EE232 Project2

Ziyun Ye
Jiahui Li
Yang Guo

June 8, 2016

1 Introduction

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. Firstly, a weighted directed graph of actors/actresses is created to explore the PageRank algorithm. Then we use Jaccard Index to construct a movie network and try to predict the rating of three movies with different methods.

2 Prob 1: Actors With More Than Five Movies

After obtaining two files which include the movies each actor/actress act in, we can merge them into one file and parse each line using the format as actor name followed by movie names, which is separated by two backslashes. When we parse the data, we find that some movie names are followed by words like "(voice)" or "(uncredited)". In order to avoid ambiguity, we remove these words and treat "Minions (2015)" and "Minions (2015) (voice)" as the same movie. Besides, the actors/actresses with less than 5 movies are removed from the list.

3 Prob 2: A Weighted Directed Network of Actors

We can create a weighted directed graph G(V,E) after merging the data and parsing it. The vertexes are all actors/actresses in the combined file and the weight of the directed edge i→j is assigned the number of movies that both i and j act in divided by the number of movies that i acts in. In order to decrease the complexity and running time of our algorithm, we calculate the weight of edge i→j and edge j→i at the same time, i.e. for each actor we only iterate through the actors with id larger than itself and calculate the weight of edges between them. The number of iterations is reduced from N^2 to $N(N-1)/2$. Then the weighted directed graph is successfully constructed.

4 Prob 3: Top 10 Pagerank Actors

The PageRank algorithm is used by Google to measure the importance of websites. It is assumed that more important websites are likely to receive more links from other websites. The outputs of PageRank algorithm is a probability distribution representing the likelihood that a person randomly clicking on links will arrive at any particular page. PageRank is initialized to the same value for all pages - the probability distribution between 0 and 1 which equals to 1 divided by the number of pages on the web. The PageRank transferred from a given page to the targets of its outbound links upon the next iteration is divided equally among all outbound links. For example, if a page has links to 4 other pages, in each iteration this page will transfer 1/4 of its existing values to these 4 pages. Thus each time we multiply the current PageRank matrix to the transition matrix. After the finite number of iterations, the PageRank of each page is converged to a fixed value. However, if a web page links to nobody else but itself, eventually all PageRank will be transferred to it and its PageRank will be 1, which makes this algorithm meaningless. To avoid this problem, we use the damping factor which means the probability that the person will continue at any step. The damping factor is generally set to be 0.85. Thus with probability 0.15 the person will stop at each step, which prevents it from walking to the page that links to itself eventually. Consequently, PageRank is calculated as

$$PR(A) = (1-d) * \frac{1}{n} + d * \sum_{p_j \in M(p_j)} \frac{PR(p_j)}{L(p_j)}$$

where d is the damping factor. In conclusion, generally speaking, the more other pages link to a web page and this page links to less pages, the higher its PageRank will be.

In this part, we first ran PageRank algorithm in our actor/actress network created before. In our network, if an actor acts in more movies (the denominator of weight), the weight of edges from it to other actors will be lower and its PageRank tends to be higher. We get the top 10 actors with highest page rank scores shown below. However, we don't know their name. The reason is that these actors/actresses were mostly born in 1920s or 1930s. Although they acted in plenty of movies, we are not familiar with them. After all, we are more familiar with young and active actors/actresses.

Top10	Name	Scores	Num. of movies	Year of Birth
1	Flowers, Bess	7.169053e-05	878	1898
2	Jeremy, Ron	6.960280e-05	1423	1953
3	Harris, Sam (II)	6.761147e-05	681	1877
4	Blum, Steve (IX)	5.274525e-05	629	1960
5	Kaufman, Lloyd	5.135600e-05	300	1945
6	Tatasciore, Fred	5.120594e-05	543	1967
7	Sayre, Jeffrey	4.979781e-05	510	1900
8	Roberts, Eric (I)	4.900497e-05	428	1956
9	Farnum, Franklyn	4.884797e-05	608	1878
10	Kemp, Kenner G.	4.854207e-05	494	1908

Table 1: Top 10 Pagerank Actors

Then we can list the top 10 famous movie celebrities as following, namely 'Leonardo DiCaprio','Scarlett Johansson' ,'Tom Cruise','Brad Pitt','Jackie Chan'(China),'Emma Watson', 'Jay Chou'(China),'Jennifer Lawrence','Meryl Streep', 'Will Smith','Binging Fan'(China). We find their PageRank scores as below.

