("%6d" % len(movDrama),"major movies are dramas.")
          print("%6d" % len(movBoth),"major movies are both, comedies and dramas.")
```

```
2553 major movies are comedies.
3370 major movies are dramas.
 814 major movies are both, comedies and dramas.
```

1.3.8 Task 2.8: Compare ratings of different categories

Now we want to compare the ratings for comedies and dramas for major movies.

Compute the average rating, maximum rating, minimum rating, standard deviation, and the median for each category. Hint: there is a function that does all of this in one line.

What do these numbers tell us? Provide an interpretation.

```
In [108]: # Task 2.8: Compare ratings of different categories
          print('Comedy Ratings:')
          print("Average\t", "%8.4f" % movComedy.rating.mean())
          print("Min\t", "%8.4f" % movComedy.rating.min())
          print("Max\t", "%8.4f" % movComedy.rating.max())
          print("Std.Dev\t", "%8.4f" % movComedy.rating.std())
          print("Median\t", "%8.4f" % movComedy.rating.median())
          print()
          print('Drama Ratings:')
          print("Average\t", "%8.4f" % movDrama.rating.mean())
          print("Min\t", "%8.4f" % movDrama.rating.min())
          print("Max\t", "%8.4f" % movDrama.rating.max())
          print("Std.Dev\t", "%8.4f" % movDrama.rating.std())
          print("Median\t", "%8.4f" % movDrama.rating.median())
```

```
Comedy Ratings:
Average            6.1214
Min            1.7000
Max            8.7000
Std.Dev            1.1961
Median           6.2000

Drama Ratings:
Average            6.7496
Min            1.3000
Max            9.1000
Std.Dev            0.9910
Median           6.9000
```

Your Interpretation: Drama rating variation is smaller than comedy rating variation (see plots at the end)

1.3.9 Task 2.9: Movies per year

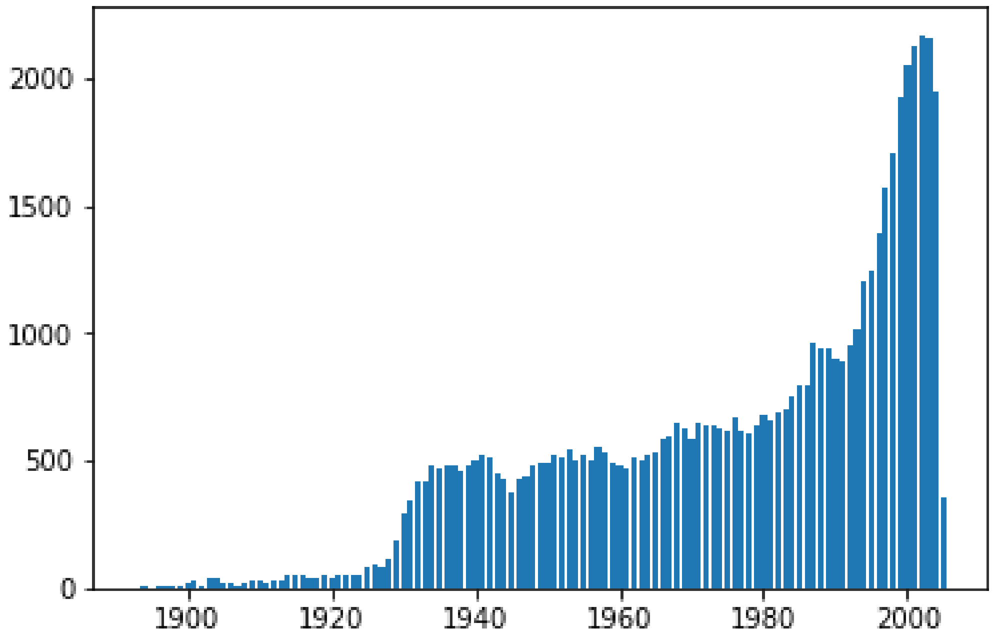
Calcluate how many major movies were made in each year.

Print the number like this, sorted by year:

```
year   number of movies
1902    xxx
1903    xxx
...     ...
```

Use this data to render a line chart of the number of movies per year.

