Data Analytics Week 5 Assignment

202011431 산업공학과 차승현

Collaborative filtering

건국대학교

Analysis Procedure

```
import pandas as pd
import numpy as np
import math
from numpy import dot
from numpy.linalg import norm
```

```
data = pd.read_csv(r'C:₩Users₩moon_₩DataAnalytics/data_week5.txt')
data
```

	User	Item	Score
0	1	2	4
1	1	4	3
2	1	6	2
3	1	8	1
4	1	9	2
...	...	...	...
61	10	4	3
62	10	5	4
63	10	8	2
64	10	9	1
65	10	10	1

66 rows × 3 columns

라이브러리를 import하고 데이터를 불러온다.

```
df1 = data.pivot_table('Score', index = 'User', columns='Item')
df1
```

Item	1	2	3	4	5	6	7	8	9	10
User										
1	NaN	4.0	NaN	3.0	NaN	2.0	NaN	1.0	2.0	3.0
2	5.0	2.0	1.0	3.0	2.0	NaN	1.0	NaN	NaN	NaN
3	1.0	1.0	NaN	2.0	3.0	5.0	NaN	2.0	4.0	4.0
4	NaN	NaN	4.0	5.0	NaN	NaN	3.0	NaN	3.0	NaN
5	2.0	1.0	NaN	NaN	4.0	3.0	NaN	1.0	1.0	2.0
6	2.0	NaN	4.0	2.0	NaN	1.0	3.0	5.0	1.0	2.0
7	1.0	3.0	NaN	2.0	5.0	NaN	2.0	NaN	1.0	4.0
8	NaN	1.0	1.0	2.0	3.0	3.0	NaN	3.0	1.0	NaN
9	3.0	NaN	2.0	NaN	1.0	NaN	4.0	NaN	5.0	NaN
10	1.0	2.0	1.0	3.0	4.0	NaN	NaN	2.0	1.0	1.0

data에서 user별 점수의 분포를 나타내는 matrix를 생성한다. 이를 df1이라는 변수에 저장한다.

```
user_x_df =pd.DataFrame({2: [2], 3: [1], 5: [3], 8: [4], 9: [3]})

user_x_df
```

	2	3	5	8	9
0	2	1	3	4	3

```
df = pd.concat([df1, user_x_df])
df
```

	1	2	3	4	5	6	7	8	9	10
1	NaN	4.0	NaN	3.0	NaN	2.0	NaN	1.0	2.0	3.0
2	5.0	2.0	1.0	3.0	2.0	NaN	1.0	NaN	NaN	NaN
3	1.0	1.0	NaN	2.0	3.0	5.0	NaN	2.0	4.0	4.0
4	NaN	NaN	4.0	5.0	NaN	NaN	3.0	NaN	3.0	NaN
5	2.0	1.0	NaN	NaN	4.0	3.0	NaN	1.0	1.0	2.0
6	2.0	NaN	4.0	2.0	NaN	1.0	3.0	5.0	1.0	2.0
7	1.0	3.0	NaN	2.0	5.0	NaN	2.0	NaN	1.0	4.0
8	NaN	1.0	1.0	2.0	3.0	3.0	NaN	3.0	1.0	NaN
9	3.0	NaN	2.0	NaN	1.0	NaN	4.0	NaN	5.0	NaN
10	1.0	2.0	1.0	3.0	4.0	NaN	NaN	2.0	1.0	1.0
0	NaN	2.0	1.0	NaN	3.0	NaN	NaN	4.0	3.0	NaN

User X에 대한 정보를 데이터프레임화하여, df1과 합병한 후 df라는 변수로 저장한다.

```
df['평균값'] = df.iloc[:, 0:10].mean(axis=1)
df
```