Name	Scores	Num. of movies	Year of Birth
Leonardo DiCaprio	1.838731e-05	37	1974
Scarlett Johansson	1.661759e-05	54	1984
Tom Cruise	2.154690e-05	45	1962
Brad Pitt	9.966488e-06	75	1963
Jackie Chan	2.356326e-05	129	1954
Emma Watson	1.124533e-05	21	1990
Jay Chou	4.751665e-06	11	1979
Jennifer Lawrence	1.046650e-05	28	1990
Meryl Streep	1.807936e-05	77	1949
Will Smith	1.682103e-05	38	1968
Binging Fan	8.230334e-06	45	1981

Table 2: Top 10 Pagerank Actors

As we can see from these two tables, there is no significant pairing among actors. The actors/actresses that we are familiar with are young and active actors. However, they may not act in hundreds of movies and cooperate with thousands of actors, which leads to the result that their PageRanks are not as high as those legendary actors/actresses. But as the number and importance of their movies increase, one day these well-known actors/actresses will show up in the high PageRank list.

5 Prob 4: A Weighted Undirected Network of Movies

From this part, we construct a movie network according to the set of actors/actresses (larger than 5). The weight of edges between movie nodes is defined as Jaccard Index: the intersection of actor sets of 2 movies divided by the union of the actor sets. Here we calculate the union of the actor sets as the sum of the sets minus the intersection of them. Thus the more common actors in two movies, the higher Jaccard Index will be. Using this method we can create an undirected network of movies.

Considering that it takes a long time to run the code in Problem 2, we decide to change our method. This time we build three dictionaries. Two of them are: 'movie to actor', whose keys are the given movies and the values are the actors; 'actor to movie', whose keys are actors and values are movies that actors have acted. By using these two dictionaries, we can get the third 'movie intersection' whose keys are the given movie and the values are movies which have common actors with given movie. Using this method, we can calculate the weight, then we write a txt file in which each line includes two nodes and the weight of edge between these two nodes

We find that if we keep all the movies with more than 5 actors/actresses, the data set will be too large to run the Fast Greedy algorithm in the following problems. So we change the threshold to 10 and 15. In this way, we have calculated the number of movies who have more than 5,10,15 actors separately. We can get the table below.

Threshold	Vertice	Edges
5	390000	75890439
10	180000	54653804
15	140000	37931582

Table 3: The Vertice and Edges of different thresholds

So we decide to use 15 (the number of actors that a movie have), instead of 5, as our threshold in order to reduce the computational work.

6 Prob 5: Communities Finding on Movie Network

Using the network created in Problem 4, we can run the Fast Greedy Newman algorithm to find communities in this network. Results show that there are 208 communities. There are so many communities in this network because some nodes are disconnected with others, which means that these movies do not have common actors with others. We can list the ID of community with size >1000 as follows:

Community ID	size	Community ID	size
1	67958	2	10322
3	5330	4	2813
5	14002	6	3125
7	3932	8	5335
9	9354	10	3534
11	1484	12	1179
14	3793	15	1861
17	1185	19	1301
28	1597		

Table 4: ID of community with size >1000

Obviously, most communities (191/208) have less than 1000 nodes. Then we can make a dictionary in Python where keys are movie names and values are their genres and write the results as the movie id order into a txt file. After that, in R we can add the genre to each movie node in the network and tag the communities with the genres that appear in 20% or more of the movies in the community. If there are many genres satisfying this condition, we choose the genre with highest frequency as the tag. Because it takes a lot of space to list the tag of each community one by one, we list the frequency of each tag of communities in the network instead:

