INTERNSHIP TASK

Topic: Recommendation System

Name:	ANKIT KUMAR SINGH

Currently Working as Data Science Intern in Navodita Infotech. Intern ID: N242507502.

This is the documentation report of Recommendation System.

>Abstract Idea

We have three datasets of movies,ratings and tags. And now we are to create a movie recommendation system using all these datasets which contains various features like Movie title, genres, ratings, tags, timestamp among others. Thus in this project we created several functions for recommending movies in general as well as for any specific user.

>Introduction

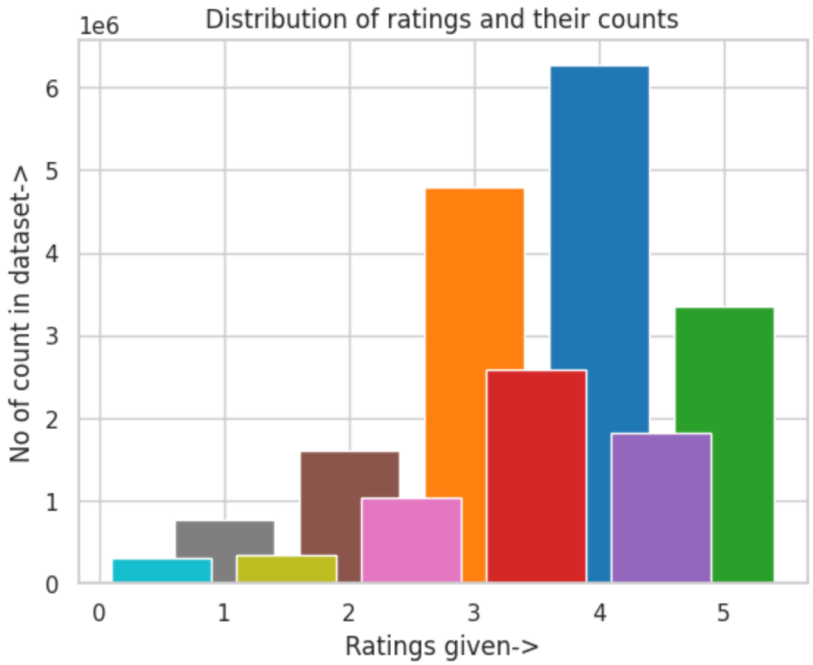
From the dataset provided we have data about movies and their respective genres, similarly we have a dataset with users giving rating to a movie. Our idea works on two principles-

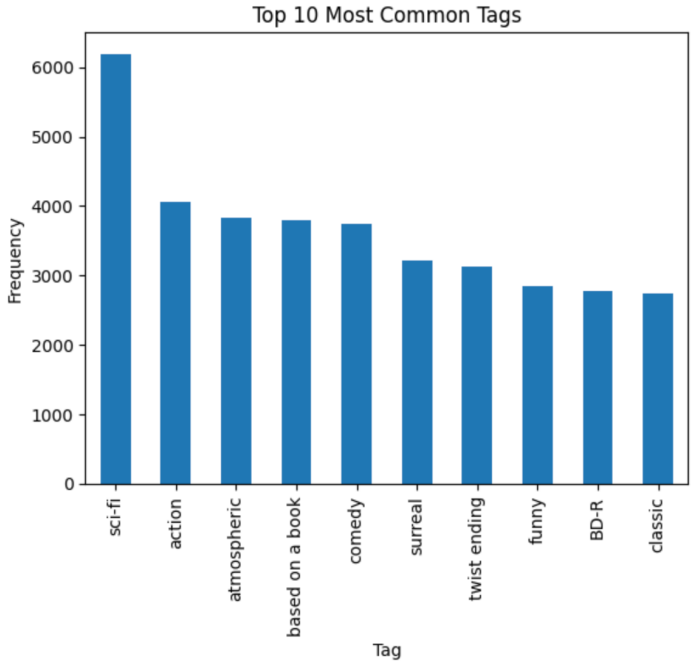
1. A user will see similar type of movies and will give high rating to similar movies
2. Similar movies based on genres are good recommendations.

