아이템 기반 최근접 이웃 협업 필터링

학습 목표

- 아이템 기반 이웃 협업 필터링에 대해 알아봅니다.

데이터

- [https://grouplens.org/datasets/movielens/ (https://grouplens.org/datasets/movielens/)](https://grouplens.org/datasets/movielens/)
- 파일명 : ml-latest-small.zip(size: 1MB)
- 10만개의 평점 정보(rating)

In [8]:

```
import pandas as pd
import numpy as np

movies = pd.read_csv("../data/grouplens/ml_small/movies.csv")
ratings = pd.read_csv("../data/grouplens/ml_small/ratings.csv")

print(movies.shape, ratings.shape)
```

```
(9742, 3) (100836, 4)
```

In [9]:

```
movies.head(3)
```

Out[9]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance

In [10]:

```
# 사용자별 영화에 대한 평점을 매긴 데이터 셋
ratings.head(3)
```

Out[10]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224

사용자와 아이템 간의 평점에 기반해 추천하는 시스템

- 사용자를 행으로, 모든 영화를 컬럼으로 구성한 데이터 셋으로 변경
  - pivot_table() 함수를 이용하면 가능.
  - columns='movidId'와 같이 부여하면 movieId 컬럼의 모든 값이 새로운 컬럼 이름으로 변경됨.

In [11]:

```
ratings.columns
```

Out[11]:

```
Index(['userId', 'movieId', 'rating', 'timestamp'], dtype='object')
```

In [12]:

```
sel = ['userId', 'movieId', 'rating']
ratings = ratings[sel]
ratings_m = ratings.pivot_table('rating', index='userId', columns='movieId')
print(ratings_m.shape) # 610명 사용자와 9724개의 영화 제목
ratings_m.head(3)
```

```
(610, 9724)
```

Out[12]:

movieId	1	2	3	4	5	6	7	8	9	10	...	193565	193567	19357
userId														
1	4.0	NaN	4.0	NaN	NaN	4.0	NaN	NaN	NaN	NaN	...	NaN	NaN	Nal
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	Nal
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	Nal

3 rows × 9724 columns

- movidId가 모두 컬럼으로 변환.
- NaN값이 많다. 사용자가 평점을 매기지 않은 영화가 컬럼으로 변환되면서 NaN값으로 변경됨.

전처리

- NaN은 0으로 변환처리
- movidId를 영화명으로 변경
  - ratings와 movies를 합치기

In [13]:

```python
# title컬럼을 얻기 위해 movies와 조인
print(ratings.shape, movies.shape)
print(ratings.columns, movies.columns)
rating_movies = pd.merge(ratings, movies, on='movieId')
print(rating_movies.shape)
rating_movies
```

```
(100836, 3) (9742, 3)
Index(['userId', 'movieId', 'rating'], dtype='object') Index(['movieId', 'title', 'g
enres'], dtype='object')
(100836, 5)
```

Out[13]:

	userId	movieId	rating	title	genres
0	1	1	4.0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	5	1	4.0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	7	1	4.5	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
3	15	1	2.5	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
4	17	1	4.5	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
...	...	...	...	...	...
100831	610	160341	2.5	Bloodmoon (1997)	Action|Thriller
100832	610	160527	4.5	Sympathy for the Underdog (1971)	Action|Crime|Drama
100833	610	160836	3.0	Hazard (2005)	Action|Drama|Thriller
100834	610	163937	3.5	Blair Witch (2016)	Horror|Thriller
100835	610	163981	3.5	31 (2016)	Horror

100836 rows × 5 columns

In [14]:

```
# columns='title'로 title컬럼으로 피벗 수행
# 실제 평점 데이터 셋
ratings_m = rating_movies.pivot_table('rating',
                                     index='userId', columns='title')
ratings_m
```

Out[14]:

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteri r includ (198
userId										
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
...	...	...	...	...	...	...	...	...	...	
606	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
607	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
608	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
609	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
610	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3.5	N

610 rows × 9719 columns

In [15]:

```
# NaN을 0으로 변경
ratings_m = ratings_m.fillna(0)
ratings_m.head(3)
```

Out[15]:

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteri r includ (198
userId										
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	(
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	(
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	(

3 rows × 9719 columns

영화간 유사도 산출해 보기

- **코사인 유사도를 기반**으로 하여 유사도 측정
  - 사이킷 런의 cosine_similarity()를 이용하여 측정
  - cosine_similarity()함수는 행을 기준으로 서로 다른 행을 비교해 유사도 산출
    - ratings_m은 userId가 기준이므로 cosine_similarity()를 적용하면 영화간의 유사도가 아닌 사용자간의 유사도가 생기므로
      - 행열 변경

In [16]:

```
ratings_m_T = ratings_m.transpose()
ratings_m_T.head(3)
```

Out[16]:

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	60
title																		
'71 (2014)	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.
'Hellboy': The Seeds of Creation (2004)	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.
'Round Midnight (1986)	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.

