# Project 2

Aozhu Chen 004773895
Ruoxi Zhang 404753334
Ning Xin 704775693

In this project, we will create and explore a network from the IMDb movie data. We are going to study the properties of actor/actress network and movies. All datasets are available online.

## Question 1:

**Download & pre-processing the data**

By loading the IMDb data, we find that the data size is large. First, we need to merge those 2 lists into one file. We do this in python by reading read line by line. The data is also filtered by following constrains:

- The actors or actresses must involve in more than or equal 10 movies.
- The names of actors, actresses and movies only contain letters, numbers and valid white space

Overall, parsing the data is not easy, we need to consider a lot of different exceptions. Such as the redundant information and irregular separators.

Total number of actors & actresses with > 10 movies = 113115

Total number of movies = 716002

## Question 2:

We will use hash maps to help us simplify the process to construct a weighted directed graph G(V, E) from the list. More specifically, we combine each actor or actress with corresponding movies. And for each movie we summary the actors and actress involved. Then iterate through each pair of actors and actress who involves in the same movie and calculate the weight as described in following: Again, we only consider the actors and actress involved in more than or equal to 10 movies.

**$V$ = all actors/actressess in list**

**$S_i = \{ m | i \in V, m$ is a movie in which $i$ has acted$\}$**

**$E = \{(i,j) | i,j \in V, S_i \cap S_j \neq \emptyset\}$ and for each directed Edge $i \to j$, a weight is assigned as $\frac{|S_i \cap S_j|}{|S_i|}$.**

**Weighted Directed Graph:**

Number of edges = 23650208

Number of vertices = 113115

## Question 3:

By running the pagerank algorithm on the actor and actress network, we can get the result of those who are among top 10.

| Name of actor/actress | Pagerank scores | Credit |
|---|---|---|
| Robert Eric I | 1.303e-04 | 462 |
| Trejo Danny | 1.108e-04 | 333 |
| Jeremy Ron | 1.095e-04 | 1444 |
| Flowers Bess | 1.028e-04 | 892 |

| | | |
|---|---|---|
| Riehle Richard | 1.003e-04 | 363 |
| David Keith I | 9.266e-05 | 116 |
| Harris Sam II | 9.175e-05 | 701 |
| Kaufman Lloyd | 9.035e-05 | 320 |
| Madsen Michael I | 8.817e-05 | 275 |
| Jackson Samuel L | 8.766.e-05 | 176 |

Frankly speaking, I do not know their name. The reason I think it should be different generation and culture. Most of them are much older than my generation, thus, I even did not see any movies related to those people. However, I spend some time on searching their movie information. I find that most of them have several masters works and that movies are really popular at their generation. I guess this can be the reason why they have such a high PageRank score.
Then I list the top 10 famous movie celebrities based on our opinion (which have most number of movies)

| Name of actor/actress | Pagerank scores | Credit |
|---|---|---|
| Depp Johnny | 5.539471e-05 | 83 |
| Schwarzenegger Arnold | 4.375223e-05 | 60 |
| Carrey Jim | 3.378052e-05 | 58 |
| Streep Meryl | 4.527094e-05 | 80 |
| Radcliffe Daniel | 1.522899e-05 | 33 |
| DiCaprio Leonardo | 3.458099e-05 | 39 |
| Cruise Tom | 4.175141e-05 | 46 |
| Pitt Brad | 4.489189e-05 | 78 |
| Chaplin Charles | 3.207075e-05 | 88 |
| Hanks Tom | 5.166254e-05 | 52 |

Most of the well-known actors or actresses do not show up in the high page rank list. As we can see from the table, the number of credits different a lot. This means that the actors or actresses familiar to us are involved in way less movies than the older actors and actress. However, what is also surprise to me is that the number of the movies participated is not necessary guarantee that the page rank score is higher. However, generally the more movie the actor or actress involved, the more likely the page rank score is higher.

In conclusion, we think the IMDb page rank scores not completely related to the fame of actors or actresses. Most of the page rank scores related to people who participate in many movies and still active in some newly released movies. In addition, the person who has more relation among other famous actors or actresses has higher page rank score.

