IBCF (item based collaborative filtering) - Recommendation System

```
import pandas as pd
import numpy as np
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
import matplotlib.pyplot as plt
```

Double-click (or enter) to edit

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

Read the dataset using Pandas library

```
import pandas as pd
# header = ['user_id','item_id','rating','timestamp']
# dataset = pd.read_csv('C:/Users/91920/Downloads/Compressed/ml-100k/ml-100k/u.data',sep =
header = ['Customer_id', 'rating','dates']

dataset=pd.read_csv('combined_data_1.txt',names=header)
# print(a.head(20))
# print(dataset.head())
```

```
df_nan=pd.DataFrame(pd.isnull(dataset.rating))
df_nan=df_nan[df_nan['rating']==True]
df_nan=df_nan.reset_index()
movie_np=[]
movie_id=1
for i,j in zip(df_nan['index'][1:],df_nan['index'][:-1]):
    temp = np.full((1,i-j-1), movie_id)
    movie_np = np.append(movie_np, temp)
    movie_id += 1
```

```
# Account for last record and corresponding length
# numpy approach
last_record = np.full((1,len(dataset) - df_nan.iloc[-1, 0] - 1),movie_id)
movie_np = np.append(movie_np, last_record)

print('Movie numpy: {}'.format(movie_np))
print('Length: {}'.format(len(movie_np)))
```

```
dataset= dataset[pd.notnull(dataset['rating'])]

dataset['Movie_Id'] = movie_np.astype(int)
print('-Dataset examples-')
print(dataset.iloc[::10000, :])
print(dataset.head(20))
b=pd.read_csv('movie_titles.csv',encoding="ISO-8859-1",header=None,names = ['Movie_Id', 'Y
print(b.head(20))
dataset=pd.merge(dataset,b,on='Movie_Id')
print(dataset.head(20))
```

```
Movie numpy: [  1.   1.   1. ... 788. 788. 788.]
Length: 4076026
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:10: SettingWithCopyW
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
  # Remove the CWD from sys.path while we load stuff.
-Dataset examples-
        Customer_id  rating       dates  Movie_Id
1           1488844     3.0  2005-09-06         1
10008       2421394     3.0  2005-06-01         8
20008       1834737     3.0  2005-10-22         8
30017       1572097     4.0  2005-07-25        17
40018       2304974     3.0  2001-11-24        18
...             ...     ...         ...       ...
4030763      564658     3.0  2005-08-27       763
4040763     1674814     3.0  2005-02-24       763
4050781     2550506     2.0  2005-03-22       781
4060788      973703     2.0  2005-10-24       788
4070788      441929     4.0  2004-10-26       788

[408 rows x 4 columns]
   Customer_id  rating       dates  Movie_Id
1      1488844     3.0  2005-09-06         1
2       822109     5.0  2005-05-13         1
3       885013     4.0  2005-10-19         1
4        30878     4.0  2005-12-26         1
5       823519     3.0  2004-05-03         1
6       893988     3.0  2005-11-17         1
7       124105     4.0  2004-08-05         1
8      1248029     3.0  2004-04-22         1
9      1842128     4.0  2004-05-09         1
10     2238063     3.0  2005-05-11         1
11     1503895     4.0  2005-05-19         1
12     2207774     5.0  2005-06-06         1
13     2590061     3.0  2004-08-12         1
14        2442     3.0  2004-04-14         1
15      543865     4.0  2004-05-28         1
16     1209119     4.0  2004-03-23         1
17      804919     4.0  2004-06-10         1
18     1086807     3.0  2004-12-28         1
19     1711859     4.0  2005-05-08         1
20      372233     5.0  2005-11-23         1
    Movie_Id    Year                                 Name
0          1  2003.0                      Dinosaur Planet
1          2  2004.0           Isle of Man TT 2004 Review
2          3  1997.0                            Character
```

```
3          4  1994.0                    Paula Abdul's Get Up & Dance
4          5  2004.0                        The Rise and Fall of ECW
5          6  1997.0                                            Sick
6          7  1992.0                                           8 Man
7          8  2004.0                       What the #$*! Do We Know!?
8          9  1991.0                       Class of Nuke 'Em High 2
9         10  2001.0                                         Fighter
10        11  1999.0                  Full Frame: Documentary Shorts
```

```
dataset.rating.value_counts()
```

```
4.0    1409989
3.0    1172804
5.0     904698
2.0     403839
1.0     184696
Name: rating, dtype: int64
```

```
dataset.head()
```

	Customer_id	rating	dates	Movie_Id	Year	Name
0	1488844	3.0	2005-09-06	1	2003.0	Dinosaur Planet
1	822109	5.0	2005-05-13	1	2003.0	Dinosaur Planet
2	885013	4.0	2005-10-19	1	2003.0	Dinosaur Planet
3	30878	4.0	2005-12-26	1	2003.0	Dinosaur Planet
4	823519	3.0	2004-05-03	1	2003.0	Dinosaur Planet

We transform the dataset into a matrix where each row represents the user and column represents the item.

```
A = dataset.pivot_table(values='rating',index=['Customer_id'],columns=['Movie_Id'])
```

The MovieLens dataset consists of ratings on a scale of 1-5 where 1 represents the lowest rating while 5 represents the highest rating. However, 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, in this blog, we will work on binary ratings instead of continuous ratings to keep ourselves in sync with the latest research.