3 rows × 610 columns

영화의 유사도 구하기

- cosine_similarity(A, B) : A와 B사이의 코사인 유사도를 계산한다.

In [17]:

```
from sklearn.metrics.pairwise import cosine_similarity
```

In [18]:

```
item_sim = cosine_similarity(ratings_m_T, ratings_m_T)
print(type(item_sim), item_sim.shape)
print(item_sim)
```

```
<class 'numpy.ndarray'> (9719, 9719)
[[1.         0.         0.         ... 0.32732684 0.         0.        ]
 [0.         1.         0.70710678 ... 0.         0.         0.        ]
 [0.         0.70710678 1.         ... 0.         0.         0.        ]
 ...
 [0.32732684 0.         0.         ... 1.         0.         0.        ]
 [0.         0.         0.         ... 0.         1.         0.        ]
 [0.         0.         0.         ... 0.         0.         1.        ]]
```

cosine_similarity()의 결과값을 데이터프레임으로 변환

In [19]:

```
col = ratings_m.columns
item_sim_df = pd.DataFrame(data=item_sim, index=col, columns=col)
print(item_sim_df.shape)
item_sim_df.head(3)
```

(9719, 9719)

Out[19]:

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*b i
title										
'71 (2014)	1.0	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.141653	
'Hellboy': The Seeds of Creation (2004)	0.0	1.000000	0.707107	0.0	0.0	0.0	0.000000	0.0	0.000000	
'Round Midnight (1986)	0.0	0.707107	1.000000	0.0	0.0	0.0	0.176777	0.0	0.000000	

3 rows × 9719 columns

영화 '71(2014)'와 유사도가 높은 상위 30개 영화 추출해보기

In [26]:

```
item_sim_df["'71 (2014)"].sort_values(ascending=False)[:30]
```

Out[26]:

```
title
'71 (2014)                                                          1.0
City of Lost Souls, The (Hyôryuu-gai) (2000)                        1.0
Clown (2014)                                                        1.0
Strange Circus (Kimyô na sâkasu) (2005)                             1.0
Ginger Snaps: Unleashed (2004)                                      1.0
Ginger Snaps Back: The Beginning (2004)                             1.0
Get on the Bus (1996)                                               1.0
Collector, The (2009)                                               1.0
Prince of Darkness (1987)                                           1.0
Gen-X Cops (1999)                                                   1.0
Stingray Sam (2009)                                                 1.0
Pulse (Kairo) (2001)                                                1.0
Cooties (2015)                                                      1.0
Frontière(s) (2007)                                                 1.0
From Beyond (1986)                                                  1.0
Rapture-Palooza (2013)                                              1.0
Stake Land (2010)                                                   1.0
Reality (2014)                                                      1.0
Crimson Peak (2015)                                                 1.0
Spring (2015)                                                       1.0
Crippled Avengers (Can que) (Return of the 5 Deadly Venoms) (1981)  1.0
Stuck (2007)                                                        1.0
Afflicted (2013)                                                    1.0
The Boy and the Beast (2015)                                        1.0
Goodnight Mommy (Ich seh ich seh) (2014)                            1.0
Heartless (2009)                                                    1.0
Pact, The (2012)                                                    1.0
Dobermann (1997)                                                    1.0
Hazard (2005)                                                       1.0
Haunter (2013)                                                      1.0
Name: '71 (2014), dtype: float64
```