## Question 4

In this question, we are asked to create a movie relationship network where the edges are represented by the number of jaccard-index, which is related to the number of overlap between each pair of movies. If there's no overlap between movies, then there's no edge. Below is the approach we used to address this problem. The data preparation process is done in python and network creation is done in R Studio.

**Data cleaning and preparation**

The first step is to prepare the data and generate edge list file for network generating process.

The algorithm we use is based on the first question, with tiny modifications. In question one, we put out attention on the actors, here we mainly deal with the movies. The files needed are still actor-movie dataset and actress movie dataset.

Firstly, we merge these 2 files into a dictionary named movie_to_actor where movie name is the key and the value is the actor list in this movie. In the selecting process, because the original file maps people to the movies they attend, we remove some “dirty” data where the name is partly lost or with some strange symbols which should not be in a normal name. Then there are **642587** movies selected.

Secondly, considering the memory limit in our laptops, we changed the original actor number limited in the spec and made it 10 instead of 5, which means we deleted movies with less than 10 actors in it. After this process, the number of left movies dropped to **216183**.

Thirdly, we create another dictionary which maps actor to his/her movies. This dictionary contains **1858417** records.

Finally, it comes to the calculation of jaccard-index and output the edge list file. Here we created a new dictionary where the keys are sorted possible movie-pair string with tab as separator and the values are the jaccard-index of this pair of movies. The jaccard-index is calculated by:

$$jaccard-indexofmoivepair = \frac{numberofmutualactors \in twomovies}{actors \in A + actor \in B - mutualactors}$$

What’s needed to mention here is that we sort the movie pairs lexicologically to reduce duplicated calculation and output for the same movie pair, in this case single edge will only be represented once, which is enough for an undirected graph, otherwise there should be redundant edges like A-B and B-A. The final statistical data is list below:

| | |
|---|---|
| # of movies | 642587 |
| # of movies with less than 10 actors | 216183 |
| # of actor – movie relations | 1858417 |
| # of movie pair edges | 53825016 |

Table 4.1 Statistical data in data cleaning process (10-actor version)

To reduce the calculation time, we also make another version with “more than 15 actors” condition. Below is the statistical data.

| | |
|---|---|
| # of movies | 642587 |
| # of movies with less than 10 actors | 140829 |
| # of actor – movie relations | 1614445 |
| # of movie pair edges | 37933082 |

Table 4.2 Statistical data in data cleaning process (15-actor version)

**Network Creation Process**

The network creation is done in R Studio. Considering the result should be a large network, so we use “fread” method to efficiently read the edge list file. Then the movie names are used as nodes, the jaccard-index between them is used as weight of corresponding edge.

For the whole process in question 4, you can refer to our source code for further details.

## Question 5:

In this question, we run fast greedy community finding algorithm on the network we already constructed (15-actor version), then we tag each community with the genre that happens 20% or more in this community, if there are multiple genres having more than 20%, we select the highest one, if their shares are the same, we select the first one.

After we run the community finding algorithm, we found 170 communities, and the modularity is 0.7632237. Below is the size of each community:

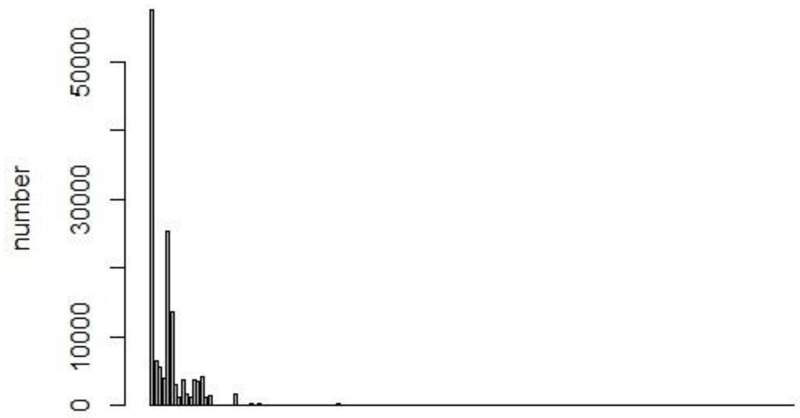
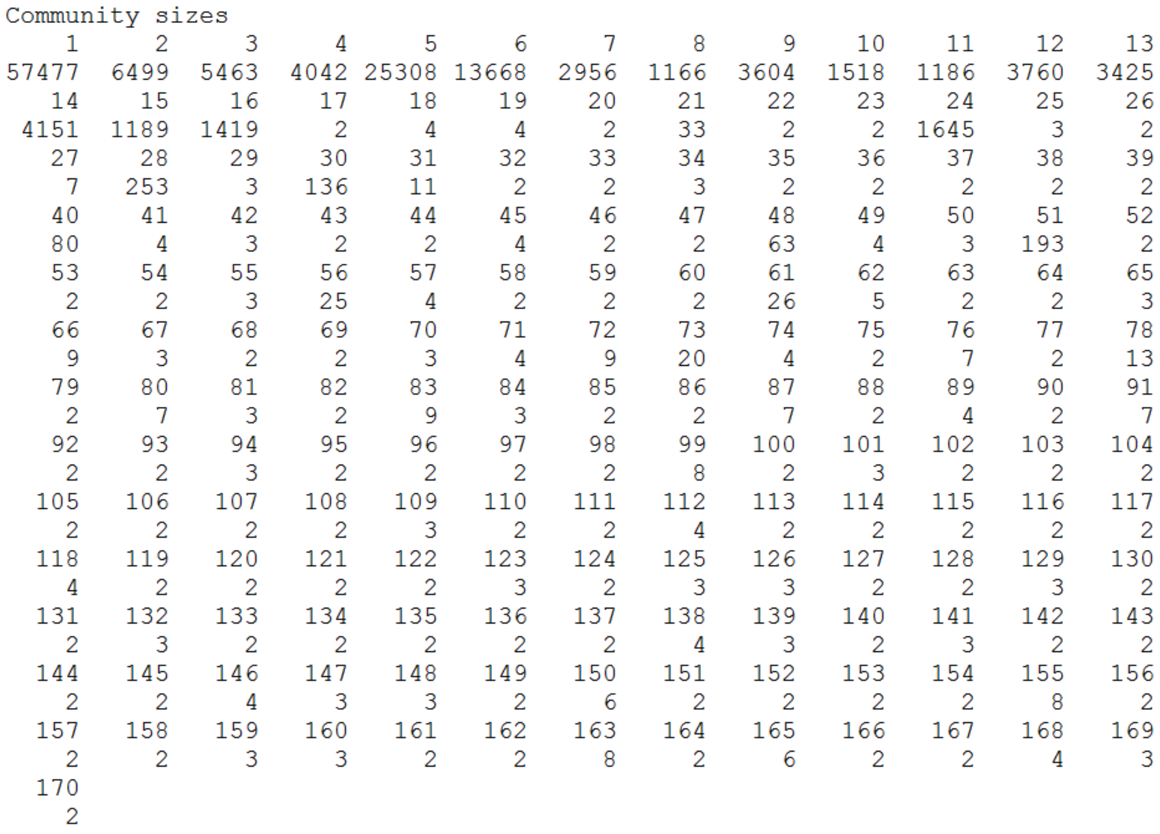