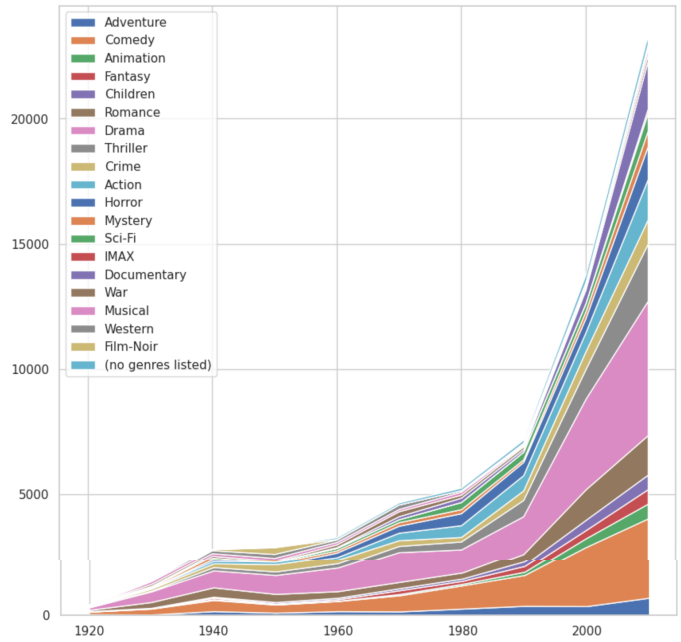
>Dataset Descriptions

We have four datasets in total namely - movies,ratings,tags and links.Movies dataset contains title and genres wrt movie Id. Similarly the ratings dataset contains ratings given by which user to which movieId at which timestamp. And the tags dataset also contains tags given by any user to any movie.The final dataset contains relation between imdbId and tmdbId. The last dataset is of not so much use because in other datasets we don't use imdbId as well as tmdbId.

