EE232E PROJECT 4

IMDb Mining

Devanshi Patel	504945601	devanshipatel@cs.ucla.edu
Ekta Malkan	504945210	emalkan@cs.ucla.edu
Pratiksha Kap	704944610	pratikshakap@cs.ucla.edu
Sneha Shankar	404946026	snehashankar@cs.ucla.edu

Introduction

The IMDb is an excellent resource to find detailed information about almost any film ever made. It contains a vast amount of data, which undoubtedly contains valuable information about general trends in films. Data mining techniques enable us to uncover information which will both confirm or disprove common assumptions about movies, and also allow us to predict the success of a future film given select information about the film before its release.

In this project, we study various properties of IMDb dataset by first exploring properties of a directed actor/actress network and then using an undirected movie network.

Part 1: Actor/Actress Network

The IMDb dataset is huge and contains a ton of information that can be mined and analyzed. In this part, we generate a directed network using data from the following text files:

1. Actor_movies.txt
2. Actress_movies.txt

In order to generate a consistent network without any ambiguity, we first need to perform some data preprocessing. The first step is to merge the above mentioned text files so that we have all the information about movie stars and their corresponding movies in a single file. We write the code to generate the merged file in Python.

Since the files are huge, it would have been efficient to read them directly as a table. However, since each movie star has acted in different number of movies, it is not possible to generate a

table with variable number of columns. Instead, we read each file line by line and then parse it by splitting based on the delimiter ("\t\t"). Since we don't want to analyze the movie stars who have acted in fewer movies, we filter our data to include only those movie stars who have acted in 10 or more movies. One might think that it is acceptable to load the data from this merged file and start analyzing it. However, it needs to be cleaned first to avoid inconsistency during network creation.

During the analysis of the merged text file, we found that it listed same movie multiple times due to the role of movie stars in that movie. For example, in some movies the movie star might not have acted but just played the voice. In some movies, the movie star might have been uncredited. To handle such scenarios, it is important to preprocess the movie names so that these kind of movies are considered the same and not treated as separate vertices during graph generation. We perform the following preprocessing as part of this:

1. Remove leading and trailing blank spaces around the movie name
2. Split the movie name to remove extraneous information after the first pair of parentheses

To ensure that we can perform quicker computation on the data in the future, we store the movies and movie stars information in 2 dictionaries. The first dictionary contains the mapping from movie star to list of movies he/she has acted in. The second contains mapping from movie name to the list of movie stars who acted in that movie. Here, we assume that each line in the file represents a unique movie star. However, if the case of duplicate movie star exists, then we just treat them as the same and append the movies to the already existing key for that movie star in the dictionary.

Question 1: Perform the preprocessing on the two text files and report the total number of actors and actresses and total number of unique movies that these actors and actresses have acted in.
Answer:

Total number of actors and actresses	**113,129**
Total number of unique movies	**468,192**

Question 2: Create a weighted directed actor/actress network using the processed text file and equation 1. Plot the in-degree distribution of the actor/actress network. Briefly comment on the in-degree distribution.

Answer: We now create a directed network from the cleaned text file. In this network, the actor/actress are represented by nodes and the edges connecting the nodes are weighted. We used the following equation to assign weights:

$$w_{i \to j} = \frac{|S_i \cap S_j|}{|S_i|}$$

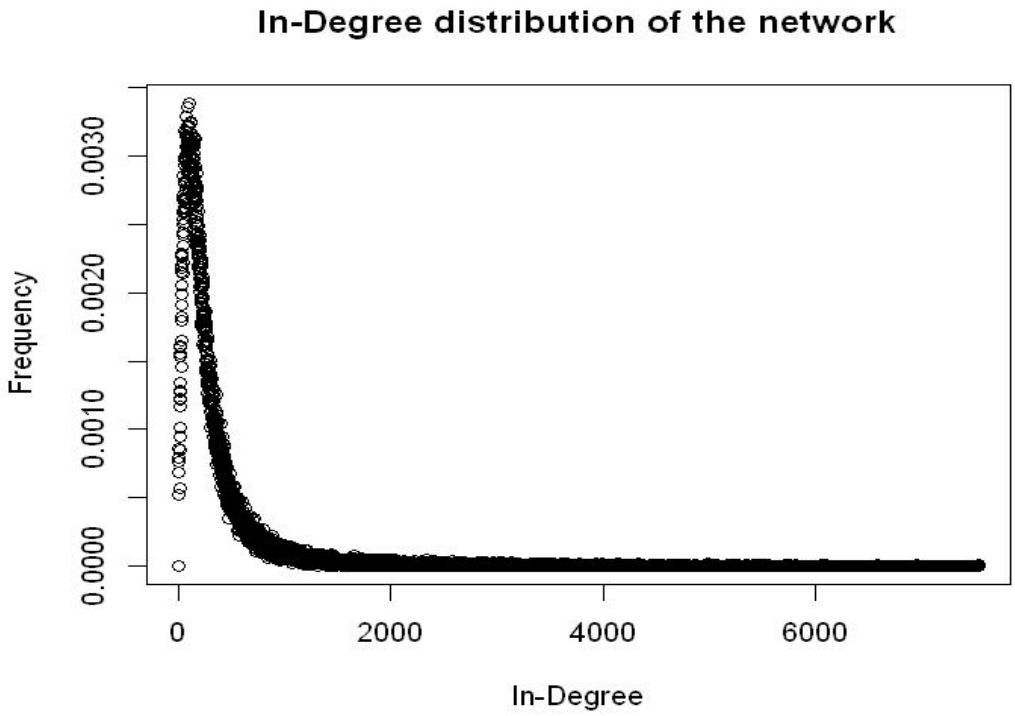
In the above equation, S_i and S_j are the set of movies starring movie star i and j respectively. Thus, we divide the number of common movies between two stars by the number of movies of that star to draw a directed weighted edge from that node.

To create the network, we first need to generate the edgelist file that contains the list of all edges together with their weights. We follow the below steps to obtain the weighted edges:

1. Iterate over each actor/actress in actor to movie dictionary.
2. Generate a dictionary that stores number of common movies with each co-star of that actor.
3. Iterate through this dictionary for each co-star to obtain weighted edges for that movie star.

We wrote the code for above steps in Python and generated an edgelist file that can be directly loaded into a graph. We used igraph library available in R to load the graph as it is very quick.

In order to get some idea about properties of the network, we plot its in-degree distribution.



From the above plot of in-degree distribution, we can observe that it follows the Power law. For majority of the nodes, the in-degree is very low. There are few nodes which have high values of in-degree. In other words, most of the movie stars have worked with less number of people throughout their career. There are only some popular movie stars and they would have acted with a huge number of people hence leading to a high in-degree.

Question 3: Design a simple algorithm to find the actor pairings. To be specific, your algorithm should take as input one of the actors listed above and should return the name of the actor with whom the input actor prefers to work the most. Run your algorithm for the actors listed above and report the actor names returned by your algorithm. Also for each pair, report the (input actor, output actor) edge weight. Does all the actor pairing make sense?

Answers: In this part, we find pairings between actors and try to determine the name of the actor that a certain actor prefers to work the most with. To obtain the pairs, we wrote an algorithm that takes an actor/actress as input and looks at all the outgoing edges for that actor. The edge with highest weight is selected and that determines the output actor that our input

actor prefers the most. The below table lists out all the pairs and their corresponding edge weights:

Input Actor	Output Actor	Edge weight
Tom Cruise	Nicole Kidman	0.1746
Emma Watson (II)	Daniel Radcliffe	0.5200
George Clooney	Matt Damon	0.1194
Tom Hanks	Tim Allen (I)	0.1013
Dwayne Johnson (I)	Steve Austin (IV) Paul Levesque (I) Mark Calaway	0.2051
Johnny Depp	Helena Bonham Carter	0.0816
Will Smith (I)	Darrell Foster	0.1224
Meryl Streep	Kevin Kline (I) Robert De Niro	0.0618
Leonardo DiCaprio	Martin Scorsese	0.1020
Brad Pitt	George Clooney	0.0986

In the above table, the edge weight for Emma Watson is 0.52 which is quite high. This indicates she strongly prefers working with Daniel Radcliffe. Thus, this pairing makes sense. Similarly, whichever pairing has fairly high edge weight make sense.

However, by carefully examining all pairings, we noticed that not all the pairs make sense. For Dwayne Johnson, our algorithm returned three actors because the edge weights for all of them were equal. This makes sense mathematically but not logically i.e. an actor should prefer to work the most with just one other actor. Even in the case of Meryl Streep, our algorithm returns two output actors but this also doesn't make sense. Moreover, the associated edge weight is 0.0618 which is very low and not a good indicator of preference between two actors. It might be due to the reason that Meryl Streep might have acted with a lot of different actors with less repeats of co-actors. Thus, the largest edge weight ended up being so low.

Since our algorithm is very simple, it takes into consideration only the edge weight i.e. the number of common movies between actors to decide the pairing. This measure is not very accurate. If some other parameters are also considered while selecting the pairs, then we won't get multiple co-actors for the same actor.

Question 4: Use the google's pagerank algorithm to find the top 10 actor/actress in the network. Report the top 10 actor/actress and also the number of movies and the in-degree of each of the actor/actress in the top 10 list. Does the top 10 list have any actor/actress listed in the previous section? If it does not have any of the actor/actress listed in the previous section, please provide an explanation for this phenomenon.

Answer: In this section, we use the pagerank algorithm developed by Google to find top 10 movie actor/actress in our network. We use the pagerank function available in igraph package of R.
The table below shows the top 10 movie stars together with the number of movie they have starred in and their in-degree.