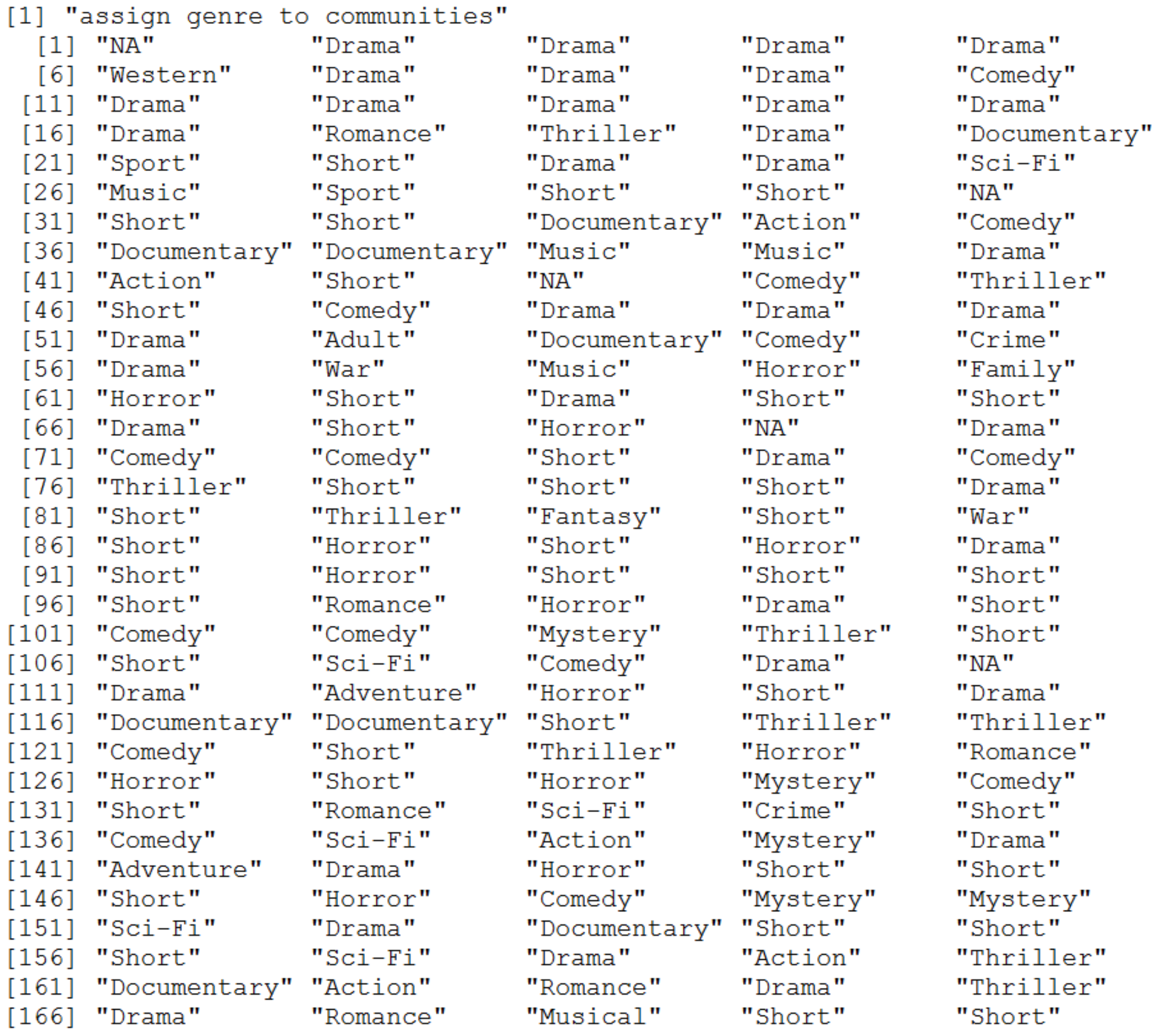
Figure 4.1 Bar plot of community sizes



```
Community sizes
    1     2     3     4     5     6     7     8     9    10    11    12    13 
57477  6499  5463  4042 25308 13668  2956  1166  3604  1518  1186  3760  3425 
   14    15    16    17    18    19    20    21    22    23    24    25    26 
 4151  1189  1419     2     4     4     2    33     2     2  1645     3     2 
   27    28    29    30    31    32    33    34    35    36    37    38    39 
    7   253     3   136    11     2     2     3     2     2     2     2     2 
   40    41    42    43    44    45    46    47    48    49    50    51    52 
   80     4     3     2     2     4     2     2    63     4     3   193     2 
   53    54    55    56    57    58    59    60    61    62    63    64    65 
    2     2     3    25     4     2     2     2    26     5     2     2     3 
   66    67    68    69    70    71    72    73    74    75    76    77    78 
    9     3     2     2     3     4     9    20     4     2     7     2    13 
   79    80    81    82    83    84    85    86    87    88    89    90    91 
    2     7     3     2     9     3     2     2     7     2     4     2     7 
   92    93    94    95    96    97    98    99   100   101   102   103   104 
    2     2     3     2     2     2     2     8     2     3     2     2     2 
  105   106   107   108   109   110   111   112   113   114   115   116   117 
    2     2     2     2     3     2     2     4     2     2     2     2     2 
  118   119   120   121   122   123   124   125   126   127   128   129   130 
    4     2     2     2     2     3     2     3     3     2     2     3     2 
  131   132   133   134   135   136   137   138   139   140   141   142   143 
    2     3     2     2     2     2     2     4     3     2     3     2     2 
  144   145   146   147   148   149   150   151   152   153   154   155   156 
    2     2     4     3     3     2     6     2     2     2     2     8     2 
  157   158   159   160   161   162   163   164   165   166   167   168   169 
    2     2     3     3     2     2     8     2     6     2     2     4     3 
  170 
    2 
```