Tag	Frequency	Tag	Frequency
Drama	0.2115385	Western	0.004807692
Comedy	0.06730769	Horror	0.07692308
Short	0.3125	Romance	0.02884615
Thriller	0.09134615	Sport	0.01923077
History	0.009615385	Documentary	0.03365385
Music	0.01923077	Action	0.01442308
Fantasy	0.009615385	Adult	0.004807692
War	0.02403846	Sci-Fi	0.03365385
Mystery	0.01442308	Adventure	0.009615385

Table 5: Frequency of Each Tag of Communities

From the table we can tell the most common tag is drama. Thus the tags are not meaningful because different communities can have the same tag. So the tag cannot represent the uniqueness of each communities. However, if we use Top 3 or Top 5 genres as the tag of the communities, it will make the tag meaningful as different communities have different tags.

7 Prob 6: Neighbor finding on Movie Network

In this part, we are required to find the top 5 nearest neighbors of three given movies, namely 'Batman v Superman: Dawn of Justice (2016)','Mission: Impossible - Rogue Nation (2015)','Minions (2015)' and find the communities they belong to.
After we get the communities of movie network by the method of Fast Greedy, we can easily do it. The results are shown below.

Movies	Community
Batman v Superman: Dawn of Justice (2016)	1
Mission: Impossible - Rogue Nation (2015)	1
Minions (2015)	1

Table 6: Rating of Top 5 Nearest Neighbors of Minions (2015)

Top 5	Neighbors of Batman	Community
1	Eloise (2015)	1
2	Into the Storm (2014)	1
3	The End of the Tour (2015)	1
4	Grain (2015)	1
5	Man of Steel (2013)	1

Table 7: Top 5 Nearest Neighbors of Batman v Superman: Dawn of Justice (2016)

Top 5	Neighbors of Mission Impossible	Community
1	Fan (2015)	1
2	Phantom (2015)	1
3	The Program (2015/II)	1
4	Breaking the Bank (2014)	1
5	Legend (2015/I)	1

Table 8: Top 5 Nearest Neighbors of Mission: Impossible - Rogue Nation (2015)

Top 5	Neighbors of Minions	Community
1	The Lorax (2012)	1
2	Inside Out (2015)	1
3	Despicable Me 2 (2013)	1
4	Up (2009)	1
5	Surf's Up (2007)	1

Table 9: Top 5 Nearest Neighbors of Minions (2015)

From these results, we can clearly see that almost all of the neighbors of Batman are about adventure, science fiction and action. Especially the movie 'Man of Steel (2013)' has one same superhero with 'Batman v Superman: Dawn of Justice (2016)'. For the movie 'Mission: Impossible - Rogue Nation (2015)', its top 5 neighbors are all about action and gunplay movies, such as 'Phantom (2015)' which is about a disgraced Indian soldier carries out a series of assassinations in the hope of restoring his honor (just like Mission Impossible movie). For the movie 'Minions (2015)', its neighbors are animation and family movies, especially 'Despicable Me 2 (2013)' which is also a interesting movie about minions. Thus we can conclude that our method is pretty good and it gives us really nice result.

Interestingly, we find that not only these three movies and their corresponding neighbor movies are in the same community, but also these three movies are is the same community. It is probably because that the community is the largest community in movie network and this community is much larger than any other communities. These movies have many common actors/actresses so the weights of edges between them are high. So the Fast Greedy algorithm divide them to one community.