Actor/Actress	Number of Movies	In-degree	Pagerank
Bess Flowers	828	7537	0.000235
Fred Tatasciore	353	3954	0.000199
Sam Harris (II)	600	6960	0.000197
Steve Blum (IX)	373	3316	0.000195
Harold Miller (I)	561	6587	0.000173
Ron Jeremy	637	2905	0.000158
Lee Phelps (I)	647	5563	0.000157
Yuri Lowenthal	317	2662	0.000157
Robin Atkin Downes	267	2953	0.000152
Frank O'Connor (I)	623	5502	0.000147

The above table does not have any of the actors in the previous section. The actors mentioned previously are extremely famous and it was expected that they would be the ones with top pagerank values. However, when we actually calculate the pagerank and sort it in descending order, the top 10 actors obtained do not match with the previous list at all. Moreover, the actors in above table are not even familiar to our generation. Most of them are very old right now and they mostly played the role of voice actors in animation movies.

The pagerank of a page is determined based on the number and quality of other important pages pointing to it. If applied to our example, the pagerank of an actor is high if there are

edges pointing to that actor from other important actors. Thus, the only justification for these unpopular actors having high pageranks is that they must be sharing connections with many other influential and famous actors. This is evident from the table above that has high values for in-degree and number of movies of these actors.

Additionally, since most of these actors have acted as voice actors in animation movies, the number of movies is high because they do not need to actually film it and just need to invest time in audio recordings. These actors seem to form a community and they have many movies in common as they have worked in mostly animation movies. As other actors with high page rank within community point to these actors, their page rank also increases.

Question 5: Report the pagerank scores of the actor/actress listed in the previous section. Also, report the number of movies each of these actor/actress have acted in and also their in-degree.

Answer: The table shown below lists the pagerank scores, number of movies as well as the in-degree of the actor/actress listed in question 3.

Actor/Actress	Number of Movies	In-degree	Pagerank
Tom Cruise	63	1651	3.974e-05
Emma Watson (II)	25	453	1.75e-05
George Clooney	67	1573	4.00e-05
Tom Hanks	79	2064	5.10e-05
Dwayne Johnson (I)	78	1357	4.20e-05
Johnny Depp	98	2144	5.38e-05
Will Smith (I)	49	1319	3.20e-05
Meryl Streep	97	1594	3.96e-05
Leonardo DiCaprio	49	1301	3.17e-05
Brad Pitt	71	1739	4.29e-05

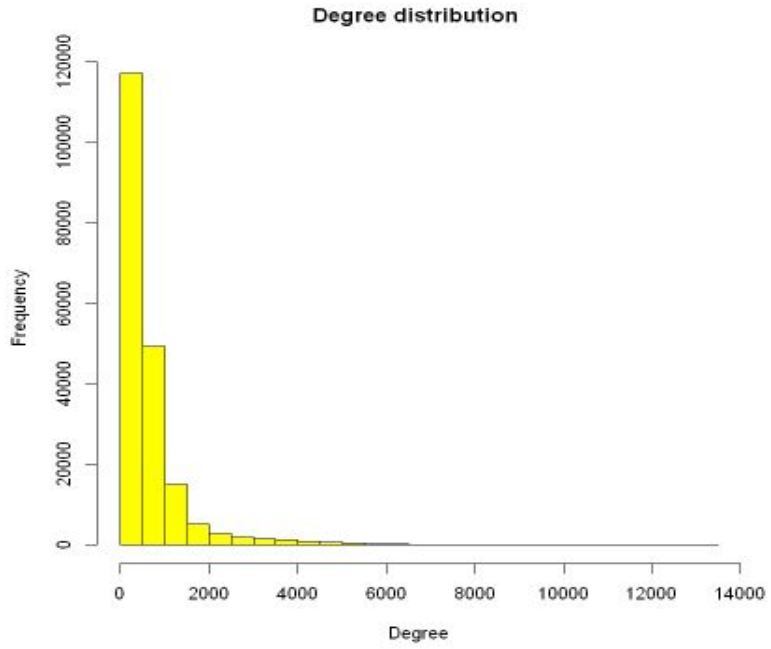
In the above table for famous actors, we observe that the pagerank values are quite low compared to the top 10 pagerank values. This is also evident from the fact that the number of movies and the in-degrees of these actors is lower in comparison.

The reason for low pagerank values might be that usually famous actors don't end up working in a lot of common movies. Their movies are with a lot of different actors. Moreover, since these actors don't prefer to work in animation movies, they don't share any movies with the top 10 actors mentioned earlier. Thus, they don't inherit the high pagerank values and end with up low values.

Another reason might be that nowadays, most actors have their own personal web pages within IMDb so, they might not have links from pages of other famous actors. Hence leading to a low pagerank.

2 Movie Network

Question 6: Create a weighted undirected movie network using equation 2. Plot the degree distribution of the movie network. Briefly comment on the degree distribution.



We observe that the above graph follows power-law distribution. This shows that clearly there are a few movies with a high degree distribution, whereas other movies have a very low degree distribution. This means that some sets of movies have many common actors, and other sets hardly have any actor common with other movies. This is expected as the IMDb database contains movies of different genres, as well as from many different languages and regions. Like we observed Hindi Bollywood (Indian) movies, Chinese movies, Hollywood movies etc.Each of these movies are seen by a different kind of audience, and the local popularity of actors may not apply globally. Hence, they might not share any actor/actresses in common, or share very few actors/actresses with each other. However movies belonging to the same community might share a lot of actors.

Question 7: Use the Fast Greedy community detection algorithm to find the communities in the movie network. Pick 10 communities and for each community plot the distribution of the genres of the movies in the community.

Answer: We ran fast greedy algorithm in the graph obtained in the previous question. Running of FastGreedy on such a huge graph is time and resource consuming. We ran it on an Amazon EC2 instance of 32 GB RAM, and it took a little over 6 hours for the computation.

Before analyzing the movie distribution of the entire genre, let us first analyze the movie distribution of all the movies in the imdb dataset.

Out of our clean and preprocessed results , we found the following statistics:

Action	4162
Adult	2664
Adventure	2469
Animation	397
Biography	160
Comedy	21504
Crime	3185
Documentary	2055
Drama	45081
Family	3437
Fantasy	2779
Film-Noir	222
Game-Show	7
History	1706
Horror	3781
Music	1451
Musical	3550
Mystery	3036
News	40
Reality-TV	9
Romance	18523
Sci-Fi	3635
Short	19978
Sport	1668
Talk-Show	5
Thriller	15894
War	4937
Western	7575

As we can clearly observe, the movies of **“Drama”** genre seem to have around **45081** movies out of 197377 movies, which is a huge count. This means that the frequency of “Drama” movies is highest in our database, followed by Comedy at 21504 movies, the “Short” at 19978 and then

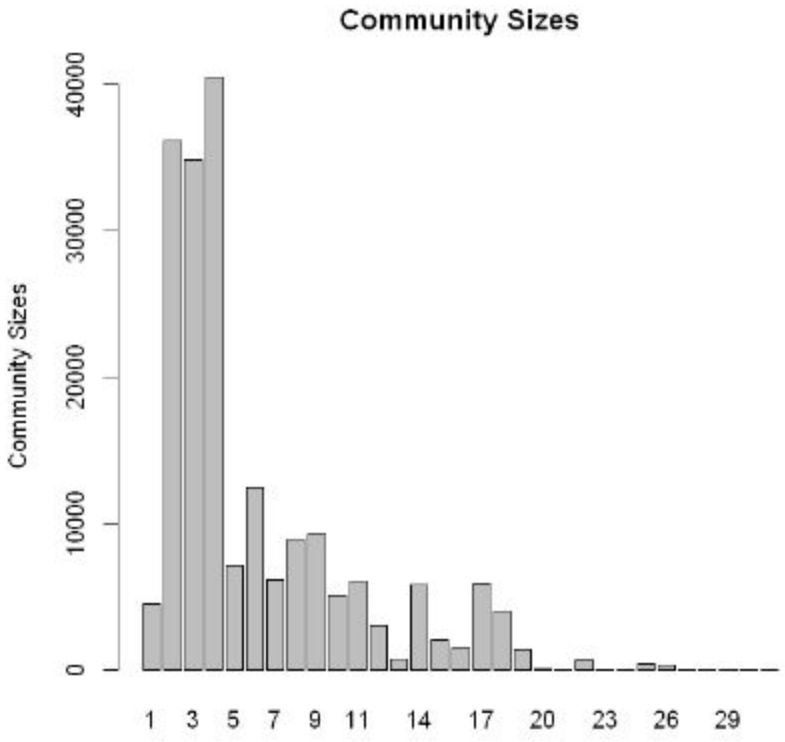
Romance ar 18523. All the other genres contain very few number of movies as compared to these 5 genres.

Also, a quick check of lengths reveals that out of 197377 movies present in our graph and communities, only 173910 movies were obtained from the movie_genre file. This means that we do not have enough data to cover all the movies, and the genre for some of the movies would be missing.

