Problem Statement

Movies taste is largely subjective. Thus, making a recommendation is an interesting challenge. One person's taste is not necessarily shared by everyone. For instance, one person may like superhero movies, but dislike movies focus on drama and dialog while another might like horror but not family films.

From a commercial perspective, entertainment is a hit driven industry. This means that a successful movie drives majority of the revenue. The return is high stake. To improve the effectiveness of the return on investment, such as paying actors, hiring the production crew, marketing and etc, I am attempting a movie recommendation system.

The movie recommendation system at the fundamental level is to recommend similar movies that a person rated highly. If successful, the algorithm could:

1. increase audience engagement for lesser known films
2. improve effectiveness on marketing

I will take the data hosted on Kaggle. The data set is the Movie Lens dataset (https://www.kaggle.com/rounakbanik/the-movies-dataset).

Data Wrangling

I downloaded the 5 datasets from the page. They are Movie Metadata, Credits, keywords, links, and ratings. Aside from some missing values, the data is clean in terms of errors or typos. The first step is to work with some of the data structure in some columns.

Some of the columns are originally in a Json format but converted to csv. Hence reading some of the columns will take the values as strings. For example, in the credits dataset, the cast column has all the cast in one row for each movie. I may want to only use the top N actors from the cast to be part of my movie recommendation engine.

To extract all the elements, say, cast for each movie and make it into a Python List, I applied the following custom function.

```
# Returns the list of names
def get_list(x):
    if isinstance(x, list):
        names = [i['name'] for i in x]
        return names
    #Return empty list in case of missing/malformed data
```

return []

The Movie Metadata dataset contains information on 45,000 movies featured in the Full MovieLens dataset. Features include posters, backdrops, budget, revenue, release dates, languages, production countries and companies.

The Credits dataset contains the cast and crew information.

The keywords dataset contains the movie id and the keywords relevant to the plot.

The ratings dataset contains each user's ratings that on the movies he or she has rated.

The link dataset is mapping the IMBD movie id and the id we are using.

I start with the Movie Metadata and transform the JSON stringfield column to a list for columns genre, production country. I also dropped 3 rows where genre is n/a. Due to memory limit, I took only a portion of the entire movie metadata.

Similarly, for the credit dataset, I extracted the top 5 actors and directors for each movie. After that, I joined the two processed datasets together on id.

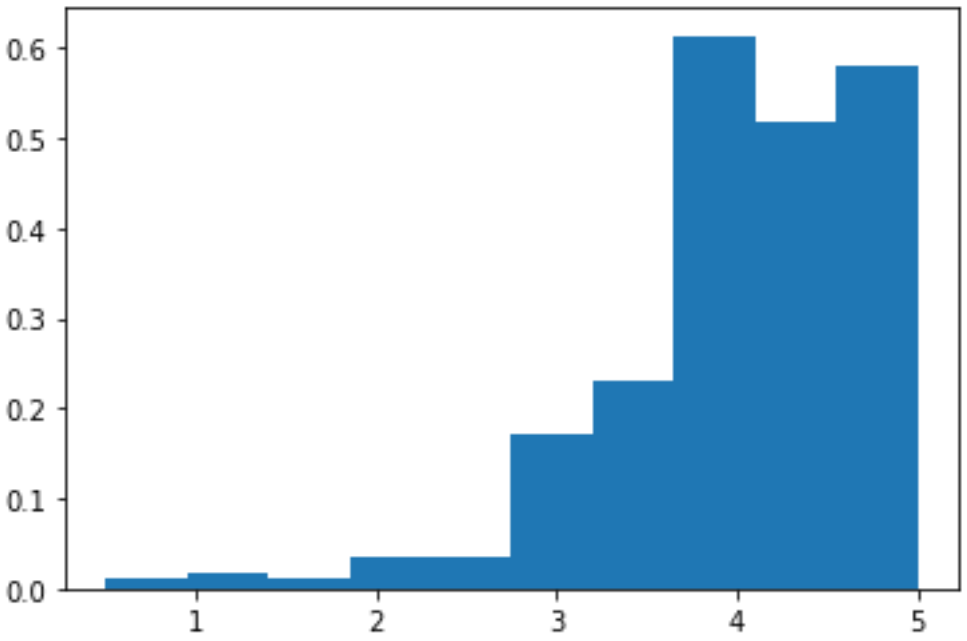
Data Analysis

Initial data exploration, I look at word cloud on title, genre, and the overview of the movie.

The title word cloud is below. The larger the text the more frequent it shows up in the title. The word cloud below shows that Love Man and Girl are the top 3 words to show up in the title.

In the genre world cloud, we can see that drama does come up a lot. The combination of Comedy Drama seems like to be the most produced genre. I am speculating because it is a popular genre.

Another exploratory analysis is with the rating data set. From looking at the overall rating histogram we see a right skewed distribution.



This is common on rating items. My guess is that the users tends to watch things they might intuitively think they would likely enjoy.

Data Modeling

For constructing my models, I have taken 3 approach. With movie recommendation systems, we can apply content base filtering, collaborative filtering or a hybrid model that combine the two previous methods.

Content base filtering, as its name suggest, is to make recommendation to the user based on the content that user have watched. Thus, maybe we may find a particular user would watch movies from a particular actor and that may be a significant signal that could help the engine find movies featuring this actor to recommend this specific user. Or we may find specific keywords in description of the movie that are similar to form a recommendation.

My first attempt at content-based filtering is to recommend based on keyword and overview. From those columns I applied TF-IDF. TF-IDF basically is a method that counts frequency of terms but also identify commonly used words like ‘the’ that has no significance to modeling and lessen their impact. From that we apply the cosine similarity scoring to identify how other movies is close to a given movie. Then the recommendation will be a list of top 30 movies in terms of similarity score.

Collaborative filtering on the other hand is based other people ratings to make a recommendation. I believe more sophisticated method would group similar users first and find recommendations for other users in that segment.

Hybrid methods are mixture of the methods mention above. The method I have adopted here is to take the recommended movie for a user via Content Base filtering first then run a collaborative filtering to predict the user’s rating to make recommendation.