8 Prob 7: Movie Rating Prediction

In this part, we are required to predict the rating of these three movies. We first use the result of last problem to get the names of top 5 nearest neighbors of these three movies and then we read the 'movie rating' text file to get the rating of these movies if they have been rated. The following results are shown below (N/A means no such movie in 'movie rating' text file):

Top 5	Neighbors of Batman	Rating
1	Eloise (2015)	N/A
2	Into the Storm (2014)	5.9
3	The End of the Tour (2015)	7.8
4	Grain (2015)	N/A
5	Man of Steel (2013)	7.2

Table 10: Rating of Top 5 Nearest Neighbors of Batman v Superman: Dawn of Justice (2016)

Top 5	Neighbors of Mission Impossible	Rating
1	Fan (2015)	N/A
2	Phantom (2015)	5.6
3	The Program (2015/II)	6.5
4	Breaking the Bank (2014)	8.6
5	Legend (2015/I)	7.0

Table 11: Rating of Top 5 Nearest Neighbors of Mission: Impossible - Rogue Nation (2015)

Top 5	Neighbors of Minions	Rating
1	The Lorax (2012)	6.5
2	Inside Out (2015)	8.9
3	Despicable Me 2 (2013)	7.5
4	Up (2009)	8.0
5	Surf's Up (2007)	6.5

Table 12: Rating of Top 5 Nearest Neighbors of Minions (2015)

In order to get the rating of these three movies, we decide to calculate the weighted average rating of their Top5 neighbors. Here, for simplicity, we set the weight of neighbors from 1 to 5 is [1.4,1.3,1.2,1.1,1.0], then we can get the rating of these three movies shown below. We can see that some of the result are predicted well and some are not, the average error is 0.57. It is mainly because that there are lots of factors that can determine whether audiences like these movies or not. Only thinking about the rating of Top5 neighbors obviously is not an ideal method for predicting the rating of that movie.

Movies	Rating Predicted	Actual Rating
Batman v Superman: Dawn of Justice (2016)	6.9	7.1
Mission: Impossible - Rogue Nation (2015)	6.8	7.5
Minions (2015)	7.4	6.6

Table 13: Prediction and Actual Rating of Three Movies

9 Prob 8: Predicting the ratings by Regression Models

To train the regression models, we merged the movie_rating list, movie_genre list, director_movie list and the results of page ranks in problem4 to get computable training dataset. There are 186906 available records in the intersection of the above tables. Caret package was used for data processing and model training.

9.1 Features

The features we used are:
1.Top 5 PageRanks of the actors (five floating point values) in each movie. We use the resulting PageRanks of actors acting in more than 5 movies. PageRanks of actors with less than 5 works are considered as 0.
2. The 101 boolean values representing if the director of the movies is one of the top 100 directors or not. Each of the first 100 values represents a top 100 director, and the last one represents no-top directors.
3.The average rating of movies of the same genre. We calculated the average score of each genre and attached the responding score with the movies of that genre.

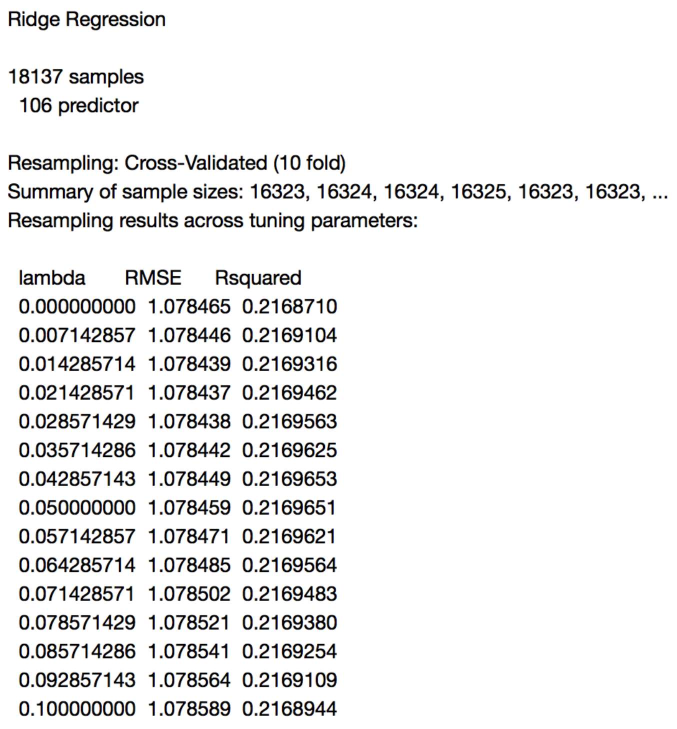
9.2 Sampling and data preprocessing

To reduce the complexity of the models, we resample the dataset to get 18137 training samples. The samples are generated by createDataPartition() randomly.