We obtained 31 communities of various sizes using FastGreedy as follows :

Community ID	Number of Movies	Community ID	Number of Movies
1	4543		
2	36140	17	5885
3	34796	18	3997
4	40489	19	1411
5	7170	20	124
6	12494	21	21
7	6146	22	644
8	8955	23	19
9	9294	24	12
10	5087	25	429
11	6040	26	379
12	3041	27	14
13	779	28	17
14	5858	29	13
15	2087	30	13
16	1479	31	1

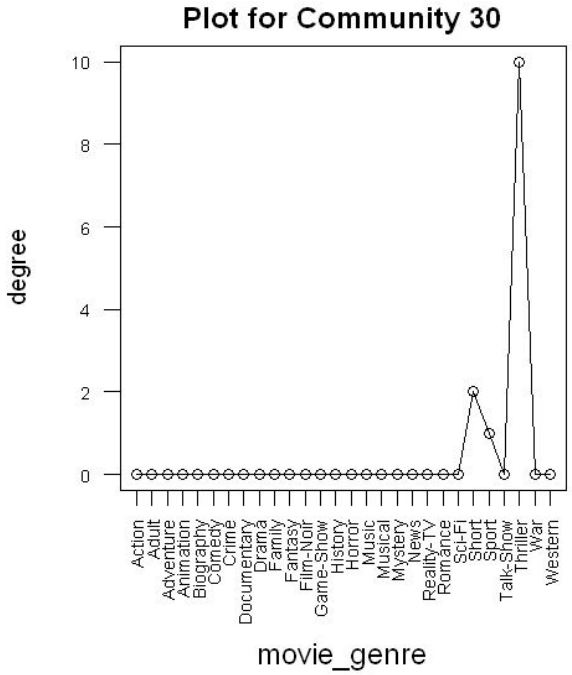
To analyze the community structure, we created a plot as follows:

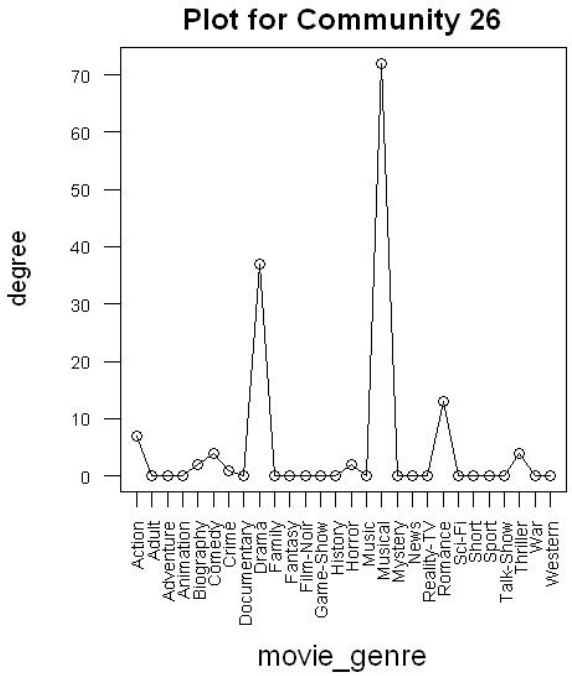


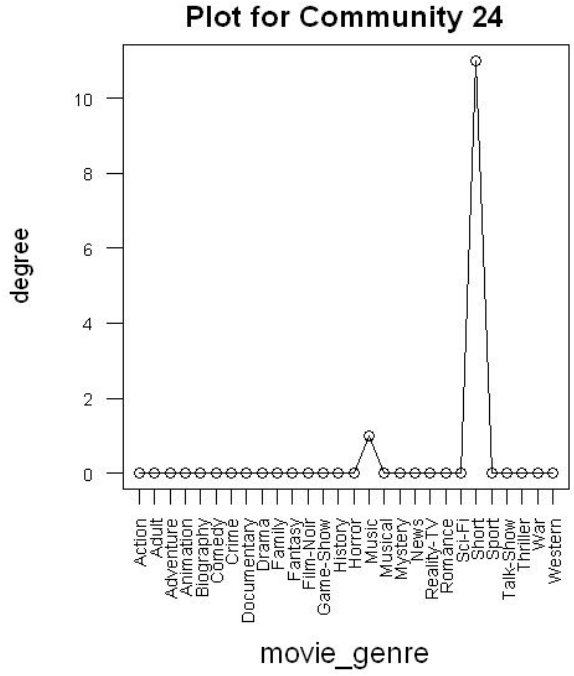
From the above plot, we observe that we have a few giant communities of sizes in range 34-40k,

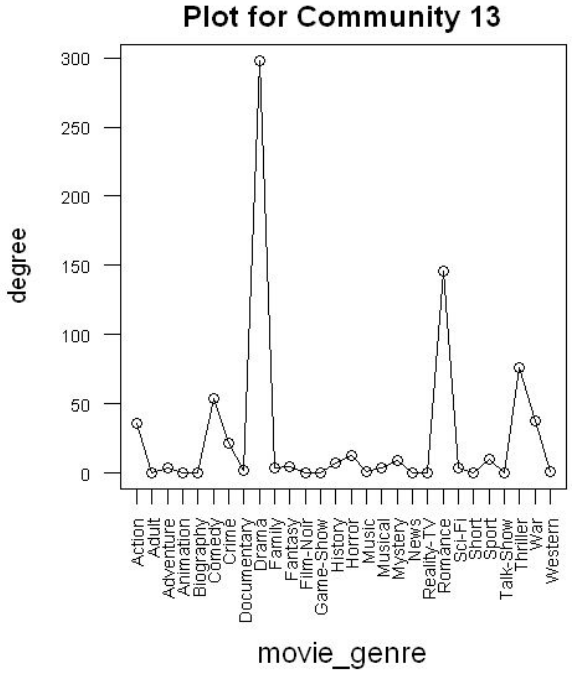
However, most of the other communities are of very small sizes, ranging from a few tens or hundreds upto 12k.

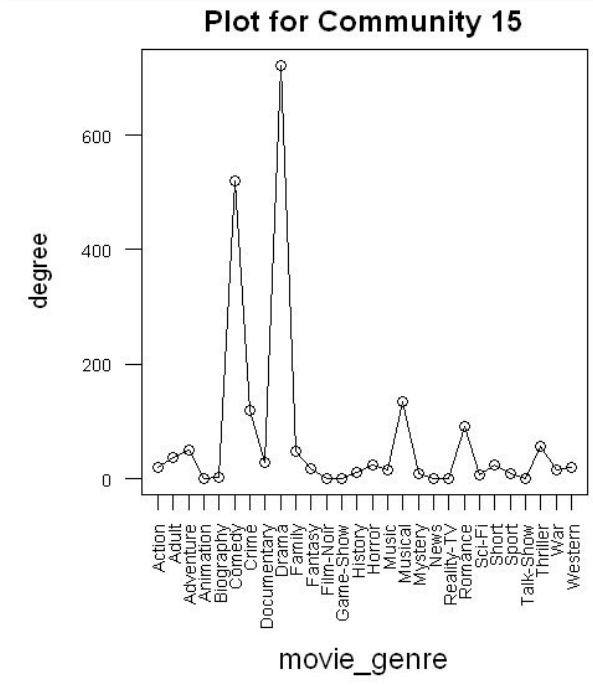
From these 31 communities, we have picked up 10 communities at random of various sizes and analyzed their genre distribution as follows:

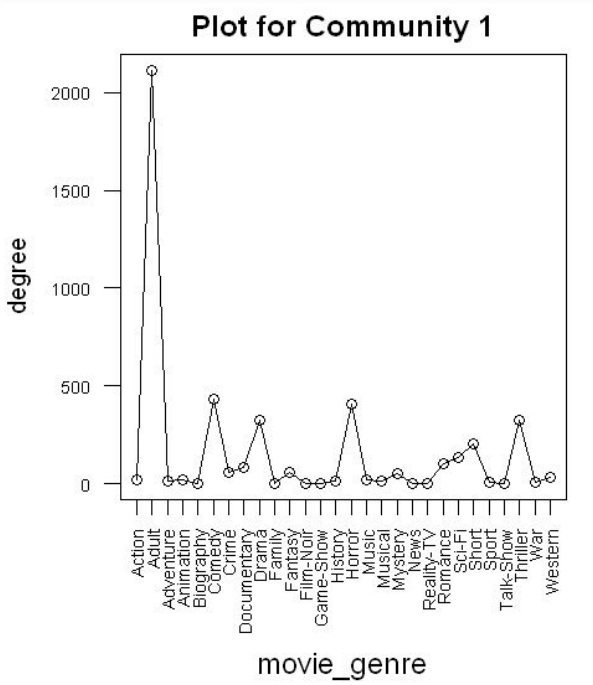

Plot for Community 30
degree
movie_genre
Action
Adult
Adventure
Animation
Biography
Comedy
Crime
Documentary
Drama
Family
Fantasy
Film-Noir
Game-Show
History
Horror
Music
Musical
Mystery
News
Reality-TV
Romance
Sci-Fi
Short
Sport
Talk-Show
Thriller
War
Western

