DS 5230 Netflix Movie and TV shows Recommender Systems

Podeti Nithish Goud, Varsha Reddy Anugu
Student, Master's in Data Science, Northeastern University
podeti.n@northeastern.edu , anugu.v@northeastern.edu

Abstract

Recommendation systems, like the Netflix Recommender System, will become essential competitive elements for every significant over-the-top video streamer as the Streaming Wars intensify. This project is Movie and TV show recommendations of Netflix based on the content of the description. How do we recommend movies or tv shows when ratings or multiple users are not given? We have solved this problem using content-based recommendation systems. Recommending movies or tv shows based on the description of movies or tv shows is one approach that we worked on. Another approach was to combine multiple metrics like Director, cast, listed-in, etc. for better recommendations. Analysis of the number of clusters that can be formed based on the description, listed-in, country, cast, and director metrics is also done using K- means algorithm and Agglomerative hierarchical clustering. That is clustering similar content by matching text-based features. The results are recommendations of movies and tv shows based on above mentioned two approaches using content-based filtering and the clusters formed using those algorithms are shown.

Introduction:

Information filtering systems include recommender systems. The algorithm of the system can precisely identify user preferences by analyzing vast data sets. You can suggest new, pertinent material to your readers once you understand what they enjoy. And that is true of everything, including romantic partners, music, and movies.

A new phase of information has emerged because of the tremendous rise in data collection. Recommendation Systems play a key role in the usage of data to build more effective systems. As they improve the effectiveness of search results and present items that are more pertinent to the search item or are related to the user's search history, recommendation systems are a type of content filtering system. Examples of recommender systems in use include Netflix, YouTube, Tinder, and Amazon. Based on their selections, the programs tempt consumers with pertinent suggestions. Amazon uses it to propose products to customers, YouTube to select the next video to play automatically, and by Facebook to suggest sites and users to follow. Additionally, the profitability of services like Netflix and Spotify depends on the effectiveness of their recommendation algorithms.

Our Project deals with getting recommendations based on content from the dataset which is in the form of text. Our main idea of the approach is to use a content-based recommendation system. Since there are no ratings given or users, our sole idea of this project is to recommend based on text. The first approach was to consider only the description metric of a movie or Tv show and based on this what better recommendations can be made. The Second approach was to consider multiple metrics like director, cast, listed-in, and description and get the best recommendations considering precise information. For both of these approaches, we used the cosine similarity metric to get similarity scores for that particular movie and the top ten movies that we get based on the similarity scores.

We have also performed cluster analysis on the dataset to get an optimal number of clusters based on text. That is clustering similar content by matching text-based features. Applied different algorithms like K-means clustering and Agglomerative hierarchical clustering to get the best cluster arrangements. We have used the elbow method to find the optimal number of clusters which is 9. Based on these clusters we have segmented into 9 segments of movies and tv shows. We made word clouds of each cluster and highlighted which genres are there in every cluster. Each cluster can recommend a particular genre of movies based on user liking.

Dataset:

This dataset consists of tv shows and movies available on Netflix as of 2021. The dataset is collected from Flixable which is a third-party Netflix search engine. It consists of 12 columns and 8807 rows.

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...

Exploratory Data Analysis:

Few insights of the data from the dataset have been explored so that a better view of what insights we get can be understandable.

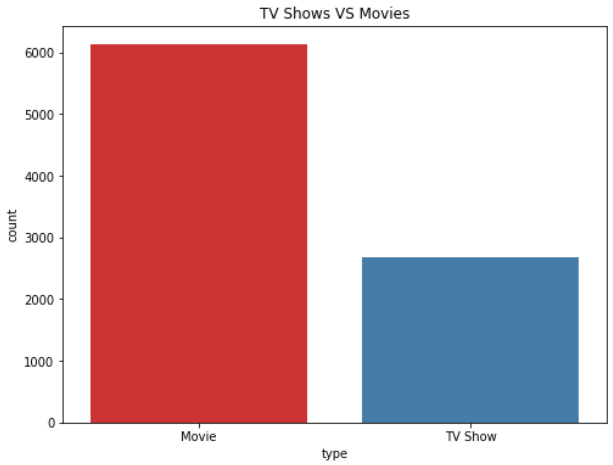

Figure 1: Bar chart of the number of tv shows and movies.

We can observe that around 6000 movies and around 2800 tv shows which are half the count of the number of movies.