In [21]:

```
col
```

Out[21]:

```
Index([''71 (2014)', ''Hellboy': The Seeds of Creation (2004)',
       ''Round Midnight (1986)', ''Salem's Lot (2004)',
       ''Til There Was You (1997)', ''Tis the Season for Love (2015)',
       ''burbs, The (1989)', ''night Mother (1986)',
       '(500) Days of Summer (2009)', '*batteries not included (1987)',
       ...
       'Zulu (2013)', '[REC] (2007)', '[REC]²  (2009)',
       '[REC]³  3 Génesis (2012)',
       'anohana: The Flower We Saw That Day - The Movie (2013)',
       'eXistenZ (1999)', 'xXx (2002)', 'xXx: State of the Union (2005)',
       '¡Three Amigos! (1986)', 'À nous la liberté (Freedom for Us) (1931)'],
      dtype='object', name='title', length=9719)
```

In [27]:

```
item_sim_df["Godfather, The (1972)"].sort_values(ascending=False)[:10]
```

Out[27]:

```
title
Godfather, The (1972)                                   1.000000
Godfather: Part II, The (1974)                          0.821773
Goodfellas (1990)                                       0.664841
One Flew Over the Cuckoo's Nest (1975)                  0.620536
Star Wars: Episode IV - A New Hope (1977)               0.595317
Fargo (1996)                                            0.588614
Star Wars: Episode V - The Empire Strikes Back (1980)   0.586030
Fight Club (1999)                                       0.581279
Reservoir Dogs (1992)                                   0.579059
Pulp Fiction (1994)                                     0.575270
Name: Godfather, The (1972), dtype: float64
```

In [28]:

```
item_sim_df["Inception (2010)"].sort_values(ascending=False)[:10]
```

Out[28]:

```
title
Inception (2010)                  1.000000
Dark Knight, The (2008)           0.727263
Inglourious Basterds (2009)       0.646103
Shutter Island (2010)             0.617736
Dark Knight Rises, The (2012)     0.617504
Fight Club (1999)                 0.615417
Interstellar (2014)               0.608150
Up (2009)                         0.606173
Avengers, The (2012)              0.586504
Django Unchained (2012)           0.581342
Name: Inception (2010), dtype: float64
```

최적화된 평점 스코어 만들기 (함수)

In [29]:

```
item_sim_df.head()
```

Out[29]:

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)
title									
'71 (2014)	1.0	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.141653
'Hellboy': The Seeds of Creation (2004)	0.0	1.000000	0.707107	0.000000	0.000000	0.0	0.000000	0.0	0.000000
'Round Midnight (1986)	0.0	0.707107	1.000000	0.000000	0.000000	0.0	0.176777	0.0	0.000000
'Salem's Lot (2004)	0.0	0.000000	0.000000	1.000000	0.857493	0.0	0.000000	0.0	0.000000
'Til There Was You (1997)	0.0	0.000000	0.000000	0.857493	1.000000	0.0	0.000000	0.0	0.000000

5 rows × 9719 columns

아이템 기반 협업 필터링에서 개인화된 예측 평점 계산

In [30]:

```
item_sim_df.head()
```

Out[30]:

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)
title									
'71 (2014)	1.0	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.141653
'Hellboy': The Seeds of Creation (2004)	0.0	1.000000	0.707107	0.000000	0.000000	0.0	0.000000	0.0	0.000000
'Round Midnight (1986)	0.0	0.707107	1.000000	0.000000	0.000000	0.0	0.176777	0.0	0.000000
'Salem's Lot (2004)	0.0	0.000000	0.000000	1.000000	0.857493	0.0	0.000000	0.0	0.000000
'Til There Was You (1997)	0.0	0.000000	0.000000	0.857493	1.000000	0.0	0.000000	0.0	0.000000

5 rows × 9719 columns

In [31]:

```
val = np.array(  [np.abs(item_sim_df).sum(axis=1)]  )  # 열기준-행의합(영화별 합)
print( val.shape )
val[0:10]
```

```
(1, 9719)
```

Out[31]:

```
array([[508.07109734,  45.47134351,  72.86766512, ..., 522.73875081,
        914.30133794,  41.71929155]])
```

In [32]:

```
def predict_rating(ratings_arr, item_sim_arr):
    val = np.array(  [np.abs(item_sim_arr).sum(axis=1)]  )  # 유사도의 열의 합
    pred = ratings_arr.dot(item_sim_arr) / val
    return pred
```

In [33]:

```
ratings_pred = predict_rating(ratings_m.values, item_sim_df.values)
print( ratings_pred.shape )
print( type(ratings_pred))
ratings_pred[0:10]
```

```
(610, 9719)
<class 'numpy.ndarray'>
```