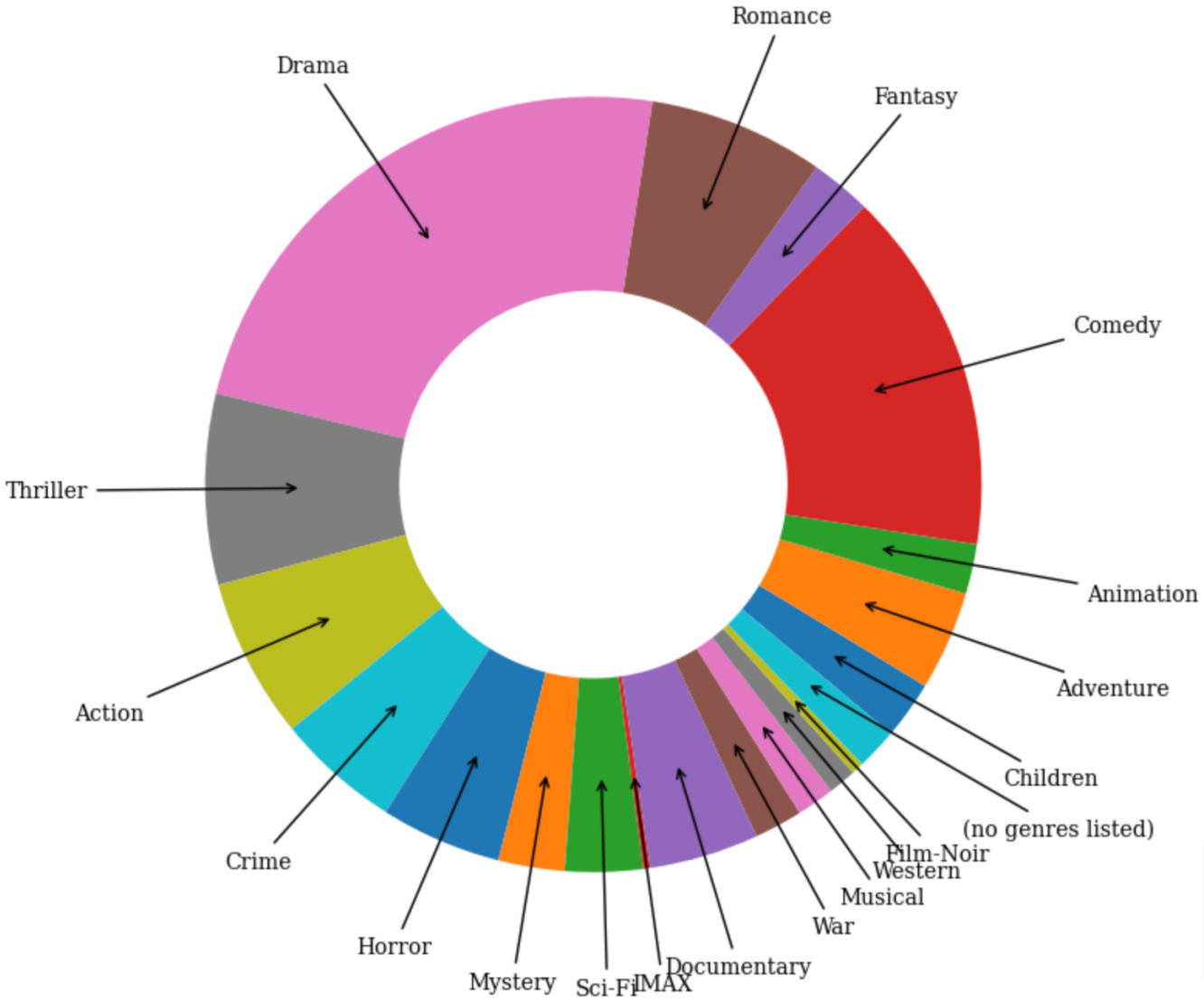
>Dataset Visualization







Pie chart showing distribution of different genres in dataset



>Pre Processing

- Movies Dataset- The movies dataset contains no null values. Now this dataset contains titles with a release year. So we created a function to extract that year from the title. Now we also have genres in the dataset. So now we created a function to make a genres list out of it and further create features = genres and making value 1 if present in list otherwise 0. While preprocessing we also noticed that some movies have two different movieId so we give their genres equal to the Union set of both movies.

```
Movies with 2 occurrence in dataset-
Darling (2007) with tags-
       movieId           title genres
18778    93279  Darling (2007)  Drama
28105   130062  Darling (2007)  Drama

Men with Guns (1997) with tags-
      movieId                 title        genres
1716     1788  Men with Guns (1997)  Action|Drama
9154    26982  Men with Guns (1997)         Drama

The Dream Team (2012) with tags-
       movieId                  title                 genres
30645   138868  The Dream Team (2012)  Comedy|Drama|Romance
30646   138870  The Dream Team (2012)           Documentary
```

Final movies Dataset-

	movieId	title	genres_list	Children	Adventure	Animation	Comedy	Fantasy	Romance	Drama	...	Horror	Mystery	Sci-Fi	IMAX	Documentary	War
0	1	Toy Story (1995)	[Children, Adventure, Animation, Comedy, Fantasy]	1	1	1	1	1	0	0	...	0	0	0	0	0	0
1	2	Jumanji (1995)	[Children, Adventure, Fantasy]	1	1	0	0	1	0	0	...	0	0	0	0	0	0
2	3	Grumpier Old Men (1995)	[Romance, Comedy]	0	0	0	1	0	1	0	...	0	0	0	0	0	0

- Rating Dataset - The ratings dataset contains basically four major features namely userId, movieId, rating and timestamp. Now we would need to obtain some data from this dataset, like avg_rating for a movie as well as for a userId. Now for calculating avg_rating for a movie we basically traverse through the dataset, and then store their respective rating value by any user in a list which is the value of movieId key of a dictionary. Now their avg_rating can be calculated using just the sum()/len() of all the movies. Similarly we did for userId and calculated avg_rating of every userId.

	movieId	avg_rating	Rating_no
0	1	3.895	60424
1	2	3.221	23950
2	3	3.180	15267
3	4	2.880	2935
4	5	3.081	14769
...	...	...	...
34203	151697	3.000	1
34204	151701	4.000	1
34205	151703	5.000	1
34206	151709	3.500	1
34207	151711	4.000	1

	userId	avg_rating	Rating_no
0	1	3.500	3
1	2	4.125	4
2	3	4.250	4
3	4	3.730	183
4	5	3.460	25
...	...	...	...
247748	247749	3.600	5
247749	247750	3.800	70
247750	247751	4.211	190
247751	247752	2.727	33
247752	247753	4.159	22

247753 rows × 3 columns

But now we came across a problem, there were several movies which had null ratings because no userId gave ratings to them. Thus we used the concept of weighted rating both for movieId's and userId's.

