ABSTRACT

Nowadays, providing useful suggestions of things to online users to extend their consumption of products on their web-sites is that the aim of the many tech companies. Users generally select a replacement item supported others recommendation or from their feedback, comparing similar items or response from other users. A recommendation system must be implemented to undertake these tasks automatically. The recommender system provides best suggestions that are suitable to the user's requirements or needs, even once they aren't conscious of that item. The offers of advice with individualized content are supported previous history and it hooks the customer to stay returning regularly to their websites. During our project, a movie recommendation mechanism using different algorithms within the info of Netflix are going to be built. The most sorts of recommender algorithm are Popularity based, Collaborative, Content-based and Hybrid engine Approaches. All of these methods are getting to be introduced during this project. We selected the algorithms that best fit the info to implement and compare them.

1 INTRODUCTION

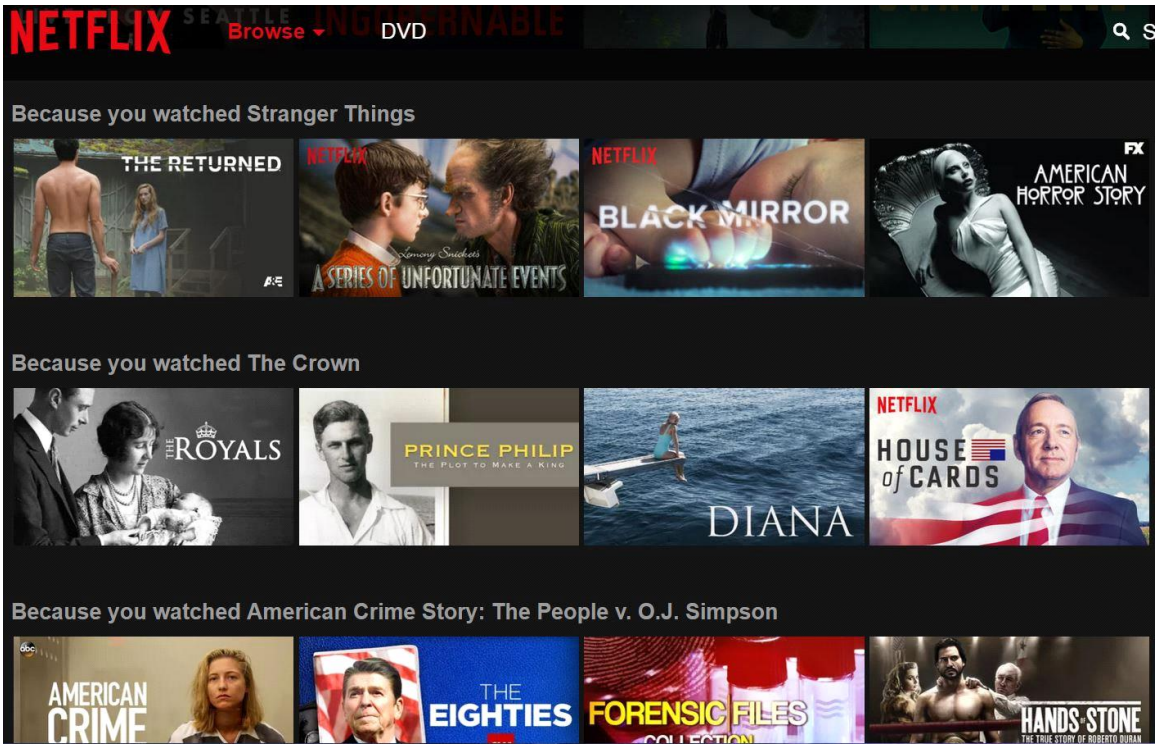
Netflix is an online streaming platform more than 75% people watch on Netflix are discovered by various Netflix recommendation systems. The majority of users what they decide to watch on Netflix or amazon any other online streaming platform is the result of decisions made by an algorithms to suggest the best movies or items to the users. Netflix is the profitable because the user's makes monthly subscription plan or yearly plan but not a trouble making since users can cancel their subscriptions at any time. Netflix uses machine learning algorithms to create a recommendation systems. Netflix uses powerful algorithms to determine each person a specific suggestion. In real time platforms their goal is to accomplish the recommendations for their items to the users. There are different ways to build a recommendation system based on videos a user has watched, we can simply suggest similar videos. We continue to improve our algorithms beyond the selection by looking for new methods we can present recommendations the algorithms we selected (i) Popularity based, in this method only popular movies are recommended (ii) content-based recommender is based on the similar items they liked and also using the keywords of the dataset (iii) collaborative filtering method recognizes the patterns and provides recommendations and (iv) Hybrid engine, it is the combination of both filtering methods. And also we used K-Means clustering with adamic adar measure. Selecting one algorithm that fits better is not an easy job, many algorithms were tested and implemented out to find which works better for the Netflix users.

2. PROBLEM DEFINITION

Netflix is all about connecting movies to the people they're interested. Netflix is understood for one among the strong recommendation engines to assist users finding the films they love, they developed world-class recommendation system: CinematchSM. Its job is predicting whether someone are going to be proud of a movie supported what proportion they liked or disliked the opposite movies. Netflix use the predictions to form personal recommendations supported each user's tastes and interests. Nowadays there are many interesting approached implemented alternatively to beat the drawbacks of Cinematch that haven't tried before on Netflix. If there's a way better approach it could make an enormous difference to users/customers and their business also. They're going to use content-based and collaborative filtering models to recommend movies and television shows. During this project, we would like to make a recommendation engine supported text/description similarity techniques. If people are stuck with the selection of movie what they want to watch next they will use outside sources like internet explorer to check out the new movies. Netflix created lots of specialization in ensuring they need an accurate algorithm for recommendation to avoid seeking help from the outside.

3. METHODOLOGY

The goal of our project is to build a Netflix movie or TV shows recommendation system using machine learning approach and based on graph by using the Adamic Adar measure. The system should be able to generate a recommended movies for a user based on his/her previous watched Movie or TV show, the similar interaction of other users with the system to make prediction.



In order to build our machine learning model, the following prerequisite was needed:

i. A development environment and tool for machine learning model.
ii. An appropriate machine learning and filtering methods to build our model.
iii. A rich library of movies & TV shows which consists of various attributes for building the dataset.

The following Algorithms was adopted to build our recommendation system:

- K-Means clustering with TF-IDF
- Adamic Adar Measure: Example if two movies are not directly connected, but they share persons, categories, clusters & countries.