```
In [109]: # Task 2.9: Movies per year
          mYear = movies_pd.groupby('year')['title'].count().reset_index(name='number of movies'
          mYear.head(2)

Out[109]:    year  number of movies
          0  1893                 1
          1  1894                 9

In [110]: %matplotlib inline
          # create the plot here
          import matplotlib.pyplot as plt
          year = mYear['year']
          nummov = mYear['number of movies']
          plt.bar(year,nummov)

Out[110]: <Container object of 113 artists>
```

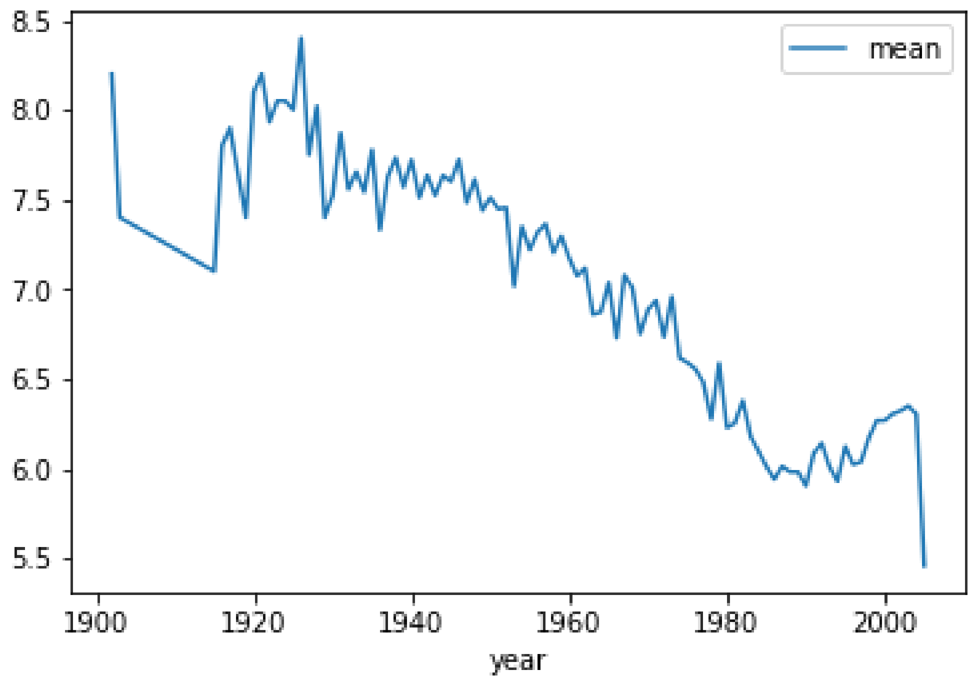


1.3.10 Task 2.10: Yearly average

Compute the average rating per year for all major movies.
Use the numbers you computed to plot a line chart. Plot the year on the x-axis and the average rating on th y axis.

```
In [111]: # Task 2.10: Yearly average
          avg_year=majorMovies.groupby('year')['rating'].agg(['mean'])
          avg_year.plot()
```