$$\text{Weighted Rating (WR)} = \left(\frac{v}{v+m} . R\right) + \left(\frac{m}{v+m} . C\right)$$

In Weighted rating we basically normalize the rating according to the no of ratings given to the movie. Thus now we calculated the weighted rating for both datasets. Further the movies which had null ratings were given a rating of 0.1. This will further help us when we predict the best movies for a user or genre.
So the final dataset obtained -

	movieId	avg_rating	Rating_no	Weighted rating
0	1	3.895	60424	3.881
1	2	3.221	23950	3.214
2	3	3.180	15267	3.172
3	4	2.880	2935	2.924
4	5	3.081	14769	3.079
...	...	...	...	...
34203	151697	3.000	1	3.052
34204	151701	4.000	1	3.053
34205	151703	5.000	1	3.054
34206	151709	3.500	1	3.052
34207	151711	4.000	1	3.053

- Tags dataset - In this dataset we basically have four features namely userId, movieId, tag and timestamp. Now we have multiple tags for a single movieId as well as for a single userId. So now we need to follow the same procedure which we followed previously in the ratings dataset. So now we stored every tag for userId as well as for every movieId and then we created two datasets from

them. We further created a concatenated string for all the tags for a movie as it would help us when we calculate the similarity between any two tags. Now we also converted timestamps into human data.

Final dataset created -

	movieId	tags	Tag no.
0	1	[é˜®ä¸€é¸£, fun, imdb top 250, witty, comedy, ...	110
1	2	[Fantasy, children, comedy, bad cgi, adapted f...	41
2	3	[grun running, Jack Lemmon, Walter Matthau, mo...	15
3	4	[chick flick, girl movie, characters, revenge,...	5
4	5	[humorous, Touching, Fantasy, watched under du...	18

>Methodologies

Now we will create a hybrid recommendation system with various methods like-

- Basic Recommendation based on ratings of movies
- Content Based Recommendation system
- Based on Collaborative Filtering
- Based on Clustering of movies and users

>Basic Recommendation System

This is a very basic recommendation system which extracts results based on the dataset we already have. Now this dataset is best suited when a new user signs up on our system or a user wants to see movies based on a particular order (probably rating of the movies).

Now in basic recommendation we created three functions each taking different input namely, movie title, genre as well as userId.

- Based on Movie Name: We take input of movie name and then we calculate the closest movie name wrt input using a python module **difflib**.This module calculates least distance between two movie titles by finding the largest contiguous matching subsequence (not containing junk elements). Now after finding 3 closest movies, we basically find all the genres they belong to and then we find the top 2 genres in every matching movie. And then we find the highest rated movie whose genre set is a subset with maximum length of all_common_genres in all the closest movies. Thus we return the top 10 movies having the maximum genre subset of all_common_genres.The results obtained for are-

```
Top Movie Reccomendations for movie Cars

Cars (2006)
Cars 2 (2011)
Toy Story (1995)
Goofy Movie, A (1995)
Aladdin (1992)
Space Jam (1996)
Oliver & Company (1988)
Wallace & Gromit: A Close Shave (1995)
Aladdin and the King of Thieves (1996)
Wallace & Gromit: The Wrong Trousers (1993)
```

```
Top Movie Reccomendations for movie Batman

Batman (1989)
Batman (1966)
Rumble in the Bronx (Hont faan kui) (1995)
Batman Forever (1995)
Hackers (1995)
Getaway, The (1994)
Hard Target (1993)
North Star (a.k.a. Tashunga) (1995)
Black Mask (Hak hap) (1996)
Breaker! Breaker! (1977)
```

Analysis: We can see that for Cars it recommended movies like- Toy Story, Space Jam and Aladdin among others which are of the same genre as Cars namely Children, Animation and Drama. Similar things can also be analyzed for Batman as Batman is of the genre Crime, Thriller, Action which synchronizes with genres of other movies.

