Capstone 3: Final Report: Movie Recommendation System.

Context of the project

Everyone loves movies irrespective of age, gender, race, colour, or geographical location. We all are connected via this fantastic medium. Yet what is most interesting is how unique our choices and combinations are regarding movie preferences. A Recommendation System is a filtration program whose prime goal is to predict a user's “rating” or “preference” towards a domain-specific item or item. In our case, this domain-specific item is a movie. Recommendation systems also learn your viewing patterns and may suggest content based on the time of the day or your watching patterns. The recommender system helps users quickly find their preferred movies without searching from the extended content catalogue. From a business perspective, the more relevant content, or movies a user finds on any platform, the higher their engagement and, as a result, increased revenue.

Problem Statement

How can data from a movie-based platform recommend other movies to users based on their preferences and activities on that platform?

The Dataset

The dataset contains the metadata (cast, crew, budget, etc..) of movies. Kaggle has removed the original version of this dataset per a DMCA takedown request from IMDB. To minimise the impact, Kaggle replaced it with a similar set of films and data fields from The Movie Database (TMDb) in accordance with their terms of use. There will be two datasets used: credits_data and movies_data. Both are CSV files, and a quick overview of the datasets are below:

	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_com
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": "Ing Film Partner
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 Pictures", "id":
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": "Co Pictures",
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	en	The Dark Knight Rises	Following the death of District Attorney Harve...	112.312950	[{"name": "Leg Pictures", "id
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	en	John Carter	John Carter is a war-weary, former military ca...	43.926995	[{"name": "Walt Pictures",

My approach to the problem

In this project, I built a recommendation system to filter and predict only the movies a user prefers.
There are three types of recommendation systems.

- Demographic Filtering: The recommendations are the same for every user. They are generalised, not personalised. These types of systems are behind sections like “Top Trending”.
- Content-based Filtering: These suggest recommendations based on the item metadata (movie, product, song, etc.). Here, the main idea is that if a user likes an item, they will also like items like it.
- Collaboration-based Filtering: These systems make recommendations by grouping users with similar interests. For this system, metadata of the item is not required.

In this project, I built a content-based and demographic recommendation engine for movies. The approach to building the movie recommendation engine consists of the following steps.

1. Perform Exploratory Data Analysis (EDA) on the data
2. Build the recommendation system
3. Get recommendations

My findings

Data Wrangling and EDA on the dataset

The first dataset contains the following features:

- movie_id - A unique identifier for each movie.
- Cast - The name of the lead and supporting actors.
- Crew - The name of the Director, Editor, Composer, Writer etc.

The second dataset has the following features:

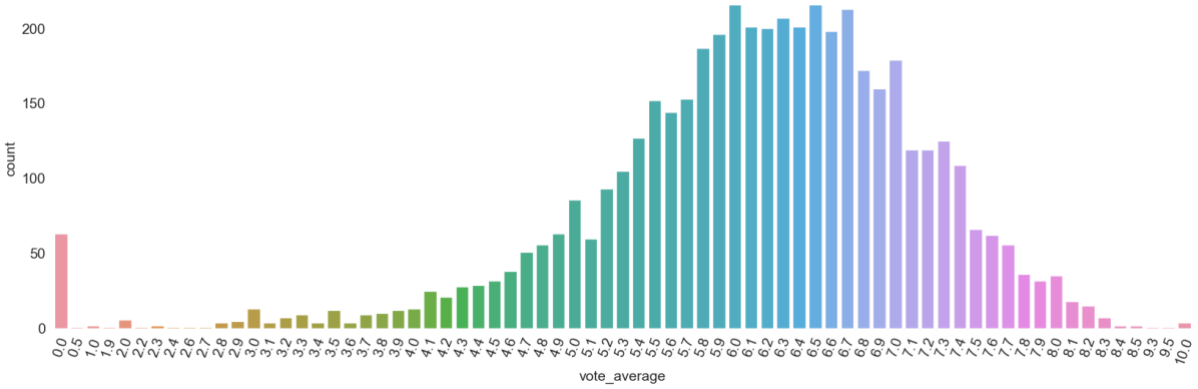
- budget - The budget in which the movie was made.
- Genre - The movie genre, Action, Comedy, Thriller etc.
- Homepage - A link to the homepage of the movie.
- id - This is, infact the movie_id as in the first dataset.
- Keywords - The keywords or tags related to the movie.
- original_language - The language in which the movie was made.
- original_title - The title of the movie before translation or adaptation.
- Overview - A brief description of the movie.
- Popularity - A numeric quantity specifying the movie popularity.
- production_companies - The production house of the movie.
- production_countries - The country in which it was produced.
- release_date - The date on which it was released.
- Revenue - The worldwide revenue generated by the movie.
- Runtime - The running time of the movie in minutes.
- Status - "Released" or "Rumored".
- Tagline - Movie's tagline.
- title - Title of the movie.
- vote_average - average ratings the movie received.
- vote_count - the count of votes received.