```
Out[111]: <matplotlib.axes._subplots.AxesSubplot at 0x12f863ba8>
```



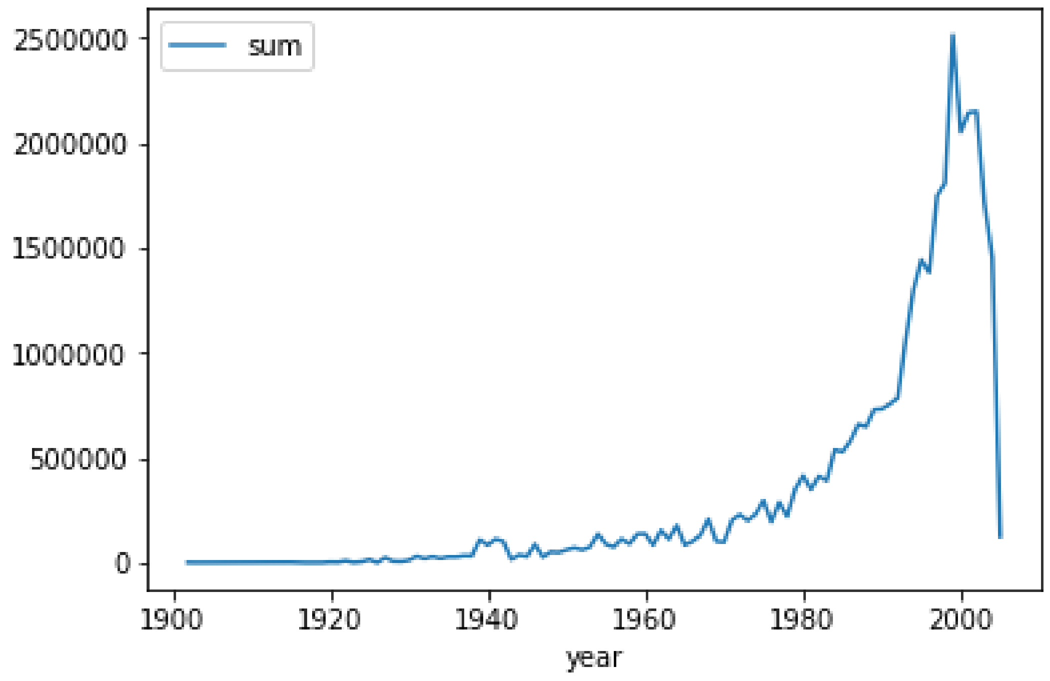
1.3.11 Task 2.11: Explore and Interpret

Are old movies better? How could you explain this? Are there differences between the rating of major movies and all movies over time? Continue to explore and use plots to inform your answer. Interesting measures to consider are the total number of votes per year, the average number of votes for a movie in a particular year, etc.

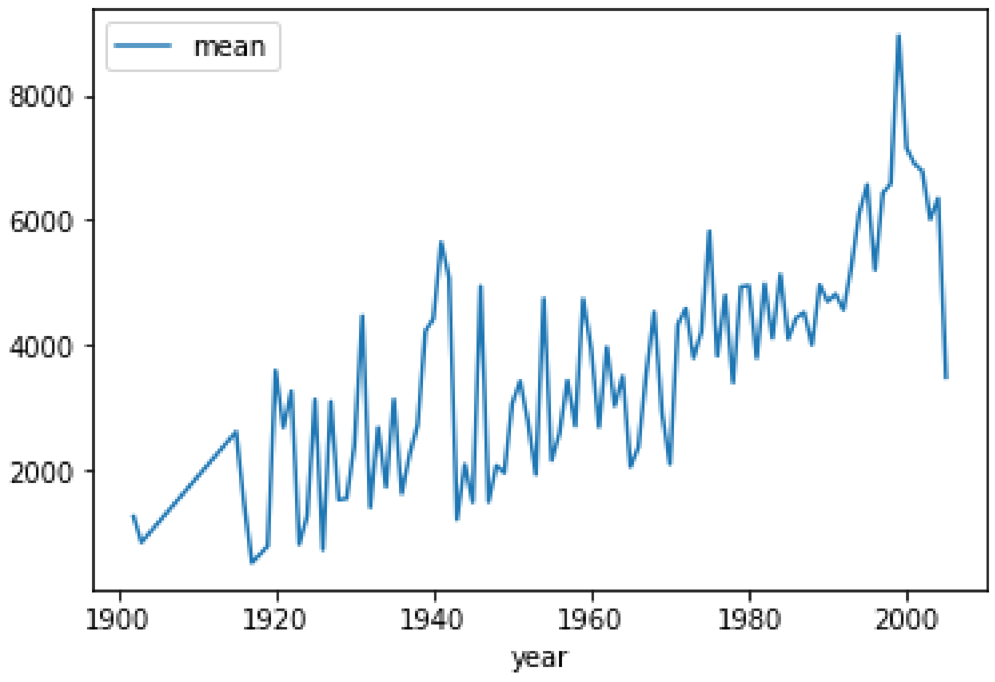
TODO: your code and your interpretation

```
In [112]: # total number of votes for movies by year
          avg_year=majorMovies.groupby('year')['votes'].agg(['sum'])
          avg_year.plot()
```