- Based on Preferred Genres: We created a function in which we take two genres and then find the movies which are of preferred genres and sorted according to ratings. The results obtained were -

```
Top 10 Movies in genres Crime Drama according to ratings-

Shawshank Redemption, The (1994) with rating 4.424
Godfather, The (1972) with rating 4.328
Godfather: Part II, The (1974) with rating 4.232
Fight Club (1999) with rating 4.21
Dark Knight, The (2008) with rating 4.164
Pulp Fiction (1994) with rating 4.149
City of God (Cidade de Deus) (2002) with rating 4.144
Goodfellas (1990) with rating 4.144
Inception (2010) with rating 4.119
American History X (1998) with rating 4.113
```

Analysis: We can see that if we give our preferred genres as "Crime" and "Drama", then we obtain results as Godfather, Dark Knight, Fight Club which are of preferred genres.

- Based on User Id: We created a function in which we take input of userId and then we calculate the movies the user has rated in our dataset. Now we take these movies and obtain the common genres from these movies. Now we take these genres and then obtain the best rated movie from the

previous function and not seen by the user and return the movies. The results obtained were -

```
Your Top 3 favourite genres are  Drama, Comedy, Adventure

Movies for you in genre Drama according to ratings-

Waiting to Exhale (1995) with rating 2.924
American President, The (1995) with rating 3.633
Nixon (1995) with rating 3.382
Casino (1995) with rating 3.749
Sense and Sensibility (1995) with rating 3.921
Money Train (1995) with rating 2.927
Copycat (1995) with rating 3.281
Powder (1995) with rating 3.17
Leaving Las Vegas (1995) with rating 3.647
Othello (1995) with rating 3.477

Movies for you in genre Comedy according to ratings-

Toy Story (1995) with rating 3.881
Grumpier Old Men (1995) with rating 3.172
Waiting to Exhale (1995) with rating 2.924
Father of the Bride Part II (1995) with rating 3.079
Sabrina (1995) with rating 3.353
American President, The (1995) with rating 3.633
Dracula: Dead and Loving It (1995) with rating 2.742
Four Rooms (1995) with rating 3.35
Ace Ventura: When Nature Calls (1995) with rating 2.641
Money Train (1995) with rating 2.927

Movies for you in genre Adventure according to ratings-

Toy Story (1995) with rating 3.881
Jumanji (1995) with rating 3.214
Tom and Huck (1995) with rating 3.104
GoldenEye (1995) with rating 3.425
Balto (1995) with rating 3.205
Cutthroat Island (1995) with rating 2.808
City of Lost Children, The (Cité des enfants perdus, La)
```

>Content Based Recommendation System

Now the previous method we created was very trivial, and the only metric we used for similarity between movies were genres but now we also use tags as another factor. Now there are no other features to consider in this like cast,crew, money_earned etc. Now for using tags we create a string which contains all the tags separated by a space. Now for calculating cosine similarity between two strings we used another library, SentenceTransformer . Now this library takes a list of all tags

with a preferred tag string and gives a list with cosine similarity between every string . Then we store the cosine similarity score with index and then sort the list, and take the top 10 movies with the highest similar tag with the input tag. A problem we encountered when creating this method was, as the tags data was very large it was taking very much time in calculating the cosine similarity between every subset. Thus we were able to train our model for cosine similarity on limited data points. The results obtained are -

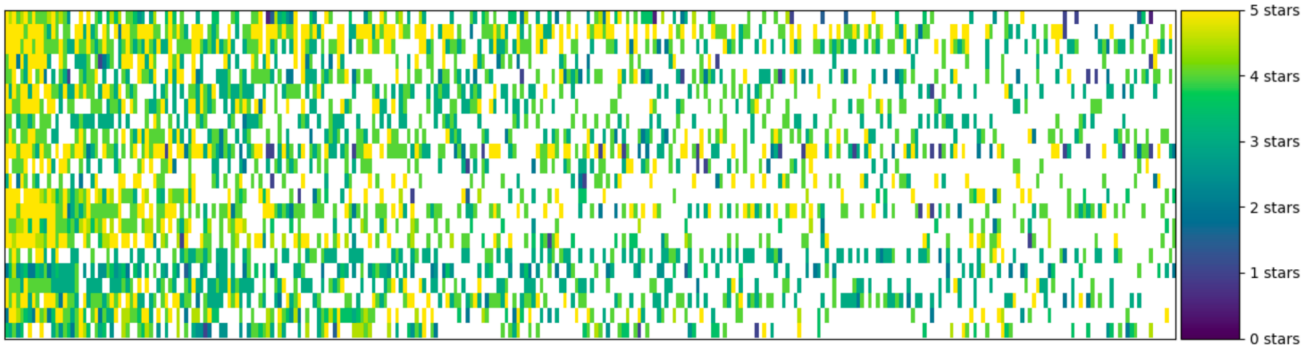
```
Top 10 Reccomendations for the tags -  war action fighting guns is
Hunted, The (1995)
Before the Rain (Pred dozhdot) (1994)
Brother Minister: The Assassination of Malcolm X (1994)
National Lampoon's Senior Trip (1995)
Red Firecracker, Green Firecracker (Pao Da Shuang Deng) (1994)
Heaven & Earth (1993)
Cops and Robbersons (1994)
Life with Mikey (1993)
Lightning Jack (1994)
Geronimo: An American Legend (1993)
```