Adamic Adar(x, y) = $\sum u \in N(x) \cap N(y)$ 1log (N (u))

Where x, y are two movies.

- In order to determine the list of recommendation movies, we are going to explore the neighbourhood of the target film since the measures between all movies are not pre-calculated.

- Machine learning models: content based filtering, collaborative, hybrid filtering recommender models.

3.1 DEVELOPMENT ENVIRONMENT:

Jupyter Notebook was adopted as the development environment writing and executing our code. We used python as a programming language since it provides a wide variety of machine learning algorithms. By using these resources, we build a machine learning model for predicting (recommendation) system to our dataset. We tried executing this on Google colaboratory it was implemented successfully.

3.2 DATA GATHERING:

In this project we used three datasets. We collected data from open sources. First dataset consists of TV shows and Movies available on Netflix. Using this dataset we implemented graph based recommendation engine using Adamic Adar measure.

We built a base Movie Recommendation System by implementing a few filtering methods (content based, popularity based and collaborative filtering) for this we have two MovieLens datasets.

The dataset contains the following attributes:-

cast - The names of leading and supporting actors for the particular movie.

crew - The names or details of the Director, Editor, Composer, Writer etc., of each movie.

movie_id – Id is a unique identifier for each movie.

The second small dataset has the mentioned attributes:-

- Budget - The budget of the movie was made.
- Genre - The genre of the movie.
- homepage - homepage link of the movie.
- Id - Id is the movie_id as in the first dataset.
- Popularity - A numeric representation specifying popularity of the movie
- production_companies - The production house of the movie.
- production_countries - The country which produced.
- release_date - The date on which it was released.
- Revenue - Revenue generated by the movie.
- Run_time – duration of movie
- status - "Released" or "Rumored".
- tag – tagline of the movie.
- title – movie name
- keywords - The keywords related to the movie.
- Original language – The origin language of the movie.
- Original title - The title of the movie.
- overview – An overview of the each movie.
- vote_average - average ratings of the movie received by users.
- vote_count - the count of votes received by the users.

3.3 DATA PREPARATION:

In this project we used three datasets. The first dataset used is gathered from open source search engine kaggle which consists of Shows and Movies list with some attributes available on Netflix. Using this dataset we implemented graph based recommendation engine using Adamic Adar measure.

Steps involved:

- Exploring the what content is available in different countries
- Identifying similar content of shows and movies available by matching text based features using TFIDF
- Network analyzing of Directors or actors.

Uncut version of films are labeled as “unrated” which contains warning messages that the uncut version won’t be suitable to under 17, it has different content available from the theatrical release version.

So we have to repair this by replacing UR category by NR within the first dataset.

(i) We get to choose the data that is relevant, which suggests to scale back the info volume by improving the quality of the info and

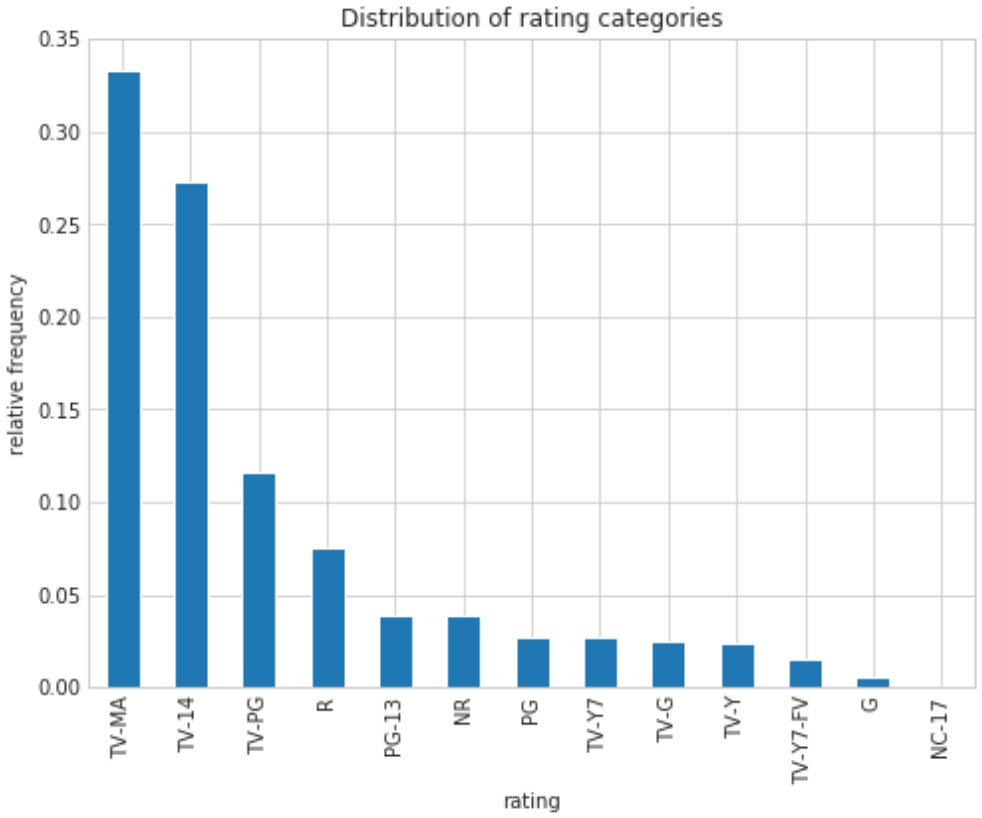
(ii) To normalize the data, removing some utmost values within the ratings per user

We joined the 2 datasets on the ‘id’ column.

Using pandas we loaded both csv files, combined both files as one dataframe. The resulting new data was utilized in performing our recommendation model experiment.

```
[ ] dt2.head(5)
```

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ingenious Film Partners", "id": 289...
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"name": "Walt Disney Pictures", "id": 2}, {"...
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	A cryptic message from Bond’s past sends him o...	107.376788	[{"name": "Columbia Pictures", "id": 5}, {"nam...



Average Rating distribution after fixing the above mentioned problem (replacing UR by NR)

3.4 DATA ANALYSIS:

To analyze the data, firstly we need to

- Understand categories in the rating column.
- Need to understand what content is available in various countries.
- Need to understand whether Netflix has been focusing on shows or movies in recent years.
- Understand the most observed rating categories in Movies and TV shows.
- Identifying the content that is alike by matching text-based features using TF-IDF.

