MOVIE RECOMMENDATIONS

Background

When I first found out about the Netflix Prize competition in 2009, I was intrigued by the notion of building some sort of recommendation system using data. At that time, I was still an undergraduate student majoring in mathematics. I took some statistics class to better understand the theory behind the winner of the recommendation system, but I did not have the capability of writing any sort of computer code in any languages.

Fast forward to today, the advancement of Python language and its numerous libraries enable me to implement my theoretical statistical knowledge to create a recommendation system. There are numerous topics that I can create a recommendation system for, but for this project I choose movies because there are so many tutorials and different means to do it. That way, I am able to learn many different techniques to apply for different topics in the future. Also, watching movies is one of my hobbies.

Problem Statement

Many of the movie recommendation systems employed by various streaming platforms have been in mature stage, I believe that there is still room for improvement. There is one particular thing that bothers me when I search for a movie to watch, where the recommendations are too heavily influenced by the previous movies that I watched. Once in a while, I would like to watch a movie that is not in the same genre as the previous ones.

Outline

For this project, I will follow the same procedure as I did for the first capstone project.

1. Download the data from https://grouplens.org/datasets/movielens/. I will choose the "MovieLens 10M Dataset"
2. Perform Exploratory Data Analysis (EDA) to check the integrity of the data.
3. Perform cross validation methods to choose which explanatory variables to be used
4. Separate the data into train and test set
5. Use either Memory-Based Collaborative Filtering based system or Model-based Collaborative Filtering system
6. Check the RMSE to see which one perform better

Data Source

For this project, many tutorials recommend using Movielens database. They have accumulated 25 million movie ratings for 62,000 movies by 162,000 users. That is a lot of data especially for this project. Fortunately, the data can be easily downloaded from the website.

Importing and Cleaning Data

This project will have 3 different raw data from Movielens

1. movies.dat (510 KB)
    a. Information of movieId, title and genres
2. ratings.dat (258,893 KB)
    a. Information of rating based on userId and movieId
3. tags.dat (3,501 KB)
    a. Information of tag based on userId and movieId

The three raw data are separated into different pandas dataframe. I also created a new dataframe based on ratings.dat to filter the user who has less than 55 reviews for modeling purposes.

To make sure that I do not experience any issue when building the model, I checked if the data has any missing values. movies.dat an ratings.dat have no missing values while tags.dat does so I replace all the missing values with blank (“”). I also checked to see if the rating type is numerical so that it can be mathematically calculated.

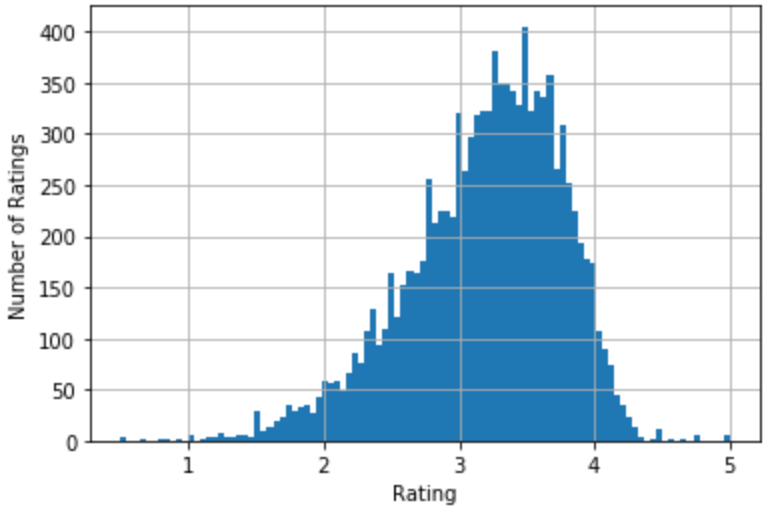
Once the data is cleaned, I combine the dataframe from movies.dat and ratings.dat based on movieId.

Exploratory Data Analysis (EDA)

These sections are divided into 2 parts:

1. By Title of the Movies

- There are 10,676 different movie titles in the dataset and below is the distribution of its rating.