Analysis: Now we can see that as our input tags were “war action fighting guns” the results obtained were also of action types, like Before the Rain, Lighting Jack etc. Though there are some weird results like Senior trip which is a comedy/drama movie. And this limitation is because of the limited data we used.

>Clustering Based Recommendation

Now we know that users watch movies and we already assumed that it is highly likely that a user will watch a similar type of movie and give higher rating to the same type of movie. And then we recommend those movies which belong to the same clusters using the KMeans. Now we have 20 genres so we create 20 clusters for this data, and then we give similar movies in its clusters.

Now for better visualization i created a heat map with movies title vs userId-



Now for every usedId we take the cluster it is in and then recommend top 10 rated movies in that cluster -

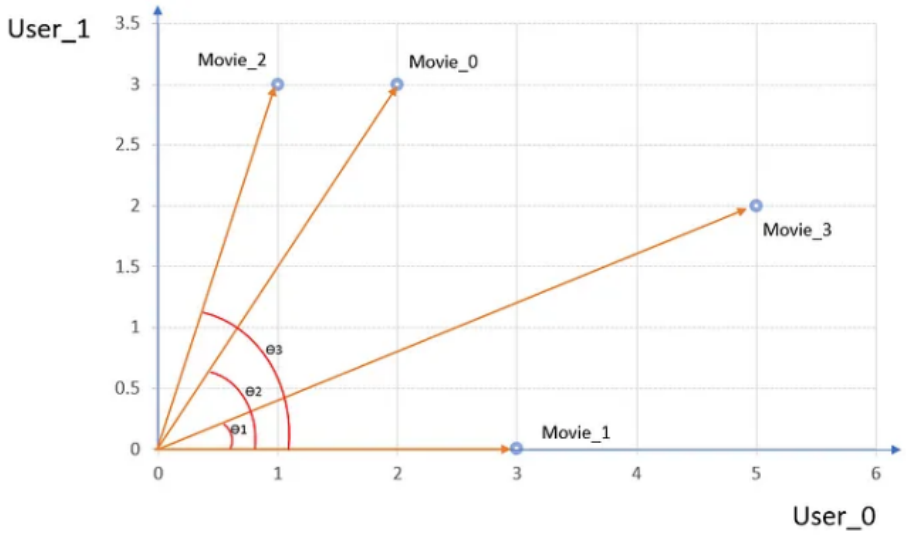
Usual Suspects, The (1995)	4.821429
Schindler's List (1993)	4.607143
Killing Fields, The (1984)	4.571429
Deer Hunter, The (1978)	4.500000
Strictly Ballroom (1992)	4.500000
Maltese Falcon, The (1941)	4.428571
As Good as It Gets (1997)	4.428571
Chasing Amy (1997)	4.416667
Big Lebowski, The (1998)	4.416667
Sting, The (1973)	4.400000
Shallow Grave (1994)	4.400000

Analysis: We can see that all the movies belonging to the same clusters are of similar types, like Schindler's List, The Usual Suspects, The Deer Hunted etc.

>Collaborative Filtering

We apply the method of item-based collaborative filtering here. For this, a dataset of movie_titles, userIDs and the corresponding rating that the user gives to a movie are required.

Item based collaborative filtering recommends a specific user, already existing in our dataset, some more new movies which he/she has not already seen.
Assuming movie_titles as row_indices and userIDs as column names, every row becomes a vector for every movie_title.
These vectors are used to find the similarity between the movies based on the cosine similarity metric.



