CSE 523 Machine Learning Project Report
Movie Recommender System
weekly Report

-Kashish Jivani(AU1940161)
-Neel Popat(AU1940165)
-Yashvi Navadia(AU1940123)

Simple Recommender System:

Task Completed :

- Previously we researched and learnt how weighted mean can be used for filtering movies and we have now completed our first filtering operation on the movies dataset with the appropriate results from the code. According to the formula, what we really need to find is the appropriate value of m (minimum number of votes required to be listed in the chart) for our dataset. To do that we will set a cutoff such that movies having votes more than the cutoff value will be qualified to be listed in the Top 250 movies chart.

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

where,

- v is the number of votes for the movie
- m is the minimum votes required to be listed in the chart
- R is the average rating of the movie
- C is the mean vote across the whole report

To proceed further in terms of coding, we first need to to change the data type of votes_count and votes_average column from NaNs to int in case of no values entered in some row. After casting it to int, the mean is calculated as shown below:

```
vote_counts = md[md['vote_count'].notnull()]['vote_count'].astype('int')
vote_averages = md[md['vote_average'].notnull()]['vote_average'].astype('int')
C = vote_averages.mean()
C
```

```
5.244896612406511
```

Further to find the appropriate value of M, the cutoff assumed is 90% and the output will give the minimum number of votes a movie must have to qualify to be listed in the top 250 chart.

```
[ ]  m = vote_counts.quantile(0.90)
     m

     160.0
```

Now that we have the value of m, we know that movies having votes less than 160 do not qualify for the top 250 list according to our sample data.

The next step is to find the total number of movies that qualify for the top 250 chart. We will do that using the .shape() function which will return the number of elements in an array in each dimension where the first value will give us the number of rows, in our case, total movies and the second value will give the number of attributes associated with those movies.

```
[ ]  md['year'] = pd.to_datetime(md['release_date'], errors='coerce').apply(lambda x: str(x).split('-')[0] if x != np.nan else np.nan)

[ ]  qualified = md[(md['vote_count'] >= m) & (md['vote_count'].notnull()) & (md['vote_average'].notnull())][['title', 'year', 'vote_count', 'vote_average', 'popularity', 'genre
     qualified['vote_count'] = qualified['vote_count'].astype('int')
     qualified['vote_average'] = qualified['vote_average'].astype('int')
     qualified.shape

     (4555, 6)
```

Here we get 4555 movies having more than 160 votes. So now to get top 250 we sort the data based on decreasing weighted rating. For that, we define a function to calculate the weighted rating and then create an extra column in the database which has the weighted rating of the movies. We then sort them in decreasing order and the first 250 movies in the list will be our desired output.

```
{x}

     def weighted_rating(x):
         v = x['vote_count']
         R = x['vote_average']
         return (v/(v+m) * R) + (m/(m+v) * C)

[ ]  qualified['wr'] = qualified.apply(weighted_rating, axis=1)

[ ]  qualified = qualified.sort_values('wr', ascending=False).head(250)
```

Outcome of the task performed:

	title	year	vote_count	vote_average	popularity	genres	wr
10309	Dilwale Dulhania Le Jayenge	1995	661	9	34.457024	[Comedy, Drama, Romance]	8.268189
15480	Inception	2010	14075	8	29.108149	[Action, Thriller, Science Fiction, Mystery, A...	7.969033
12481	The Dark Knight	2008	12269	8	123.167259	[Drama, Action, Crime, Thriller]	7.964533
22879	Interstellar	2014	11187	8	32.213481	[Adventure, Drama, Science Fiction]	7.961151
2843	Fight Club	1999	9678	8	63.869599	[Drama]	7.955192
4863	The Lord of the Rings: The Fellowship of the Ring	2001	8892	8	32.070725	[Adventure, Fantasy, Action]	7.951302
292	Pulp Fiction	1994	8670	8	140.950236	[Thriller, Crime]	7.950077
314	The Shawshank Redemption	1994	8358	8	51.645403	[Drama, Crime]	7.948249
7000	The Lord of the Rings: The Return of the King	2003	8226	8	29.324358	[Adventure, Fantasy, Action]	7.947434
351	Forrest Gump	1994	8147	8	48.307194	[Comedy, Drama, Romance]	7.946934
5814	The Lord of the Rings: The Two Towers	2002	7641	8	29.423537	[Adventure, Fantasy, Action]	7.943492
256	Star Wars	1977	6778	8	42.149697	[Adventure, Action, Science Fiction]	7.936463

Activate Windows
Go to Settings to activate Wi

As shown in the image above, a new column is created and the list of movies is sorted with respect to decreasing weighted rating value.

Task to be performed in the upcoming week:

The output we see here is sorted as per the rating but still messed up in terms of its content for example genres, directors or cast and crew. Next we will try to modify our approach in a way that the users get recommendations based on the genre they like or the artist they might like or other such specifications.

References:

- https://www.analyticsvidhya.com/blog/2020/03/what-are-lambda-functions-in-python/
- https://www.geeksforgeeks.org/python-pandas-dataframe-astype/
- https://www.analyticsvidhya.com/blog/2020/11/create-your-own-movie-movie-recommendation-system/