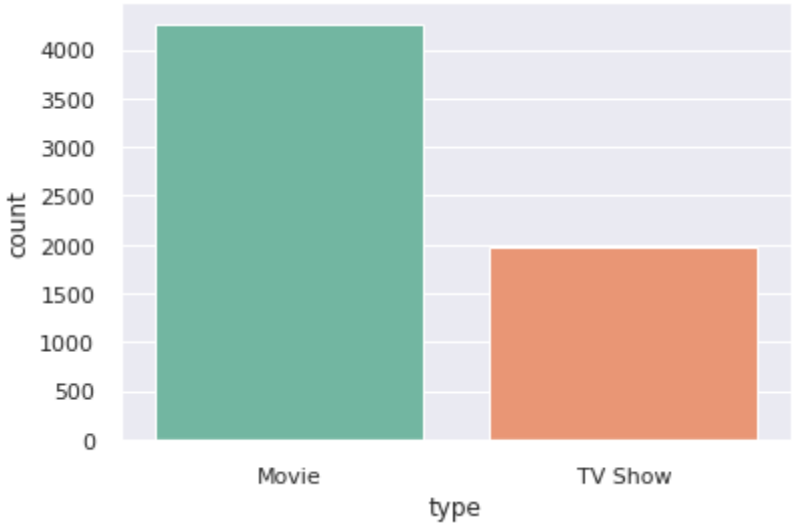
Now explore the categories of ratings from Google:

- TV-MA: It is designed to adults only, hence it may be restricted for children under 17 which is not suitable to them.
- TV-14: This program was hardly cautioned to parents since it is unsuitable for children under 14 years which is similar to PG-13 or R.
- TV-PG: Parental assistance to the children is suggested for this program which may be unsuitable for younger children.
- R: It is not suitable to the children under 17, restricted and parent's guidance is required.
- PG-13: Some contents may not be suitable for children aged under 13. Parents are requested to be cautious before their children watching it.
- NR or UR: Suppose a movie or show has not been submitted to MPAA for a rating, labels not rated or is an uncut version of a movie maybe submitted.
- PG: In this program some content may not be suitable for children, It may contain some content parents might not like to show interest for their young children.
- TV-Y7: This program is created for children age 7 and above.
- TV-G: This indicates a film contains nothing so it is useful for all age's people.
- TV-Y: Programs rated TV-Y are outlined to be relevant for younger children of all age groups, as well as children ages 2-6.
- TV-Y7-FV: is recommended for ages 7 and older, the program which contains fantasy violence.
- G: Suitable to all the audience means no age restrictions.
- NC-17: No One aged 17 and under are admitted only for adults. Children are not admitted.

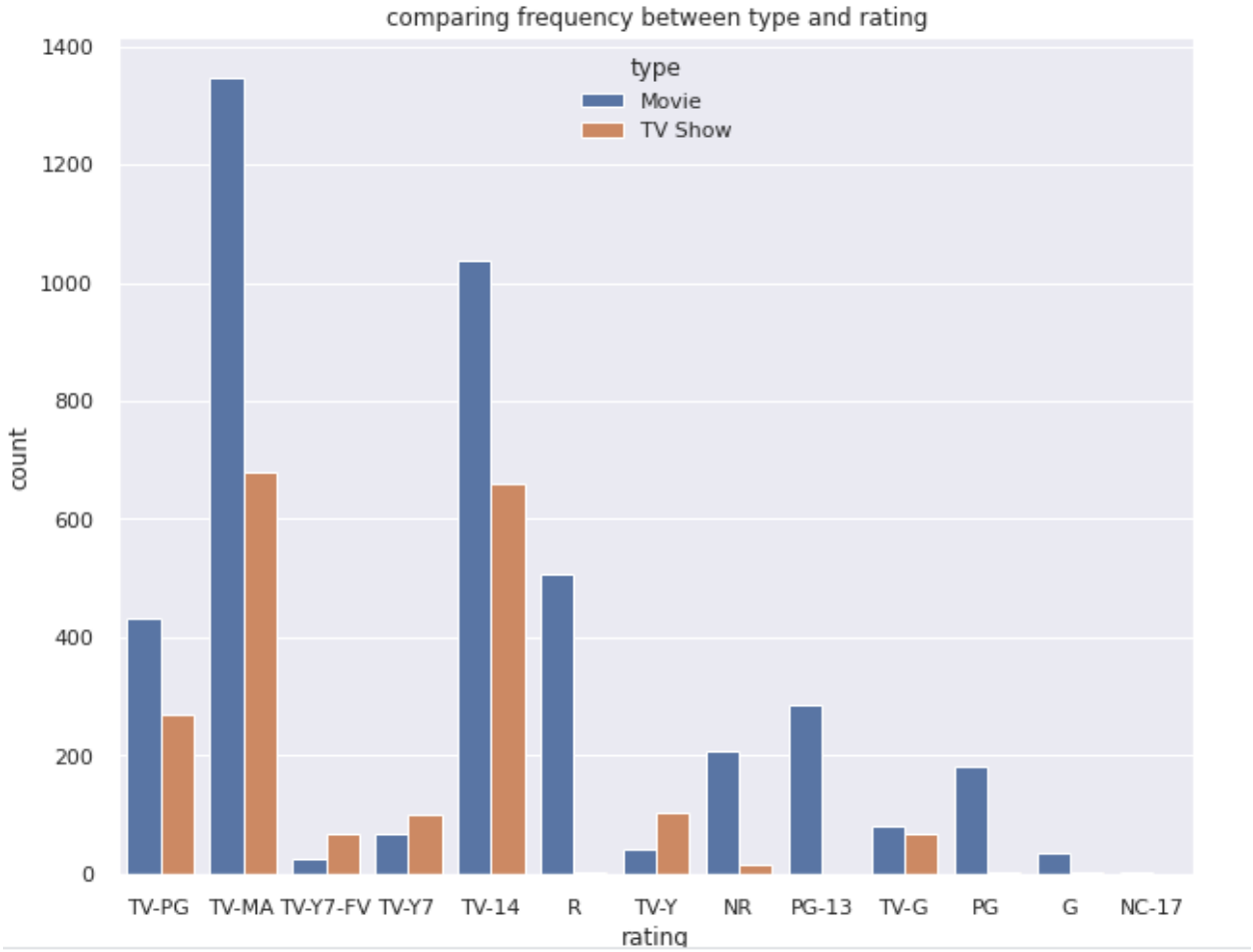
Now let's see the output analysis for our dataset:

i. Comparing Rating distributions between TV shows and Movies
ii. Comparison between the types that the top countries produce
iii. Netflix Genre in countries