- The ratings range from 0.5 to 5 with mean of 3.19 and standard deviation of 0.567
- On average, there are 936 reviews per movie with standard deviation of 2,487.43
- The movie with most rating is Pulp Fiction (2004) with 34,864 reviews and rating of 4.15

title	rating	no_of_ratings
Pulp Fiction (1994)	4.157426	34864
Forrest Gump (1994)	4.013582	34457
Silence of the Lambs, The (1991)	4.204200	33668
Jurassic Park (1993)	3.661564	32631
Shawshank Redemption, The (1994)	4.457238	31126

3. By Year the Movies are Released

- The movies released year span from 1915 to 2007 (92 years)
- The ratings range from 0.5 to 5 with mean of 3.72 and standard deviation of 0.211
- On average, there are 106,383.55 reviews per year with standard deviation of 172,558.88
- The year with most rating is 1995 with 874,436 reviews and rating of 3.44

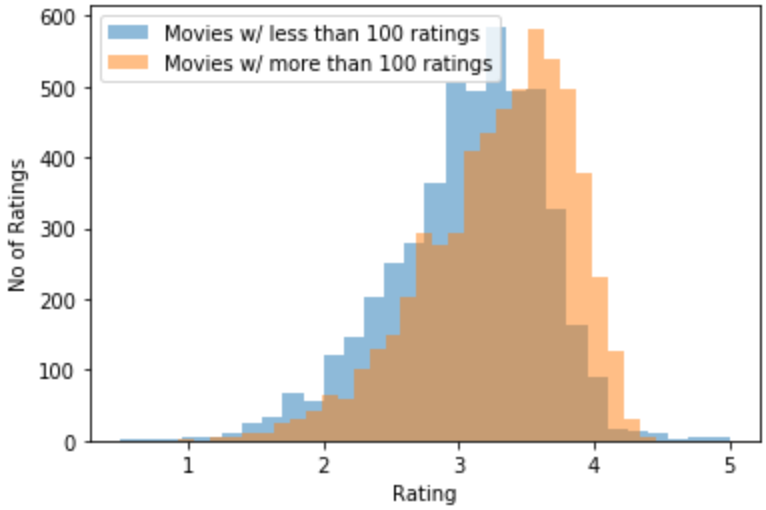
year	rating	no_of_ratings
1995	3.442817	874436
1994	3.472097	746042
1996	3.360754	659425
1999	3.453832	543990
1993	3.496386	534899

- The year with the highest rating is 1946 with 18,719 reviews and rating of 4.05

year	rating	no_of_ratings
1946	4.054036	18719
1934	4.051894	6600
1942	4.043820	22353
1931	4.025816	8483
1941	4.013690	26589

Inferential Statistics

From the ratings data, I see that many movies have less ratings compared to the others. I want to break the data into 2 based on movie titles to test if movies with less reviews will tend to have the same average with movies with more reviews.



I break it to movies with 100 ratings or less and movies with more than 100 ratings. Visually from the graph, I see that there is a slight difference in the rating distribution between the two, but I want to check statistically if this is the case. I tested using scipy.ttest_ind and from the result, I get a really low p-value which indicates that both have identical average values.

Recommendation System

There are many different methods to find a correlation between movies that you watched or rated to movies that should be recommended to you. In this project, I will use 3 different methods:

1. Simple Correlation
2. Memory-based
3. Model-based

Simple Correlation

First, I want to see if using simple correlation is good enough to recommend a movie based on another movie. For this example, I use “Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)” to see what other movies correlate with this movie.

I pivot the data table putting userId as index and movie titles as columns. Next, I find the correlations between the selected movie to other movies based on userId’s ratings. I sorted the

results based on correlations, picked movies with more than 100 ratings, and sorted from the highest correlations. The result is shown below.

