

데이터 과학

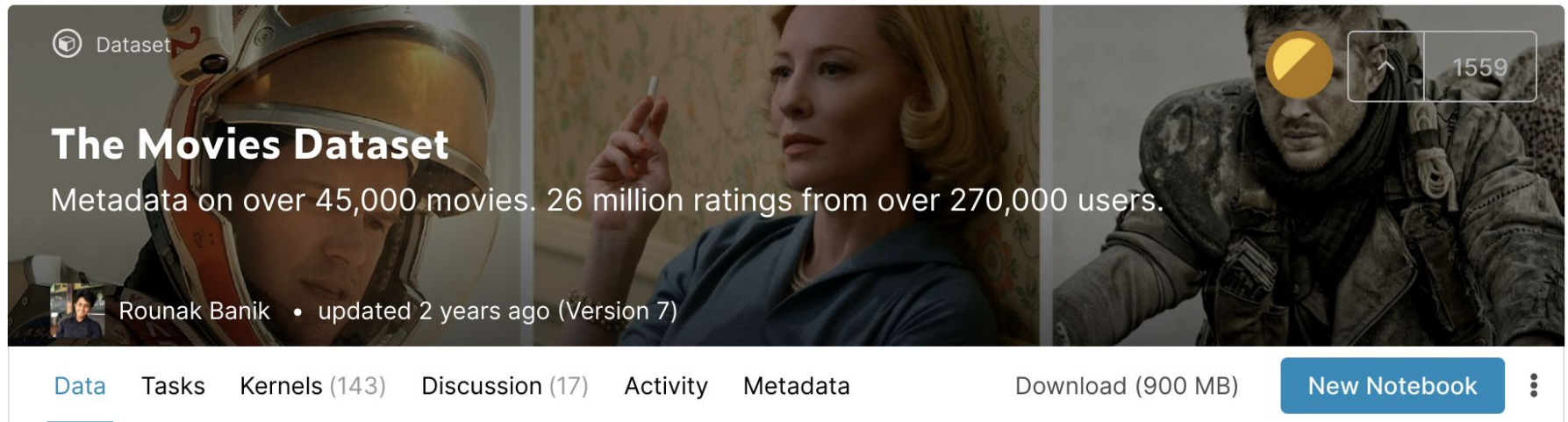
L08.1: Similarity Practice


Kookmin University

The Movies Dataset

<https://www.kaggle.com/rounakbanik/the-movies-dataset>


- 영화 평점 데이터셋
- kaggle에서 notebook 환경 제공




 Dataset

The Movies Dataset

Metadata on over 45,000 movies. 26 million ratings from over 270,000 users.

 Rounak Banik • updated 2 years ago (Version 7)

[Data](#) [Tasks](#) [Kernels \(143\)](#) [Discussion \(17\)](#) [Activity](#) [Metadata](#)

[Download \(900 MB\)](#) [New Notebook](#) 

The Movies Dataset

- `ratings.csv`: 평점 데이터
- `ratings_small.csv`: 평점 데이터 (작은 버전)
- `keywords.csv`: 영화 키워드 데이터
- `movies_metadata.csv`: 영화 정보 데이터
- `credits.csv`: 영화 제작 정보
- `links.csv`: imdb와 tmdb에서의 영화 id 정보
- `links_small.csv`: imdb와 tmdb에서의 영화 id 정보 (작은버전)

비슷한 드라마 찾기

- **목표:** The Movies Data 를 이용하여 비슷한 드라마 찾기
- Kaggle에서 제공하는 notebook을 이용하여 실습
- 방법 및 순서
 - Pandas를 이용하여 데이터 불러오기
 - Pandas를 이용하여 데이터 정제하기
 - Python의 집합을 활용하여 Jaccard 유사도 구하기
 - Numpy를 이용하여 Pearson 유사도 구하기
 - 모든 영화에 대해서 추천점수 계산하고, 가장 추천점수가 높은 영화 고르기

Pandas

- Python용 데이터분석 라이브러리
- MS Office의 Excel과 같이, **행과 열로 구성된 데이터 객체**를 다룸

100% \$ % .0 .00 123 Arial				
fx	5			
	A	B	C	D
1				
2	#rounds			
3		PACC-opt4	PACC-opt3	PACC-opt2
4	livejournal	5	9	5
5	patent	6	10	6
6	friendster	6	11	6
7	skitter	5	9	5
8	subdomain	6	10	6
9	twitter	5	8	5
10	yahooweb	7	16	16
11	clueweb09	8	13	13



