pearson_correlation_coefficient_recommender_system

October 30, 2019

[1]:
```
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from tqdm import tqdm
import heapq
```

[2]:
```
df=pd.read_csv('ratings.csv')
```

[3]:
```
user_movie_rating = df.pivot_table(index='userId', columns='movieId',␣
 ↪values='rating')
```

[4]:
```
# klen=[]
```

[5]:
```
# for j in tqdm(range(1,len(user_movie_rating.index)+1)):
#     dist_temp=[]
#     klen_temp=[]
#     for i in range(1,len(user_movie_rating.index)+1):
#         user1_rating=user_movie_rating.iloc[j-1][user_movie_rating.iloc[j-1].
 ↪isna()==False]
#         user1_movieId=list(user_movie_rating.iloc[j-1][user_movie_rating.
 ↪iloc[j-1].isna()==False].index)
#         user_i_rating=user_movie_rating.iloc[i-1][user_movie_rating.iloc[i-1].
 ↪isna()==False]
#         user_i_1_rating=user_i_rating[user_i_rating.index.
 ↪isin(user1_movieId)]
#         user_i_1_movieId=list(user_i_1_rating.index)
#         user1_i_rating=user1_rating[user1_rating.index.
 ↪isin(user_i_1_movieId)]
#         user1_mean=np.average(user1_rating)
#         user_i_mean=np.average(user_i_rating)
#         a=sum((user1_i_rating-user1_mean)*(user_i_1_rating-user_i_mean))
#         b=np.sqrt(sum(np.square(user_i_rating-user_i_mean)))
#         c=np.sqrt(sum(np.square(user1_rating-user1_mean)))
#         k=a/(b*c)
#         klen_temp.append(k)
#     klen.append(klen_temp)
```

[6]:
```
# df1=pd.DataFrame(klen)
```

[7]:
```
# df1
```

[8]:
```
# df1.to_csv('pearson_similarity.csv', index=False, header=False)
```

[9]:
```
similarity_df=pd.read_csv('pearson_similarity.csv',header=None)
```

[10]:
```
similarity_df.fillna(0.0,inplace=True)
```

[11]:
```
similarity_df
```

[11]:

```
          0         1         2         3         4         5         6    \
0    1.000000  0.001265  0.000553  0.048419  0.021847 -0.045497 -0.006200
1    0.001265  1.000000  0.000000 -0.017164  0.021796 -0.021051 -0.011114
2    0.000553  0.000000  1.000000 -0.011260 -0.031539  0.004800  0.000000
3    0.048419 -0.017164 -0.011260  1.000000 -0.029620  0.013956  0.058091
4    0.021847  0.021796 -0.031539 -0.029620  1.000000  0.009111  0.010117
..        ...       ...       ...       ...       ...       ...       ...
605  0.012016  0.006226 -0.037289  0.020590  0.026319 -0.009137  0.028326
606  0.055261 -0.020504 -0.007789  0.014628  0.031896  0.045501  0.030981
607  0.075224 -0.006001 -0.013001 -0.037569 -0.001751  0.021727  0.028414
608 -0.025713 -0.060091  0.000000 -0.017884  0.093829  0.053017  0.008754
609  0.010932  0.024999  0.019550 -0.000995 -0.000278  0.009603  0.068430

          7         8         9    ...       600       601       602  \
0    0.047013  0.019510 -0.008754   ...  0.018127 -0.017172 -0.015221
1   -0.048085  0.000000  0.003012   ... -0.050551 -0.031581 -0.001688
2   -0.032471  0.000000  0.000000   ... -0.004904 -0.016117  0.017749
3    0.002065 -0.005874  0.051590   ... -0.037687  0.063122  0.027640
4   -0.012284  0.000000 -0.033165   ...  0.015964  0.012427  0.027076
..        ...       ...       ...   ...       ...       ...       ...
605  0.022277  0.031633 -0.039946   ...  0.053683  0.016384  0.098011
606  0.048822 -0.012161 -0.017656   ...  0.049059  0.038197  0.049317
607  0.071759  0.032783 -0.052000   ...  0.069198  0.051388  0.012801
608  0.077180  0.000000 -0.040090   ...  0.043465  0.062400  0.015334
609  0.017144  0.051898 -0.026004   ...  0.021603  0.030061  0.051255

          603       604       605       606       607       608       609
0   -0.037059 -0.029121  0.012016  0.055261  0.075224 -0.025713  0.010932
1    0.000000  0.000000  0.006226 -0.020504 -0.006001 -0.060091  0.024999
2    0.000000 -0.001431 -0.037289 -0.007789 -0.013001  0.000000  0.019550
3   -0.013782  0.040037  0.020590  0.014628 -0.037569 -0.017884 -0.000995
4    0.012461 -0.036272  0.026319  0.031896 -0.001751  0.093829 -0.000278
..        ...       ...       ...       ...       ...       ...       ...
605  0.061078  0.019678  1.000000  0.017927  0.056676  0.038422  0.075464
606  0.002355 -0.029381  0.017927  1.000000  0.044514  0.019049  0.021860
607  0.006319 -0.007978  0.056676  0.044514  1.000000  0.050714  0.054454
608  0.094038 -0.054722  0.038422  0.019049  0.050714  1.000000 -0.012471
609  0.015621  0.069837  0.075464  0.021860  0.054454 -0.012471  1.000000
```

```
[610 rows x 610 columns]
```