Here , The vectors for a two-user dataset have been depicted. The NaN values in the dataset are replaced by zeros and vectors are plotted. The smallest angle between two movie vectors

represents high cosine value i.e high similarity value. This implies more similarity between those movies.

First we make a dataset with movie_titles as indices, userIds as columns and cell values as ratings.
The dataset looks like:-

userId	4	5	6	7	8	9	10	11	12	13	...	247744	247745	247746	247747	247748	247749	247750	247751	247752	247753
title																					
(500) Days of Summer (2009)	NaN	NaN	NaN	NaN	NaN	3.5	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10 Things I Hate About You (1999)	NaN	NaN	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
101 Dalmatians (1996)	NaN	NaN	3.5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
12 Angry Men (1957)	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4.5
8 Mile (2002)	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Walk in the Clouds, A (1995)	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
World Is Not Enough, The (1999)	NaN	NaN	4.0	NaN	3.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN
X-Men (2000)	NaN	NaN	NaN	NaN	NaN	NaN	4.0	3.5	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4.5	NaN	NaN
You Can Count on Me (2000)	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Zoolander (2001)	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

After filling NaN values with zeros:-

userId	4	5	6	7	8	9	10	11	12	13	...	247744	247745	247746	247747	247748	247749	247750	247751	247752	247753
title																					
(500) Days of Summer (2009)	0.0	0.0	0.0	0.0	0.0	3.5	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10 Things I Hate About You (1999)	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
101 Dalmatians (1996)	0.0	0.0	3.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12 Angry Men (1957)	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.5
8 Mile (2002)	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Walk in the Clouds, A (1995)	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
World Is Not Enough, The (1999)	0.0	0.0	4.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0
X-Men (2000)	0.0	0.0	0.0	0.0	0.0	0.0	4.0	3.5	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.5	0.0	0.0
You Can Count on Me (2000)	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Zoolander (2001)	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Thus, based on the cosine similarity, we find k nearest neighbors(Here,movies) using sklearn's Nearestneighbors() library function. Furthermore, we remove the movie itself and it's corresponding distance from the list of nearest movies.
The nearest neighbors output is as follows:-

```
The Nearest Movies to  Gone Girl (2014)  are:  ['Whiplash (2014)', 'Interstellar (2014)', 'Forrest Gump (1994)', 'Seven (a.k.a. Se7en) (1995)']
The Distance from  Gone Girl (2014)  are:  [0.5363246420861708, 0.7144971438323768, 0.8246425560905208, 0.8278557947469614]
Indices of Nearest Movies to  Gone Girl (2014)  are:  [22, 13, 8, 19]
```

Now, we update all the NaN values in the dataset. This is done by using the below formula:-

$R(m, u) = \{\sum_{j} S(m, j)R(j, u)\} / \sum_{j} S(m, j)$.

Where :-

R(m, u): the rating for movie m by user u

S(m, j): the similarity between movie m and movie j

$j \in J$ where J is the set of similar movies to movie m.

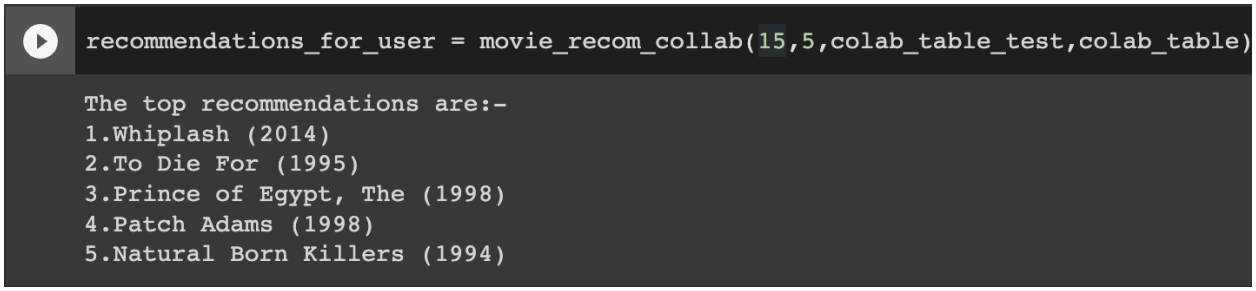
The R(j,i) values are used from the zero filled dataset. And the values are finally updated into the NaN valued dataset.

