SIMPLE RECOMMENDER SYSTEM WITH PYTHON

PROJECT OVERVIEW

There are three major types of recommender system and they are simple, content-based and collaborative filter recommender systems. While the content-based and collaborative filter are more effective, sophisticated and user specific, they have one common challenge which is that they cannot be used to recomend items to new or non regular users as past history of activities of the target user is required. Hence, simple recommender system could be adopted to recommend items to new users so as to get a clue of their preferences before more sophisticated recommender systems is be adopted. Also, e-commerce and classified ads websites for example could adopt simple recommender system to create a monthly or weekly table of trending ads for various ads categories based on interation of website visitors with items. Music or video streaming websites could also adopt this approach to create periodic list of top music or videos in various categories. In this work, I shall build a simple recommender system to determine the 100 most prefered movie based on users rating score and number of ratings for each movie in the dataset.

DATASET

The data used for this project is the MovieLense for Education and Development Dataset downloaded from Grouplens website (https://grouplens.org/datasets/movielens/latest/). The dataset was last updated in September 2018. GroupLens is a research lab in the Department of Computer Science and Engineering at the University of Minnesota, Twin Cities specializing in recommender systems, online communities, mobile and ubiquitous technologies, digital libraries, and local geographic information systems. The grouplense movie dataset are broken into full and summary datasets for convenience. The summary version conatin just a subset of the the dataset. The full dataset which would be used in this project has 6 tables, namely the genome-scores.csv, genome-tags.csv, links.csv, movies.csv, ratings.csv and tags.csv. Only the ratings.csv and movies.csv files will be used in this project because they have the information on rating counts, rating scores and movie title which are the information needed.

STEP 1: LIBRARY, DATA IMPORTATION AND DATA PREPARATION

In [2]:
```python
#we importation of pandas and the ratings.csv file
import pandas as pd
ratings = pd.read_csv('ratings.csv')
ratings.head()
```

Out[2]:

	userId	movieId	rating	timestamp
0	1	307	3.5	1256677221
1	1	481	3.5	1256677456
2	1	1091	1.5	1256677471
3	1	1257	4.5	1256677460
4	1	1449	4.5	1256677264

In [3]:
```python
#there are over 27 million records implying over 27 million ratings
ratings.shape
```

```
(27753444, 4)
```

Out[3]:

In [4]:
```
#importation of the movies.csv file which has the movie titles and would be needed
movieTitle = pd.read_csv('movies.csv')
movieTitle.head()
```

Out[4]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [5]:
```
#observe that there are 53,889 different movies in the dataset
movieTitle.shape
```

Out[5]:
```
(58098, 3)
```

In [6]:
```
#creating a new table which will have average_ratings and total_ratings as columns again
df1 = ratings.copy()[['userId','movieId']]
df2 = ratings.copy()[['movieId','rating']]
```

In [7]:
```
df1.head()
```

Out[7]:

	userId	movieId
0	1	307
1	1	481
2	1	1091
3	1	1257
4	1	1449

In [8]:
```
df2.head()
```

Out[8]:

	movieId	rating
0	307	3.5
1	481	3.5
2	1091	1.5
3	1257	4.5
4	1449	4.5

In [9]:
```
#Here I shall create the movie1 table which is the table containing all unique movies an
movie1  = df1.groupby(['movieId'],as_index=False)['userId'].count()
movie1.columns = ['movieId','numberOfRatings']
movie1.head()
```

Out[9]:

	movieId	numberOfRatings
0	1	68469

1	2	27143
2	3	15585
3	4	2989
4	5	15474

In [10]:
```
#we create movie2 table which is the table containing all unique movies and the total nu
movie2  = df2.groupby(['movieId'],as_index=False)['rating'].mean()
movie2.columns = ['movieId','averageRating']
movie2.head()
```

Out[10]:

	movieId	averageRating
0	1	3.886649
1	2	3.246583
2	3	3.173981
3	4	2.874540
4	5	3.077291

In [11]:
```
#creating a new table by merge movie1 and movie2 on movieId
movieRating = pd.merge(movie1,movie2, on='movieId')
movieRating.head()
```

Out[11]:

	movieId	numberOfRatings	averageRating
0	1	68469	3.886649
1	2	27143	3.246583
2	3	15585	3.173981
3	4	2989	2.874540
4	5	15474	3.077291

In [12]:
```
#observe that movieRating table has 53889 unique movies while the movieTitle table has 5
```

In [14]:
```
#merging this movieRating table with the moveTitle table
basicRecSys_df = pd.merge(movieTitle,movieRating, on='movieId')
basicRecSys_df.head()
```