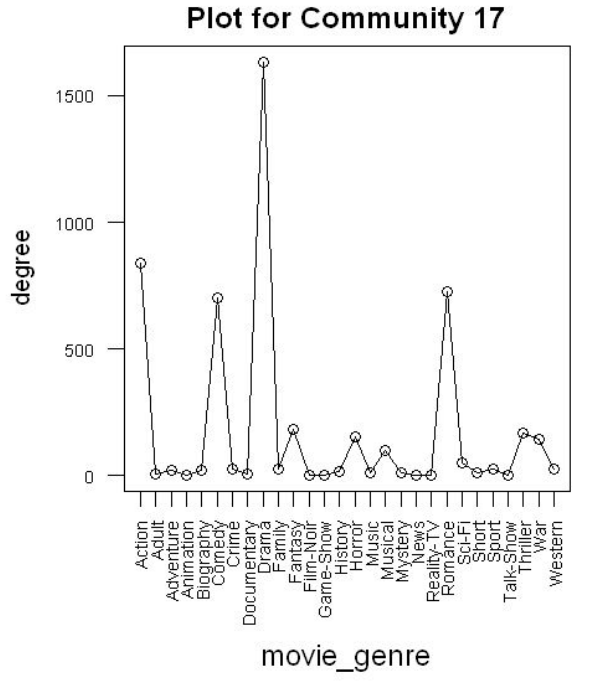

Plot for Community 26
degree
movie_genre

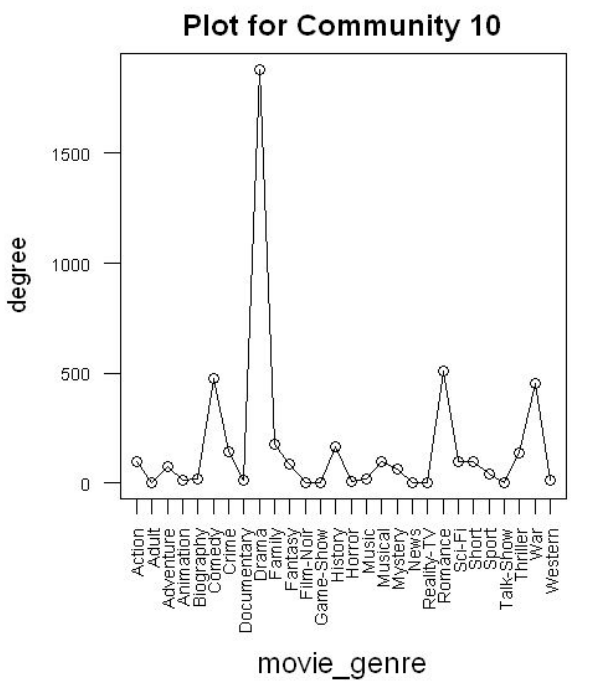

Plot for Community 24
degree
movie_genre

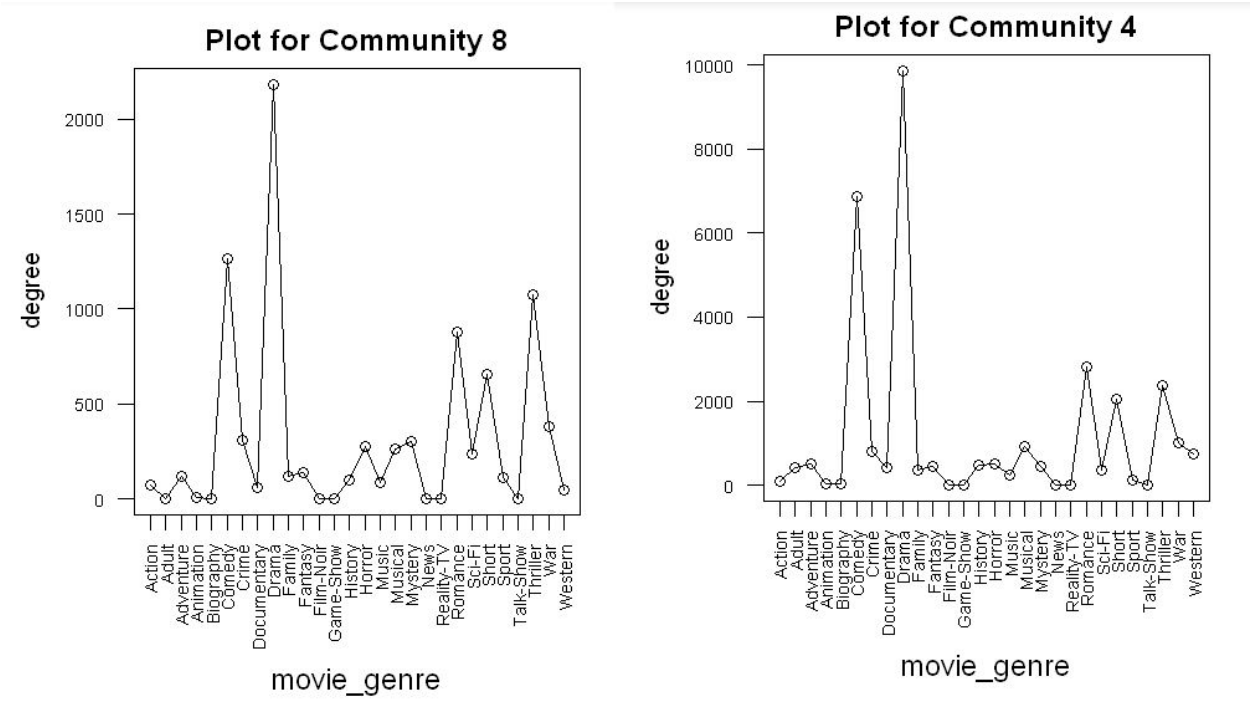

Plot for Community 13
degree
movie_genre


Plot for Community 15
degree
0
200
400
600
Action
Adult
Adventure
Animation
Biography
Comedy
Crime
Documentary
Drama
Family
Fantasy
Film-Noir
Game-Show
History
Horror
Music
Musical
Mystery
News
Reality-TV
Romance
Sci-Fi
Short
Sport
Talk-Show
Thriller
War
Western
movie_genre


Plot for Community 1
degree
0
500
1000
1500
2000
Action
Adult
Adventure
Animation
Biography
Comedy
Crime
Documentary
Drama
Family
Fantasy
Film-Noir
Game-Show
History
Horror
Music
Musical
Mystery
News
Reality-TV
Romance
Sci-Fi
Short
Sport
Talk-Show
Thriller
War
Western
movie_genre


Plot for Community 17
degree
0
500
1000
1500
Action
Adult
Adventure
Animation
Biography
Comedy
Crime
Documentary
Drama
Family
Fantasy
Film-Noir
Game-Show
History
Horror
Music
Musical
Mystery
News
Reality-TV
Romance
Sci-Fi
Short
Sport
Talk-Show
Thriller
War
Western
movie_genre


Plot for Community 10
degree
0
500
1000
1500
Action
Adult
Adventure
Animation
Biography
Comedy
Crime
Documentary
Drama
Family
Fantasy
Film-Noir
Game-Show
History
Horror
Music
Musical
Mystery
News
Reality-TV
Romance
Sci-Fi
Short
Sport
Talk-Show
Thriller
War
Western
movie_genre



Analysis:

For this question, we deliberately chose communities with various size, ranging from the smallest community of size 1 to the largest community of size ~40k.

One important thing to note here is that even though the community sizes may be very large, the actual number of movies from these communities which are present in the movie_genre file may not be the same. The plots and all the analysis here onwards take into consideration only those movies whose genres are available to us from the movie_genre file.

We observe that when the size of community is small (<100) , there is a vast difference in the movie genre distribution of movies in that community. However, as the size of the community goes on increasing, the movie_genre distribution is consistent across communities. This behaviour has been observed for communities ranging from size of 1 upto the size of 45k.

Also a few common trends observe include the following:

1. **Drama movies** are the most frequent genre for highest number of movies across communities.
2. **Comedy** forms the second-highest genre.
3. **Short** forms the third highest genre, followed by **Realty** and the **Thriller**.
4. All other genres have very few movies.

Question 8(a): In each community determine the most dominant genre based simply on frequency counts. Which genres tend to be the most frequent dominant ones across communities and why?

And

Question 8(b): In each community, for the ith genre assign a score of ln(c(i))* p(i)/ q(i) where: c(i) is the number of movies belonging to genre i in the community; p(i) is the fraction of genre i movies in the community, and q(i) is the fraction of genre i movies in the entire data set. Now determine the most dominant genre in each community based on the modified scores. What are your findings and how do they differ from the results in 8(a).

Answer:

Checking only for ten communities first as above, we see that :

Answer :

		Question 8a		Question 8b	
Community ID	**Number of Movies**	**Frequency Count**	**Freq-Dominant Genre**	**Modified Score**	**Modified dominant Genre**
1	4543	2113	Adult	237.1545	Adult
2	36140	6695	Thriller	19.78402	Documentary
3	34796	10868	Short	32.28363	Western
4	40489	9863	Drama	15.32416	Comedy
5	7170	1754	Drama	27.78843	Adventure
6	12494	2633	Drama	27.10024	Family
7	6146	1737	Drama	15.3217	History

8	8955	2182	Drama	11.23784	Mystery
9	9294	2378	Drama	12.62681	Drama
10	5087	1882	Drama	20.87113	War
11	6040	1816	Drama	13.74459	Musical
12	3041	934	Drama	13.11631	Comedy
13	779	298	Drama	9.307087	Romance
14	5858	1593	Drama	27.80878	Family
15	2087	722	Drama	16.34304	Musical
16	1479	570	Drama	32.82124	War
17	5885	1633	Drama	48.28182	Action
18	3997	1134	Drama	34.8016	Adventure
19	1411	426	Drama	16.24796	Family
20	124	55	Drama	7.659954	Drama
21	21	8	Drama	16.36345	Sport
22	644	160	Romance	17.32732	Romance
23	19	7	Romance	7.522897	Romance
24	12	11	Short	19.13437	Short
25	429	238	Drama	16.09823	Action
26	379	72	Musical	106.2296	Musical
27	14	14	Short	22.97319	Short
28	17	9	Short	10.12606	Short
29	13	1	Short	0	Action

30	13	10	Thriller	19.38044	Thriller
31	1	1	Short	0	Action

Observations: As we compare the charts for Q8a and 8b,

For 8a, the metric chosen is the Frequency of each genre in a community. With this metric, we see that **Drama** is the most frequent genre observed across communities.
However, this is a false indicator that "Drama" is the most widely loved movie-genre . The reason being that the statistics get skewed. As we saw earlier, (the plot is shown again below), Drama forms the highest number of movies in our dataset. Also, this is a false indicator, as it does not depict the true importance that a community of movies would have given to a genre, had there been an almost equal distribution in the original dataset.