	1	2	3	4	5	6	7	8	9	10	평균값
1	NaN	4.0	NaN	3.0	NaN	2.0	NaN	1.0	2.0	3.0	2.500000
2	5.0	2.0	1.0	3.0	2.0	NaN	1.0	NaN	NaN	NaN	2.333333
3	1.0	1.0	NaN	2.0	3.0	5.0	NaN	2.0	4.0	4.0	2.750000
4	NaN	NaN	4.0	5.0	NaN	NaN	3.0	NaN	3.0	NaN	3.750000
5	2.0	1.0	NaN	NaN	4.0	3.0	NaN	1.0	1.0	2.0	2.000000
6	2.0	NaN	4.0	2.0	NaN	1.0	3.0	5.0	1.0	2.0	2.500000
7	1.0	3.0	NaN	2.0	5.0	NaN	2.0	NaN	1.0	4.0	2.571429
8	NaN	1.0	1.0	2.0	3.0	3.0	NaN	3.0	1.0	NaN	2.000000
9	3.0	NaN	2.0	NaN	1.0	NaN	4.0	NaN	5.0	NaN	3.000000
10	1.0	2.0	1.0	3.0	4.0	NaN	NaN	2.0	1.0	1.0	1.875000
0	NaN	2.0	1.0	NaN	3.0	NaN	NaN	4.0	3.0	NaN	2.600000

Df에 '평균값'이라는 column을 새로 생성하여 각 행마다 평균을 계산한다. 이때, Nan은 없는 값으로 간주하고 표본의 개수에 포함하지 않는다. 이후 1열부터 10열까지 (item 1 to 10)의 rating 값에서 평균값을 제거하여 tendency를 제거한다.

```
for i in range(len(df)):
    for j in range(10):
        if math.isnan(df.iloc[i, j]) == True:
            pass
        else:
            df.iloc[i, j] = df.iloc[i, j] - df.iloc[i, 10]
```

애초에 i행 j열의 값이 null인 것들은 계산을 할 필요가 없다. Null이 아닌 값에 평균값을 빼주어 tendency를 제거한 값을 산출한다.

	1	2	3	4	5	6	7	8	9	10
1	NaN	1.500000	NaN	0.500000	NaN	-0.50	NaN	-1.500	-0.500000	0.500000
2	2.666667	-0.333333	-1.333333	0.666667	-0.333333	NaN	-1.333333	NaN	NaN	NaN
3	-1.750000	-1.750000	NaN	-0.750000	0.250000	2.25	NaN	-0.750	1.250000	1.250000
4	NaN	NaN	0.250000	1.250000	NaN	NaN	-0.750000	NaN	-0.750000	NaN
5	0.000000	-1.000000	NaN	NaN	2.000000	1.00	NaN	-1.000	-1.000000	0.000000
6	-0.500000	NaN	1.500000	-0.500000	NaN	-1.50	0.500000	2.500	-1.500000	-0.500000
7	-1.571429	0.428571	NaN	-0.571429	2.428571	NaN	-0.571429	NaN	-1.571429	1.428571
8	NaN	-1.000000	-1.000000	0.000000	1.000000	1.00	NaN	1.000	-1.000000	NaN
9	0.000000	NaN	-1.000000	NaN	-2.000000	NaN	1.000000	NaN	2.000000	NaN
10	-0.875000	0.125000	-0.875000	1.125000	2.125000	NaN	NaN	0.125	-0.875000	-0.875000
0	NaN	-0.600000	-1.600000	NaN	0.400000	NaN	NaN	1.400	0.400000	NaN

```
df23589 = df[[2, 3 ,5 , 8 , 9]]
df23589
```

	2	3	5	8	9
1	1.500000	NaN	NaN	-1.500	-0.500000
2	-0.333333	-1.333333	-0.333333	NaN	NaN
3	-1.750000	NaN	0.250000	-0.750	1.250000
4	NaN	0.250000	NaN	NaN	-0.750000
5	-1.000000	NaN	2.000000	-1.000	-1.000000
6	NaN	1.500000	NaN	2.500	-1.500000
7	0.428571	NaN	2.428571	NaN	-1.571429
8	-1.000000	-1.000000	1.000000	1.000	-1.000000
9	NaN	-1.000000	-2.000000	NaN	2.000000
10	0.125000	-0.875000	2.125000	0.125	-0.875000
0	-0.600000	-1.600000	0.400000	1.400	0.400000