Out[14]:

	movieId	title	genres	numberOfRatings	averageRating
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	68469	3.886649
1	2	Jumanji (1995)	Adventure|Children|Fantasy	27143	3.246583
2	3	Grumpier Old Men (1995)	Comedy|Romance	15585	3.173981
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	2989	2.874540
4	5	Father of the Bride Part II (1995)	Comedy	15474	3.077291

In [15]:
```
#observe that the movies without ratings have been dropped
basicRecSys_df.shape
```

Out[15]: (53889, 5)

In [16]:
```
#we check to conform that there is no null value in columns with numerical values
basicRecSys_df[['movieId','numberOfRatings','averageRating']].isna().sum().sum()
```

Out[16]: 0

STEP 2: THE RECOMMENDER SYSTEM

One may be tempted to extract the 100 most prefered movies by sorting the average rating but that would be a wrong approach as only average rating values are not sufficient to rate acceptance level. The total number of users that rate the movies is also a very important metric. The simple recommender system combine the 2 metric by calculating the weighted average of the 2 metrics with the following expression.

- weighted_rating = (v/(v+m)R)+(m/(m+v)C) where
  - wgere v = number of ratings for each movie
  - m = minimum number of ratings required to be listed
  - R = average rating for each movie
  - C = average rating across all movies note that v and R are already known, and C can easily be calculated with the by using the inbuit mean() function. The choice of m depends of individuals. For this project, we take m to be 90 percentile of the numberOfRating column.

In [17]:
```
# calculating the minimum number of votes required to be included in the movies from whe
m = basicRecSys_df['numberOfRatings'].quantile(0.90)
print(m)
```

```
531.0
```

In [18]:
```
#calculating the average ratings across all movies
C = basicRecSys_df['averageRating'].mean()
print(C)
```

```
3.0685927253973246
```

In [19]:
```
#I shall now filter out all qualified movies into a new DataFrame. Observe that the sele
q_basicRecSys_df = basicRecSys_df.copy().loc[basicRecSys_df['numberOfRatings'] >= m]
q_basicRecSys_df.shape
```

Out[19]: (5392, 5)

In [20]:
```
#creating a function that computes the weighted_rating of each movie
def weighted_rating(x, m=m, C=C):
    v = x['numberOfRatings']
    R = x['averageRating']
    # Calculation based on the IMDB formula
    return (v/(v+m) * R) + (m/(m+v) * C)
```

In [21]:
```
#I shall now define a new column/feature and call it wightedRating
q_basicRecSys_df['wightedRating'] = q_basicRecSys_df.apply(weighted_rating, axis=1)
```

In [22]:
```
q_basicRecSys_df.head()
```

Out[22]:

	movieId	title	genres	numberOfRatings	averageRating	wightedRating
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	68469	3.886649	3.88035
1	2	Jumanji (1995)	Adventure|Children|Fantasy	27143	3.246583	3.24316

2	3	Grumpier Old Men (1995)	Comedy|Romance	15585	3.173981	3.17050
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	2989	2.874540	2.90381
4	5	Father of the Bride Part II (1995)	Comedy	15474	3.077291	3.07700

In [23]:
```python
q_basicRecSys_df = q_basicRecSys_df.sort_values('wightedRating', ascending=False)
```

In [24]:
```python
#observe that movies with 5 start rating are not even in the top 50. This is most likely
q_basicRecSys_df.head(100)
```

Out[24]:

	movieId	title	genres	numberOfRatings	averageRating	wightedRating
315	318	Shawshank Redemption, The (1994)	Crime|Drama	97999	4.424188	4.416882
843	858	Godfather, The (1972)	Crime|Drama	60904	4.332893	4.321965
49	50	Usual Suspects, The (1995)	Crime|Mystery|Thriller	62180	4.291959	4.281600
523	527	Schindler's List (1993)	Drama|War	71516	4.257502	4.248739
1195	1221	Godfather: Part II, The (1974)	Crime|Drama	38875	4.263035	4.246940
...	...	...	...	...	...	...
1183	1209	Once Upon a Time in the West (C'era una volta ...	Action|Drama|Western	5952	4.105091	4.020195
952	969	African Queen, The (1951)	Adventure|Comedy|Romance|War	11937	4.059646	4.017438
5897	5995	Pianist, The (2002)	Drama|War	18209	4.044456	4.016805
933	950	Thin Man, The (1934)	Comedy|Crime	3704	4.150783	4.015094
30569	134130	The Martian (2015)	Adventure|Drama|Sci-Fi	16160	4.043812	4.012787

100 rows × 6 columns