Action	4162
Adult	2664
Adventure	2469
Animation	397
Biography	160
Comedy	21504
Crime	3185
Documentary	2055
Drama	45081
Family	3437
Fantasy	2779
Film-Noir	222
Game-Show	7
History	1706
Horror	3781
Music	1451
Musical	3550
Mystery	3036
News	40
Reality-TV	9
Romance	18523
Sci-Fi	3635
Short	19978
Sport	1668
Talk-Show	5
Thriller	15894
War	4937
Western	7575

This means that if the results obtained show “Drama” as most important, it is because of very large entries of “Drama” movies, about 45k. This is more than the double the size of second-highest movie-genre “Comedy” of size 21k, and more than 9000 times the movies with least presence, namely “Talk-Show”.

As we learnt in our previous course “Large Scale Data-Mining”, a clean dataset is of equitable entries is very crucial before performing analysis of any kind. Hence, if we want to use our graphs and communities generated for a more sophisticated analysis, we first need to do one of these 2 things:

a) clean/ modify the dataset to contain equitable entries of all genres.
b) Use a Normalization metric.

Using of a normalization metric is precisely what we do in our next step, Question 8b.

Our Modified metric is :
Score= Log(c[i]) *p(i)/q(i)
where:
c(i) is the number of movies belonging to genre i in the community;
p(i) is the fraction of genre i movies in the community
q(i) is the fraction of genre i movies in the entire data set.

This new metric is beneficial because it normalizes the bias due to differences of number of movies in each genre present in our original dataset.

Changes Observed:

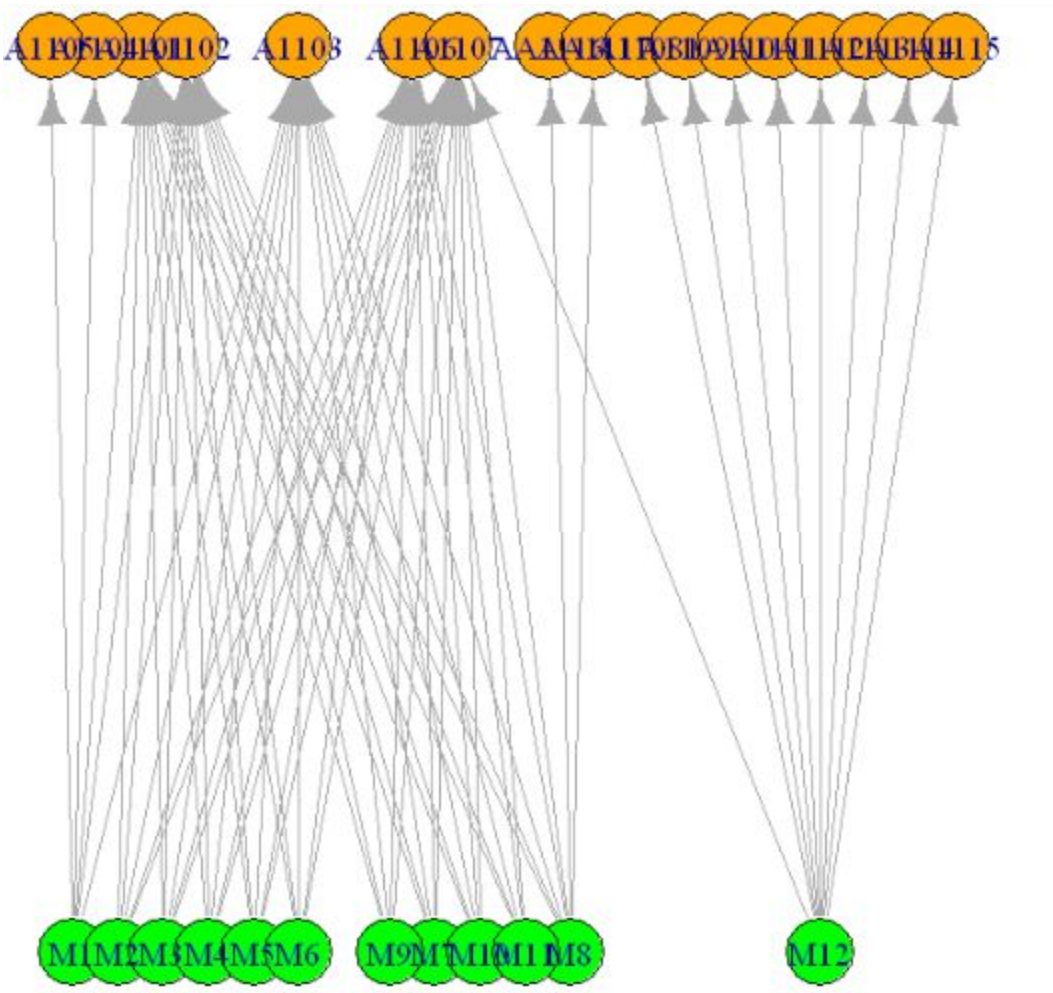
1. As per this modified metric, we observe that the skewness effect of "Drama" and other genres of high frequency is now lost.
2. The new genre representation is normalized, and may be a true indicator of genre-popularity in that community.
3. Some of the examples of this change are as follows:

The most dominant genre of

- Community 2 changed from Thriller to documentary
- Community 3 changed from Short to Western.
- Community 4,5,6,7,8,etc. changed from Drama to Comedy, Adventure, Family, History, Mystery, etc.
- All changes have been highlighted in a darker colour in the table.
- We observe that there is no single genre that appears most dominant across all communities using metrics 2. Due to the loss of the skewness effect, we are now in a better position to analyze the genres truly significant for a community.

Question 8(c): Find a community of movies that has size between 10 and 20. Determine all the actors who acted in these movies and plot the corresponding bipartite graph (i.e. restricted to these particular movies and actors). Determine three most important actors and explain how they help form the community. Is there a correlation between these actors and the dominant genres you found for this community in 8(a) and 8(b).

Answer: For this question, we selected the community of ID 24.
It contains 12 movies and 17 unique Actors.



We have plot a Bipartite graph indicating movies and actors in community 24. A bipartite graph, is a graph such that the node set may be partitioned into two disjoint sets, and each edge has one endpoint in the first set of nodes and the other in the second set of nodes . In many situations, bipartite graphs are related to the analysis of contingency tables [11, 92, 101], also called frequency tables or co-occurrence data. In our case, we use the bipartite graph to map actor-movie relationship within a community of size between 10 and 20 more closely. We also analyze which actors are common across most movies, which gives us an intuition about how these communities might have been formed in the first place.

The three most important Actors in the above community are:

1101 : Marquis,Brother
1102 : Ice,FreshKid
1103 : 2LiveCrew

1107 : Campbell,Luther

Actor	No. Of Movies
1101	11
1102	11
1103	11
1104	1
1105	1
1106	10
1107	11
1108	1
1109	1
1110	1
1111	1
1112	1
1113	1
1114	1
1115	1
1116	1
1117	1

The list of actors present in this community (as per the movies listed above) include:

‘Marquis,Brother', 'Ice,FreshKid', '2LiveCrew', 'Flav,Flavor', 'Rida,Flo'

'Ice,FreshKid', 'Campbell,Luther', 'Mixx,Mr.', '2LiveCrew', 'Marquis,Brother'

'Ice,FreshKid', 'Campbell,Luther', 'Mixx,Mr.', '2LiveCrew', 'Marquis,Brother'

'Ice,FreshKid', 'Campbell,Luther', 'Mixx,Mr.', '2LiveCrew', 'Marquis,Brother'

'Ice,FreshKid', 'Campbell,Luther', 'Mixx,Mr.', '2LiveCrew', 'Marquis,Brother'

'Ice,FreshKid', 'Campbell,Luther', 'Mixx,Mr.', '2LiveCrew', 'Marquis,Brother'

'Ice,FreshKid', 'Campbell,Luther', 'Mixx,Mr.', '2LiveCrew', 'Marquis,Brother'

'DeVito,Danny', 'Mixx,Mr.', '2LiveCrew', 'Marquis,Brother', 'Webb,Chloe(I', 'Campbell,Luther', 'Ice,FreshKid'

'Ice,FreshKid', 'Campbell,Luther', 'Mixx,Mr.', '2LiveCrew', 'Marquis,Brother'

'Ice,FreshKid', 'Campbell,Luther', 'Mixx,Mr.', '2LiveCrew', 'Marquis,Brother'

'Ice,FreshKid', 'Campbell,Luther', 'Mixx,Mr.', '2LiveCrew', 'Marquis,Brother'

'Rubin,Rick', 'Collins,Phil(I)', 'Albarn,Damon', 'Mixx,Mr.', 'Bambaataa,Afrika', 'Thompson,Ahmir-Khalib', 'Jon,Lil', 'Williams,Pharrell', 'Frantz,Chris(I)'

