GIFT UNIVERSITY
Converting Knowledge into Practical Experience

12/1/2023

Assignment 01

Instructor: Umer Ramzan

Group Members

Sufyan Ahmad:	**201980013**
Muhammad Bilal Butt:	**191980015**
Muhammad Nadeem:	**201980050**
Muhammad Naeem:	**201980009**

The age of recommender Systems

The rapid growth of data collection has led to a new era of information. Data is being used tocreate more efficient systems and this is where Recommendation Systems come into play. Recommendation Systems are a type of **information filtering systems** as they improve thequality of search results and provides items that are more relevant to the search item or are related to the search history of the user.

They are used to predict the **rating** or **preference** that a user would give to an item. Almost every major tech company has applied them in some form or the other: Amazon uses it to suggest products to customers, YouTube uses it to decide which video to play next on auto play, and Facebook uses it to recommend pages to like and people to follow. Moreover, companies like Netflix and Spotify depend highly on the effectiveness of their recommendationengines for their business and success.

In this Assignment we'll be building a baseline Movie Recommendation System using ***TMDB5000 Movie Dataset.*** This assignment will pretty much serve as a foundation in recommendation systems and will provide you with something to start with.

There are basically three types of recommender systems:-

Demographic Filtering-

They offer generalized recommendations to every user, based on movie popularity and/or genre.The System recommends the same movies to users with similar demographic features. Since each user is different, this approach is considered to be too simple. The basic idea behindthis system is that movies that are more popular and critically acclaimed will have a higher probabilityof being liked by the average audience.

Content Based Filtering- They suggest similar items based on a particular item. This system uses item metadata, such as genre, director, description, actors, etc. for movies, to make theserecommendations. The general idea behind these recommender systems is that if a person liked a particular item, he or she will also like an item that is similar to it.

Collaborative Filtering- This system matches persons with similar interests and provides recommendations based on this matching. Collaborative filters do not require item metadata likeits content-based counterparts

Task:

Design a movie recommender model using Demographic Filtering

There is a folder named ‘input’ along with the assignment contain the following data files:

1. **tmdb_5000_credits.csv**
2. **tmdb_5000_movies.csv**

The first dataset contains the following features:-

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

The second dataset has the following features:-

- budget - The budget in which the movie was made.
- genre - The genre of the movie, 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 recieved.
- vote_count - the count of votes recieved.

Print the first five rows of the data.

Before getting started with this -

- we need a metric to score or rate movie
- Calculate the score for every movie
- Sort the scores and recommend the best rated movie to the users.

We can use the average ratings of the movie as the score but using this won't be fair enough since a movie with 8.9 average rating and only 3 votes cannot be considered better than the

movie with 7.8 as as average rating but 40 votes. So, we'll be using IMDB's weighted rating (wr) which is given as :-

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

where,

- v is the number of votes for the movie;
- m is the minimum votes required to be listed in the chart(trending list or recommendations);
- R is the average rating of the movie; And
- C is the mean vote across the whole report

We already have v (**vote_count**) and R (**vote_average**) and C can be calculated:

First Step import the libraries and import dataset in df1and df2 .

```
27]: #Import the Libraries
     import pandas as pd
     import numpy as np

     # Load the data sets 'input/tmdb_5000_credits.csv' in df1 and 'input/tmdb_5000_movies.csv' in df2
     df1=pd.read_csv('input/tmdb_5000_credits.csv')
     df2=pd.read_csv('input/tmdb_5000_movies.csv')
```

In this step df1.head () means the first five records are show.

```
]: df1.head()
```

]:

	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...

In this step we show the features names of df1.

```
9]: df1.columns
```

```
9]: Index(['movie_id', 'title', 'cast', 'crew'], dtype='object')
```

In this step we show the features of df2 .

```
[30]: df2.columns

[30]: Index(['budget', 'genres', 'homepage', 'id', 'keywords', 'original_language',
       'original_title', 'overview', 'popularity', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'tagline', 'title', 'vote_average',
       'vote_count'],
      dtype='object')
```

In this Step Rename the movie_id column into id in df 1.

```
[31]: #code here to rename 'movie_id' column to 'id' in df1.
      df1.columns = ['id','tittle','cast','crew']
```

In this step The merge of df1 data is attached the df2 data on the basis of id column.

```
[32]: #Lets join the two datasets on 'ID' column and save joined dataframe in df2 (First make

      df2 = df2.merge(df1,on='id')
```

In this step df2.head (5) means the first five records are show in df2.

```
In [33]: # the results should look like this.

         df2.head(5)
```

Out[33]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_comp
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 Partners
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 D Pictures", "id": 2
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": "Col 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": "Lege Pictures", "id": 92
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 D Pictures", "

Task 1:

In this Step Calculate C by computing the mean by using df2 vote average.

```
In [34]: #Calculate C by computing the mean of vote_average.

         C= df2['vote_average'].mean()
         C
```

```
Out[34]: 6.092171559442011
```

Value of C should be 6.092171559442011

Code: C= df2['vote_average'].mean()

So, the mean rating for all the movies is approx. 6 on a scale of 10.The next step is to determine

an appropriate value for **m**, the minimum votes required to be listed in the chart. We will use 90th percentile as our cutoff. In other words, for a movie to feature in the charts, it must have more votes than at least 90% of the movies in the list.