Figure 4.1 Community sizes – table

After we found the community structure, we tag each community with its label, the label result are in the below figure.



```
[1] "assign genre to communities"
  [1] "NA"          "Drama"       "Drama"       "Drama"       "Drama"      
  [6] "Western"     "Drama"       "Drama"       "Drama"       "Comedy"     
 [11] "Drama"       "Drama"       "Drama"       "Drama"       "Drama"      
 [16] "Drama"       "Romance"     "Thriller"    "Drama"       "Documentary"
 [21] "Sport"       "Short"       "Drama"       "Drama"       "Sci-Fi"     
 [26] "Music"       "Sport"       "Short"       "Short"       "NA"         
 [31] "Short"       "Short"       "Documentary" "Action"      "Comedy"     
 [36] "Documentary" "Documentary" "Music"       "Music"       "Drama"      
 [41] "Action"      "Short"       "NA"          "Comedy"      "Thriller"   
 [46] "Short"       "Comedy"      "Drama"       "Drama"       "Drama"      
 [51] "Drama"       "Adult"       "Documentary" "Comedy"      "Crime"      
 [56] "Drama"       "War"         "Music"       "Horror"      "Family"     
 [61] "Horror"      "Short"       "Drama"       "Short"       "Short"      
 [66] "Drama"       "Short"       "Horror"      "NA"          "Drama"      
 [71] "Comedy"      "Comedy"      "Short"       "Drama"       "Comedy"     
 [76] "Thriller"    "Short"       "Short"       "Short"       "Drama"      
 [81] "Short"       "Thriller"    "Fantasy"     "Short"       "War"        
 [86] "Short"       "Horror"      "Short"       "Horror"      "Drama"      
 [91] "Short"       "Horror"      "Short"       "Short"       "Short"      
 [96] "Short"       "Romance"     "Horror"      "Drama"       "Short"      
[101] "Comedy"      "Comedy"      "Mystery"     "Thriller"    "Short"      
[106] "Short"       "Sci-Fi"      "Comedy"      "Drama"       "NA"         
[111] "Drama"       "Adventure"   "Horror"      "Short"       "Drama"      
[116] "Documentary" "Documentary" "Short"       "Thriller"    "Thriller"   
[121] "Comedy"      "Short"       "Thriller"    "Horror"      "Romance"    
[126] "Horror"      "Short"       "Horror"      "Mystery"     "Comedy"     
[131] "Short"       "Romance"     "Sci-Fi"      "Crime"       "Short"      
[136] "Comedy"      "Sci-Fi"      "Action"      "Mystery"     "Drama"      
[141] "Adventure"   "Drama"       "Horror"      "Short"       "Short"      
[146] "Short"       "Horror"      "Comedy"      "Mystery"     "Mystery"    
[151] "Sci-Fi"      "Drama"       "Documentary" "Short"       "Short"      
[156] "Short"       "Sci-Fi"      "Drama"       "Action"      "Thriller"   
[161] "Documentary" "Action"      "Romance"     "Drama"       "Thriller"   
[166] "Drama"       "Romance"     "Musical"     "Short"       "Short"      
```

Figure 4.2 Genre tag of each community

**Analysis:**

Now that we have the community structure and its tags, we need to analyze the logic behind our result and reason whether we can trust these results, and furthermore, whether we can use this model to predict other instances. We think that we can take the result as a reference but we should be careful and not fully trust it. Remember when we construct the network, the edge weight is the jaccard index, which represent the actor overlap between each pair of movie. The reason why we can use it as a reference is that in many cases, an actor is more good at certain genre of movie, especially when there are a large portion of overlap (high jaccard index). In the other hand, in essence, there is no direct logical relation between the overlap of actor portion and the movie genre, there may be some coincidence, so we should not rely too much on this model.