I merged both datasets for easier manipulation of the data. There were no missing values in the datasets. I did remove a column of duplicates of the titles of the movies.

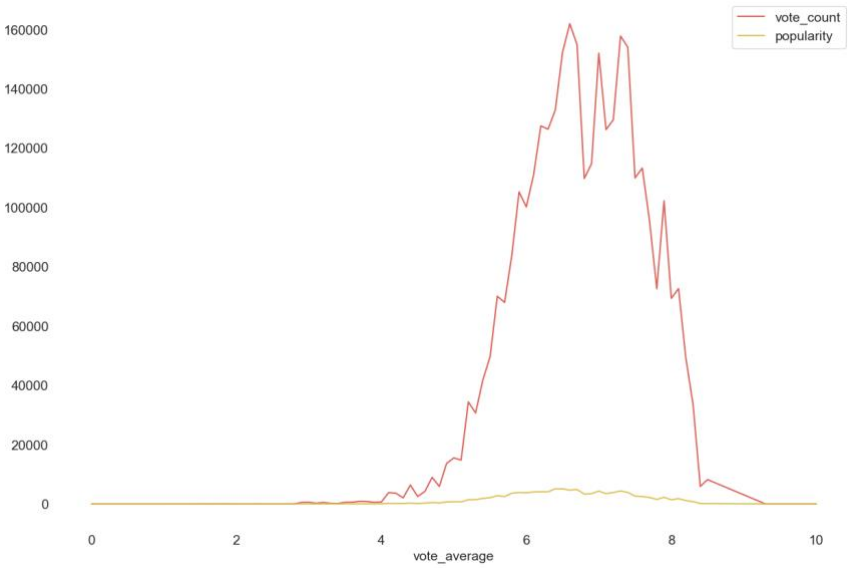
	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_com
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "In Film Partner
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 Pictures", "id":
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": "C Pictures",
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	en	The Dark Knight Rises	Following the death of District Attorney Harve...	112.312950	[{"name": "Leg Pictures", "id
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	en	John Carter	John Carter is a war-weary, former military ca...	43.926995	[{"name": "Walt Pictures",

5 rows × 23 columns

I analysed how the data was spread regarding the vote average in the bar plot below.



Findings: In the line plot below, it is evident that the most popular movies have the most votes even though the average vote range from 5 to 7.5. Movies with a vote average of 9 to 10 are less popular than movies with a vote count in the range of 5 to 7.



The accuracy of predictions made by the recommendation system can be adapted using the “plot/description” of the movie. But the quality of suggestions can be further improved using the movie's metadata. Let’s say the query to our movie recommendation system is “The pirates of the Caribbean”. Then the predictions should also include movies directed by the film's director, and it should also have movies with the cast of the given query movie. I utilised the following features to personalise the recommendation: cast, crew, keywords, and genres. The movie data is in the form of lists containing strings, and I would need to convert the data into a safe and usable structure. I applied the literal_eval() function to those features and created a soup of all the metadata information extracted to input into the vectoriser.

Pre- Processing and Modelling

My original idea was to create a recommendation system focused on Content-based filtering, but I will also include demographic filtering. I will not be including collaboration-based filtering because I do not have a userID, which is necessary for collaborative filtering, and therefore another dataset would have to be used.

Content-Based Filtering

Plot description-based Recommender

I converted the word vector of each overview. I used the Term Frequency-Inverse Document Frequency (TF-IDF) vectors for each overview. Frequency is related to the frequency of a word in a document. Inverse Document Frequency is the relative count of documents containing the term given as a 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 you a matrix where each column represents a word in the overview vocabulary (all the words that appear in at least one document), and each row represents a movie, as before. This reduces the importance of words frequently occurring in plot overviews and their significance in computing the final similarity score. To save time, TfIdfVectorizer from scikit-learn was used.