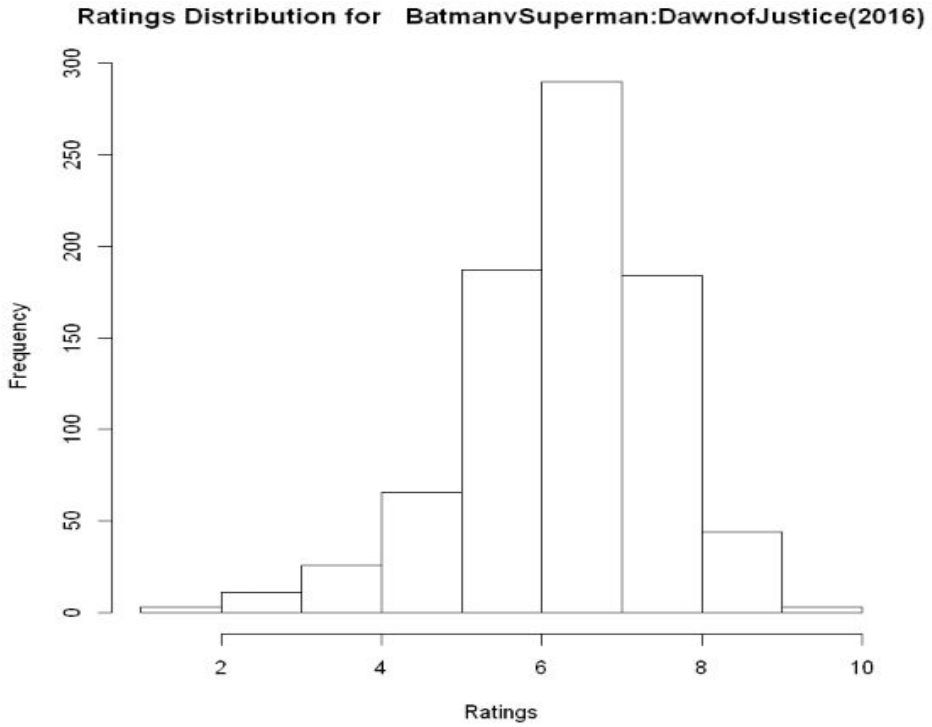
We observe that the more crucial three actors above have acted in almost every movie present in this community (11 out of 12 movies) . Hence the are the building blocks of this community.

As per our results in **8a and 8b both, we observe that the genre of community 24 is “Short” in both the cases.** On verifying with the original movie_genre file, we see that for the movies in community 24 the genre is indeed “Short”. This indicates that both of our metrics performed well for this community.

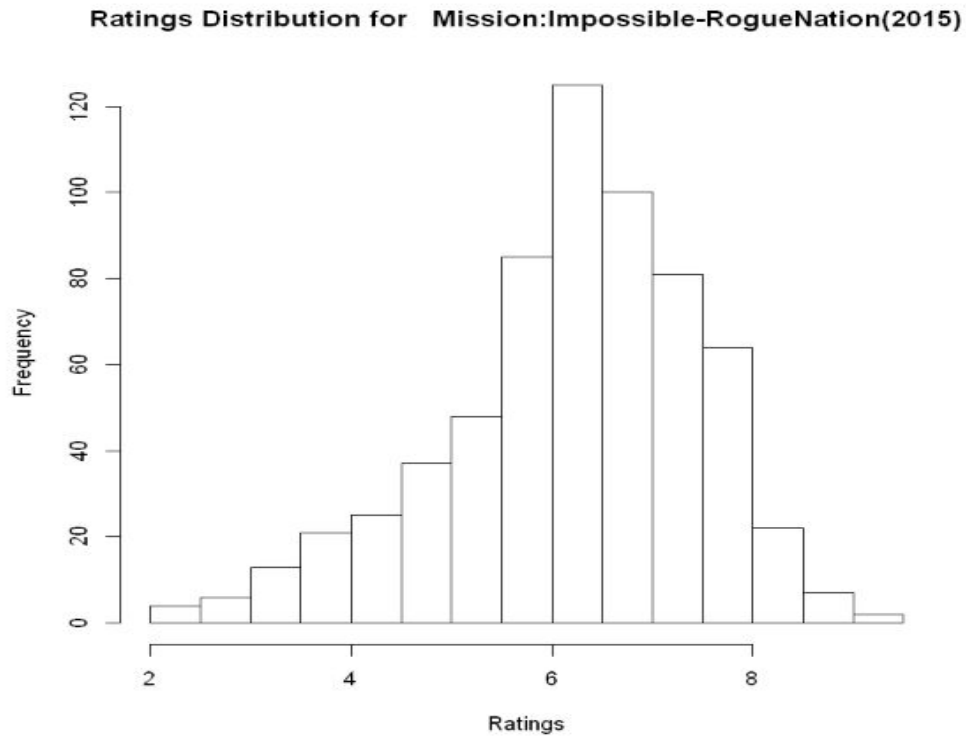
Question 9: For each of the movies listed above, extract it’s neighbors and plot the distribution of the available ratings of the movies in the neighborhood. Is the average rating

of the movies in the neighborhood similar to the rating of the movie whose neighbors have been extracted?

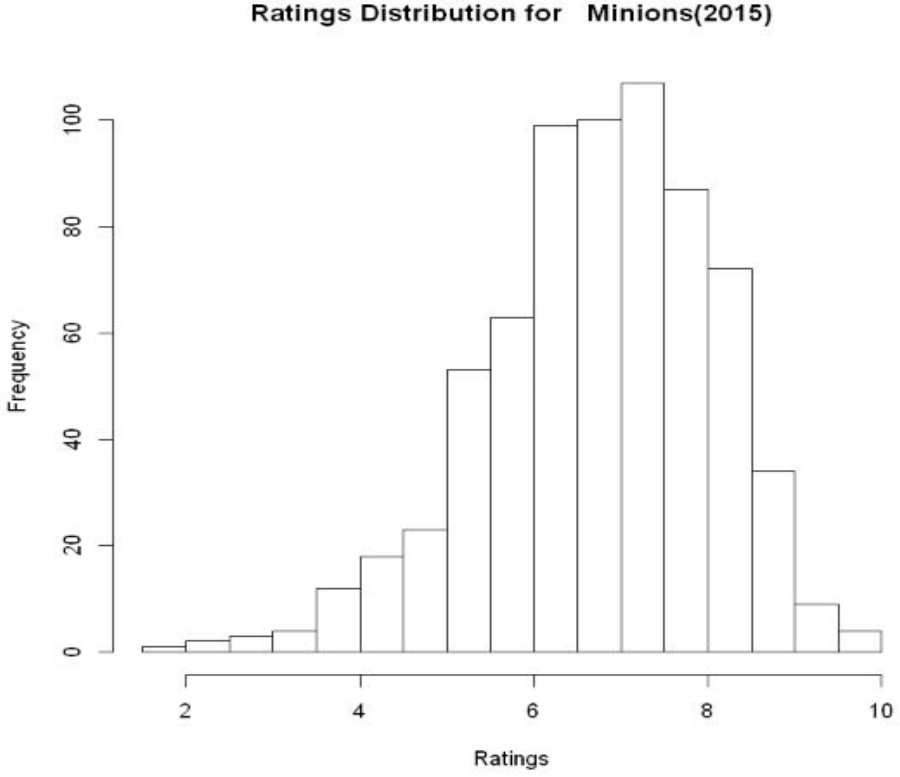
Distribution of ratings for the movies is as follows:



We can see that the ratings vary from 1 to 10 for the neighbours of Batman v Superman: Dawn of Justice (2016), with majority of movies having the rating between 6 and 7 which is closer to the actual rating of the movie.



For the movie, Mission Impossible, we can see that ratings of neighbours vary from 2 to 10 with majority of the movies having the rating between 6 and 6.5 and the average being 6.24. For this movie the average is not closer to the actual rating of the movie ,i.e.,7.4