userId	2	3	4	5	11	12	13	14	15	16	...	247737	247738	247739	247741	247742	247744	247747	247750	247751	247753
title																					
Adventures of Priscilla, Queen of the Desert, The (1994)	0.0	1.041335	4.000000	0.0	0.0	4.000000	0.0	0.0	3.500000	0.0	...	4.000000	0.0	0.0	0.0	0.0	0.781001	0.0	0.0	0.0	0.0
At First Sight (1999)	0.0	4.000000	3.115633	0.0	0.0	1.691197	0.0	0.0	1.479797	0.0	...	1.691197	0.0	0.0	0.0	0.0	3.000000	0.0	0.0	0.0	0.0
Barcelona (1994)	0.0	1.041335	4.000000	0.0	0.0	1.691197	0.0	0.0	1.479797	0.0	...	1.691197	0.0	0.0	0.0	0.0	0.781001	0.0	0.0	0.0	0.0
Beautiful Girls (1996)	0.0	1.041335	5.000000	0.0	0.0	1.691197	0.0	0.0	1.479797	0.0	...	1.691197	0.0	0.0	0.0	0.0	0.781001	0.0	0.0	0.0	0.0
Bottle Rocket (1996)	0.0	1.041335	4.000000	0.0	0.0	1.691197	0.0	0.0	1.479797	0.0	...	1.691197	0.0	0.0	0.0	0.0	0.781001	0.0	0.0	0.0	0.0
Casino (1995)	0.0	1.041335	4.000000	0.0	0.0	1.691197	0.0	3.0	1.479797	0.0	...	1.691197	4.0	0.0	4.5	0.0	0.781001	0.0	0.0	0.0	0.0
Clueless (1995)	0.0	1.041335	4.000000	0.0	0.0	4.000000	0.0	3.0	5.000000	0.0	...	5.000000	0.0	0.0	0.0	0.0	0.781001	0.0	0.0	0.0	0.0
Dave (1993)	0.0	1.041335	4.000000	0.0	0.0	1.691197	0.0	0.0	3.500000	0.0	...	1.691197	3.0	0.0	0.0	0.0	0.781001	0.0	0.0	0.0	0.0
Forrest Gump (1994)	0.0	4.000000	2.000000	0.0	0.0	1.691197	0.0	5.0	4.500000	0.0	...	1.691197	3.0	0.0	0.0	0.0	0.781001	5.0	0.0	4.5	5.0
Four Weddings and a Funeral (1994)	0.0	1.041335	4.000000	0.0	0.0	1.691197	0.0	1.0	4.500000	0.0	...	1.691197	0.0	0.0	0.0	0.0	0.781001	0.0	0.0	0.0	0.0
Fugitive, The (1993)	0.0	1.041335	4.000000	0.0	0.0	1.691197	0.0	4.0	1.479797	0.0	...	1.691197	4.0	0.0	0.0	4.0	0.781001	0.0	4.0	0.0	0.0
Gone Girl (2014)	4.0	1.041335	3.115633	4.5	0.0	1.691197	0.0	0.0	1.479797	0.0	...	1.691197	0.0	0.0	0.0	0.0	0.781001	0.0	0.0	0.0	0.0
Hoop Dreams	0.0	1.041335	4.000000	0.0	0.0	1.691197	0.0	0.0	1.479797	0.0	...	1.691197	0.0	0.0	0.0	0.0	0.781001	0.0	0.0	0.0	0.0

Now, if we are asked to suggest the movies for any user, we suggest him/her top n number of highest rated movies, which he/she has not watched yet. This prediction would be much more accurate because the ratings that we have newly predicted for the NaN values are personalized for each user.

In this way, the item based collaborative filtering algorithm predicts personalized recommendations for each user based on his/her own predicted ratings.

Example for the same is:-



Here we have got personalized recommendations of movies for the user with userId = 15.
We can refer to the movies which he/she has seen from the dataset above.

>References

https://towardsdatascience.com/unsupervised-classification-project-building-a-movie-recommender-with-clustering-analysis-and-4bab0738efe6

https://www.kaggle.com/code/rounakbanik/movie-recommender-systems

https://towardsdatascience.com/item-based-collaborative-filtering-in-python-91f747200fab

https://www.mygreatlearning.com/blog/masterclass-on-movie-recommendation-system/