With this matrix in hand, we can now compute a similarity score. Many similarity methods could be used, such as the Euclidean, Pearson and cosine similarity scores. I used the cosine similarity to calculate a numeric quantity that denotes the similarity between two movies. I am using the cosine similarity score since it is independent of magnitude, relatively easy, and fast to calculate. I defined a function that takes a movie title as an input and outputs a list of the ten most similar movies. I also needed a mechanism to identify the index of a move in the metadata Data frame, given its title.

```
In [22]: get_recommendations('The Dark Knight Rises')

Out[22]: 65                                The The Dark Knight
         299                                Batman Forever
         428                                Batman Returns
         1359                                     Batman
         3854    Batman: The Dark Knight Returns, Part 2
         119                                Batman Begins
         2507                                   Slow Burn
         9            Batman v Superman: Dawn of Justice
         1181                                       JFK
         210                                Batman & Robin
         Name: title_x, dtype: object
```

This system needs to do a better job with recommendations and returns movies of the same/similar title instead of broadening the range.

Our recommendation system would be improved if there were better metadata. I want to explore and build a recommendation system based on the following metadata: the director, related genres, actors and movie plot keywords. Therefore, developing a credit, genres, and keywords-based recommender was the better option.

Credits, Genres and Keywords-Based Recommender
I applied the same/similar functions as the above recommender and then computed the Cosine Similarity matrix based on the count_matrix. I then reused the get_recommendations() function by passing in the new cosine_sim2 matrix as the second argument.

```
In [36]: get_recommendations('The Dark Knight Rises', cosine_sim2)

Out[36]: 65               The Dark Knight
         119                Batman Begins
         4638    Amidst the Devil's Wings
         1196                The Prestige
         3073           Romeo Is Bleeding
         3326              Black November
         1503                      Takers
         1986                      Faster
         303                     Catwoman
         747               Gangster Squad
         Name: title_x, dtype: object
```

Now, the recommendation system is much more successful in capturing more information due to the metadata and have given better recommendations.

Demographic Filtering
For this demographic filter, I needed a metric to score or rate a movie, calculate the score for every movie, sort the scores and recommend the best-rated movies to users. To do this, I used IMDB's weighted rating (wr), which is given as follows:

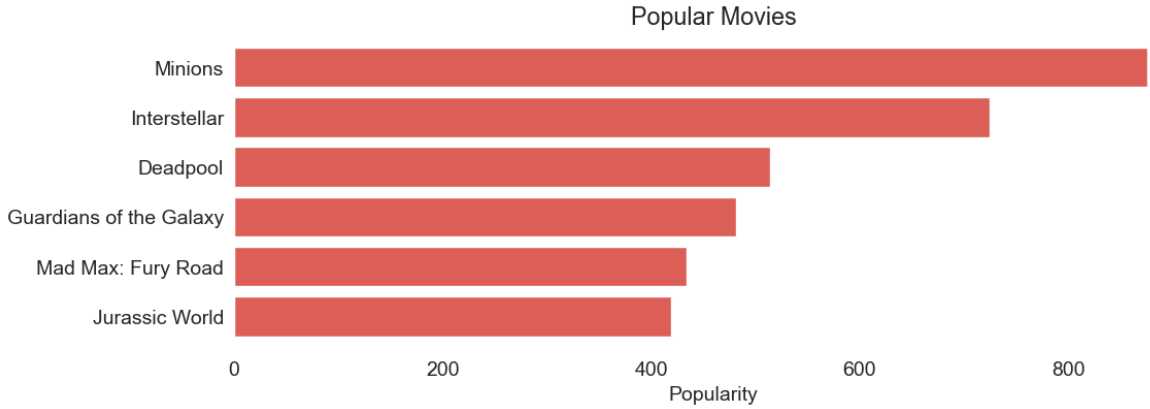
$$\text{Weighted Rating (WR)} = \left(\frac{v}{v+m}.R\right) + \left(\frac{m}{v+m}.C\right)$$

- v is the number of votes for the movie;
- m is the minimum votes required to be listed in the chart;

- R is the average rating of the movie; And
- C is the mean vote across the whole report

The mean rating for all the movies is approximately six on a scale of 10. The next step was determining an appropriate value for m, the minimum votes required to be listed in the chart, and I will use the 90th percentile as our cut-off. In other words, for a movie to feature in the charts, it must have more votes than at least 90% of the film in the list.

I defined a function called weighted_rating() and defined a new feature score, which calculates the value by applying this function to the data frame of qualified movies. After further processing, I was able to create a plot of the most popular movies as below:



Recommendations

Demographic recommenders provide a general chart of recommended movies to all the users, and they are not sensitive to the interests and tastes of a particular user. Therefore, a more intuitive system, like Content-based filtering, would be much better with the use of cosine similarity to calculate a numeric quantity that denotes the similarity between two movies.

Further research into the topic

I would love to create a collaborative filtering-based recommendation engine, as I could have explored machine learning models for such a recommendation engine. Still, due to the limitations of my dataset that did not contain a UserID I could not have created one. Exploring and building a collaborative filtering-based recommendation system would be another project I would love to focus on next.