코사인 유사도를 산출해야 하는데, User X는 Item 2, 3, 5, 8, 9에 관해서만 ratio를 부여하였으므로 해당 열의 값의 df를 뽑아낸다.

```
df23589 = df23589.replace(np.nan, 0)
df23589
```

	2	3	5	8	9
1	1.500000	0.000000	0.000000	-1.500	-0.500000
2	-0.333333	-1.333333	-0.333333	0.000	0.000000
3	-1.750000	0.000000	0.250000	-0.750	1.250000
4	0.000000	0.250000	0.000000	0.000	-0.750000
5	-1.000000	0.000000	2.000000	-1.000	-1.000000
6	0.000000	1.500000	0.000000	2.500	-1.500000
7	0.428571	0.000000	2.428571	0.000	-1.571429
8	-1.000000	-1.000000	1.000000	1.000	-1.000000
9	0.000000	-1.000000	-2.000000	0.000	2.000000
10	0.125000	-0.875000	2.125000	0.125	-0.875000
0	-0.600000	-1.600000	0.400000	1.400	0.400000

Nan(np.nan)값은 유사도를 산출할 수 없으므로, 0으로 대체해준다.

```
user_x = []
for i in range(10):
    user_x.append(df23589.iloc[-1, :])

user_x = np.array(user_x)
user_x
```

```
array([[-0.6, -1.6,  0.4,  1.4,  0.4],
       [-0.6, -1.6,  0.4,  1.4,  0.4],
       [-0.6, -1.6,  0.4,  1.4,  0.4],
       [-0.6, -1.6,  0.4,  1.4,  0.4],
       [-0.6, -1.6,  0.4,  1.4,  0.4],
       [-0.6, -1.6,  0.4,  1.4,  0.4],
       [-0.6, -1.6,  0.4,  1.4,  0.4],
       [-0.6, -1.6,  0.4,  1.4,  0.4],
       [-0.6, -1.6,  0.4,  1.4,  0.4],
       [-0.6, -1.6,  0.4,  1.4,  0.4]])
```

```
for i in range(10):
    for j in range(len(user_x[0])):
        if (df23589.iloc[i, j]) == 0:
            user_x[i][j] = 0
```

```
user_x
```

```
array([[-0.6,  0. ,  0. ,  1.4,  0.4],
       [-0.6, -1.6,  0.4,  0. ,  0. ],
       [-0.6,  0. ,  0.4,  1.4,  0.4],
       [ 0. , -1.6,  0. ,  0. ,  0.4],
       [-0.6,  0. ,  0.4,  1.4,  0.4],
       [ 0. , -1.6,  0. ,  1.4,  0.4],
       [-0.6,  0. ,  0.4,  0. ,  0.4],
       [-0.6, -1.6,  0.4,  1.4,  0.4],
       [ 0. , -1.6,  0.4,  0. ,  0.4],
       [-0.6, -1.6,  0.4,  1.4,  0.4]])
```

User X에 관한 array를 만들어준다. 이 때, user 1부터 10까지 2, 3, 5, 8, 9 item 중에서 점수를 부여하지 않거나 tendency 제거 후 0점을 나타내는 값에 대해서는 똑같이 user_x에 0으로 값을 넣어준다.

```
#코사인 유사도를 구하는 함수
def cos_sim(a, b):
    return dot(a, b) / (norm(a) * norm(b))
```

```
user1_10 = df23589.iloc[:10, :]
user1_10
```

	2	3	5	8	9
1	1.500000	0.000000	0.000000	-1.500	-0.500000
2	-0.333333	-1.333333	-0.333333	0.000	0.000000
3	-1.750000	0.000000	0.250000	-0.750	1.250000
4	0.000000	0.250000	0.000000	0.000	-0.750000
5	-1.000000	0.000000	2.000000	-1.000	-1.000000
6	0.000000	1.500000	0.000000	2.500	-1.500000
7	0.428571	0.000000	2.428571	0.000	-1.571429
8	-1.000000	-1.000000	1.000000	1.000	-1.000000
9	0.000000	-1.000000	-2.000000	0.000	2.000000
10	0.125000	-0.875000	2.125000	0.125	-0.875000