[12]:
```
user_similarity_dict={}
for i in tqdm(range(similarity_df.shape[0])):
    sorted_similarity=heapq.nlargest(10,similarity_df.iloc[i])[1:]
    dict1={}
    for j in sorted_similarity:
        dict1[similarity_df.iloc[i][similarity_df.iloc[i]==j].index.
 ↪values[0]+1]=j
    user_similarity_dict[i+1]=dict1
```

```
100%|| 610/610 [00:07<00:00, 83.16it/s]
```

[13]:
```
abs_error_sum=0
root_square_sum=0
count=0
for i in tqdm(user_movie_rating.index):
    if len(list(user_similarity_dict[i].keys()))==1:
        non_null_movies=list(user_movie_rating.loc[i][user_movie_rating.loc[i].
 ↪isna()==False].index)
        k0=user_movie_rating.loc[list(user_similarity_dict[i].keys())[0]]
        k0=k0[k0.index.isin(non_null_movies)]
        k0.fillna(0,inplace=True)
        predicted_data=k0
        predicted_data=np.ceil(predicted_data*2)/2
        predicted_data=predicted_data.replace(6.0,5.0)
        predicted_data=predicted_data.replace(5.5,5.0)
        actual_data=user_movie_rating.loc[i][user_movie_rating.loc[i].
 ↪isna()==False]
        actual_data=np.round(actual_data)
        abs_error_sum=abs_error_sum+sum(np.abs(predicted_data-actual_data))
        root_square_sum=root_square_sum+sum(np.
 ↪square(predicted_data-actual_data))
        count=count+len(non_null_movies)
    if len(list(user_similarity_dict[i].keys()))==2:
        non_null_movies=list(user_movie_rating.loc[i][user_movie_rating.loc[i].
 ↪isna()==False].index)
        k0=user_movie_rating.loc[list(user_similarity_dict[i].keys())[0]]
        k0=k0[k0.index.isin(non_null_movies)]
        k0.fillna(0,inplace=True)
        k1=user_movie_rating.loc[list(user_similarity_dict[i].keys())[1]]
        k1=k1[k1.index.isin(non_null_movies)]
        k1.fillna(0,inplace=True)
        predicted_data=(k0+k1)/2
        predicted_data=np.ceil(predicted_data*2)/2
        predicted_data=predicted_data.replace(6.0,5.0)
```

```
        predicted_data=predicted_data.replace(5.5,5.0)
        actual_data=user_movie_rating.loc[i][user_movie_rating.loc[i].
↪isna()==False]
        actual_data=np.round(actual_data)
        abs_error_sum=abs_error_sum+sum(np.abs(predicted_data-actual_data))
        root_square_sum=root_square_sum+sum(np.
↪square(predicted_data-actual_data))
        count=count+len(non_null_movies)
    if len(list(user_similarity_dict[i].keys()))>=3:
        non_null_movies=list(user_movie_rating.loc[i][user_movie_rating.loc[i].
↪isna()==False].index)
        k0=user_movie_rating.loc[list(user_similarity_dict[i].keys())[0]]
        k0=k0[k0.index.isin(non_null_movies)]
        k0.fillna(0,inplace=True)
        k1=user_movie_rating.loc[list(user_similarity_dict[i].keys())[1]]
        k1=k1[k1.index.isin(non_null_movies)]
        k1.fillna(0,inplace=True)
        k2=user_movie_rating.loc[list(user_similarity_dict[i].keys())[2]]
        k2=k2[k2.index.isin(non_null_movies)]
        k2.fillna(0,inplace=True)
        predicted_data=(k0+k1+k2)/3
        predicted_data=np.ceil(predicted_data*2)/2
        predicted_data=predicted_data.replace(6.0,5.0)
        predicted_data=predicted_data.replace(5.5,5.0)
        actual_data=user_movie_rating.loc[i][user_movie_rating.loc[i].
↪isna()==False]
        actual_data=np.round(actual_data)
        abs_error_sum=abs_error_sum+sum(np.abs(predicted_data-actual_data))
        root_square_sum=root_square_sum+sum(np.
↪square(predicted_data-actual_data))
        count=count+len(non_null_movies)
```

```
100%|| 610/610 [00:04<00:00, 122.08it/s]
```

[16]:
```
np.sqrt(root_square_sum/count)
```

[16]:
```
2.8356210706442226
```

[17]:
```
abs_error_sum/count
```

[17]:
```
2.4707792851759294
```

[15]:
```
abs_error_sum/count
```

[15]:
```
2.4707792851759294
```

[ ]:
```

```