Total Number of TV show and Movies available on Dataset

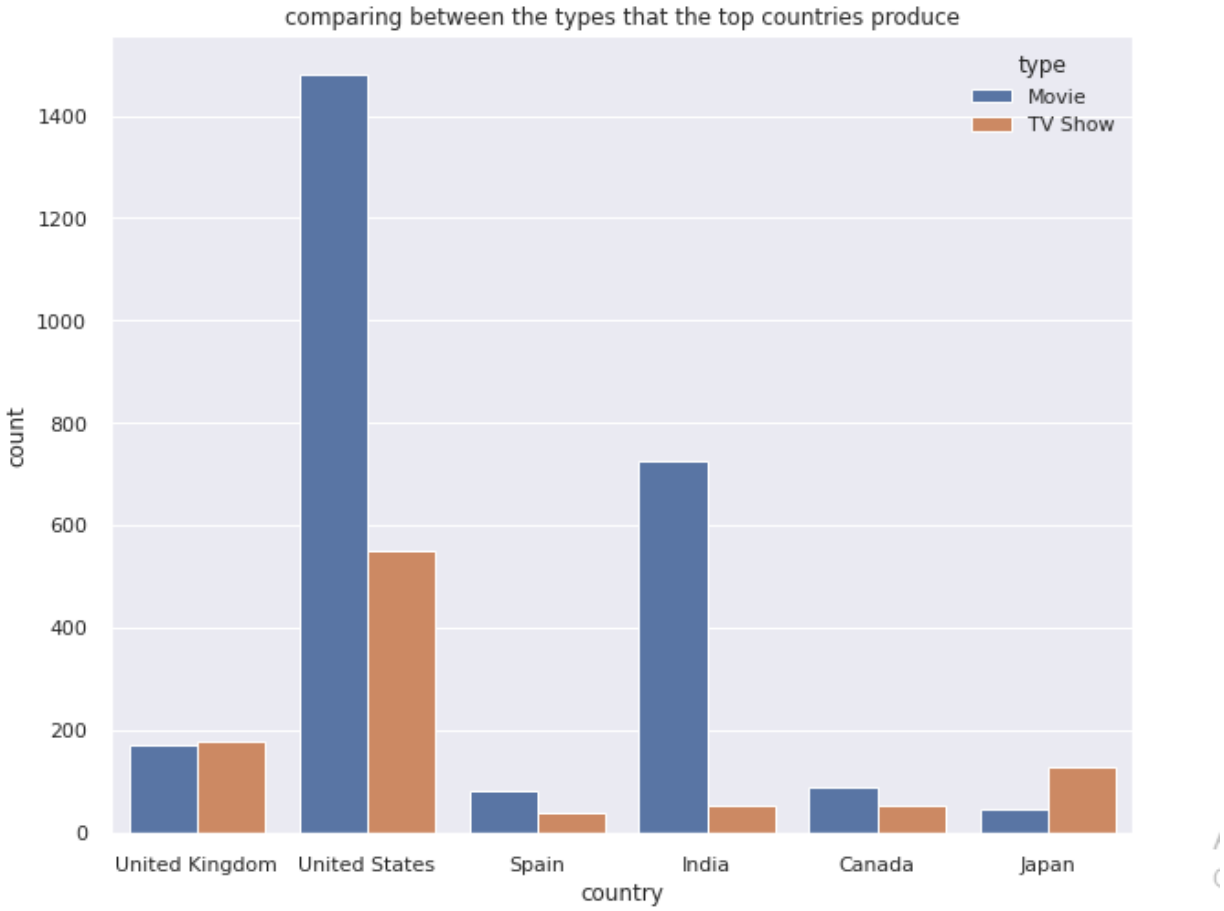


Comparing Rating distributions between TV shows and Movies

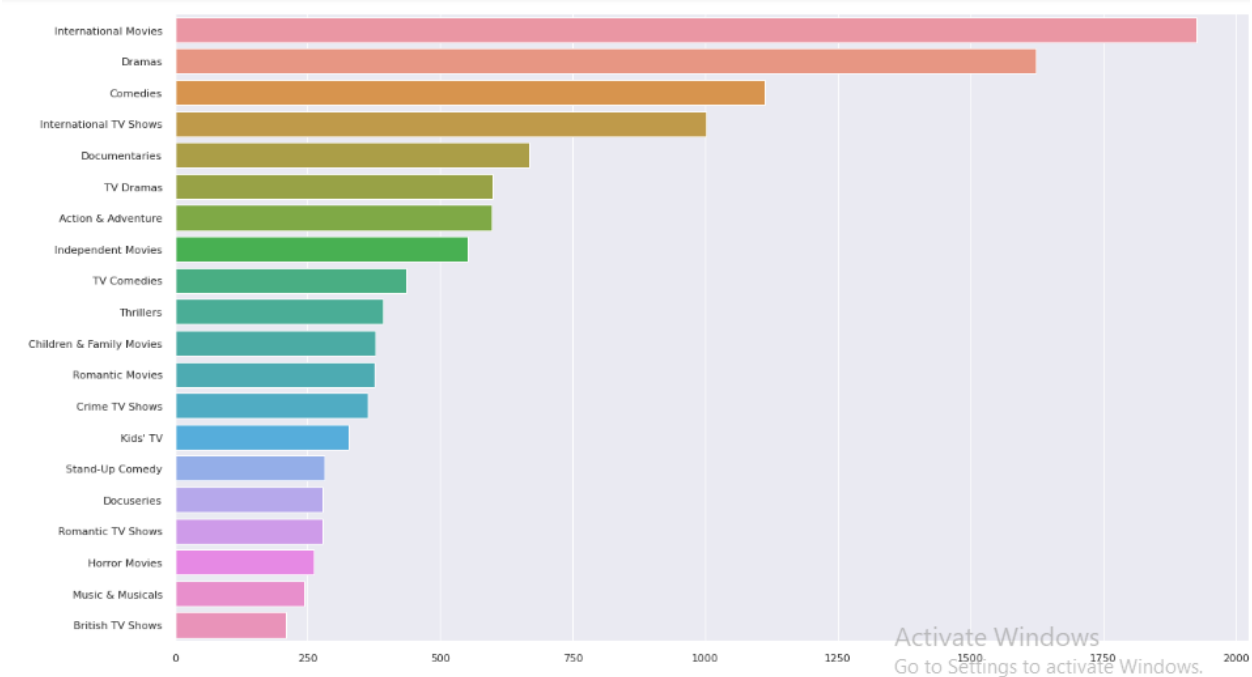


Comparison between the types that the top countries produce

```
United States                                2032
India                                         777
United Kingdom                                348
Japan                                         176
Canada                                        141
                                             ...
Germany, United Kingdom, United States          1
Netherlands, Denmark, France, Germany           1
France, Luxembourg, Canada                      1
United Kingdom, Ukraine, United States          1
United States, Greece, Brazil                   1
Name: country, Length: 554, dtype: int64
```