User x의 ratio를 제외한 user 1 to user 10까지의 값의 ratio(tendency가 제거된)를 정리한 데이터프레임을 user1_10에 저장해준 후 코사인 유사도를 계산한다.

```
for i in range(len(df)-1):
    print('User X와 User', i+1, '의 코사인 유사도: ', cos_sim(user1_10.iloc[i, :], user_x[i]))
```

```
User X와 User 1 의 코사인 유사도:  -0.9323464747969039
User X와 User 2 의 코사인 유사도:  0.8864052604279183
User X와 User 3 의 코사인 유사도:  0.16116459280507617
User X와 User 4 의 코사인 유사도:  -0.5368754921931592
User X와 User 5 의 코사인 유사도:  -0.09304842103984706
User X와 User 6 의 코사인 유사도:  0.07049248907296893
User X와 User 7 의 코사인 유사도:  0.03554592191701464
User X와 User 8 의 코사인 유사도:  0.7060180864974624
User X와 User 9 의 코사인 유사도:  0.31426968052735443
User X와 User 10 의 코사인 유사도:  0.35574885436199155
```

User X와 User n (n= 1, 2, …, 10)의 코사인 유사도는 다음과 같다.
여기서, 코사인 유사도가 높은 관계를 보이는 user를 선발해본다면 User 2, User 8, User 10, User 9를 선발할 수 있다. 이 네 user는 모두 0.3 이상의

코사인 유사도를 가진다.

여기서, User 2, 8, 10, 9의 iteam 1, 4, 6, 7, 10(아직 User X가 ratio를 부여하지 않은)에 대한 ratio는 다음과 같다.

	Item 1	Item 4	Item 6	Item7	Item 10
User 2	2.6666	0.6666	Nan	-1.3333	Nan
User 8	Nan	0	1	Nan	Nan
User 10	-0.875	1.125	Nan	Nan	-0.875
User 9	0	Nan	Nan	1	Nan

User 2, 8, 10, 9와 User X의 코사인 유사도는 다음과 같다.

	Cosine Similarity
User 2	0.866405
User 8	0.706018
User 10	0.355749
User 9	0.31427

$$r_{u,i} = k \sum_{u' \in U} sim(u,u') \times r_{u',i}$$

where k is a normalizing factor defined as $k = \frac{1}{\sum_{u' \in U} sim(u,u')}$

다음 공식을 사용하여 item1, 4, 6, 7, 10에 대해 prediction score를 산출하면

Item 1: (2.6666 * 0.866405 – 0.875 * 0.355749) / (0.866405 + 0.355749) = 1.635698

Item 4: (0.6666 * 0.866405 + 0 * 0.706018 + 1.125 * 0.355749) / (0.866405 + 0.706018 + 0.355749) = 0.507093

… 와 같이 계산할 수 있다. 이를 item 6, 7, 10 에도 적용한다면 다음과 같이 결과가 나온다.

Item 1	1.635698
Item 4	0.507093
Item 6	1
Item 7	-0.71223
Item 10	-0.875

해당 값에서, User X의 tendency(평균 값)였던 2.6을 더해주면 prediction score를 산출할 수 있다.

Item	**1**	**2**	**3**	**4**	**5**
User X	4.235698	2	1	3.107093	3
Item	**6**	**7**	**8**	**9**	**10**
User X	3.6	1.887774	4	3	1.725

User X에게 추천해줄 수 있는 item 1, 4, 6, 7, 10의 추천 순위는 다음과 같다.

Rank	Item	Prediction Score
1	Item 1	4.235698
2	Item 6	3.6
3	Item 4	3.107093
4	Item 7	1.887774
5	Item 10	1.725