Content Based Movie Recommender System

Team: Rishav, Shivanshu Kumar, Dishant Yadav, Raman Yadav

1. Motivation: The online streaming services are one of the most important things on the internet and have over a million things for one to watch. There is a need for a smart system that looks over the choices of the user, simplifies the search space and recommends the best possible things available that suits interest.

2. Literature Review:

The papers bereaved below talk about different types of implementations one can have for a movie recommendation system, namely Content-Based, Collaborative-Filtering based or Hybrid based system.

a) A research paper, by prof. Manoj Kumar, prof. DK Yadav, prof. Vijay KR Gupta and Mtech. Ankur Singh, published in 2015 explains the technique of content-based filtering used for recommendation of movies where not only the likes/dislikes are considered, but also the past history of the user (in the form of ratings etc) is taken into account for predicting the best set of movies. The algorithm seen in implementation after the filtering is the K-means clustering algorithm where based on the number of votes (or rating) provided for a movie, is clustered accordingly and picked up for the user's recommendation.
For reference:
https://drive.google.com/file/d/1QiTcXJsP0zSlRCEGUYMgufbEnrS0Im20/view?usp=sharing

b) A research paper, by Prince Praveen, Praveen Goud and Sagar Parmar, published in 2020 explains the concept of a Movie Recommendation System using the technique of Neural Network (other than the previously described K-Means Clustering algorithm) where a multi layer perceptron, each made out of a matrix of the useful attributes, is used to establish a relation between the user's choice and the recommended choice (using a similarity algorithm in ML), and get's improved at each step by the gradient descent algorithm. The matrix containing the best possible recommendations serves as an output to the algorithm.
For reference:
https://drive.google.com/file/d/1ThKM5goUIvB1jfExjgpYG9vlvTeX9I5g/view?usp=sharing

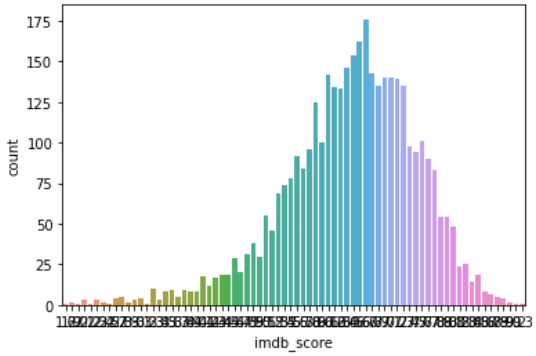
3. Dataset Description:

The dataset consisted of 5000 data entries with over 25 different features, useful among them being the *movie_title, director_name, plot_keywords, actors_name, genres*.

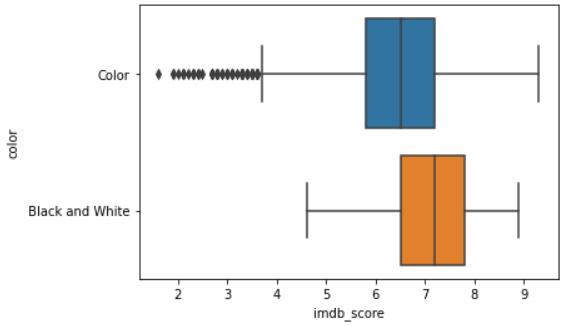
The visualizations for the dataset is done in a few ways:

a) The count vs. imdb_scores countplot, a color vs. imdb_score box plot and a genre word cloud were generated using the inbuilt python library *NLTK, Seaborn* and *Matplotlib's* functions. Here are the plots:

Count-Plot (imdb score vs count)



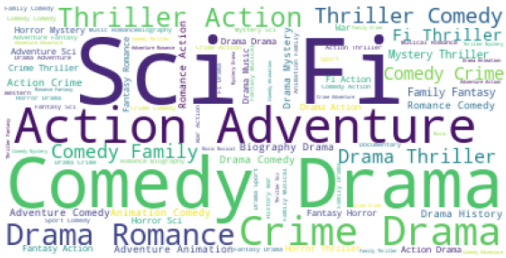
Box-Plot (imdb score vs color of movie)



b) The dataset required cleaning because it contained NaN (Null/empty/irrelevant) values at many places, which were removed using an inbuilt python library pandas' function dropna().

Dataset:https://drive.google.com/file/d/1WFaJ5CtcV9NghYbsbRYSbaTO-MbWBvLu/view?usp=sharing

Word Cloud for genres of movies



4. Methodology: The implementation for the content-based recommendation system are divided into three steps:

a) **Data Cleaning**: The dataset was big and contained too much non-usable data. So we had to remove unnecessary features. Firstly, we deleted the null data-points then the NaN data-points. Also there were some features that needed to be merged like director's name, actor's name, genres, plot keywords to one attribute named tags, which individually had to be cleared for the spaces between names etc. Then we deleted the stop words, removed irrelevant symbols, commas etc. In the movie descriptions, some words contained duplicacy; for that filtration we use *nltk.stem.porter* library.

b) **Text to Vector embedding**:
We converted the tags of every movie to vectors so that we can use them in further processes. We used the function CountVectorizer() in *sklearn.feature_extraction.text* library for this. After this we filtered the top 4000 words from the bag of unique words. These will be our vectors for the cosine-similarity.

c) **Model implementation**: Our model (the cosine-similarity) will take the name of the movie which the user will give as an input and it will suggest the top 10 movies most related to the input movie, based on the highest similarities in the respective similarity matrix; for that the (*sklearn.metrics.pairwise*) library was used. We used the cosine similarity method which compares the tag vectors from the input movie and finds the movies having the maximum of the similarity between the vectors that are generated on the basis of the unique words appearing in the description for each of the movies.

5. Related Work:

a) Project by MC Keshava et al focuses mainly on the idea that a movie is recommended to the user based on the past reputation of the movie, including the features such as the rating and popularity etc.
https://www.researchgate.net/publication/341876202_Machine_Learning_Model_for_Movie_Recommendation_System

b) Project by Ms A. Kashyap et al focuses on the idea of recommending the movies to the user based on the likes/dislikes of the user besides the ratings and popularities of the movie.
https://ijesc.org/upload/f0d1e3f5683da81c9018ff3308495420.A%20Movie%20Recommender%20System%20MOVREC%20using%20Machine%20Learning%20Techniques.pdf

c) Project by N. Smitha et al besides taking into account the features like popularity, ratings, likes/dislikes, it also focuses on the idea of the previous choices of the user before recommending the appropriate set of movies/shows to him/her.
https://ieeexplore.ieee.org/document/9388619

6. Results/Analysis:

The model is successfully suggesting the best movies (as we could evaluate from human perception since this model doesn't have any real evaluation metric) which are related to the movie that are given as the input, for example, the movie 'The Dark Knight Rises' is similar to the below recommended movies:

```
recommend('The Dark Knight Rises')
```

```
Premium Rush
Inception
The Devil's Own
Deadfall
The One
Die Hard 2
Lethal Weapon 3
Live Free or Die Hard
Triple 9
Under Siege 2: Dark Territory
```

7. Timeline:

Week 1-2: Data Collection & Preprocessing

Week 3: Features Selection

Week 4-6: Model Selections with the help of some domain knowledge

Week 7-10: Model tryouts

8. Individual Tasks:
Equally contributed by everyone (mostly work done during meetings and calls).

(Refer here for Colab notebook
GitHub Link)