Netflix Genre in countries



3.5. VARIOUS METHODS OF BUILDING A MOVIE RECOMMENDER SYSTEM

3.5.1. KMeans clustering with TF-IDF

KMeans clustering was implemented by creating a cluster of movies in the dataset. The clustering was based on user watch history. Based on the cluster, recommendation was then done, one of the advantage of this clustering method is that it scales well with large dataset.

TFIDF (term frequency-inverse document frequency): It is used to find similar description, it is a numerical statistic that demonstrates how important a word is to a corpus. IDF is used as a measure of how much information the token/word provides in the document.

I. Save objects on filing system

II. Load objects on filing system

III. Search the top_n articles from an invitation

IV. Find almost like p_n articles similar to a piece of writing

V. Clustering the corpus of texts with K-means

VI. Accurate prediction of comparable texts

The tf-idf is that the term frequency and therefore the inverse document frequency represented as:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

Scikit-learn implements the TfidfVectorizer that takes one dimensional array of texts as input and transforms them into tf-idfs:

3.5.2 ADAMIC ADAR MEASURE:

It is used to measure the closeness between nodes based on their shared neighbors.

Example if two movies are not directly connected, but they share persons, categories, clusters & countries. Adamic Adar measure can be calculated as:

$$adamicAdar(x, y) = \sum_{u \in N(x) \cap N(y)} \frac{1}{log(N(u))}$$

Where x, y are two movies.

- In order to determine the list of recommendation movies, we are going to explore the neighbourhood of the target film since the measures between all movies are not pre-calculated.

- if x & y share a node example u which has a lot of adjacent nodes, this node is relevant

→ N (u) is high → 1/log (N (u)) is not high.

If x & y share a node u which doesn't contains a lot of adjacent nodes, this node is really relevant. → N(u) is not high → 1/log(N(u)) is higher

THE RECOMMENDATION FUNCTION:

Firstly we loaded the data and followed steps as will

- Explore the neighborhood of the targeted movie/TV show → this can be a list of actor, director, country, and category

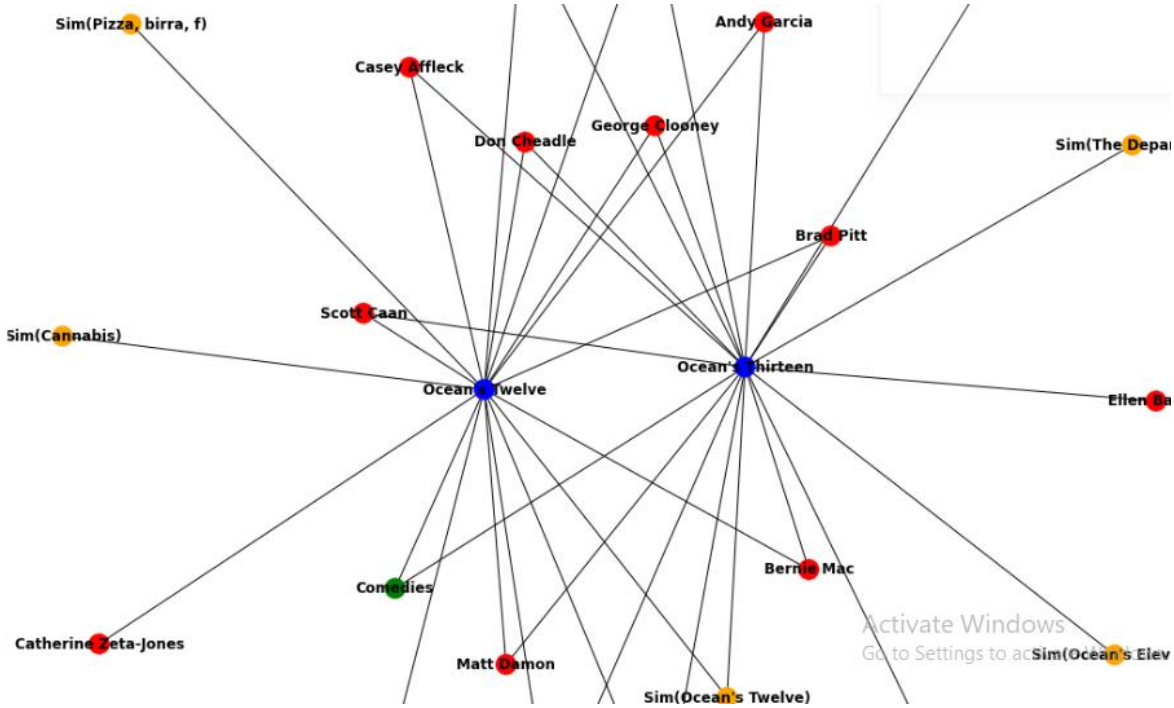
- Explore the neighborhood of each neighbor → discover the movies/TV shows that share a node with the target.

- Compute Adamic Adar measure → final results.

- Two movies are not connected directly but they share clusters, actors, categories etc.

Result for the experiment on adamic adar measure and KMeans clustering with TF-IDF:

```
****************************************
 Recommendation for 'Transformers Prime'
****************************************
Transformers: Robots in Disguise    3.613282
Beyond Skyline                      1.801518
Spirit: Riding Free                 1.742503
Chappaquiddick                      1.645837
Would You Rather                    1.569881
dtype: float64
```



3.5.3 MACHINE LEARNING

We employed the tactic of a recommender that's supported clustering and various filtering methods to create our recommendation systems. There are basically four kinds of recommender systems:

i. Demographic Filtering Recommender
ii. Content-based Filtering Recommender
iii. Collaborative Filtering Recommender and
iv. Hybrid recommender system

DEMOGRAPHIC RECOMMENDER:

This approach offers a simple recommendations to each user or client, based on general basic details of movie like popularity or rating etc. This approach recommends the same movies to all users with similar demographic features. Since each user is different they have their individual tastes, this approach is considered to be too simple so this approach is also called as a simple recommender. The basic motto behind this simple recommender system is that the movies which are more popular will have a higher

probability of being liked by the average users/audience. The shortfall of this method is it is not personalized to individual users, all users will get the same recommendation.