Out[33]:

```
array([[0.07034471, 0.5778545 , 0.32169559, ..., 0.13602448, 0.29295452,
        0.72034722],
       [0.01826008, 0.04274424, 0.01886104, ..., 0.02452792, 0.01756305,
        0.        ],
       [0.01188449, 0.03027871, 0.06443729, ..., 0.00922874, 0.01041982,
        0.08450144],
       ...,
       [0.0095766 , 0.0843035 , 0.0476134 , ..., 0.02417663, 0.03387813,
        0.07509685],
       [0.01634194, 0.0818049 , 0.04304403, ..., 0.02187106, 0.0271145 ,
        0.02983867],
       [0.04418904, 0.1559537 , 0.07550071, ..., 0.08178662, 0.05505341,
        0.01902574]])
```

예측 평점

In [34]:

```
ratings_pred_m = pd.DataFrame(data=ratings_pred,
                              index=ratings_m.index,
                              columns = ratings_m.columns)
ratings_pred_m.head()
```

Out[34]:

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(5 Days Summ (20
userId									
1	0.070345	0.577855	0.321696	0.227055	0.206958	0.194615	0.249883	0.102542	0.1570
2	0.018260	0.042744	0.018861	0.000000	0.000000	0.035995	0.013413	0.002314	0.0322
3	0.011884	0.030279	0.064437	0.003762	0.003749	0.002722	0.014625	0.002085	0.0056
4	0.049145	0.277628	0.160448	0.206892	0.309632	0.042337	0.130048	0.116442	0.0997
5	0.007278	0.066951	0.041879	0.013880	0.024842	0.018240	0.026405	0.018673	0.0215

5 rows × 9719 columns

- 기존에 관람되지 않아 NaN인(0으로 결측치 처리)를 했던 것에도 영화 평점이 부여되는 경우가 발생.

평가 - 예측 결과가 실제 평점과 얼마나 차이가 있는지 확인해 보기

- MSE결과 확인

In [35]:

```
from sklearn.metrics import mean_squared_error
```

In [36]:

```
type(ratings_pred), type(ratings_m)
```

Out[36]:

```
(numpy.ndarray, pandas.core.frame.DataFrame)
```

In [37]:

```
ratings_m_val = ratings_m.values
```

In [38]:

```
## 원래 평점 : ratings_m
## 예측 결과 : ratings_pred

# pred : 예측, actual : 실제
pred = ratings_pred[ ratings_m_val.nonzero() ].flatten()
print(pred.shape)
actual = ratings_m_val[ratings_m_val.nonzero()].flatten()
print(actual.shape)
```

```
(100832,)
(100832,)
```

In [39]:

```
mean_squared_error(pred, actual)
```

Out[39]:

```
9.895354759094706
```

In [40]:

```
# 사용자가 평점을 부여한 영화에 대해서만 예측 성능 평가 MSE 를 구함.
def get_mse(pred, actual):
    # Ignore nonzero terms.
    pred = pred[actual.nonzero()].flatten()
    actual = actual[actual.nonzero()].flatten()
    return mean_squared_error(pred, actual)
```

TOP-N 유사도를 가지는 영화 유사도 벡터만 예측

In [41]:

```
def predict_rating_top(ratings_arr, item_sim_arr, n=20):
    # 사용자-아이템 평점 행렬 크기만큼 0으로 채운 예측 행렬 초기화
    pred = np.zeros(ratings_arr.shape)

    # 사용자-아이템 평점 행렬의 열 크기만큼 Loop 수행.
    for col in range(ratings_arr.shape[1]):

        # 유사도 행렬에서 유사도가 큰 순으로 n개 데이터 행렬의 index 반환
        top_n_items = [np.argsort(item_sim_arr[:, col])[:-n-1:-1]]

        # 개인화된 예측 평점을 계산
        for row in range(ratings_arr.shape[0]):
            pred[row, col] = item_sim_arr[col, :][top_n_items].dot(ratings_arr[row, :][top_n_items].
            pred[row, col] /= np.sum(np.abs(item_sim_arr[col, :][top_n_items]))

    return pred
```