## Question 6:

In this question, we analyzed the network we already constructed and find the neighbors for given 3 movies. Below is the approach we used and the result.

We are asked to find the top 5 nearest neighbors for 3 movies:

- *Batman v Superman: Dawn of Justice (2016)*
- *Mission: Impossible  Rogue Nation (2015)*
- *Minions (2015)*

Because the edges between vertices are weighted, so the distance are represented by the edges weight. What's needed mentioning is that, due to the memory limit of our laptop, we cannot support the computation for the graph of "movie with more than 10 actors", so we had to make some modification. In this question, the network we used consists of movie with more than 15 actors.

The approach to this question is relatively intuitive. I'll use Batman v Superman (BvS) as example to briefly demonstrate our method. First, we find the node "BvS" in the movie network, then we call the function in igraph package to find all the edges connecting to BvS vertices, then find the 5 smallest weight in this group, these are the nearest vertices of BvS. Then use the community structure we already constructed to find which community these vertices belong to and corresponding genre of them.

Below are the results we got for these 3 movies:

- *Batman v Superman: Dawn of Justice (2016)*

| **Neighbors** | **Genre of Community** |
|---|---|
| Eloise (2015) | Thriller |
| Into the Strom (2014) | Thriller |
| The End of the Tour (2015) | Drama |
| Grain (2015) | Drama |
| Man of Steel (2013) | Sci-Fi |

- *Mission: Impossible  Rogue Nation (2015)*

| **Neighbors** | **Genre of Community** |
|---|---|
| Fan (2015) | Short |
| Phantom (2015) | Thriller |
| The Program (2015/II) | Sport |
| Breaking the Bank (2014) | Comedy |
| Legend (2015/I) | Thriller |

- *Minions (2015)*

| **Neighbors** | **Genre of Community** |
|---|---|
| The Lorax (2012) | Fantasy |
| Inside Out (2015) | RealityTV |
| Despicable Me 2 (2013) | Fantasy |

| | |
|---|---|
| Up (2009) | Family |
| Surf’s Up (2007) | Short |

**Observation and Conclusions:**

Based on our result, we found that the genres of community which neighbors belongs to are not very consistent, for example, for Mission Impossible, its nearest 5 neighbors belong to 5 different genres. Therefore, we can say that the genre of community to which neighbors belongs are is not very instructive. The reason is that the weight of each edge basically represents how much overlap between 2 movies’ actors. So, the fact that two movies are closed to each other just means their actors overlap is the largest, from which we cannot conclude that these 2 movies necessarily have the same genre. Conversely, they can be totally different genre.

Besides, we also noticed that the year differences between target movie and its neighbors are never more than 8 years, which can also be explained by the essence of edge weight. They are closed in the network because they share a large portion of actors, and these actors are normally active in the same period. Conversely, it’s impossible for a movie made in 1960s being close to another movie made in 2010s in the network even if they should be categorized in the same genre.

Thus, for a new movie, we can have a brief overview of what a genre this movie might belong to base on the genres of its nearest neighbors in the network. But we may need further information to determine it.

## Question 7:

In this question, we want to find a proper way to use the rating of neighbors to properly predict the un-rated movie. In this question, we used the movie network we already constructed and the movie rating file.

Considering we will use fread() function in R to increase the efficiency, so we first change original file into single tab version, where movie and its rating are separated by a single tab rather than 2.

We are still going to use the 3 movie we used in question 6, Batman v Superman: Dawn of Justice (2016), Mission: Impossible  Rogue Nation (2015), and Minions (2015). Below is the algorithm we use for this question:

a. Like what we did in question 6, we extract all the neighbors of the target movie from the network.

b. Sort edges using the weights to descending order.

c. Select the first 15 movies with highest weight from the start of list, and locate these movies in the rating list.

d. Compute the weighted average ranting of these 15 movies, and use this weighted average as the predicted rating of target movie.