POPULARITY BASED RECOMMENDATION:

This is a type of recommender system that recommends top ten or most popular movies to the users/audience based on how popular it is. The popularity ranking can be based on how many times each movie was viewed or the number of times that movie was watched. The system keeps a list of all the movies ranked from top to bottom, with the total number of votes to the movie with the least votes.

Steps involved:

- We need to measure score or rate the movie
- Once movie details are ready we need to calculate the score for each movie
- Sort the scores and recommend the best rated movie to the audience.

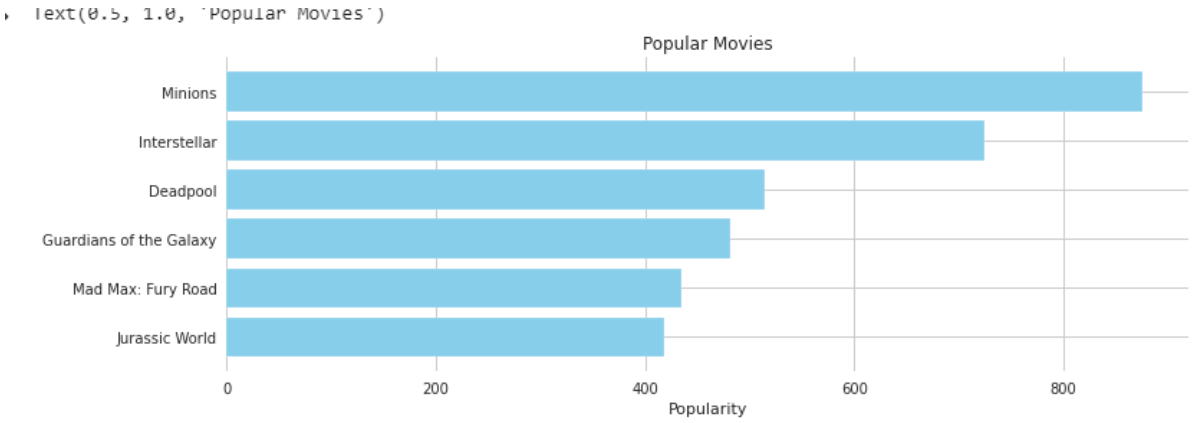
We can use the average ratings of the each movie as the score but using this won't be fair to provide recommendations as it will be biased.

So we used IMDB's weighted rating (WR) given as:

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

- v – is the number of votes for the each movie
- m – is the minimum votes required
- R – is average rating of every movie
- C – is the mean votes across the whole report

Using this approach we can find top 6 popular movies by sorting the dataset as observed:

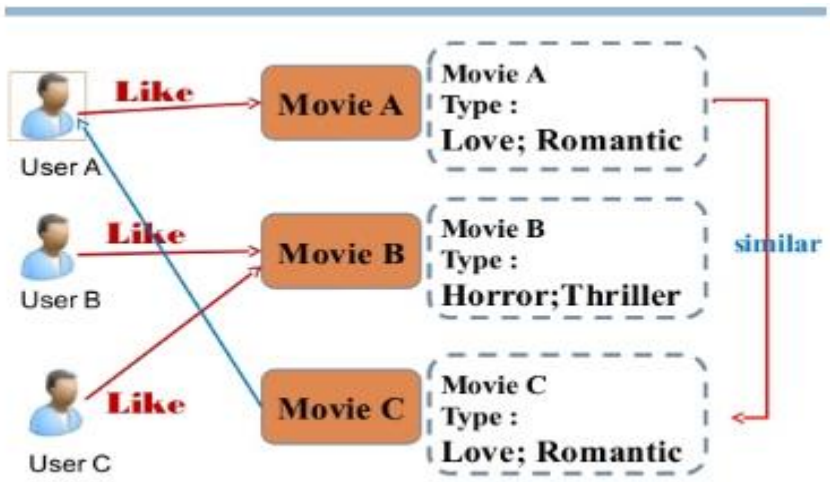


This recommender system provide a general simple overview of recommended movies to all the users based on popularity. But this is not upto the mark of the interests and tastes of the users. So we move on to a Content Based Filtering.

CONTENT BASED FILTERING:

In the previous approach they suggest based on the attributes of an item and the users preferred profile. But in this approach we are using item metadata such as cast, crew, keywords, genre, director, description, etc. for movies to provide appropriate recommendations to the users. The general concept behind these recommender systems is by looking at user specific classification problem, it should understand if a person liked a particular item and same person will also like an item that is similar to it.

There are several classification methods used for content-based such as the Euclidean, the Pearson and the cosine similarity scores. No particular classification method determined as the best method. Different scores work well in different scenarios to find its similarity with other movies. The movies that are likely to be similar will be recommended as follows:



Plot description based Recommender

We compute similarity scores for all movies in pairwise to recommend the movies supported their plot descriptions and score. The plot description of movies is provided within the overview feature of our dataset. TFIDF is used to reduce the significance of words that occur frequently in plot overviews and their importance in calculating the similarity scores also. We used the cosine similarity to compute a numeric value that denotes the similarity between two nodes means movies or shows. Since it's independent of size and is comparatively easy to calculate and fast too.

Mathematically, it is represented as follows:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

We see that in our dataset for 4803 movies over 20,978 different words were used to describe them.

```
[ ]  df2['overview'].head(5)
```

```
0    In the 22nd century, a paraplegic Marine is di...
1    Captain Barbossa, long believed to be dead, ha...
2    A cryptic message from Bond’s past sends him o...
3    Following the death of District Attorney Harve...
4    John Carter is a war-weary, former military ca...
Name: overview, dtype: object
```

Credits, Genres and Keywords Based Recommender

This recommender goes with the usage of some better metadata. We are getting to build a recommender model supported the metadata as follows:

• The top 3 cast and the director,

• Correlated genres and thence the movie keywords.

