UE19CS332
AIWIR Project Report

Recommendation Systems using Netflix Movies and TV shows Dataset

Team:

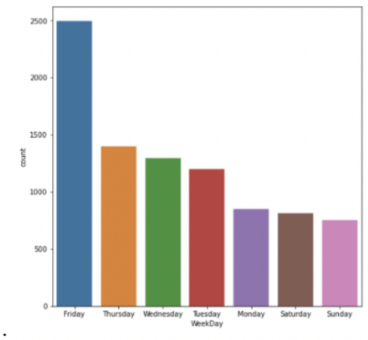
Aarushi Agarwal	PES2UG19CS004
Anwesha Kelkar	PES2UG19CS058
Arjun Harish	PES2UG19CS062

Introduction:

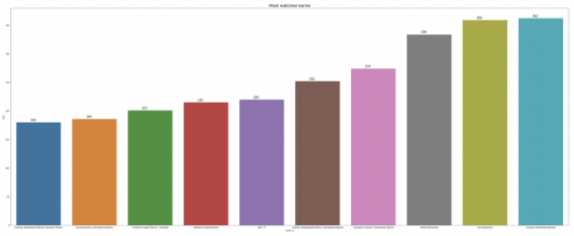
A recommendation system using two approaches of context-based modelling is built and analysed. The dataset used here is that of “Netflix TV shows and movies” consisting of records of over 6000 movies and 3000 TV Shows. While one model explores various features like director, country of origin, cast, and genre the other leverages keywords from the description provided by Netflix to recommend a TV Show/Movie to a user. The assumption is that the input to the model is a TV show/movie that the user has already watched and likes. We utilised concepts of cosine similarity that best fit the algorithms and the data to implement the models and compare them. We see that recommendations given based on features like director, country of origin, cast, and genre are more fruitful to a user.

Preprocessing and EDA:

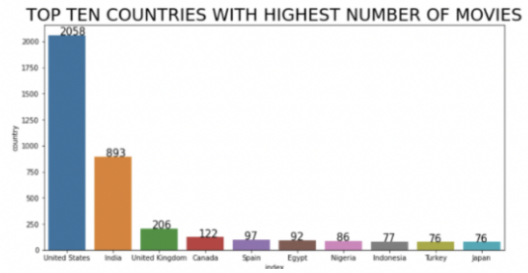
Exploratory data analysis was performed to understand user behavior in order to properly recommend movies and TV shows. During preprocessing, we detected the existing NULL values present and removed all the corresponding records for the same in order to avoid any discrepancies. Post this, our dataset was ready to use. Our EDA suggests that most of the content in Netflix is movies, of which, most of it is generated from the United States. The top genres are Drama and International Movies. Also, the majority of viewers use Netflix to watch Movies or TV-Shows on Fridays.



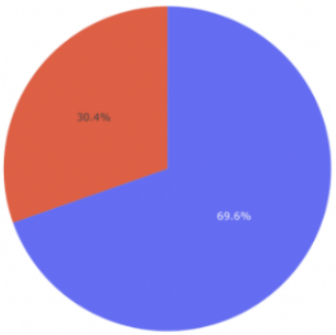
Most of the users use Netflix to watch TV-Show or Movies on Fridays.



Drama and International Movies are the most-watched Genres



The United States has the highest contribution to movies.



69.6% content is Movies and 30.4% Content is TV-Shows

First Model Approach:

There are five features of the dataset that we will make use of in the approach director, country, cast, and listed in which is nothing but the genre. The Input is a single movie or TV show that the user currently likes and the output is a list of five other movies or TV shows that he can watch next based on these features that we've considered. The idea is to convert the data of these into binary data frames such that these combined values of the features can be compared for each movie. We can then compute a similarity between them and select the top five similarities as our result. The similarity measure that we made use of is cosine similarity which was explained previously.