According to our observation, the original data are unevenly distributed regarding the parameter representing top directors. That is to say, the portion of movies directed by top 100 directors is very small. To reduce the imbalance, we repeated the records directed by the top 100 directors in the resampling process and train the models with the balanced samples. Besides, we scale and center the data, reduce the skewness and get rid of the zero-variance predictors by preProcess() function.

9.3 Model training and predictions

We trained several regression models and here take two of them to illustrate our observation. The first one is the ridge regression model, which is a regularized linear regression model with penalty parameters, stands out for its fast computing path. The other one is the cubist model, a rule-based model that is an amalgamation of several methodologies, which gives the most accurate prediction. We created parameter grids to help tune models and the results are showed as follows.



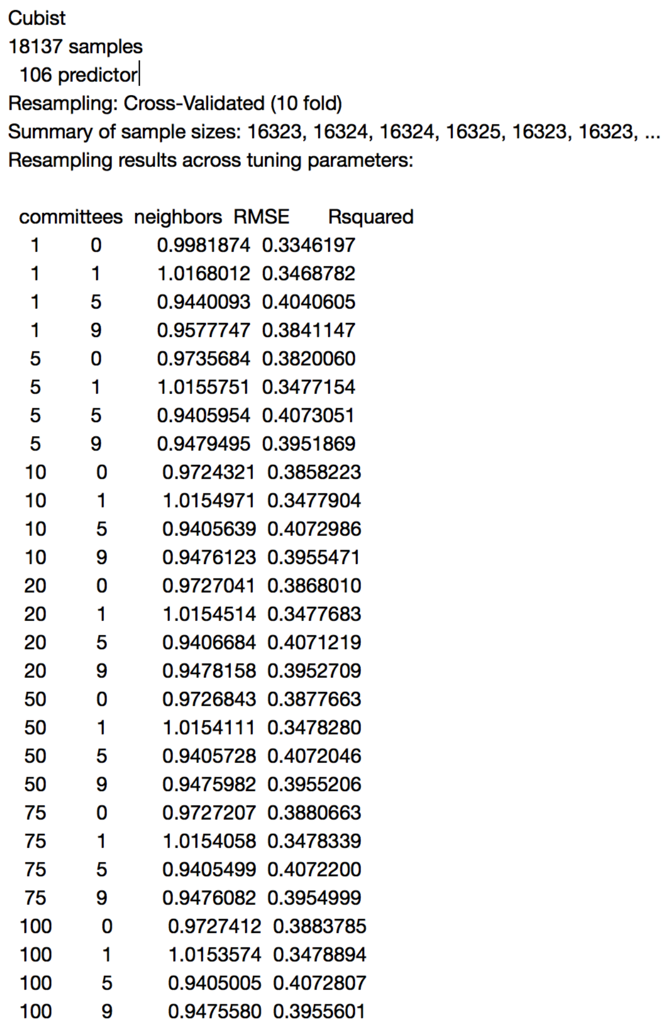
```
Ridge Regression

18137 samples
  106 predictor

Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 16323, 16324, 16324, 16325, 16323, 16323, ...
Resampling results across tuning parameters:

  lambda       RMSE      Rsquared
  0.000000000  1.078465  0.2168710
  0.007142857  1.078446  0.2169104
  0.014285714  1.078439  0.2169316
  0.021428571  1.078437  0.2169462
  0.028571429  1.078438  0.2169563
  0.035714286  1.078442  0.2169625
  0.042857143  1.078449  0.2169653
  0.050000000  1.078459  0.2169651
  0.057142857  1.078471  0.2169621
  0.064285714  1.078485  0.2169564
  0.071428571  1.078502  0.2169483
  0.078571429  1.078521  0.2169380
  0.085714286  1.078541  0.2169254
  0.092857143  1.078564  0.2169109
  0.100000000  1.078589  0.2168944
```