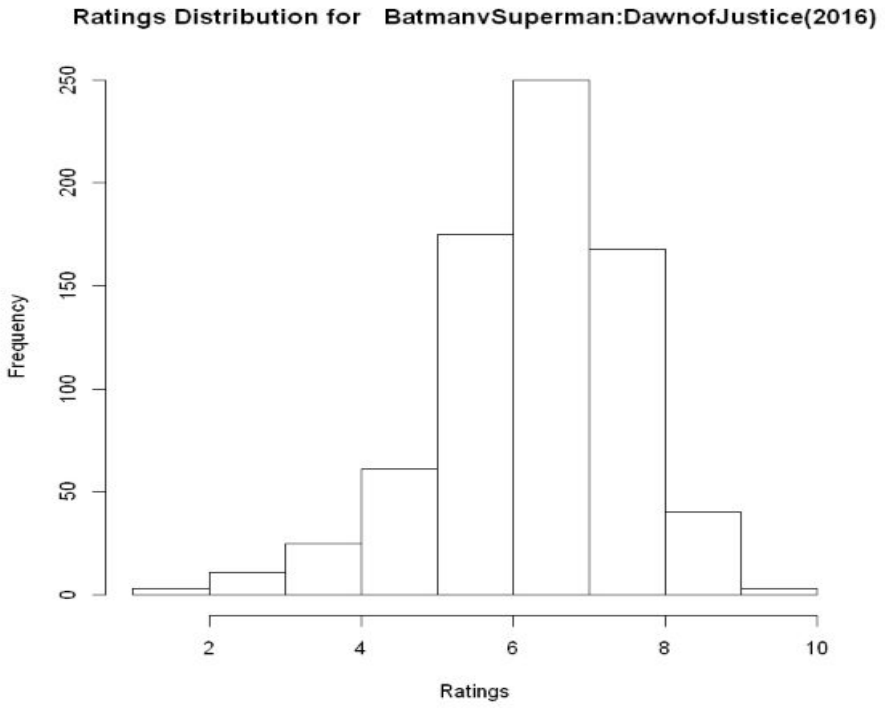


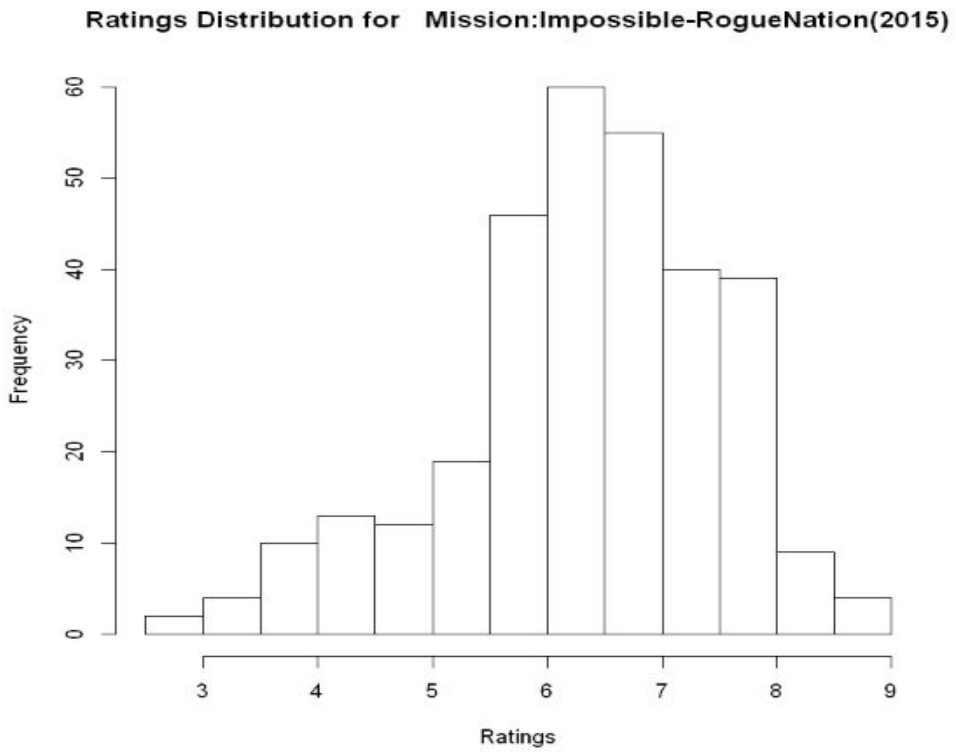
For the movie, minions, majority of the neighbours have the rating between 7 and 7.5 which increase the average. The average is 6.7 which is greater than the actual rating of the movie.

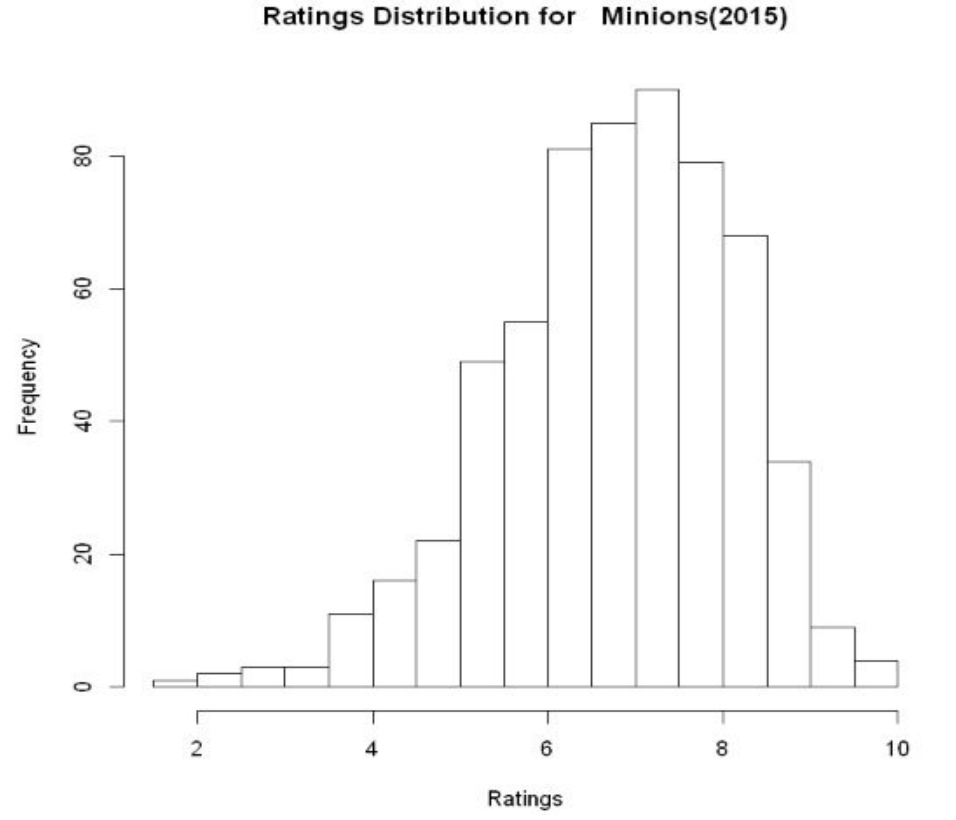
Movie Name	Actual Rating	Average Rating
BatmanvSuperman:DawnofJustice(2016)	6.6	6.3451
Mission:Impossible-RogueNation(2015)	7.4	6.2466
Minions(2015)	6.4	6.7945

From the above figures, we can observe that the average rating of the neighbors is almost closer to the actual rating of the movie.

Question 10: Repeat question 10, but now restrict the neighborhood to consist of movies from the same community. Is there a better match between the average rating of the movies in the restricted neighborhood and the rating of the movie whose neighbors have been extracted.




Ratings Distribution for Mission:Impossible-RogueNation(2015)
Frequency
0
10
20
30
40
50
60
3
4
5
6
7
8
9
Ratings


Ratings Distribution for Minions(2015)
Frequency
0
20
40
60
80
2
4
6
8
10
Ratings

Movie Name	Actual Rating	Average Rating
BatmanvSuperman:DawnofJustice(2016)	6.6	6.325543
Mission:Impossible-RogueNation(2015)	7.4	6.380831
Minions(2015)	6.4	6.815686

The above graph were constructed by considering only those neighbours that belong to the same community. We can observe that for the movie Mission Impossible, the average rating gets a bit closer to the actual rating of the movie but for the other two movies, the average rating does not get closer.

Question 11: For each of the movies listed above, extract it's top 5 neighbors and also report the community membership of the top 5 neighbors. In this question, the sorting is done based on the edge weights.

The top 5 movies and their communities for the three films are:

Top 5 movies	Community
BatmanvSuperman:DawnofJustice(2016)	**2**
Eloise(2015)	2
TheJusticeLeaguePartOne(2017)	2
IntotheStorm(2014)	2
LoveandHonor(2013)	2
ManofSteel(2013)	2

Top 5 movies	Community
Mission:Impossible-RogueNation(2015)	**2**
Fan(2015)	8
Phantom(2015)	8
Suffragette(2015)	8
BreakingtheBank(2014)	8
NowYouSeeMe:TheSecondAct(2016)	2

Top 5 movies	Community
Minions(2015)	2
TheLorax(2012)	2
InsideOut(2015)	2
Up(2009)	2
SurfsUp(2007)	2
DespicableMe2(2013)	2

We can observe that for Minions and Batman VS Superman, all the top 5 neighbours belong to the same communities. But for Mission Impossible, top 4 neighbors belong to community 8 where 5th one belongs to community 2. I believe that community 8 and 2 are closely related.

Question 12: Train a regression model to predict the ratings of movies: for the training set you can pick any subset of movies with available ratings as the target variables; you have to specify the exact feature set that you use to train the regression model and report the root mean squared error (RMSE). Now use this trained model to predict the ratings of the 3 movies listed above (which obviously should not be included in your training data).

In this part of the project, we use two different regression models to predict the ratings of the following three movies:

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

The models we use are- linear regression and random forest regression models.

Feature Set:

We first create a decent feature set to train a regression model. To construct the feature set, we make use of all the text files provided i.e. actor_movies.txt, actress_movies.txt, director_movies.txt, movie_rating.txt and movie_genre.txt. In order to reduce the dataset, we work on only those movies whose ratings are provided in movie_rating.txt. Also, our regression model will not take categorical variables for training. Thus, to reduce the complexity of the model as well as the time taken, we construct and use numerical features only. The features we use are:

- Movie names converted to integers
- Number of actors and actress in the movie (for sake of simplicity, we will use the word ‘actor’ henceforth to denote actors as well as actress)
- Number of movies directed by the director
- Average Rating of the genre of the movie

- Top 5 PageRank scores of the actors in the movie

Throughout this part, we work on a subset of data which makes sure that the movie in our feature set is present in movie_rating.txt, movie_genre.txt, director_movies.txt as well as in the merged file of actor_movies.txt and actress_movies.txt. Furthermore, we consider only those movies which have only one director. Also, since we get the top 5 page ranks of the actors in the movie, we consider only those movies which have at least 5 actors as the cast. There is no further pruning on the dataset.

Regression Model(s):
We then construct a linear regression and Random forest model to train this feature set. The random forest regression model was created with the following options:
n_estimators = 15, max_features=9, max_depth=15

The predictions obtained are as follows:

Movies	Actual Rating	Predictions from Linear Regression	Predictions from Random Forest
Batman v Superman: Dawn of Justice (2016)	6.6	6.48629083	6.78313808
Mission: Impossible - Rogue Nation (2015)	7.4	6.37754583	6.66292151
Minions (2015)	6.4	6.1447285	6.19749686

RMSE obtained from Linear Regression Model = 0.6119
RMSE obtained from Random Forest Model = 0.4538

Inference and Analysis:
In this model, we take into account most of the things related with the movie:

- The number of actors which denoted how big the movie cast is and therefore how big the movie is
- The number of movies directed by the director of the movie which gives the credibility of the director in terms of movies directed
- The average rating of the genre of the movie which gives an idea of how a particular genre is accepted by IMDb and the population
- The top 5 PageRank of the actors of a movie which incorporates the popularity of the movie actors

All these numerical features, along with the movie name itself (converted to a number) help build a set which describe a movie (from all the available data). We also observe that the Random Forest model gives slightly better results than the linear regression. This means that our data has more complex dependency than linear. This is caught by the random forest model and the best predictions were obtained by setting the options of the model as stated above.