	title	Correlation	no_of_ratings	movieId	genres
0	Star Wars: Episode IV - A New Hope (a.k.a. Sta...	100.000000	28566	260	Action Adventure Sci-Fi
1	Star Wars: Episode V - The Empire Strikes Back...	72.164990	23091	1196	Action Adventure Sci-Fi
2	Star Wars: Episode VI - Return of the Jedi (1983)	66.312501	25098	1210	Action Adventure Sci-Fi
3	Beyond Silence (Jenseits der Stille) (1996)	48.432453	106	1893	Drama
4	Raiders of the Lost Ark (Indiana Jones and the...	46.240733	21803	1198	Action Adventure
5	Futurama: Bender's Game (2008)	46.023348	141	62956	Animation Comedy Sci-Fi
6	My Name Is Nobody (Il Mio nome Ã¨ Nessuno) (1973)	45.841934	119	26294	Comedy Western
7	Apartment, The (L'Appartement) (1996)	43.308614	132	6789	Drama Mystery Romance
8	Hairdresser's Husband, The (Mari de la coiffeu...	43.285454	138	8270	Comedy Drama Romance
9	Blackboard Jungle (1955)	42.578024	163	8451	Drama
10	Strange Love of Martha Ivers, The (1946)	42.047967	125	3965	Drama Film-Noir

Interpreting the result, I can see obviously that the chosen movie will have 100% correlation with itself. I also see 2 other Star Wars movies top the chart which are expected but I are quite surprised that the other Star Wars series are not heavily correlated with the first movie from the users' ratings.

As Star Wars fans myself, other than the Indiana Jones movie, I have not watched the other movies in the list. I am curious if this is due to the low number of ratings so I refine it instead of more than 100 reviews, I do more than 500 and 1,000 reviews.

Result for more than 500 reviews

	title	Correlation	no_of_ratings	movieId	genres
0	Star Wars: Episode IV - A New Hope (a.k.a. Sta...	100.000000	28566	260	Action Adventure Sci-Fi
1	Star Wars: Episode V - The Empire Strikes Back...	72.164990	23091	1196	Action Adventure Sci-Fi
2	Star Wars: Episode VI - Return of the Jedi (1983)	66.312501	25098	1210	Action Adventure Sci-Fi
3	Raiders of the Lost Ark (Indiana Jones and the...	46.240733	21803	1198	Action Adventure
4	Star Wars: Episode III - Revenge of the Sith (...	41.444946	5193	33493	Action Adventure Fantasy Sci-Fi
5	Star Wars: Episode I - The Phantom Menace (1999)	40.280006	15744	2628	Action Adventure Sci-Fi
6	Star Wars: Episode II - Attack of the Clones (...	39.495531	7934	5378	Action Adventure Sci-Fi
7	Lord of the Rings: The Two Towers, The (2002)	35.655860	14389	5952	Action Adventure Fantasy
8	Lord of the Rings: The Fellowship of the Ring,...	34.425024	15938	4993	Action Adventure Fantasy
9	Lord of the Rings: The Return of the King, The...	34.237021	12366	7153	Action Adventure Fantasy
10	Indiana Jones and the Last Crusade (1989)	33.121802	16145	1291	Action Adventure

Result for more than 1,000 reviews

	title	Correlation	no_of_ratings	movieId	genres
0	Star Wars: Episode IV - A New Hope (a.k.a. Sta...	100.000000	28566	260	Action Adventure Sci-Fi
1	Star Wars: Episode V - The Empire Strikes Back...	72.164990	23091	1196	Action Adventure Sci-Fi
2	Star Wars: Episode VI - Return of the Jedi (1983)	66.312501	25098	1210	Action Adventure Sci-Fi
3	Raiders of the Lost Ark (Indiana Jones and the...	46.240733	21803	1198	Action Adventure
4	Star Wars: Episode III - Revenge of the Sith (...	41.444946	5193	33493	Action Adventure Fantasy Sci-Fi
5	Star Wars: Episode I - The Phantom Menace (1999)	40.280006	15744	2628	Action Adventure Sci-Fi
6	Star Wars: Episode II - Attack of the Clones (...	39.495531	7934	5378	Action Adventure Sci-Fi
7	Lord of the Rings: The Two Towers, The (2002)	35.655860	14389	5952	Action Adventure Fantasy
8	Lord of the Rings: The Fellowship of the Ring,...	34.425024	15938	4993	Action Adventure Fantasy
9	Lord of the Rings: The Return of the King, The...	34.237021	12366	7153	Action Adventure Fantasy
10	Indiana Jones and the Last Crusade (1989)	33.121802	16145	1291	Action Adventure

I see that by removing movies with less than 1,000 reviews, I see more movies that in our opinion are more related to the first Star Wars movie.