Below is the result of our algorithm for the three above mentioned movies.

| ***Movie Name*** | ***Predicted Rating*** | ***Real Rating*** |
|---|---|---|
| Batman v Superman: Dawn of Justice (2016) | 6.4 | 7.1 |
| Mission: Impossible Rogue Nation (2015) | 6.9 | 7.5 |
| Minions (2015) | 6.2 | 6.4 |

From the table we can see that the predicted rating are all a little bit lower than the real rating, however, the differences is within our tolerance. Basically, it means that if I only know the predicted rating, I can safely use it to judge whether I should take some time to watch this movie. If we dig a little bit deeper, we can see that this kind of approach is reasonable because the large weight means there is a large overlap of actors between the reference movie and the target movie. Thus, if the reference movie has a high score, we may also think the target movie will also have a good quality. This conclusion, from another perspective, proved that a common sense that some actors are the guarantee of the quality of movies. Besides, this model could be better if we take into director and producer into consideration.

## Question 8:

The following two features are used to make the training and prediction

1. **Top 5 page ranks of the actors or actresses in the movie:** We used page ranks calculated by the part 3 and pick the top 5 scores among the actors or actresses involved in the movie. Again, the movie and acting people pair list is filtered by removing:
   - Movies contains less than 10 actors or actresses
   - Actor or actress participates in less than 10 movies

   Also, the name of movies and acting people is cleaned by removing noncharacter, nondigital and nonwhite space chars.
2. **101 Boolean values of top 100 directors:** We rank the movies in the movie_rating.txt in descending order. Picked the first 100 directors' names involved in the high score movies. The value is store as a binary list. Each position or bit represent the director in top 100 director whether involved in the movie or not. Which means the movie contains more top 100 director, the more 1 will shows up in corresponding position

**Model analysis and result:**

We combine above features together as a list and convert to a data frame. Also, convert the corresponding real rating as a data frame. The features and labels are feed to a linear regression model. The result for the 3 movies is as below:

| | Batman v Superman: Dawn of Justice (2016) | Mission: Impossible - Rogue Nation (2015) | Minions (2015) |
|---|---|---|---|
| Actual | 7.1 | 7.5 | 6.4 |
| Prediction | 5.95 | 5.94 | 6.33 |
| RMSE | 1.1173 | | |

From the table we can see that the performance of the regression model is not very good. The calculated RMSE is 1.1172 which is very large.

## Question 9:

In this part, we create a bipartite graph consists of two subsets: The acting people and the movies. The edges are created between the actor or actress and each movie they participated. In each individual subset, there are no connections, which means there are no edge between the movies and there are no edges between actors or actress. We use Graph.TupleList in python to create the bipartite graph since we have stored all the edges in the tuples. The data we used to make the prediction is filtered as before. Removing all the actor or actress participated in less than 10 movies. Also, the data is cleaned by removing all non-characters, non-digits and non-white spaces. We tried two method to make the prediction.

Then, we find all the actors connected to the movie. And assign the movie rate based on the value of the actress, we used two method to make the prediction:

1. Average all the actors and actresses evolved
2. Average only top 10 rating actors and actresses evolved

. Also, we used two method to calculate the rate of each actor or actress.

1. Average all the rating of the movies that actor or actress participated
2. Average only top 10 rating of the movies that actor or actress participated

So, there are total 4 different rating strategies. The result is as following

Method 1: Average all the actors and actresses evolved & Average all the rating of the movies that actor or actress participated

Method 2: Average all the actors and actresses evolved & Average only top 10 rating of the movies that actor or actress participated

Method 3: Average only top 10 actors and actresses evolved & Average all the rating of the movies that actor or actress participated

Method 4: Average all the actors and actresses evolved & Average only top 10 rating of the movies that actor or actress participated

| | Batman v Superman: Dawn of Justice (2016) | Mission: Impossible - Rogue Nation (2015) | Minions (2015) | RMSE |
|---|---|---|---|---|
| Actual | 7.1 | 7.5 | 6.4 | |
| Method 1 | 6.32211 | 6.24451 | 6.41437 | 0.8527 |
| Method 2 | 6.65006 | 6.50779 | 7.50667 | 0.8966 |
| Method 3 | 7.31236 | 7.51667 | 6.41437 | 0.1232 |
| Method 4 | 7.86733 | 7.76200 | 7.50667 | 0.7921 |

From the table, we can see that the performance of method 3 is very good. The RMSE is only 0.1232. Also, compare to the regression model the performance in this part is greatly increased.