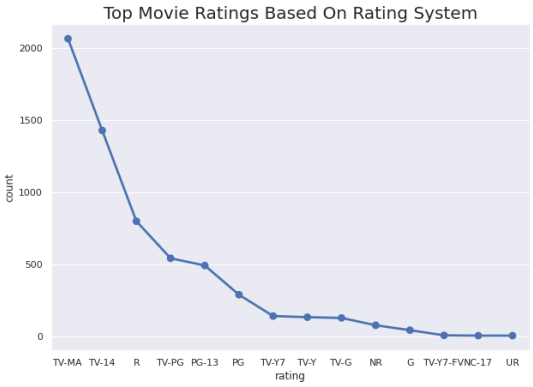

Figure 2: Pointing ratings of movies on a point plot

We can observe that TV-MA (Mature Audiences) has the highest count and is used the majority in films.

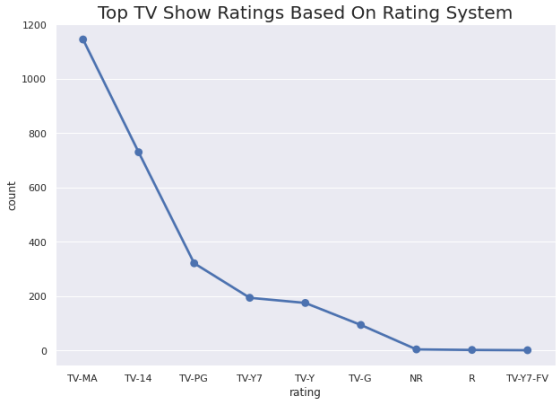

Figure 4: Pointing ratings of tv shows on a point plot

We can observe that TV-MA rating is used in the majority of films.

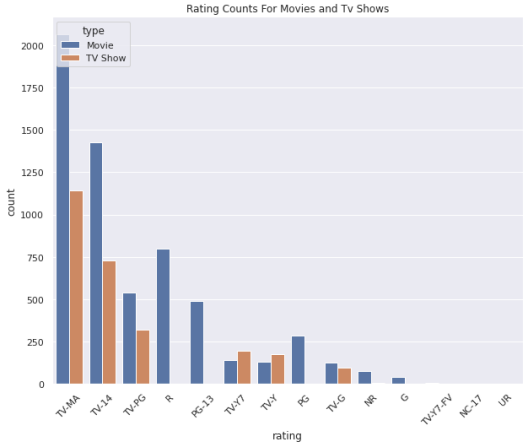

Figure 5: Rating Counts for movies and tv shows.

We can observe that the highest ratings go as follows, TV-MA, TV-14, and TV-PG.

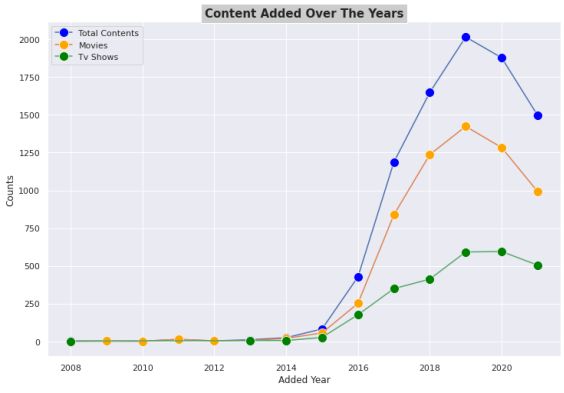

Figure 6: Content added over the years.

Most content is added to Netflix in the year 2019.

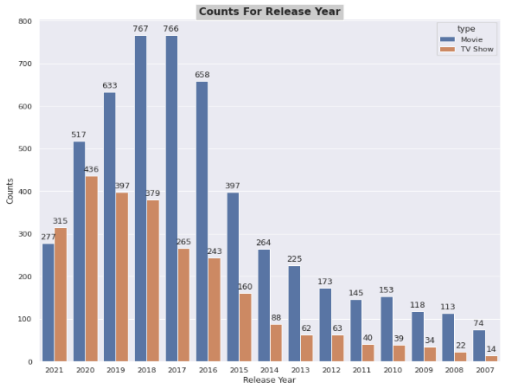

Figure 7: Counts for release year

In the year 2018, 767 movies are released which is the greatest number of releases in a year. However, 436 tv shows were released in 2020 which is the highest number.