The below code converts the MovieLens dataset into the binary MovieLens dataset. We have considered items whose ratings are greater or equal to 3 being liked by the user and others being disliked 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.

```
A.fillna(0 , inplace=True)
A.value_counts()
```

```
1    2    3    4    5    6    7    8    9    10   11   12   13   14   15   16   17
0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

Length: 308389, dtype: int64
```

```
for coloums in A:
    A[coloums].mask(A[coloums] <3 ,0, inplace=True)
    A[coloums].mask(A[coloums] >=3 ,1, inplace=True)

print(A)
```

```
Movie_Id     1    2    3    4    5    6    ...  783  784  785  786  787  788
Customer_id                                 ...
10           0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0
1000027      0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0
1000033      0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0
1000035      0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0
1000038      0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0
...          ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
999964       0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0
999972       0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0
999977       0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0
999984       0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0
999988       0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0

[393089 rows x 788 columns]
```

how many movie like by user 1 (for prediction)

Unsupported Cell Type. Double-Click to inspect/edit the content.

To save the memory, we convert the dense rating matrix into a sparse matrix using the csr_matrix() function.

```
csr_sample = csr_matrix(A)
print(csr_sample)
```

```
  (0, 174)	1.0
  (0, 190)	1.0
  (0, 196)	1.0
  (0, 284)	1.0
  (0, 467)	1.0
  (0, 472)	1.0
  (0, 570)	1.0
  (1, 467)	1.0
  (1, 606)	1.0
  (1, 666)	1.0
  (2, 29)	1.0
  (2, 110)	1.0
  (2, 188)	1.0
  (2, 190)	1.0
  (2, 251)	1.0
  (2, 272)	1.0
  (2, 333)	1.0
  (2, 482)	1.0
  (2, 493)	1.0
  (2, 562)	1.0
  (2, 570)	1.0
  (2, 757)	1.0
  (3, 29)	1.0
  (3, 174)	1.0
  (3, 272)	1.0
  :	:
  (393084, 606)	1.0
  (393084, 699)	1.0
  (393085, 155)	1.0
  (393085, 174)	1.0
  (393085, 198)	1.0
  (393085, 240)	1.0
  (393085, 404)	1.0
  (393085, 570)	1.0
  (393085, 719)	1.0
  (393086, 240)	1.0
  (393086, 268)	1.0
  (393086, 412)	1.0
  (393087, 83)	1.0
  (393087, 174)	1.0
  (393087, 190)	1.0
  (393087, 251)	1.0
  (393087, 456)	1.0
  (393087, 467)	1.0
  (393087, 482)	1.0
  (393088, 29)	1.0
  (393088, 186)	1.0
  (393088, 190)	1.0
  (393088, 264)	1.0
  (393088, 442)	1.0
  (393088, 627)	1.0
```

Compute similarity between items of csr_sample using cosine similarity for simple task of finding the nearest neighbors

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

```
NearestNeighbors(metric='cosine', n_jobs=-1, n_neighbors=3)
```

Generate Recommendations

Here, we are generating recommendations for the user_id: 1.

We generate recommendations for user_id:10 based on 20 items being liked by him. So, we first get the 20 items being liked/consumed by the user as shown below:

```
pd.to_numeric(dataset_sort_des.Customer_id)
dataset_sort_des = dataset.sort_values(['Customer_id'], ascending=[True])
filter1 = dataset_sort_des[dataset_sort_des['Customer_id'] == '10'].Movie_Id
filter1 = filter1.tolist()
filter1 = filter1[:20]
print("Items liked by user: ",filter1) #these 20 movies have been watch by user recently.
```

```
Items liked by user:  [191, 483, 299, 197, 571, 473, 285, 175, 468]
```

```
dataset_sort_des[dataset_sort_des['Customer_id'] == 10]
```

Customer_id	rating	dates	Movie_Id	Year	Name

Next, for each item being liked by the user1, we recommend 1 similar items. The number of similar items to be recommended can vary depending on the need of the system.

```
distances1=[]
indices1=[]
for i in filter1:
    distances , indices = knn.kneighbors(csr_sample[i],n_neighbors=3) # 1 movie for each m
    indices = indices.flatten()
    indices= indices[1:]
    indices1.extend(indices)
print("Items to be recommended: ",indices1)
```

```
Items to be recommended:  [196611, 305194, 38, 68, 340097, 976, 14566, 139228, 32789
```