Second Model Approach:

Netflix and countless other product-based companies also give content descriptions. In the case of Netflix, this content is usually the summary of the movie or the tv show. These Netflix descriptions or synopses are concise and usually something that captures the eye of the audience. In some cases, this is taken positively by the public and in other cases, fans are offended as these descriptions are not apt for the said movie/TV show. Our second approach takes advantage of these descriptions. We extract all the words from the description and convert them into binary data frames for easier comparison. This can be easily achieved using natural language processing libraries such as Punkt from nltk package. This is done for each record. We then compare these and find the movies that have the most words in common, this again is done using a similarity measure called cosine similarity.

Our proposed solution makes use of python3 and related libraries to preprocess our dataset, visualize it and build the recommendation model.

Code:

Recommender model 1:

```
def recommender_model_1(search):
    cs_list = []
    binary_list = []
    if search in movies['title'].values:
        idx = movies[movies['title'] == search].index.item()
        for i in binary.iloc[idx]:
            binary_list.append(i)
        point1 = np.array(binary_list).reshape(1, -1)
        point1 = [val for sublist in point1 for val in sublist]
        for j in range(len(movies)):
            binary_list2 = []
            for k in binary.iloc[j]:
                binary_list2.append(k)
            point2 = np.array(binary_list2).reshape(1, -1)
            point2 = [val for sublist in point2 for val in sublist]
            dot_product = np.dot(point1, point2)
            norm_1 = np.linalg.norm(point1)
            norm_2 = np.linalg.norm(point2)
            cos_sim = dot_product / (norm_1 * norm_2)
            cs_list.append(cos_sim)
        movies_copy = movies.copy()
        movies_copy['cos_sim'] = cs_list
        results = movies_copy.sort_values('cos_sim', ascending=False)
        results = results[results['title'] != search]
        top_results = results.head(5)
        return(top_results)
    elif search in tv['title'].values:
        idx = tv[tv['title'] == search].index.item()
        for i in binary2.iloc[idx]:
            binary_list.append(i)
        point1 = np.array(binary_list).reshape(1, -1)
        point1 = [val for sublist in point1 for val in sublist]
        for j in range(len(tv)):
            binary_list2 = []
            for k in binary2.iloc[j]:
                binary_list2.append(k)
            point2 = np.array(binary_list2).reshape(1, -1)
            point2 = [val for sublist in point2 for val in sublist]
            dot_product = np.dot(point1, point2)
            norm_1 = np.linalg.norm(point1)
```

```
            norm_2 = np.linalg.norm(point2)
            cos_sim = dot_product / (norm_1 * norm_2)
            cs_list.append(cos_sim)
        tv_copy = tv.copy()
        tv_copy['cos_sim'] = cs_list
        results = tv_copy.sort_values('cos_sim', ascending=False)
        results = results[results['title'] != search]
        top_results = results.head(5)
        return(top_results)
    else:
        return("Title not in dataset. Please check spelling.")
```

Recommender Model 2:

```
def recommender2(search):
    cs_list = []
    binary_list = []
    if search in movies_des['title'].values:
        idx = movies_des[movies_des['title'] == search].index.item()
        for i in movie_word_binary.iloc[idx]:
            binary_list.append(i)
        point1 = np.array(binary_list).reshape(1, -1)
        point1 = [val for sublist in point1 for val in sublist]
        for j in range(len(movies_des)):
            binary_list2 = []
            for k in movie_word_binary.iloc[j]:
                binary_list2.append(k)
            point2 = np.array(binary_list2).reshape(1, -1)
            point2 = [val for sublist in point2 for val in sublist]
            dot_product = np.dot(point1, point2)
            norm_1 = np.linalg.norm(point1)
            norm_2 = np.linalg.norm(point2)
            cos_sim = dot_product / (norm_1 * norm_2)
            cs_list.append(cos_sim)
        movies_copy = movies_des.copy()
        movies_copy['cos_sim'] = cs_list
        results = movies_copy.sort_values('cos_sim', ascending=False)
        results = results[results['title'] != search]
        top_results = results.head(5)
        return(top_results)
    elif search in tv_des['title'].values:
        idx = tv_des[tv_des['title'] == search].index.item()
        for i in tv_word_binary.iloc[idx]:
            binary_list.append(i)
        point1 = np.array(binary_list).reshape(1, -1)
        point1 = [val for sublist in point1 for val in sublist]
        for j in range(len(tv)):
            binary_list2 = []
            for k in tv_word_binary.iloc[j]:
                binary_list2.append(k)
            point2 = np.array(binary_list2).reshape(1, -1)
            point2 = [val for sublist in point2 for val in sublist]
            dot_product = np.dot(point1, point2)
            norm_1 = np.linalg.norm(point1)
            norm_2 = np.linalg.norm(point2)
```

```
            cos_sim = dot_product / (norm_1 * norm_2)
            cs_list.append(cos_sim)
        tv_copy = tv_des.copy()
        tv_copy['cos_sim'] = cs_list
        results = tv_copy.sort_values('cos_sim', ascending=False)
        results = results[results['title'] != search]
        top_results = results.head(5)
        return(top_results)
    else:
        return("Title not in dataset. Please check spelling.")
```