From the cast, crew and keywords features, we would like to extract the three top most actors, the director and therefore the keywords related to that movie. Our data is present within the sort of "string" lists, we would like to convert it into a secure and ease of use structure.

```
[ ]  get_recommendations('Catwoman', cosine_sim2)
```

```
4638    Amidst the Devil's Wings
3          The Dark Knight Rises
65               The Dark Knight
72                 Suicide Squad
119                Batman Begins
299               Batman Forever
468                    Swordfish
1043           Miss Congeniality
1456              Bound by Honor
210               Batman & Robin
Name: title, dtype: object
```

Collaborative Filtering:

The system finds users with alike interests/tastes and provides suggestions supported this matching. Collaborative filters don't require similar like in content-based filtering model.

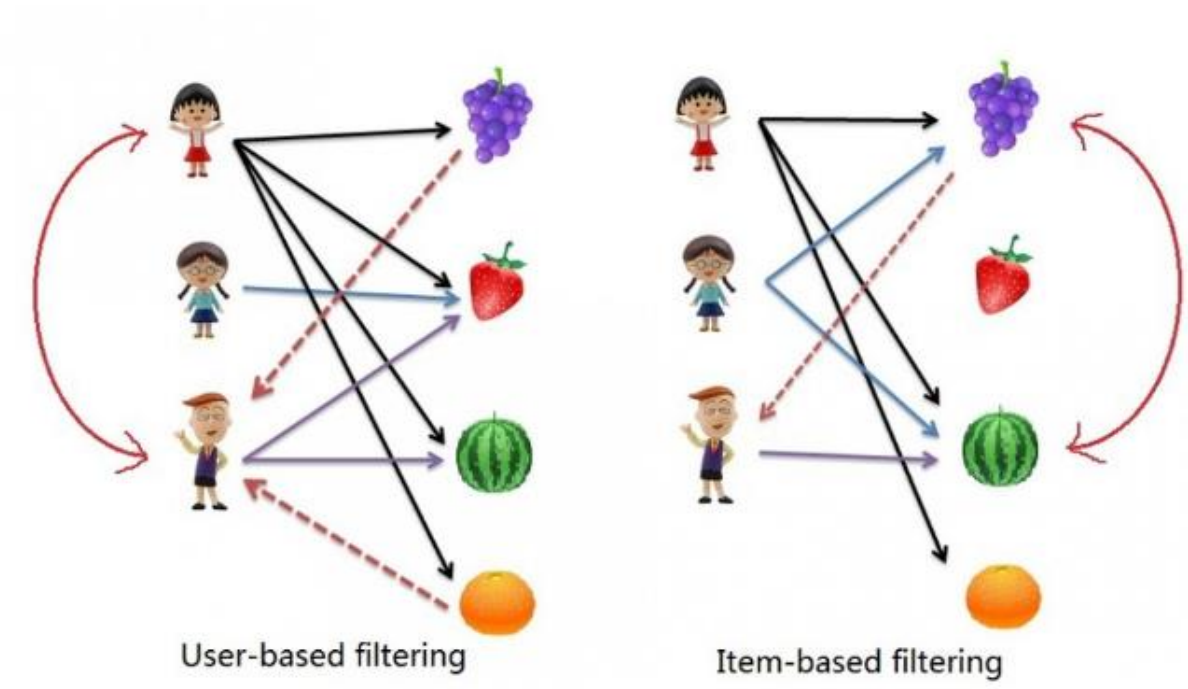
Our content based engine has some limitations. This model is merely able to suggest movies which are on the brink of a particular movie. Which suggests, it's unable to catch specific tastes and provide suggestions across particular genres. And therefore the content based recommendation engine that isn't exactly personal therein it doesn't observe the individual tastes and biases of a user. Users using our recommendation system will get a similar suggestions for the required movie.

Therefore, we will use a way called collaborative filtering to provide recommendations to users. Collaborative Filtering is predicated on the thought that users almost like me are often wont to predict what proportion i will be able to sort of a particular movie. We are using three methods during this filtering.

i. User based filtering

ii. Items based filtering and

iii. SVD(Single value decomposition)



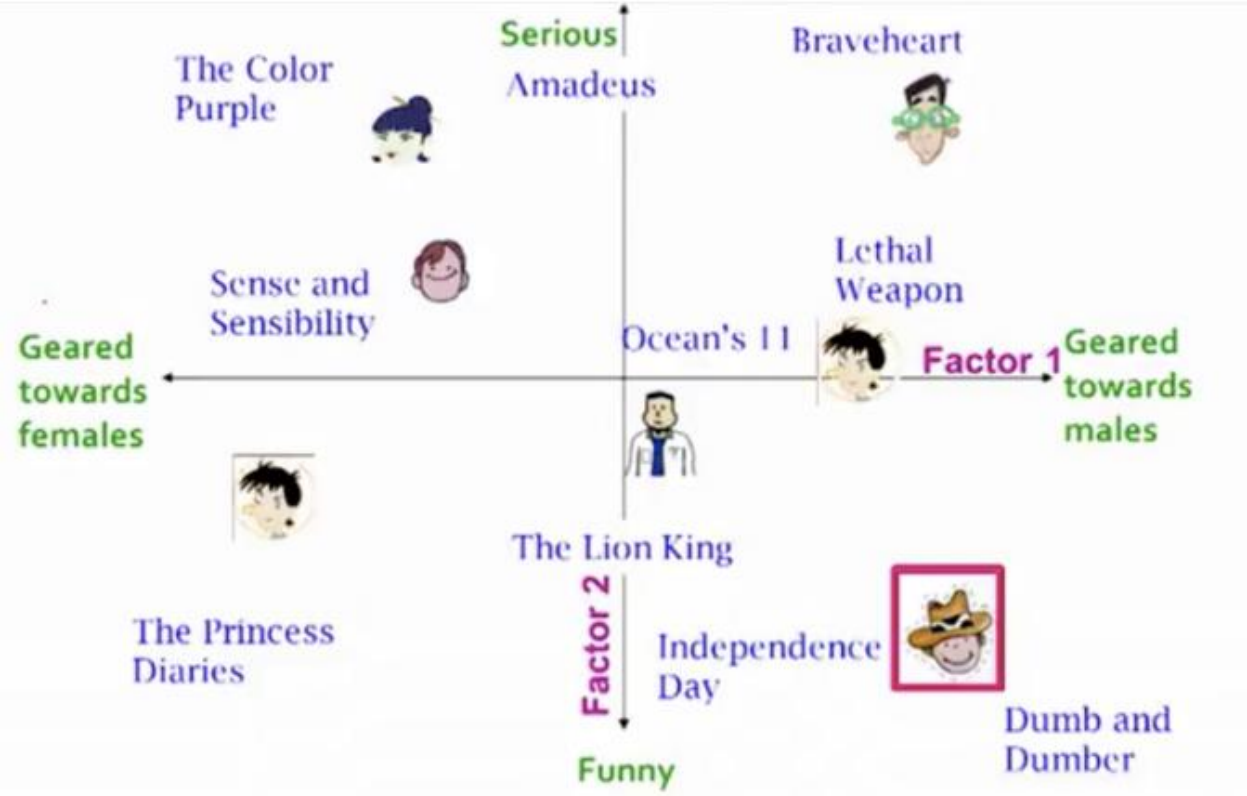
User based filtering- User based systems suggest the products to a user that similar users have liked. For computing the similarity between two users we will either use Pearson correlation or cosine similarity. Although measuring user-based Collaborative filtering is extremely simple, it has some drawbacks. One among the most issue is that user's choice or taste can be changed over time. It indicates that pre-measuring the matrix which is supporting their neighboring users may give bad output. To tackle the drawback, we are considering an item-based filtering

