Movie Recommendation System

I - Introduction

a) Problem Statement & Objective

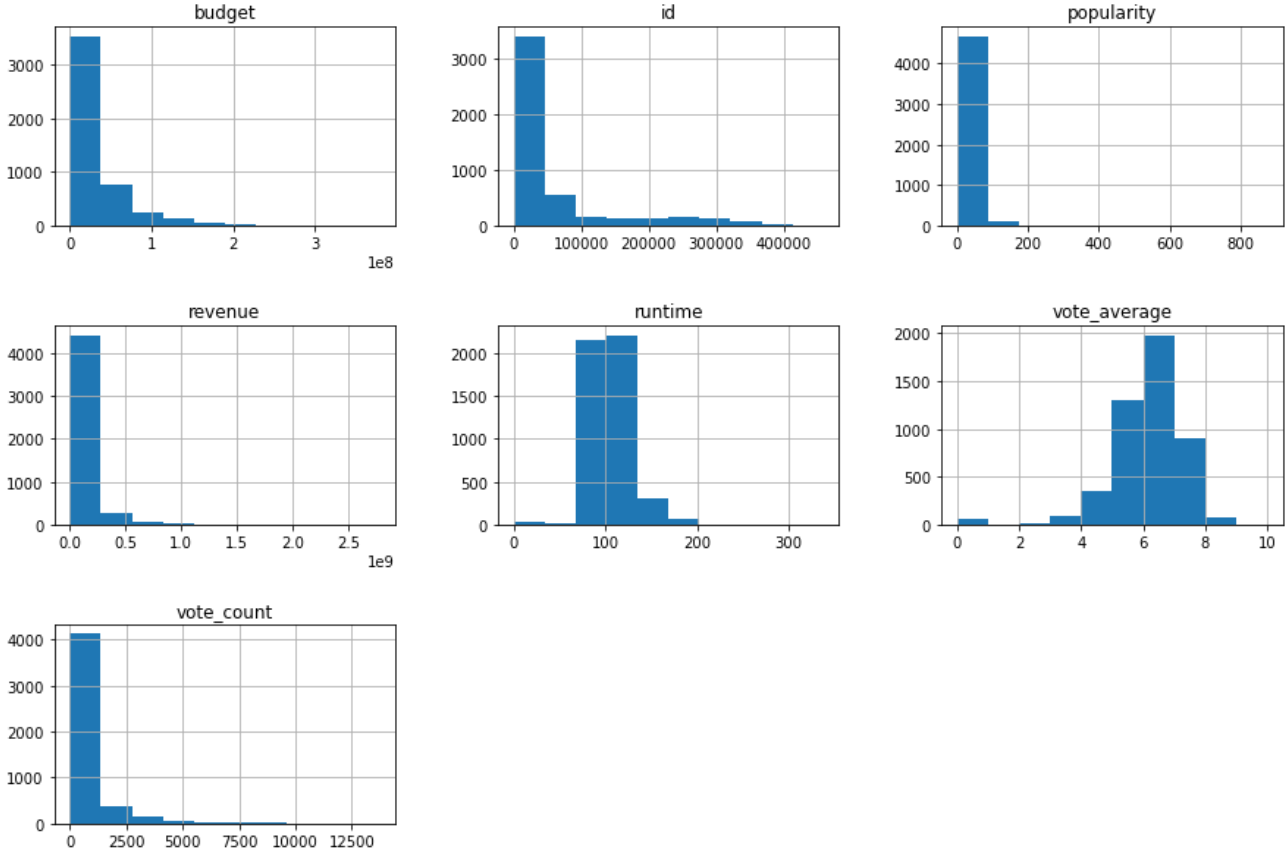
A movie streaming platform receives complaints about irrelevant movie recommendations from users. Improving their recommendation system is highly important to decrease the retention rate and keep their current subscribers more engaged.

This project aims to create a movie recommendation system that improves the quality of search results and provides more relevant movies according to users' search history.

II - Data Wrangling & EDA

a) Movies Dataset

Initially, there were some missing values on homepage, tagline, release date, overview, and runtime columns. I removed these columns from the dataset because they were irrelevant to our analysis. There was just one movie with a missing release data value. After looking up this movie, I saw that most of the information was missing. Therefore I dropped this movie altogether. Then I manually filled the missing values in overview and runtime columns by making a google search.



After graphically visualizing the data, I realized that 3376 of 4803 movies didn't make any revenue. That meant either this information is not available or they were not shown in theaters. I removed the revenue column from the dataset to make the analysis more coherent.

b) Credits Dataset

There were no null and duplicate values in the credit dataset.