In [42]:

```
ratings_pred = predict_rating_top(ratings_m.values , item_sim_df.values, n=20)
print('아이템 기반 인접 TOP-20 이웃 MSE: ', get_mse(ratings_pred,
                                               ratings_m.values ))


# 계산된 예측 평점 데이터는 DataFrame으로 재생성
ratings_pred_m = pd.DataFrame(data=ratings_pred,
                              index= ratings_m.index,
                              columns = ratings_m.columns)
```

```
C:₩Users₩TOTOFR~1₩AppData₩Local₩Temp/ipykernel_13232/3067033362.py:13: FutureWarnin
g: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr
[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an ar
ray index, `arr[np.array(seq)]`, which will result either in an error or a different
result.
  pred[row, col] = item_sim_arr[col, :][top_n_items].dot(ratings_arr[row, :][top_n_i
tems].T)
C:₩Users₩TOTOFR~1₩AppData₩Local₩Temp/ipykernel_13232/3067033362.py:14: FutureWarnin
g: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr
[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an ar
ray index, `arr[np.array(seq)]`, which will result either in an error or a different
result.
  pred[row, col] /= np.sum(np.abs(item_sim_arr[col, :][top_n_items]))
```

```
아이템 기반 인접 TOP-20 이웃 MSE:  3.694957479362603
```

In [43]:

```
user_rating_id = ratings_m.loc[9, :]
user_rating_id[ user_rating_id > 0].sort_values(ascending=False)[:10]
```

Out[43]:

```
title
Adaptation (2002)                                                         5.
0
Citizen Kane (1941)                                                       5.
0
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)    5.
0
Producers, The (1968)                                                     5.
0
Lord of the Rings: The Two Towers, The (2002)                             5.
0
Lord of the Rings: The Fellowship of the Ring, The (2001)                 5.
0
Back to the Future (1985)                                                 5.
0
Austin Powers in Goldmember (2002)                                        5.
0
Minority Report (2002)                                                    4.
0
Witness (1985)                                                            4.
0
Name: 9, dtype: float64
```

평점을 준 영화를 제외하고 추천할 수 있도록 평점을 주지 않은 영화를 리스트 객체로 반환

In [44]:

```
def get_unseen_movies(ratings_matrix, userId):
    # userId로 입력받은 사용자의 모든 영화정보 추출하여 Series로 반환함.
    # 반환된 user_rating 은 영화명(title)을 index로 가지는 Series 객체임.
    user_rating = ratings_matrix.loc[userId,:]

    # user_rating이 0보다 크면 기존에 관람한 영화임. 대상 index를 추출하여 list 객체로 만듬
    already_seen = user_rating[ user_rating > 0].index.tolist()

    # 모든 영화명을 list 객체로 만듬.
    movies_list = ratings_matrix.columns.tolist()

    # list comprehension으로 already_seen에 해당하는 movie는 movies_list에서 제외함.
    unseen_list = [ movie for movie in movies_list if movie not in already_seen]

    return unseen_list
```

최종적으로 사용자 추천

In [45]:

```
def recomm_movie_by_userid(pred_df, userId, unseen_list, top_n=10):
    # 예측 평점 DataFrame에서 사용자id index와 unseen_list로 들어온 영화명 컬럼을 추출하여
    # 가장 예측 평점이 높은 순으로 정렬함.
    recomm_movies = pred_df.loc[userId, unseen_list].sort_values(ascending=False)[:top_n]
    return recomm_movies

# 사용자가 관람하지 않는 영화명 추출
unseen_list = get_unseen_movies(ratings_m, 9)

# 아이템 기반의 인접 이웃 협업 필터링으로 영화 추천
recomm_movies = recomm_movie_by_userid(ratings_pred_m, 9, unseen_list, top_n=10)

# 평점 데이타를 DataFrame으로 생성.
recomm_movies = pd.DataFrame(data=recomm_movies.values,index=recomm_movies.index,columns=['pred_sco
recomm_movies
```

Out[45]:

title	pred_score
Shrek (2001)	0.866202
Spider-Man (2002)	0.857854
Last Samurai, The (2003)	0.817473
Indiana Jones and the Temple of Doom (1984)	0.816626
Matrix Reloaded, The (2003)	0.800990
Harry Potter and the Sorcerer's Stone (a.k.a. Harry Potter and the Philosopher's Stone) (2001)	0.765159
Gladiator (2000)	0.740956
Matrix, The (1999)	0.732693
Pirates of the Caribbean: The Curse of the Black Pearl (2003)	0.689591
Lord of the Rings: The Return of the King, The (2003)	0.676711

history

- 2022-08 ver 0.1