Question 13: Create a bipartite graph following the procedure described above. Determine and justify a metric for assigning a weight to each actor. Then, predict the ratings of the 3 movies using the weights of the actors in the bipartite graph. Report the RMSE. Is this rating mechanism better than the one in question 12? Justify your answer.

In this part of the project, we predict the ratings using the weights of the actors in the bipartite graph. One set of vertices is the actors and the other is the movies whose ratings are available. For the actors, we consider all the unique actors in actror_movies.txt and actress_movies.txt. Let's call the vertex set of actors as V1 and the vertex set of movies as V2. Count(V1) = 3349871 and Count(V2) = 348545. We then connect edges from V1 to V2 based on the movies each actor has acted in. To assign a metric to each actor, we take the average of the movie ratings in which the actor has played a role and assign that as a weight to each actor.

To predict the ratings of the three movies, we now make use of this weight as a feature in our regression model. To predict the ratings, we consider the following three approaches:

- Take the average of all the actor weights. This will be considered as one feature of a movie.
- Take the top 5 actor weights of the movie. These will account for 5 features of the movie.
- Take the average of the top 5 actor weights of the movie. Again, this will be considered as one feature of a movie.

We train both, linear regression model and a random forest model with the above features to predict the movie ratings. Additionally, we append the above features to our original feature set of Q12 and observe the results. Thus, the predictions were made for the following:

- Average of all actor weights (only new feature) + Linear Regression model
- Average of all actor weights (only new feature) + Random Forest model
- Average of all actor weights + old features + Linear Regression model
- Average of all actor weights + old features + Random Forest model
- Top 5 actor weights (only new feature) + Linear Regression model
- Top 5 actor weights (only new feature) + Random Forest model
- Top 5 actor weights + old features + Linear Regression model

- Top 5 actor weights + old features + Random Forest model
- Average of top 5 actor weights (only new feature) + Linear Regression model
- Average of top 5 actor weights (only new feature) + Random Forest model
- Average of top 5 actor weights + old features + Linear Regression model
- Average of top 5 actor weights + old features + Random Forest model

The prediction results and RMSE obtained are as follows. For sake of simplicity, we denote Batman v Superman: Dawn of Justice (2016) as Movie #1, Mission: Impossible - Rogue Nation (2015) as Movie #2 and Minions (2015) as Movie #3.

Movie	Model	Feature	Old Features	Prediction	RMSE
1	Linear Reg.	Avg. of all actor weights	No	4.82325287	1.8032
2	Linear Reg.	Avg. of all actor weights	No	4.83612051	
3	Linear Reg.	Avg. of all actor weights	No	6.55607221	
1	Random Forest	Avg. of all actor weights	No	5.4822908	1.2906
2	Random Forest	Avg. of all actor weights	No	5.4822908	
3	Random Forest	Avg. of all actor weights	No	6.66472592	
1	Linear Reg.	Avg. of all actor weights	Yes	4.20040261	2.2688
2	Linear Reg.	Avg. of all actor weights	Yes	4.28800963	
3	Linear Reg.	Avg. of all actor weights	Yes	6.39874849	
1	Random Forest	Avg. of all actor weights	Yes	5.0417169	1.6118
2	Random Forest	Avg. of all actor weights	Yes	5.08410161	

3	Random Forest	Avg. of all actor weights	Yes	6.44778048	
1	Linear Reg.	Top 5 actor weights	No	6.07236873	1.0502
2	Linear Reg.	Top 5 actor weights	No	5.77575915	
3	Linear Reg.	Top 5 actor weights	No	7.02627503	
1	Random Forest	Top 5 actor weights	No	5.65123327	1.1953
2	Random Forest	Top 5 actor weights	No	5.65123327	
3	Random Forest	Top 5 actor weights	No	6.97286964	
1	Linear Reg.	Top 5 actor weights	Yes	4.7994779	1.8301
2	Linear Reg.	Top 5 actor weights	Yes	4.86540381	
3	Linear Reg.	Top 5 actor weights	Yes	7.01821334	
1	Random Forest	Top 5 actor weights	Yes	5.31611125	1.5638
2	Random Forest	Top 5 actor weights	Yes	5.05889043	
3	Random Forest	Top 5 actor weights	Yes	6.85560677	
1	Linear Reg.	Avg. of top 5 actor weights	No	6.207673	**0.8177**
2	Linear Reg.	Avg. of top 5 actor weights	No	6.11994006	
3	Linear Reg.	Avg. of top 5 actor weights	No	6.86238725	

1	Random Forest	Avg. of top 5 actor weights	No	6.05488066	**0.8871**
2	Random Forest	Avg. of top 5 actor weights	No	6.05488066	
3	Random Forest	Avg. of top 5 actor weights	No	6.90468582	
1	Linear Reg.	Avg. of top 5 actor weights	Yes	4.4467801	1.9585
2	Linear Reg.	Avg. of top 5 actor weights	Yes	4.80391842	
3	Linear Reg.	Avg. of top 5 actor weights	Yes	6.76301851	
1	Random Forest	Avg. of top 5 actor weights	Yes	5.26006749	1.5248
2	Random Forest	Avg. of top 5 actor weights	Yes	5.19839031	
3	Random Forest	Avg. of top 5 actor weights	Yes	6.9772335	

Observations and Analysis:

- Of all the 36 combinations we tried, we see that the feature of average of top 5 actor weights gives the best result (i.e. the lowest RMSE of all). However, this is still not better than the RMSE we obtained in Q12.
- In most of the cases, we see that the Random Forest model performs better than the Linear Regression model (except for a couple of times). This is intuitive because we are working on the same movies as that of Q12. In Q12, we had observed that the Random Forest model gives better predictions than the Linear Regression model. This justifies the observations in Q13 as well.
- The predictions and RMSE observed in Q13 is better when we consider only the new feature(s).
- So why is the result of Q13 not as good as Q12? We will answer this question in 2 parts:
    - Why is the result of Q13 while considering only the new feature not as good as Q12? This is mainly because in Q13, we are considering only the weights of the actors. This means we are considering only the average ratings of the movies in which the actors have worked. Thus, an actor will have good rating if all the

movies he/she has acted in has decently good rating. Some actors who have acted in many movies are likely to get good ratings than the actors who have acted in relatively less number of movies. This may not give us an accurate result. On the other hand, in our model of Q12, we consider many things associated with a movie. They are: the number of actors (which gives us an idea of how big the movie is), the movies directed by the director (which gives the credibility of the director), the average rating of the genre of the movie (which says on an average how is a particular genre rated by IMDb) and the top 5 page rank of actors (gives us credibility of the actors). All these features together help to constitute a good model.

    - Why is the result of Q13 while considering the old features not as good as while considering only the new features? Because in the old feature set, we have already considered the ratings of the movies (in determining the average rating of the genre) as well as the credibility of the actors in terms of PageRank. When we add our new feature to this set, we are doing no improvement. That's because, in the new feature, we are indirectly including the movie rating (by assigning weights to the actors). This we have already done in the old feature set and moreover we have already found the credibility of the actors in terms of PageRank. Including the same features in some other form does no good. Hence, the reason.

- Comparison of different types of features used in Q13: We see that the feature of 'Average of top 5 actor weights of a movie' gives the best prediction and RMSE (lowest) of all features used. The average of all actors' weights gives the worst prediction and RMSE (highest). This is because some movies may have a lot of actors and some very less. Movie #1 has 178 actors, Movie #2 has 132 actors and Movie #3 has 25 actors. And as we see from the table, the prediction is more skewed for Movie #1 and #2. This is because, since these movies contain a lot of actors most of them would be supporting and not in the main lead. The weights of these actors are definite to be very less as compared to the actors in the lead. It is because of these actors that the feature gets distorted and hence the prediction is not good. This is not observed in case of Movie #3 (Minions) because mostly all of the characters are animated (the actor count is very less). This problem can be resolved by taking the top 5 weights of the actors wherein we can ignore all the non-influential actors. The results in the table above stand as a witness to this. Again, if there is a lot of deviation in the weights of these 5 actors, the prediction will not be that good. This can be further improved by taking the average of these top 5 weights and assigning this as one feature to the movie. Therefore, the results obtained by using this feature is better than by using any other feature.
- Thus, our observations of Q13 are justified by all of the above reasons.

Conclusions

In this project, we have successfully achieved the following:

1. We performed preprocessing on huge files from imdb movie dataset.
2. We identified actor pairings, and understood how some actors always prefer tow ork with some other actors, forming successful relationships and movies.
3. We utilized google's PageRank Algorithm to identify the top 10 actors/ actresses in the network.
4. We created an undirected weighted network of movies, which are linked to each other on basis of common actors who have worked in both the movies. We created a degree distribution plot and inferred that it follows power-law distribution.
5. We applied FastGreedy Community Detection algorithm on the movie network. We randomly picked up 10 communities and analyzed genre distribution plots of these communities.
6. We compared two metrics , frequency count and a modified score to analyze how well they capture the dominance of a genre within a community.
7. We created a bipartite graph for a small community and analyzed relationships between the actors and the movies in these community.
8. We extracted neighbours of 3 specific movies, and analyzed the ratings of the movies in the neighborhood.
9. We compared the previous neighbourhood analysis with the neighbood graph containing only members of the community to which our movies belong to. We even extracted top five neighbors based on edge weights.
10. Also , we predicted ratings of movies using different regression models like linear regression and random forest.
11. We constructed a bipartite graph to predict the rantings of the 3 movies.