In [1]:
```
import pandas as pd
import numpy as np
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
import matplotlib.pyplot as plt
import seaborn as sns
```

In [185]:
```
movies = pd.read_csv('D:/RISE - WPU/Internship/archive (1)/ratings_small.csv')
movies
```

Out[185]:

	userId	movieId	rating	timestamp
0	1	31	2.5	1260759144
1	1	1029	3.0	1260759179
2	1	1061	3.0	1260759182
3	1	1129	2.0	1260759185
4	1	1172	4.0	1260759205
5	1	1263	2.0	1260759151
6	1	1287	2.0	1260759187
7	1	1293	2.0	1260759148
8	1	1339	3.5	1260759125
9	1	1343	2.0	1260759131
10	1	1371	2.5	1260759135
11	1	1405	1.0	1260759203
12	1	1953	4.0	1260759191
13	1	2105	4.0	1260759139
14	1	2150	3.0	1260759194
15	1	2193	2.0	1260759198
16	1	2294	2.0	1260759108
17	1	2455	2.5	1260759113
18	1	2968	1.0	1260759200
19	1	3671	3.0	1260759117
20	2	10	4.0	835355493
21	2	17	5.0	835355681
22	2	39	5.0	835355604
23	2	47	4.0	835355552
24	2	50	4.0	835355586
25	2	52	3.0	835356031
26	2	62	3.0	835355749
27	2	110	4.0	835355532
28	2	144	3.0	835356016
29	2	150	5.0	835355395
...	...	...	...	...
99974	671	4034	4.5	1064245493
99975	671	4306	5.0	1064245548
99976	671	4308	3.5	1065111985
99977	671	4880	4.0	1065111973
99978	671	4886	5.0	1064245488
99979	671	4896	5.0	1065111996
99980	671	4963	4.5	1065111855
99981	671	4973	4.5	1064245471
99982	671	4993	5.0	1064245483
99983	671	4995	4.0	1064891537
99984	671	5010	2.0	1066793004
99985	671	5218	2.0	1065111990
99986	671	5299	3.0	1065112004
99987	671	5349	4.0	1065111863
99988	671	5377	4.0	1064245557
99989	671	5445	4.5	1064891627
99990	671	5464	3.0	1064891549
99991	671	5669	4.0	1063502711
99992	671	5816	4.0	1065111963
99993	671	5902	3.5	1064245507
99994	671	5952	5.0	1063502716
99995	671	5989	4.0	1064890625
99996	671	5991	4.5	1064245387
99997	671	5995	4.0	1066793014
99998	671	6212	2.5	1065149436
99999	671	6268	2.5	1065579370
100000	671	6269	4.0	1065149201
100001	671	6365	4.0	1070940363
100002	671	6385	2.5	1070979663
100003	671	6565	3.5	1074784724

100004 rows × 4 columns

In [186]:
```
#Transforming data into matrix where each row represents the user and column represents movie

users = movies.userId.unique().shape[0]
movie = movies.movieId.unique().shape[0]
movie = movies['movieId'].max()
num = np.zeros((users,movie))
for line in movies.itertuples():  # The loop iterates thrugh each row of dataframe and extracts ratings from user-movie matrix
    num[line[1]-1,line[2]-1] = line[3]
print("Original Rating Matrix: ",num)
```

```
Original Rating Matrix:  [[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [4. 0. 0. ... 0. 0. 0.]
 [5. 0. 0. ... 0. 0. 0.]]
```

The dataset consists of ratings from 1 to 5, where 1 represents the lowest rating & 5 represents the highest. Different ratings could have different meanings to users. For instance, a rating of 3 might be good for one user while average for another user.

To solve this ambiguity, big giants such as Netflix or YouTube have moved to binary ratings. Therefore, we will work on binary ratings instead of continuous ratings.

In [187]:
```
# This code converts the dataset to a binary dataset. Here only those item are considered whose ratings are greater or equal to 3 being liked by the user and others being dislike
d by the user.
#As we are only considerate about the liking of users, making ratings less than 3 as 0 would not impact the recommendation process.

for i in range(len(num)):
    for j in range(len(num[0])):
        if num[i][j]>=3:
            num[i][j]=1
        else:
            num[i][j]=0
```

In [188]:
```
# We are converting dense rating matrix to a sparse matrix using csr_matrix function.

sample = csr_matrix(num)
print(sample)
```

```
  (0, 1028)     1.0
  (0, 1060)     1.0
  (0, 1171)     1.0
  (0, 1338)     1.0
  (0, 1952)     1.0
  (0, 2104)     1.0
  (0, 2149)     1.0
  (0, 3670)     1.0
  (1, 9)        1.0
  (1, 16)       1.0
  (1, 38)       1.0
  (1, 46)       1.0
  (1, 49)       1.0
  (1, 51)       1.0
  (1, 61)       1.0
  (1, 109)      1.0
  (1, 143)      1.0
  (1, 149)      1.0
  (1, 152)      1.0
  (1, 160)      1.0
  (1, 164)      1.0
  (1, 167)      1.0
  (1, 184)      1.0
  (1, 185)      1.0
  (1, 207)      1.0
  :     :
  (670, 4033)   1.0
  (670, 4305)   1.0
  (670, 4307)   1.0
  (670, 4879)   1.0
  (670, 4885)   1.0
  (670, 4895)   1.0
  (670, 4962)   1.0
  (670, 4972)   1.0
  (670, 4992)   1.0
  (670, 4994)   1.0
  (670, 5298)   1.0
  (670, 5348)   1.0
  (670, 5376)   1.0
  (670, 5444)   1.0
  (670, 5463)   1.0
  (670, 5668)   1.0
  (670, 5815)   1.0
  (670, 5901)   1.0
  (670, 5951)   1.0
  (670, 5988)   1.0
  (670, 5990)   1.0
  (670, 5994)   1.0
  (670, 6268)   1.0
  (670, 6364)   1.0
  (670, 6564)   1.0
```

In [189]:
```
# Computing similarity between movies of sample using cosine similarity.

knn = NearestNeighbors(metric='cosine', algorithm='brute', n_neighbors=3, n_jobs=-1)
knn.fit(sample)
```

Out[189]:
```
NearestNeighbors(algorithm='brute', leaf_size=30, metric='cosine',
                 metric_params=None, n_jobs=-1, n_neighbors=3, p=2, radius=1.0)
```

In [209]:
```
# Here we are finding the movies liked by the person whose userId = 2.

movies_sort_des = movies.sort_values(['userId', 'timestamp'])
filter1 = movies_sort_des[movies_sort_des['userId'] == 2].movieId
filter1 = filter1.tolist()
filter1 = filter1[:20]
print("Movies liked by user: ",filter1)
```

```
Movies liked by user:  [150, 296, 590, 592, 153, 165, 349, 588, 292, 339, 10, 161, 185, 208, 253, 457, 593, 110, 300, 410]
```

In [210]:
```
# Based on the likes of the userId 2 the following code depicts that what movies should be recommended to that user.

distances1=[]
indices1=[]
for i in filter1:
    distances , indices = knn.kneighbors(sample[i],n_neighbors=3)
    indices = indices.flatten()
    indices= indices[1:]
    indices1.extend(indices)
print("Movies to be recommended: ",indices1)
```

```
Movies to be recommended:  [368, 399, 566, 427, 215, 129, 160, 287, 360, 99, 561, 294, 597, 561, 567, 143, 670, 123, 123, 115, 137, 437, 17, 255, 455, 151, 173, 592, 589, 84, 28
1, 123, 341, 33, 561, 653, 28, 563, 453, 26]
```

In [ ]:
```

```