Item Based Collaborative Filtering- Rather than measuring the comparison between similar users, the item-based Collaborative recommends items supported their closeness with the things that the target user rated. The similarity is calculated with Pearson Correlation or Cosine Similarity methods.

It will avoid the issue created by dynamic user preference where item-based Collaborative Filtering is more static. However, some issues remain same for this method. First, the main problem is scalability. The measuring grows with both the user and the product/item. O (mn) with m users and n items that is the worst case complexity. Sparsity is another concern.

Singular Value Decomposition (SVD) - SVD is a Scipy algebra function, using this we need to predict unknown ratings. To handle the scalability and sparsity problem created by Collaborative Filtering to hold a latent factor model to considerate the similarity between users and items/products. Firstly, we would like to show the advice issue into an optimization problem. While predicting the rating for items/products given by a user we will look into it to check how good or bad we are in. One regular metric is Root Mean Square Error (RMSE) we want to split the rating matrix into constituent user matrix with minimum sum. The less the RMSE, the higher will be the performance

Latent factor, it's a broad idea which describes a property or concept that a user or an item or product have. In our project, for movie, latent factor can mention to the genre that the movie belongs to. By extracting its latent factors SVD decreases the dimension of the utility matrix. Firstly, we match each user and an item into a latent space with dimension. So, it helps us better to understand the relationship joining users and items since they will become directly comparable. The below figure explains this idea. We used the **Surprise** library to implement SVD.



HYBRID RECOMMENDER ENGINE:

Content-based and collaborative filtering together to build an engine is called hybrid engine. This engine will provide movie recommendations to a particular user based on the estimated ratings that is internally measured before for that user using svd. The union of content-based and collaborative filtering methods together then we can overcome the issues of users. Example if we are using collaborative filtering it will provide recommendations using matrix but there is no rating given by a new user in such cases the contents of user-item means content-based filtering can be used for best recommendations.

Below is the output in our project we gave input as user-id and movie title as 'Avatar', the resulted output we get is similar movies sorted on expected ratings by that specific user.

```
hybrid(500, 'Avatar')
```

	title	vote_count	vote_average	year	id	est
8401	Star Trek Into Darkness	4479.0	7.4	2013	54138	3.238226
974	Aliens	3282.0	7.7	1986	679	3.203066
7265	Dragonball Evolution	475.0	2.9	2009	14164	3.195070
831	Escape to Witch Mountain	60.0	6.5	1975	14821	3.149360
1668	Return from Witch Mountain	38.0	5.6	1978	14822	3.138147
1376	Titanic	7770.0	7.5	1997	597	3.110945
522	Terminator 2: Judgment Day	4274.0	7.7	1991	280	3.067221
8658	X-Men: Days of Future Past	6155.0	7.5	2014	127585	3.043710
1011	The Terminator	4208.0	7.4	1984	218	3.040908
2014	Fantastic Planet	140.0	7.6	1973	16306	3.018178

4. EXPERIMENTS AND DISCUSSIONS:

We implemented the advice system using various algorithms. For this project we compared and implemented 3 different methods using various models. Also, we used Adamic adar measure by implementing KMeans clustering with TF-IDF.

MACHINE LEARNING EXPERIMENTS:

The first model we implemented was a K-Means clustering using Adamic adar measure to represent the recommendations during a graphical form afterward we implemented demographic filtering which is employed to recommend the favored movies within the dataset is additionally named as Popularity recommender model, this was achieved by listing top 6 movies with the very best watch count within the dataset, but this model gives an equivalent recommendation to each user without personalization, so we decided to implement Content-based, collaborative and hybrid filtering recommendation models. These models suggest supported the attributes of an item and therefore the users preferred profile using item metadata like cast, crew, keywords, genre, director, description, etc. for movies to know if an individual liked a specific item and same person also will like an item that's almost like it. And eventually we come up with hybrid recommender it's the mixture of both content based and collaborative filtering.

5. Conclusion:

In our project, we built 5 different recommendation engines based on different algorithms with different ideas. They are as follows:

1. **Demographic Filtering:** This is a simple recommender it used total TMDB vote averages and TMDB vote counts to build Top popular Movies Charts, in general and for a specific genre to all the users we will get same output. To sort final ratings we used the IMDB Weighted Rating System for better performance.
2. **Content Based Filtering:** We built two content based engines; plot based recommender and the other which took metadata such as cast, crew, genre and keywords to come up with predictions using matrix factorization. We also implemented a simple filter to give greater preference to movies with more votes and higher ratings.
3. **Collaborative Filtering:** We used the powerful Surprise Library to build a collaborative filter based on single value decomposition. The RMSE obtained was less than 1 and the engine gave estimated ratings for a given user and movie.
4. **Hybrid Recommender:** Content -based filtering and collaborative filtering together to build an engine is called hybrid recommender engine. This system will provide movie suggestions to a specific user by measuring the estimated ratings that it had internally measured for that particular user.
5. **K-Means clustering with TF-IDF Using Adamic Adar Measure:** K-Means clustering was implemented by creating a cluster of movies in the dataset. The clustering was based on user watch history. Based on the cluster, recommendation was then done, one of the advantage of this clustering method is that it scales well with large dataset.