Recommending movies using Collaborative Filtering

In [ ]:
```
```

In [2]:
```
import numpy as np
import pandas as pd
from scipy.spatial.distance import hamming
```

In [3]:
```
import warnings
warnings.simplefilter(action='ignore',category=Warning)
```

In [5]:
```
df_ratings=pd.read_csv('recent_ratings.csv')
df_movies=pd.read_csv('recent_movies.csv')
```

In [6]:
```
df_ratings.shape,df_movies.shape
```

Out[6]:
```
((552, 4), (188, 4))
```

In [7]:
```
df_ratings.userId.unique().size
```

Out[7]:
```
63
```

In [10]:
```
df_ratings.movieId.unique().size
```

Out[10]:
```
188
```

In [11]:
```
df_ratings.sample(5)
```

Out[11]:

	userId	movieId	rating	timestamp

	userId	movieId	rating	timestamp
329	380	122898	3.0	1536872751
362	414	122918	4.0	1513958808
254	249	184253	4.0	1518388863
316	338	189043	2.5	1530148447
184	210	122906	4.5	1537632293

In [12]:
```
df_ratings.drop(columns=['timestamp'],inplace=True)
```

In [13]:
```
df_ratings.sample(5)
```

Out[13]:

	userId	movieId	rating
65	89	176805	4.0
453	567	122898	2.0
180	184	193587	3.5
534	599	178061	3.0
301	318	188833	4.5

In [14]:
```
ratings=df_ratings.pivot(index='userId',columns='movieId',values='rating')
```

In [15]:
```
ratings.sample(5)
```

Out[15]:

movieId	122896	122898	122906	122912	122916	122918	122926	143355	166534	167064	...	189381	189713	190183	190209	190215	191005
userId																	
401	NaN	NaN	NaN	NaN	NaN	4.5	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
433	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN

movieId	122896	122898	122906	122912	122916	122918	122926	143355	166534	167064	...	189381	189713	190183	190209	190215	191005
userId																	
511	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
551	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
414	NaN	NaN	4.0	NaN	4.0	4.0	4.5	4.5	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 188 columns

Hamming Distance Measures how different two sequences are. It is % of disagreement. A value of 1 indicates sequences are very different, 0 indicates they are very similar.

In [16]:
```
l1=(1,2,3)
l2=(1,2,3)
l3=(1,4,5)
print(hamming(l1,l2))
print(hamming(l1,l3))
```

```
0.0
0.6666666666666666
```

In [17]:
```
def hamming_distance(user1,user2):
    try:
        user1_ratings=ratings.loc[user1,:]
        user2_ratings=ratings.loc[user2,:]
        distance=hamming(user1_ratings,user2_ratings)
    except:
        distance=np.NaN
    return distance
```

In [19]:
```
def get_nearest_users(active_user,k=10):
    all_users=pd.DataFrame(ratings.index)
    other_users=all_users[all_users.userId!=active_user]
    other_users['distance'] = other_users['userId'].apply(lambda x: hamming_distance(active_user,x))
```

```
        # find out hamming distance and return users with low hamming distance from active user
    return  other_users.sort_values(['distance'], ascending = True).userId[:k]
```

In [20]:

```
def get_recommended_movies(ratings,movies, user,top=5):
    # Find out nearest neighbours based on hamming distance
    nn_users = get_nearest_users(user,10)

    # Get ratings of other nearest neighbours(users)
    nn_ratings = ratings[ratings.index.isin(nn_users)]

    # Average ratings gived by nearest neighbours for all movies
    avg_ratings = nn_ratings.apply(np.nanmean).dropna()

    # Find out movies that user had already watched
    movies_watched = ratings.transpose()[user].dropna().index

    # remove movies that user already watched
    avg_ratings = avg_ratings[~ avg_ratings.index.isin(movies_watched)]

    # Findout top n movies based on avg ratings given by other nearest neighbours
    top_movies_ids = avg_ratings.sort_values(ascending=False).index[:top]

    # Return recommended movies
    return movies[movies.movieId.isin(top_movies_ids)].title
```

In [21]:

```
df_ratings.userId.unique()    # Unique userids
```

Out[21]:

```
array([ 18,  21,  25,  49,  50,  62,  68,  89,  98, 103, 105, 111, 119,
       125, 153, 184, 205, 209, 210, 212, 233, 248, 249, 252, 258, 272,
       279, 296, 305, 306, 318, 331, 338, 339, 362, 363, 380, 400, 401,
       414, 417, 433, 448, 459, 462, 471, 475, 491, 511, 514, 515, 517,
       523, 548, 551, 556, 561, 567, 586, 596, 599, 601, 610], dtype=int64)
```

In [22]:

```
get_recommended_movies(ratings,df_movies,249,5)
```

Out[22]:

```
22                                The Boss Baby (2017)
56                              Tickling Giants (2017)
97     Three Billboards Outside Ebbing, Missouri (2017)
```

```
105                                 Paddington 2 (2017)
145                                 Isle of Dogs (2018)
Name: title, dtype: object
```

In [23]:
```
get_recommended_movies(ratings,df_movies, 433,5)
```

Out[23]:
```
3     Avengers: Infinity War - Part I (2018)
12               The Lego Batman Movie (2017)
15              John Wick: Chapter Two (2017)
20                        The Big Sick (2017)
23                Call Me by Your Name (2017)
Name: title, dtype: object
```

In [ ]:
```

```