III - Pre-processing & Modeling

Using cleaned data, I built demographic, content based and collaborative recommender systems.

a) Demographic Filtering

I used vote average as the only decisive factor to rank the movies. However, some movies have very low vote count which makes the ranking unfair. To solve the issue I used 99 percentile as my cut off. As a result, I included 481 movies in the ranking.

	title	vote_count	vote_average
1881	The Shawshank Redemption	8205	8.5
3337	The Godfather	5893	8.4
2294	Spirited Away	3840	8.3
662	Fight Club	9413	8.3
1818	Schindler's List	4329	8.3
3865	Whiplash	4254	8.3
3232	Pulp Fiction	8428	8.3
2731	The Godfather: Part II	3338	8.3
4601	12 Angry Men	2078	8.2
690	The Green Mile	4048	8.2

This ranking can be used for simple movie recommendation to users who wants to watch high ranked movies. It is not influenced by any user preference or choice therefore It will be basic and same for everyone.

b) Content Based Filtering

I built an engine that computes similarity between movies based on certain metrics and suggests movies that are most similar to a particular movie that a user liked.

Firstly, I used similarity scores of words in overview column that uses movie descriptions to recommend movies that includes the higher number of similar words. I excluded the most common english words from the vectorizer to increase the accuracy of the engine.

```
get_recommendations('The Godfather', cosine_sim)
```

```
2731     The Godfather: Part II
1873                  Blood Ties
867     The Godfather: Part III
3727                  Easy Money
3623                        Made
Name: title, dtype: object
```

I received similar results to The Godfather movie mostly but Easy Money and Made movies doesn't really have similar genre and cast with The Godfather.

I decided to add cast, crew and genre columns to my recommendation engine to increase the quality of the results.

```
get_recommendations('The Godfather', cosine_sim2)
```

```
2731      The Godfather: Part II
867      The Godfather: Part III
1525              Apocalypse Now
3012               The Outsiders
4637    Amidst the Devil's Wings
Name: title, dtype: object
```

Adding more features significantly improved the results. The new recommendation engine suggested movies with relevant genre and cast.

c) Collaborative Filtering

Content based recommendation system has some disadvantages. The engine that we built is not really personal in that it doesn't capture the personal tastes and biases of a user. Anyone querying our engine for recommendations based on a movie will receive the same recommendations for that movie, regardless of who s/he is.

Collabrative filtering aims to predict how much a user is expected to like a product by considering preferences of similar users. Therefore, I usec collabrative filtering to build a more personilzed recommendation engine considering users' preferences.

I did not implement Collaborative Filtering from scratch. Instead, I used the Surprise library that used powerful algorithms like Singular Value Decomposition (SVD) to minimise RMSE (Root Mean Square Error) and give good recommendations. SVD calculates similarty between both users and movies to give predictions which enables us to use advantages of both user and item based collabrative filtering.

After building the algorithm. I randomly picked a user to look at his/her past movie ratings and make predictions using collabrative filtering I created.

	userId	movieId	rating	title
396	2	165	3.0	Back to the Future Part II
2126	2	480	4.0	Monsoon Wedding
2400	2	497	3.0	The Green Mile
2460	2	500	4.0	Reservoir Dogs
2613	2	508	4.0	Love Actually
2699	2	509	4.0	Notting Hill
2777	2	539	3.0	Psycho
100	2	62	3.0	2001: A Space Odyssey
2902	2	550	3.0	Fight Club
2954	2	586	3.0	Wag the Dog
3083	2	587	3.0	Big Fish
3209	2	588	3.0	Silent Hill
3424	2	590	5.0	The Hours
3626	2	592	5.0	The Conversation
3822	2	593	3.0	Solaris
2913	2	585	5.0	Monsters, Inc.
316	2	161	3.0	Ocean's Eleven
2093	2	468	4.0	My Own Private Idaho
1803	2	377	3.0	A Nightmare on Elm Street

User 2 mostly rates action movies. Therefore I gave the recommendation engine one action and one romance movie to predict the rating based on the User 2’s past ratings.

	title	id
322	The Fifth Element	18

```
svd.predict(2, 322, 3)
```

```
Prediction(uid=2, iid=322, r_ui=3, est=3.7593585620966308, details={'was_impossible': False})
```

Estimated rating for The Fifth Element is 3.76.

```
movie=df['title']=='The Notebook'
df[movie][['title','id']]
```

	title	id
1559	The Notebook	11036

```
svd.predict(2, 11036, 3)
```

```
Prediction(uid=2, iid=11036, r_ui=3, est=3.4433906189646724, details={'was_impossible': False})
```

Estimated rating for The Notebook is 3.44.

User 2 rates Action movie The Fifth Element (3.76) higher than Romance movie The Notebook (3.44) which makes sense if we look at his/her rating history.

One great feature of this recommender system is that it doesn't care what metadata movie contains. It works purely on the basis of an assigned movie ID and tries to predict ratings based on how the other users have predicted the movie.

IV - Conclusion

For the movie streaming platform to give more relevant results to subscribers, two recommendation system could be implemented: Content based recommender and Collaborative filtering.

Content Based Recommender: The company can use this algorithm if they want to give recommendations to customers using similarities of movies such as overview, cast, crew and genre. This engine will be the same for all users and will not take into consideration users' past activity and individual preferences. This engine is a good option if the company doesn't have a lot of active users, therefore data, to process.

Collaborative Filtering: If the company would like to give more personalized recommendations according to users' likes, dislikes and view & search history, collaborative filtering is the best option. Also this engine is dynamic and improves as we collect more data about users. This option would be great if the company has significant number of active users which would make the recommendations more relevant.