Experimental Results:

After implementing the proposed methods of building the recommendation models, we tested both models using the same inputs. For both models, the input is the Bollywood movie "Kal Ho Na Ho".

Using the first model i.e recommendation based on features like director, country, cast, and genre we get five other Bollywood movies originated from India with very similar genres. This is a satisfactory result based on our personal experience and public opinion.

Using the second model i.e recommendation based on the keywords in the description we get five movies, some of which are Bollywood and some are Hollywood. This result is not satisfactory according to our personal experience and public opinion. The reason behind this is that the descriptions may have similar keywords but not be related at all. as seen in this example.

Output screenshots:

Recommender model 1:

```
[1] recommender_model_1('Kal Ho Naa Ho')
```

	title	director	cast	country	rating	listed_in	cos_sim
1057	Soldier	Abbas Mustan	Rakhee Gulzar, Bobby Deol, Preity Zinta	India	TV-14	Comedies, Dramas, International Movies	0.553399
3619	Chal Dhar Pakad	Aatmaram Dharne	Nagesh Bhonsle	India	TV-14	Comedies, Dramas, International Movies	0.530330
4079	Irada Pakka	Kedar Shinde	Smita Jaykar, Siddarth Jadhav	India	TV-14	Comedies, Dramas, International Movies	0.527645
4184	Kya Kehna	Kundan Shah	Preity Zinta, Saif Ali Khan, Anupam Kher, Fari...	India	TV-PG	Dramas, International Movies, Romantic Movies	0.505181
718	AK vs AK	Vikramaditya Motwane	Anil Kapoor, Anurag Kashyap	India	TV-MA	Comedies, Dramas, International Movies	0.500000

Recommender model 2:

```
recommender2('Kal Ho Naa Ho')
```

	title	description	description_filtered	cos_sim
161	Team America: World Police	In this musical satire, an all-marionette police force takes on the challenging role of keeping the peace on a troubled planet.	in musical satire , all-marionette police force takes challenging role keeping peace troubled planet .	0.410993
577	Mariposa	New student Acha falls for Iqbal, a high-achieving student who's torn between love and family pressure.	new student acha falls iqbal , high-achieving student 's torn love family pressure .	0.410242
202	Midnight Sun	Born with a fatal sensitivity to sunlight, a sheltered teen girl falls for her neighbor, but hides her condition from him as their romance blossoms.	born fatal sensitivity sunlight , sheltered teen girl falls neighbor , hides condition romance blossoms .	0.407223
1157	Chashme Baddoor	When pretty new neighbor Seema falls for their shy roommate Sid, jealous womanizers Omi and Jai plot to break up the new lovebirds.	when pretty new neighbor seema falls shy roommate sid , jealous womanizers omi jai plot break new lovebirds .	0.403757
3629	Chashme Buddoor	When pretty new neighbor Seema falls for their shy roommate Sid, jealous womanizers Omi and Jai plot to break up the new lovebirds.	when pretty new neighbor seema falls shy roommate sid , jealous womanizers omi jai plot break new lovebirds .	0.403757