Devasy Patel

Practical 8

20BCE057

Date : 10/04/2023

In [ ]:
```
# build recommendation system using collaborative filtering and predict the ratings
import pandas as pd

dict = {'U1':{'M1':3,'M3':4, 'M6':1},
        'U2':{'M3':2, 'M4':2},
        'U3':{'M1':2, 'M2':4, 'M4':4, 'M5':4, 'M6':3},
        'U4':{'M1':1, 'M2':5, 'M3':2, 'M5':3},
        'U5':{'M3':1, 'M4':5, 'M5':4, 'M6':3},
        'U6':{'M1':4, 'M5':2},
        'U7':{'M2':4, 'M3':3}
        }

df = pd.DataFrame(dict)
```

In [ ]:
```
df
```

Out[ ]:

	U1	U2	U3	U4	U5	U6	U7
M1	3.0	NaN	2.0	1.0	NaN	4.0	NaN
M3	4.0	2.0	NaN	2.0	1.0	NaN	3.0
M6	1.0	NaN	3.0	NaN	3.0	NaN	NaN
M4	NaN	2.0	4.0	NaN	5.0	NaN	NaN
M2	NaN	NaN	4.0	5.0	NaN	NaN	4.0
M5	NaN	NaN	4.0	3.0	4.0	2.0	NaN

In [ ]:
```
# calculate the similarity between two users via centered cosine similarity


# user-user similarity matrix
# calculate the similarity between two users via centered cosine similarity
# user-user similarity matrix
import numpy as np
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity

def user_similarity(df):
    # calculate the similarity between two users via centered cosine similarity
    # user-user similarity matrix
    df = df.T
    df = df - df.mean(axis=1).values.reshape(-1,1)
    df = df.fillna(0)
    similarity = cosine_similarity(df)
    similarity = pd.DataFrame(similarity, index=df.index, columns=df.index)
```

```
    return similarity

df = pd.DataFrame(dict)
similarity = user_similarity(df)
```

In [ ]:
```
# similarity using pearson correlation
import numpy as np
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity

def user_similarity(df):
    # calculate the similarity between two users via pearson correlation
    # user-user similarity matrix
    # df = df.T
    # substraction of mean from each user ratings
    df = df - df.mean(axis=1).values.reshape(-1,1)
    df = df.fillna(0)
    similarity = df.corr()
    return similarity


def item_similarity(df):
    # calculate the similarity between two items via pearson correlation
    # item-item similarity matrix
    df= df.T
    df = df - df.mean(axis=0).values.reshape(1,-1)
    df = df.fillna(0)
    similarity = df.corr()
    return similarity
ussim = user_similarity(df)
itsim = item_similarity(df)
ussim
```

Out[ ]:

	U1	U2	U3	U4	U5	U6	U7
U1	1.000000	-0.118914	-0.512238	-0.354768	-0.739339	0.173793	0.644812
U2	-0.118914	1.000000	-0.138903	-0.147289	-0.380785	0.029600	-0.146525
U3	-0.512238	-0.138903	1.000000	0.306765	0.510586	-0.764792	0.089733
U4	-0.354768	-0.147289	0.306765	1.000000	0.209860	-0.602367	-0.366186
U5	-0.739339	-0.380785	0.510586	0.209860	1.000000	-0.241840	-0.630194
U6	0.173793	0.029600	-0.764792	-0.602367	-0.241840	1.000000	-0.008408
U7	0.644812	-0.146525	0.089733	-0.366186	-0.630194	-0.008408	1.000000

In [ ]:
```
# predict the ratings for 'M3' for user 'U3' consider the top 3 similar users

def predict_rating(df, similarity, user, item):
    # predict the ratings for 'item' for 'user' consider the top 3 similar users
    df = df.T
    # similarity = similarity.T
    # get the top 3 similar users
    top3 = similarity[user].sort_values(ascending=False)[1:4]
    # print(top3)
    # get the ratings of the top 3 similar users for item
    # df
    top3rating = df.loc[top3.index, item]
    # print(top3rating)
    # calculate the predicted rating
    rating = np.dot(top3, top3rating) / 3
```

```
    return rating


def predict_rating2(df, similarity, user, item):
    #  predict based on item-item similarity
    # predict the ratings for 'item' for 'user' consider the top 3 similar users
    df = df.T
    # similarity = similarity.T
    # get the top 3 similar users
    top3 = similarity[item].sort_values(ascending=False)[1:4]
    print(top3)
    # get the ratings of the top 3 similar users for item
    # df
    top3rating = df.loc[user, top3.index]
    print(top3rating)
    # calculate the predicted rating
    rating = np.dot(top3, top3rating) / 3
    return rating

predict_rating2(df, itsim, 'U1', 'M4')
```

```
M5    0.348932
M6    0.314970
M1   -0.034503
Name: M4, dtype: float64
M5    NaN
M6    1.0
M1    3.0
Name: U1, dtype: float64
nan
```

Out[ ]:

In [ ]:

```
# calculate the predicted ratings for all the items for all the users and fill the


def fill_matrix(df, similarity, method='user'):
    # df = df.T
    original = df.copy()
    # calculate the predicted ratings for all the items for all the users and fill
    if method == 'user':
        for user in df.columns:
            for item in df.index:
                if np.isnan(df.loc[item, user]):
                    df.loc[item, user] = predict_rating(original, similarity, user,
    else:
        for user in df.columns:
            for item in df.index:
                if np.isnan(df.loc[item, user]):
                    ans = predict_rating2(original, similarity, user, item)
                    print("Predicting for user: ", user, " and item: ", item, "is
                    df.loc[item, user] = ans
    return df


fill_matrix(df, ussim, method='user')
```

Out[ ]:

	U1	U2	U3	U4	U5	U6	U7
M1	3.0	-0.172049	2.000000	1.000000	0.087891	4.000000	0.693422
M3	4.0	2.000000	0.464438	2.000000	1.000000	0.243049	3.000000
M6	1.0	NaN	3.000000	NaN	3.000000	NaN	NaN
M4	NaN	2.000000	4.000000	0.660594	5.000000	NaN	NaN
M2	NaN	NaN	4.000000	5.000000	NaN	NaN	4.000000
M5	NaN	NaN	4.000000	3.000000	4.000000	2.000000	NaN

In [ ]:

Out[ ]:

	M1	M3	M6	M4	M2	M5
U1	3.292166	1.790863	-0.057549	-1.814337	-2.777620	-1.935186
U2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
U3	-0.260533	2.713863	3.958870	5.511858	8.681082	5.867787
U4	-0.162744	3.374580	2.634517	4.057348	10.854468	5.886879
U5	-1.110923	1.425044	3.502843	6.209486	4.432326	5.493775
U6	2.268097	-0.699031	-2.800452	-4.058699	-5.552735	-3.313689
U7	-0.117829	3.226384	1.888635	3.637948	8.534369	5.251506