Task 2:

In Compute the value of m (the minimum votes required to be listed in the chart) by using 90th percentile.

```
In [35]: #Compute the value of m (the minimum votes required to be listed in the chart)  by using 90th percentile.

         m= df2['vote_count'].quantile(0.9)
         m

Out[35]: 1838.4000000000015
```

Code: m= df2['vote_count'].quantile(0.9)

Value of m should be 1838.4000000000015

Task 3:

Now, filter out the movies whose value is greater than m.

```
[36]: # Now, filter out the movies whose value is greater than m.
      q_movies = df2.copy().loc[df2['vote_count'] >= m]
      q_movies.shape

:[36]: (481, 23)
```

Code :q_movies = df2.copy().loc[df2['vote_count'] >= m]

Output (481, 26)

Task 4:

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

By using weighted_rating() function we'll calculate the value by applying this function to our DataFrame of qualified movies (whose values is greater than m).

In this Step weighted_rating() and define a new feature (column) score by using the IMDB formula:

```
In [37]: def weighted_rating(x, m=m, C=C):
             v = x['vote_count']
             R = x['vote_average']
             # Calculation based on the IMDB formula
             return (v/(v+m) * R) + (m/(m+v) * C)
```

Task 5:

In this step DataFrame based on the score feature and output the title, vote count, vote average and weighted rating or score of the top 10 movies to display the head(10).

```
[41]: #Top 10 movies
      q_movies[['title', 'vote_count', 'vote_average', 'score']].head(10)
```

[41]:

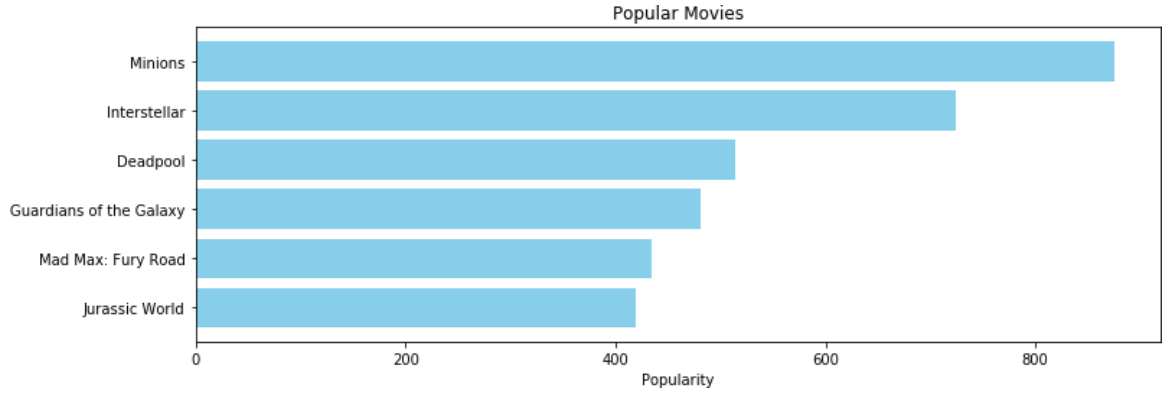
	title	vote_count	vote_average	score
0	Avatar	11800	7.2	7.050669
1	Pirates of the Caribbean: At World's End	4500	6.9	6.665696
2	Spectre	4466	6.3	6.239396
3	The Dark Knight Rises	9106	7.6	7.346721
4	John Carter	2124	6.1	6.096368
5	Spider-Man 3	3576	5.9	5.965250
6	Tangled	3330	7.4	6.934805
7	Avengers: Age of Ultron	6767	7.3	7.041968
8	Harry Potter and the Half-Blood Prince	5293	7.4	7.062856
9	Batman v Superman: Dawn of Justice	7004	5.7	5.781535

Code: q_movies[['title', 'vote_count', 'vote_average', 'score']].head(10)

In this step Sort the Movies based on the on Score .

```
In [42]: #Sort movies based on score calculated above
         q_movies = q_movies.sort_values('score', ascending=False)
```

Hurray! We have made our first (though very basic) recommender system. Under the **Trending Now** tab of these systems we find movies that are very popular and they can just be obtained by sorting the dataset by the popularity column.



Code All Program:

```
#Import the Libraries
import pandas as pd
import numpy as np

# Load the data sets 'input/tmdb_5000_credits.csv' in df1 and
'input/tmdb_5000_movies.csv' in df2
df1=pd.read_csv('input/tmdb_5000_credits.csv')
df2=pd.read_csv('input/tmdb_5000_movies.csv')

df1.head()

df1.columns

df1.columns = ['id','tittle','cast','crew']

df2 = df2.merge(df1,on='id')

df2.head(5)
```

```
C= df2['vote_average'].mean()
C
m= df2['vote_count'].quantile(0.9)
m

q_movies = df2.copy().loc[df2['vote_count'] >= m]
q_movies.shape

def weighted_rating(x, m=m, C=C):
    v = x['vote_count']
    R = x['vote_average']
    # Calculation based on the IMDB formula
    return (v/(v+m) * R) + (m/(m+v) * C)
   q_movies['score'] = q_movies.apply(weighted_rating, axis=1)
   q_movies[['title', 'vote_count', 'vote_average', 'score']].head(10)
   q_movies = q_movies.sort_values('score', ascending=False)
   pop= df2.sort_values('popularity', ascending=False)
   import matplotlib.pyplot as plt
   plt.figure(figsize=(12,4))


   plt.barh(pop['title'].head(6),pop['popularity'].head(6), align='center',
           color='skyblue')
   plt.gca().invert_yaxis()
   plt.xlabel("Popularity")
   plt.title("Popular Movies")
```