In conclusion, simple correlation is good enough to recommend movies if I have enough users' ratings but this will exclude movies that not a lot of people watch. Sometimes those movies can be entertaining as well.

Memory-based Methods

Before building a model for the recommendation system, I will try to predict a recommended movie using memory-based methods which is a pre-computed matrix of similarities will be used.

First I combined the movie titles information with the tags that users put for each movie. Then I created a new column called 'metadata' that combined all the tags from the users with the genres for each movie. Next step would be to vectorize this 'metadata' using the TfidVectorizer from sklearn library and reduce the dimensionality of the vectors.

Next, I use the ratings data to find all users that score similar ratings for a specified movie and see what other movies that these users score highly. In a simple lay term, what other movies do people like you would watch and score a high rating. In order to do that, I use the cosine_similarity from sklearn library.

	hybrid
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	1.000000
Star Wars: Episode V - The Empire Strikes Back (1980)	0.874397
Star Wars: Episode VI - Return of the Jedi (1983)	0.866502
Star Wars: Episode I - The Phantom Menace (1999)	0.752074
Alien (1979)	0.735173
Aliens (1986)	0.727314
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)	0.670736
Star Wars: Episode II - Attack of the Clones (2002)	0.660800
Star Trek II: The Wrath of Khan (1982)	0.640659
2001: A Space Odyssey (1968)	0.629786
Star Wars: Episode III - Revenge of the Sith (2005)	0.623952

From the top 10 results, I can see that 4 other Star Wars movies top the recommendation system which is expected. The other recommended movies include Alien franchises, Star Trek franchises and Indiana Jones series. I can see that the recommendation works pretty well as I would want to watch these movies after I watch the first Star Wars movie.

Model-based Methods

Lastly, I want to build a model that will have a user as input and using the algorithm, it will spit out a list of movies that is recommended based on the ratings that a user gave for various movies.

I use a library called Surprise that according to the official website, it is based on scikit. This library builds and analyzes recommender systems using explicit rating data.

I initialize the reader, split the data to test and train, then fit the train data to the algorithm. For this movie database, I get Root Mean Square Error of 0.7910 which is pretty decent.

I create a function detailed below which has input of user ID and use the algorithm to predict what movies recommended to this particular user.

```
# Create a function to return 10 most recommended movies for a selected user

def predict_user(user_id):
    if user_id in df_f.userId.unique():
        ui_list = df_f[df_f.userId == user_id].movieId.tolist()
        d = {k: v for k,v in Mapping_file.items() if not v in ui_list}
        predicted_list = []
        for i, j in d.items():
            predicted = svd.predict(user_id, j)
            predicted_list.append((i, predicted[3]))
        pdf = pd.DataFrame(predicted_list, columns = ['movies', 'ratings'])
        pdf.sort_values('ratings', ascending=False, inplace=True)
        pdf.set_index('movies', inplace=True)
        return print(pdf.head(10))
    else:
        print("Cannot find User Id in the list")
        return None
```

```
# Using the function above to display the top 10 recommended movies for a selected user

user_id = 1991
predicted_ratings = predict_user(user_id)
```

```
                                                ratings
movies
Life Is Beautiful (La Vita Ã¨ bella) (1997)    4.232855
Braveheart (1995)                               4.228899
Mr. Holland's Opus (1995)                       4.180408
Shawshank Redemption, The (1994)                4.170698
Green Mile, The (1999)                          4.165669
Lord of the Rings: The Return of the King, The ...  4.143111
Crash (2004)                                    4.095977
Lord of the Rings: The Two Towers, The (2002)   4.093613
Sixth Sense, The (1999)                         4.079201
Schindler's List (1993)                         4.075007
```

I see that these are the top 10 movies that are being recommended for userId 1991.

Conclusion

After finishing up the coding up of this project, I am quite surprised by myself that I am able to create some sort of recommendation system using the MovieLens database. I feel like that this is a good basis for more complicated models of a recommendation system. I will research more on how to be able to recommend movies using the simple correlation that can handle movies with little reviews.

References

https://towardsdatascience.com/how-to-build-a-simple-recommender-system-in-python-375093c3fb7d

https://blog.codecentric.de/en/2019/07/recommender-system-movie-lens-dataset/