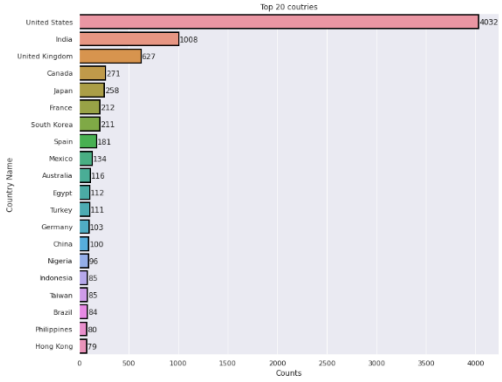

Figure 8: Count of movies and tv shows based on the country name.

The United States tops the list followed by India.

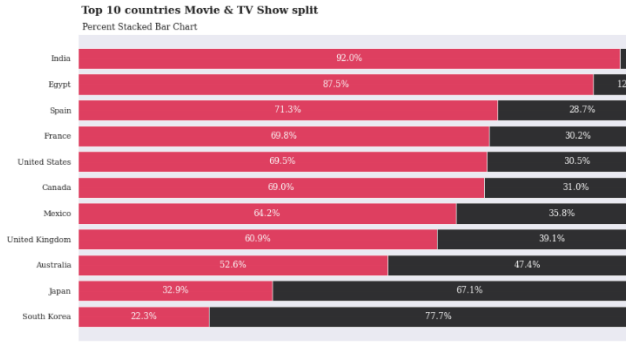

Figure 9: Percentages of movies and tv shows released from top 10 countries.

For India, 92% of movies and 8.04% of tv shows exist in the Netflix dataset.

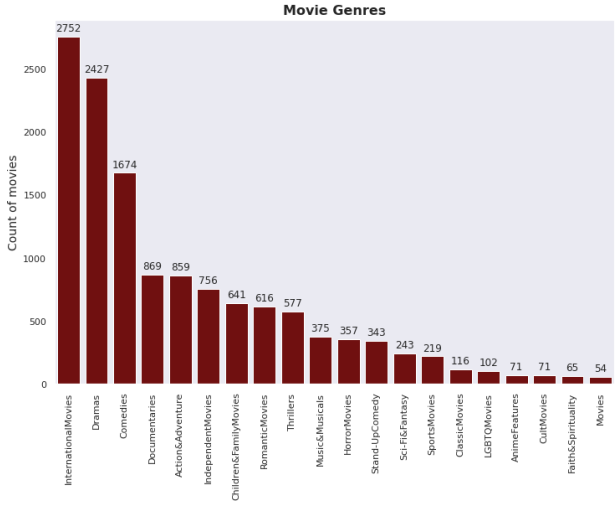

Figure 10: Top 10 Movie genres

International movies have the highest count followed by the Drama genre.

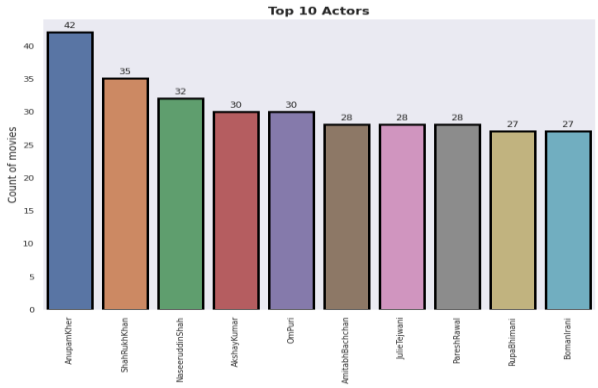

Figure 11: Top 10 Actors in movies

Anupam Kher has acted in most of the movies whereas Bomani rani with the least.

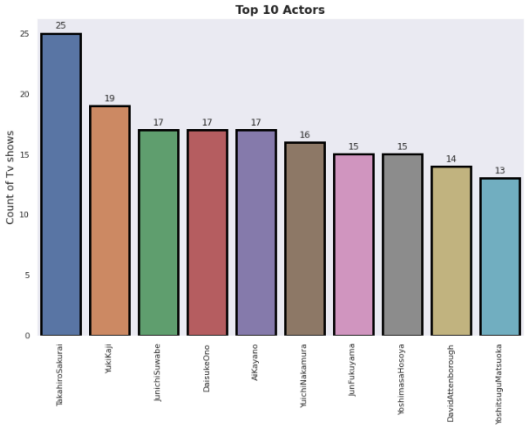

Figure 12: Top 10 actors in tv shows.

Takahiro Sakurai has acted in most of the tv shows whereas Yoshitsugu Matsuoaka acted the least.