```
Out[112]: <matplotlib.axes._subplots.AxesSubplot at 0x12f597f98>
```



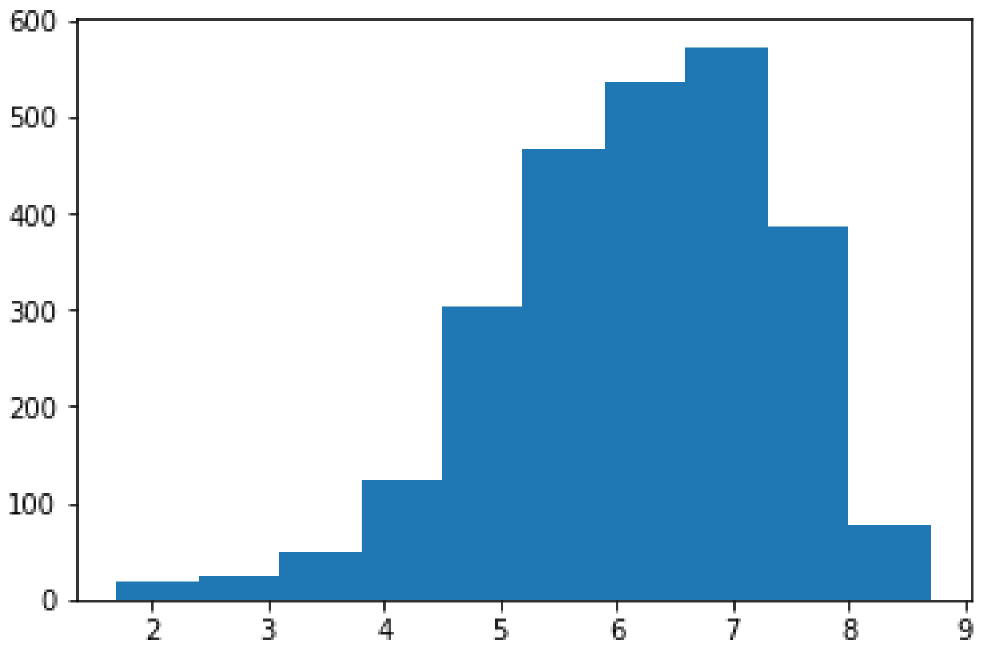
```
In [113]: # average number of votes for movies by years
          avg_year=majorMovies.groupby('year')['votes'].agg(['mean'])
          avg_year.plot()

Out[113]: <matplotlib.axes._subplots.AxesSubplot at 0x12fc84710>
```



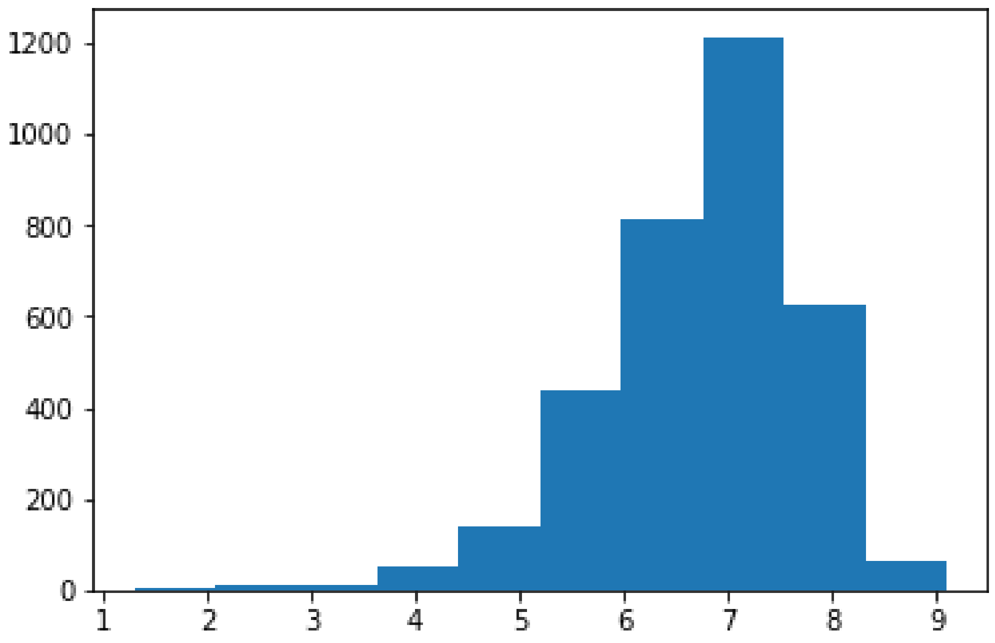
1.3.12 Note: Looks like the average number of votes are increasing by years, and the average rating of the movies are decreasing.

```
In [114]: import matplotlib.pyplot as plt
```

```
In [115]: n, bins, patches = plt.hist(movComedy['rating'])
          plt.show()
```



```
In [116]: n, bins, patches = plt.hist(movDrama['rating'])
          plt.show()
```


1200
1000
800
600
400
200
0
1
2
3
4
5
6
7
8
9