Figure 1: Ridge regression model tuning results



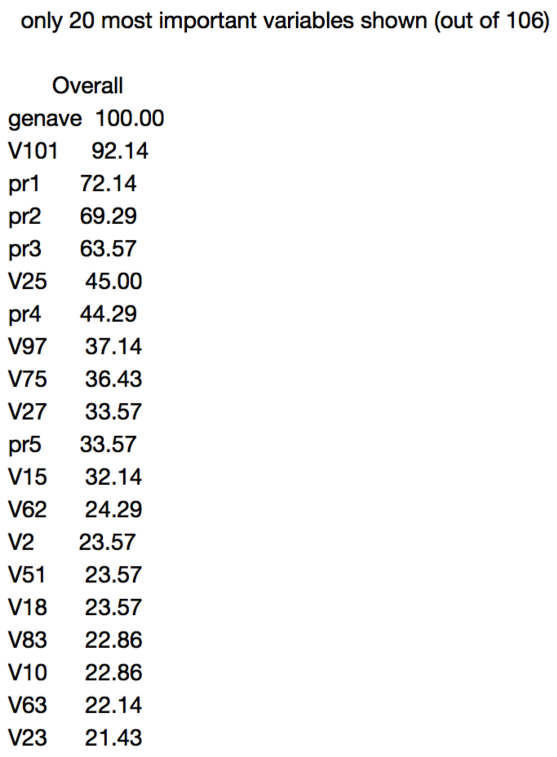
```
Cubist
18137 samples
  106 predictor
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 16323, 16324, 16324, 16325, 16323, 16323, ...
Resampling results across tuning parameters:

  committees  neighbors  RMSE       Rsquared
    1          0         0.9981874  0.3346197
    1          1         1.0168012  0.3468782
    1          5         0.9440093  0.4040605
    1          9         0.9577747  0.3841147
    5          0         0.9735684  0.3820060
    5          1         1.0155751  0.3477154
    5          5         0.9405954  0.4073051
    5          9         0.9479495  0.3951869
   10          0         0.9724321  0.3858223
   10          1         1.0154971  0.3477904
   10          5         0.9405639  0.4072986
   10          9         0.9476123  0.3955471
   20          0         0.9727041  0.3868010
   20          1         1.0154514  0.3477683
   20          5         0.9406684  0.4071219
   20          9         0.9478158  0.3952709
   50          0         0.9726843  0.3877663
   50          1         1.0154111  0.3478280
   50          5         0.9405728  0.4072046
   50          9         0.9475982  0.3955206
   75          0         0.9727207  0.3880663
   75          1         1.0154058  0.3478339
   75          5         0.9405499  0.4072200
   75          9         0.9476082  0.3954999
  100          0         0.9727412  0.3883785
  100          1         1.0153574  0.3478894
  100          5         0.9405005  0.4072807
  100          9         0.9475580  0.3955601
```

Figure 2: Cubist regression model tuning results

We've tried many methods like resampling in different ways, combining features and tuning model parameters and above is the best model can we get. The R-squared value is no more than 0.408, which indicates that the percentage of the response variable variation that is explained by the model is low. For one thing, we think the low R-squared value owes to the lack of informative features used for training .For example, the actor's and director's reputation can influence the movie ratings a lot, which cannot be indicated by our training features. An intuitive thought is that the average movie scores in which an actor starred in may indicate his reputation. For another thing, according to our investigation, in some fields like psychology and sociology, it is common to have a low R-squared regression model because of the subtle nature of the problem. When it comes to our problem, it can be interpreted as the hard-to-predict factors influencing movie ratings. The review of a movie is highly related to the current social environment and popular culture. Besides, People's taste of movie are changing time to time, so are the ratings. A famous example is that the Top 1 movie *The Shawshank Redemption* only received middle reviews when it was first published and hadn't make a hit till one year later. So it is fair to say that we can hardly predict the movie ratings following some fixed patterns.