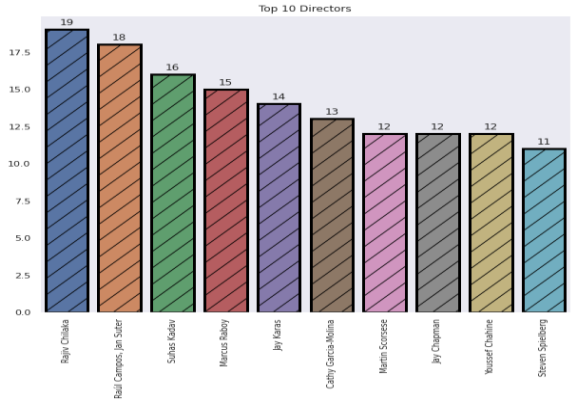

Figure 13: Top 10 directors

Rajiv Chilaka directed most of the movies whereas Steven Spielberg directed the least.

Background:

TF-IDF Vectorizer:

The TF-IDF (Term Frequency-Inverse Document Frequency (TF-IDF)) score is the frequency of a word occurring in a document, down-weighted by the number of documents in which it occurs. This is done to reduce the importance of words that occur frequently in plot overviews and therefore, their significance in computing the final similarity score.

Now if you are wondering what Term Frequency (TF) is, it is the relative frequency of a word in a document and is given as (term instances/total instances). Inverse Document Frequency (IDF) is the relative count of documents containing the term is given as log(number of documents/documents with the term) The overall importance of each word to the documents in which they appear is equal to TF * IDF This will give us a matrix where each column represents a word in the description vocabulary (all the words that appear in at least one document) and each row represents a movie, as before. This is done to reduce the importance of words that occur frequently in plot descriptions and therefore, their significance in computing the final similarity score.

Count vectorizer:

A text is converted into a vector using a count vectorizer based on the frequency (count) of each word that appears in the full text. When we have several of these texts and want to turn each word into a vector, this is useful (for use in further text analysis).

Each unique word is represented by a column in a matrix generated by Count Vectorizer, and each sample of text from the document is represented by a row in the matrix.

This is the reason for using the CountVectorizer () instead of TF-IDF. This is because we do not want to down-weight the presence of an actor/director if he or she has acted or directed in relatively more movies. It does not make much intuitive sense.

Algorithms and results:

Content-based Recommendation system:

- Content-based filtering methods are based on the description of an item and a profile of the user's preferred choices.
- In content-based filtering, keywords are used to describe the items, whereas a user profile is built to state the type of item this user likes.
- For example, if a user likes to watch movies such as Mission Impossible, then the recommender system recommends movies of the action genre or movies of Tom Cruise.
- The critical premise of content-based filtering is that if you like an item, you will also like a similar item. This approach has its roots in information retrieval and information filtering research.
- One of the main advantages of the collaborative filtering approach is that it can recommend complex items accurately, such as movies, without requiring an understanding of the item itself as it does not depend on machine-analyzable content.

Since we do not have rating values to rely on, this is the best recommender system to use for this dataset.

Based on the plot summaries for each film, we will compute pairwise similarity scores and make recommendations for films.

We have used the TD-IDF metric for the first approach in the content-based recommended system whereas the count vectorizer for the second approach.

First approach:

Now that we have the TD-IDF matrix, we can calculate a similarity score. There are other ways to do this, including using the Euclidean, Pearson, and cosine similarity scores. Which score is the best? There is no correct response. Different scores perform well in various contexts.

To determine the degree of resemblance between two films, we shall compute the cosine similarity. Since it is independent of magnitude and can be calculated rather quickly, we utilize the cosine similarity score.

Mathematically, it is defined as follows:

$$similarity = cos(x, y) = x \,.\, y \,/\, ||x|| * ||y||$$ *(Dot product A.B)*

Since we have used the TF-IDF vectorizer, calculating the dot product will directly give us the cosine similarity score.

Therefore, we will use sklearn's linear_kernel() instead of cosine_similarities() since it is faster.

We've created a function that accepts a movie title as an input and returns a list of the ten films that are most comparable to it.

```
get_recommendations('Black Panther')
```

```
228                The November Man
691                 So Not Worth It
2549                      John Henry
3855                      The Writer
6402                            Cake
2944          Who Killed Malcolm X?
8471                The Pyramid Code
98        Octonauts: Above & Beyond
6570          Daughters of the Dust
4366                        Fugitiva
```

```
get_recommendations('3 Idiots')
```

```
3463                         College Romance
3464                       Engineering Girls
4907                               Candy Jar
7521                               Mr. Young
259                                   Pahuna
3314    100 Things to do Before High School
6287                           Best Neighbors
5762                               Be with Me
2688                              Moms at War
3468                                Limitless
```

Figure 12: Results of the First Approach

Even while our system did a respectable job of locating movies with comparable narrative summaries, the recommendations' quality isn't that high. The movie "3 idiots" brings back similar-themed films (College Life). However, it fails if someone requests the same director or cast.

As a result, the model's performance should be enhanced by the addition of more metrics.

Second Approach:

Content-based filtering on multiple metrics. Content-based filtering on the following factors:

- Title
- Cast
- Director
- Listed in
- Description

The names and keyword instances would then be changed to lowercase, and any spaces between them would be removed. This is done to prevent our vectorizer from treating "Tony Anthony" and "Tony Stark" as the same Tony.

One significant distinction is that we employ the CountVectorizer() rather than TF-IDF. This is done so that the presence of an actor or director who has acted in or directed a disproportionately large number of films won't be devalued. It doesn't make a lot of sense logically.

```
get_recommendations_new('Black Panther', cosine_sim2)
```

```
143                Green Lantern
4773                 How It Ends
779             Battlefield Earth
7078                    Inkheart
949              The Darkest Hour
8025                   Singularity
6605    Di Renjie zhi Sidatianwang
6911       Halo: The Fall of Reach
7581                          Next
8512          The Space Between Us
Name: title, dtype: object
```

```
get_recommendations_new('3 Idiots', cosine_sim2)
```

```
4660                                                PK
8708                                 We're No Animals
8404    The Lord of the Rings: The Return of the King
4732                                 Rang De Basanti
2766                                           Maska
3147                                         Talaash
6915                                         Hangman
1022                                Taare Zameen Par
1115                              Ferrari Ki Sawaari
4507                                           Sanju
```

Figure 13: Results of the Second Approach

K-means Clustering:

One of the most popular methods for gaining a general understanding of the data's structure is clustering. It can be summed up as the process of finding data subgroups where data points in the same subgroup (cluster) are extremely similar and other data points in other clusters are very dissimilar. To put it another way, we look for homogeneous subgroups within the data so that the data points in each cluster are as comparable as feasible based on a similarity metric like the Euclidean-based distance or the correlation-based distance. Choosing the similarity metric to employ depends on the application.

Clustering is an unsupervised learning method to try to investigate the structure of the data by grouping the data points into distinct subgroups.

The iterative K-means algorithm attempts to divide the dataset into K unique, non-overlapping subgroups (clusters), each of which contains a single data point. While keeping the clusters as distinct (far) apart as possible, it aims to make the intra-cluster data points as comparable as possible. It distributes data points to clusters in a way that minimizes the sum of the squared distances between the data points and the cluster centroid, which is the average value of all the data points in the cluster. The homogeneity (similarity) of the data points within a cluster increase as the amount of variance within the cluster decreases.

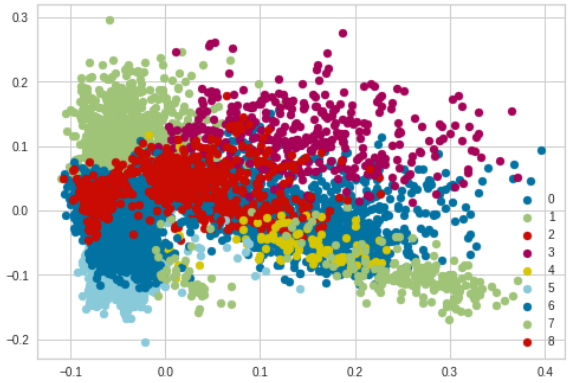

Figure 14: 9 clusters showing different genres of movies.

We have performed PCA for dimensionality reduction to reduce the dimensions of the dataset before doing the clustering for better results. We chose to have n_components=3000 which will cover the 80% of the variance.

Interpretations from the clusters are mentioned below in the form of word clouds.

Figure 15: Cluster 0 indicates Drama.

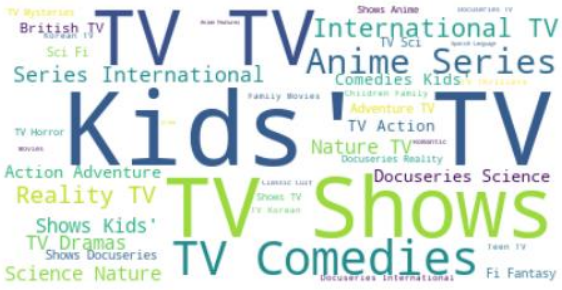

Figure 16: Cluster 1 indicates kids tv and anime.