However, the general trends regarding the significant predictors still exist. We analyzed the importance of variables and the top 20 features and their scores of cubist model are showed as follows.



```
only 20 most important variables shown (out of 106)

        Overall
genave  100.00
V101     92.14
pr1      72.14
pr2      69.29
pr3      63.57
V25      45.00
pr4      44.29
V97      37.14
V75      36.43
V27      33.57
pr5      33.57
V15      32.14
V62      24.29
V2       23.57
V51      23.57
V18      23.57
V83      22.86
V10      22.86
V63      22.14
V23      21.43
```

Figure 3: top 20 significant features

It makes sense that the average score of movies of same genre, whether the director is top directors or not and the top 3 page ranks of the movie's actors are the determinative predictors to movie ratings. Therefore, though the cubist model cannot predict the movie score accurately, it can still reflect the general relation between the predictors and rating scores. The approximate movie ratings we predicted with cubist model are listed below. The average error is 0.72.

Movies	Rating Predicted	Actual Rating
Batman v Superman: Dawn of Justice (2016)	6.056345	7.1
Mission: Impossible - Rogue Nation (2015)	6.835598	7.5
Minions (2015)	6.124046	6.6

Table 14: Prediction and Actual Rating of Three Movies

10 Prob 9: Movie Rating Prediction by Bipartite Graph

In this part we try to predict the rating of these three movies by bipartite. Firstly we construct a bipartite graph where actors/actresses representing the vertices of one part and movies representing the vertices of other part. An actor/actress is connected to all the movies he has played in. In Python, we can assign each actor/actress and each movie with an unique ID. Then we write the type of each vertice in the network (0 means actor, 1 means movie) into 'type_file.txt' file and the edges between them into 'edge_file.txt'. Then we can construct a bipartite graph in R by using these files as inputs of graph.bipartite function to create a bipartite graph.

After constructing the bipartite graph, we can calculate the average rating of movies that each actor acts in. In order to decrease the complexity of this algorithm, we load the rating of movies into a dictionary in Python using movie names as keys and ratings as values. So for each actor/actresses, we find the ratings of movies he/she acts in and calculate the average rating. The results are written in the file ave_rating.txt.

To quickly predict the rating of three movies, when we look at the movies each actor acts in and calculate the average rating, we add the average rating of this actor to the corresponding list if we find he acts in the target movies. When we finish iterating through all the actors and actresses, we just need to calculate the average of corresponding lists and use the result as the prediction of rating. The results are as follows:

Movies	Rating Predicted	Actual Rating
Batman v Superman: Dawn of Justice (2016)	6.60	7.1
Mission: Impossible - Rogue Nation (2015)	6.87	7.5
Minions (2015)	6.57	6.6

Table 15: Prediction and Actual Rating of Three Movies

Results show that the average error is 0.38, which is better than using Top 5 neighbors for prediction. The reason for this error is that we just simply calculate the average of all actors in this movie. However, some actors (like main characters or famous actors/actresses) are more important to movies and have greater influence on the rating of movies. Thus it is more reasonable to use the weighted average. Besides, for each actor, it is better to use weighted average of ratings of movies he acts in because it is some movies (not all of them) that makes them good or bad actors. Sometimes good actor may also plays a role in a movie that receives low rating, but this does not change the fact that he is a good and popular actor. Thus to improve the accuracy of prediction, we should use the weighted average of move ratings of each actor and the weighted average of actors in target movies. However, it is really hard to choose the weights accurately.

Moreover, actors/actresses are only one factor that influences the rating of movies. Other factors like plot, release date, genre of movie... will also influence the rating of movies. In conclusion, we should take every factor into consideration so we can accurately predict the rating of movies.