Figure 17: Cluster 2 indicates international movies in general.

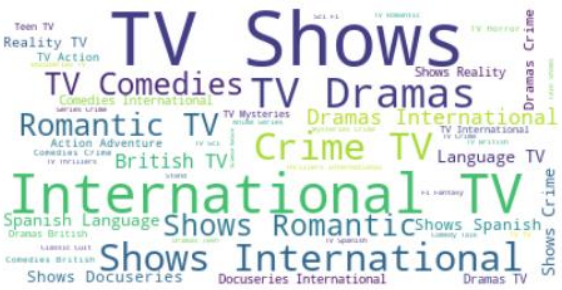

Figure 18: Cluster 3 indicates all types of Comedy, Crime TV shows.

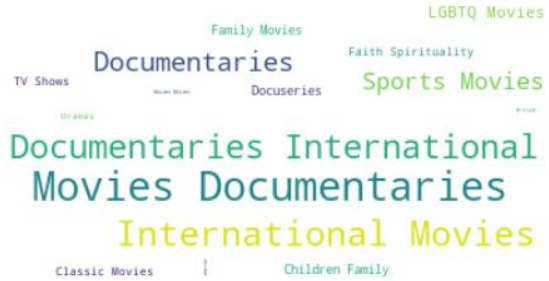

Figure 19: Cluster 4 indicates documentaries.

Figure 20: Cluster 5 indicates Romantic Korean Tv shows.

Figure 20: Cluster 6 indicates Children and Families.

Figure 21: Cluster 7 indicates Musical.

Figure 22: Cluster 8 indicates Action, Adventure.

Agglomerative hierarchical Clustering:

In hierarchical clustering, each group (or "node") relates to two or more succeeding groups in a cluster tree (a dendrogram) used to display the data. The groupings are nested and arranged in a tree-like structure, which should produce a useful categorization system.

Agglomerative Each data point is treated as a separate cluster, also known as a leaf, and is where clustering or bottom-up clustering effectively begins. From there, each cluster determines how far apart it is from the others. We referred to this as a node since it resulted from the merger of the two clusters that were closest to one another. In order to calculate the distance between a member of their cluster and a cluster outside of their own, newly created clusters do so once again. After assigning each data point to the root cluster, the procedure is repeated.

In this algorithm, we have used Euclidean distance and ward linkage as metrics to fit the data. Single and complete linkages are giving not that great results in forming the clusters compared to ward linkage. So, for analysis ward linkage is used.

We have used the elbow method to decide the optimal number of clusters.

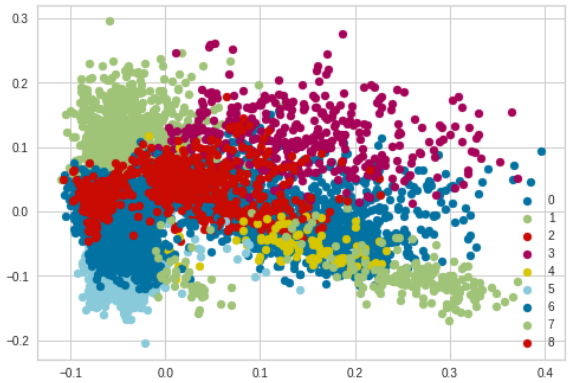

Figure 23: Clusters using Agglomerative algorithm.

Conclusion and Future work:

To conclude, from this project we performed content-based filtering to recommend movies and tv shows based on text. Different approaches are also performed in that for better results. A more metric approach works better than using only one metric. A precision score of 0.9 is obtained using more metrics when compared to one metric with 0.85. Apart from recommendations, we have performed an analysis based on clustering for this dataset which could help us in recommending movies in the future. Good insights are also derived through this approach.

In the future, we can include the rating column and users column in the dataset which contains numerical values, so that it will be easy for recommending using collaborative filtering and better results can be obtained by using more metrics. This above factor also helps us to recommend movies through model-based algorithms which are clustering. We could just analyze the clusters, but proper recommendations can’t be made, so those values help in giving accurate results through this form as well.

Our advice to DS 5230 students is to have a bigger group to discuss on different unsupervised machine learning algorithms and to implement them and discuss regarding the results they obtained after applying dimensionality reductions and clustering. This will help the students to know when we can apply the algorithms.