Pandas 사용하기

- Pandas, Numpy 패키지 import

```
import numpy as np  
import pandas as pd
```

Pandas로 Metadata 읽고 정제하기

- metadata 읽기

로 구분되어있는 데이터

```
meta = pd.read_csv( '/kaggle/input/the-movies-dataset/movies_metadata.csv' )
meta
```

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...}	300000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}]	http://toystory.disney.com/toy-story	862	tt0114709	en	Toy Story
1	False	NaN	650000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, 'name': 'Fantasy'}]	NaN	8844	tt0113497	en	Jumanji
2	False	{'id': 119050, 'name': 'Grumpy Old Men Collect...	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, 'name': 'Comedy'}]	NaN	15602	tt0113228	en	Grumpier Old Men
3	False	NaN	160000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'name': 'Drama'}]	NaN	31357	tt0114885	en	Waiting to Exhale

Pandas로 Metadata 읽고 정제하기

- metadata에서 필요한 열(column)만 추려내기

```
meta = meta[ ['id', 'original_title', 'original_language', 'genres'] ]
meta
```

	id	original_title	original_language	genres
0	862	Toy Story	en	[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}]
1	8844	Jumanji	en	[{'id': 12, 'name': 'Adventure'}, {'id': 14, 'name': 'Fantasy'}]
2	15602	Grumpier Old Men	en	[{'id': 10749, 'name': 'Romance'}, {'id': 35, 'name': 'Comedy'}]
3	31357	Waiting to Exhale	en	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'name': 'Drama'}]
4	11862	Father of the Bride Part II	en	[{'id': 35, 'name': 'Comedy'}]
...
45461	439050	رگ خواب	fa	[{'id': 18, 'name': 'Drama'}, {'id': 10751, 'name': 'Romance'}]
45462	111109	Siglo ng Pagluluwal	tl	[{'id': 18, 'name': 'Drama'}]

Pandas로 Metadata 읽고 정제하기

- 열(column) 이름 변경하기

```
meta = meta.rename(columns={'id': 'movieId',  
                           'original_title': 'title',  
                           'original_language': 'language'})
```

meta

	movieId	title	language	genres
0	862	Toy Story	en	[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}]
1	8844	Jumanji	en	[{'id': 12, 'name': 'Adventure'}, {'id': 14, 'name': 'Fantasy'}]
2	15602	Grumpier Old Men	en	[{'id': 10749, 'name': 'Romance'}, {'id': 35, 'name': 'Comedy'}]
3	31357	Waiting to Exhale	en	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'name': 'Drama'}]
4	11862	Father of the Bride Part II	en	[{'id': 35, 'name': 'Comedy'}]
...
45461	439050	رگ خواب	fa	[{'id': 18, 'name': 'Drama'}, {'id': 10751, 'name': 'Fantasy'}]
45462	111109	Siglo ng Pagluluwal	tl	[{'id': 18, 'name': 'Drama'}]

Pandas로 Metadata 읽고 정제하기

- language가 'en'인 데이터만 속아내기

```
meta = meta.loc[meta['language'] == 'en', :]  
meta
```

	movieId	title	language	genres
0	862	Toy Story	en	[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}]
1	8844	Jumanji	en	[{'id': 12, 'name': 'Adventure'}, {'id': 14, 'name': 'Fantasy'}]
2	15602	Grumpier Old Men	en	[{'id': 10749, 'name': 'Romance'}, {'id': 35, 'name': 'Comedy'}]
3	31357	Waiting to Exhale	en	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'name': 'Drama'}]
4	11862	Father of the Bride Part II	en	[{'id': 35, 'name': 'Comedy'}]
...
45459	222848	Caged Heat 3000	en	[{'id': 878, 'name': 'Science Fiction'}]
45460	30840	Robin Hood	en	[{'id': 18, 'name': 'Drama'}, {'id': 28, 'name': 'Action'}]

Pandas로 Metadata 읽고 정제하기

- movieId의 타입을 숫자로 변경하기

```
meta.movieId = pd.to_numeric(meta.movieId)
meta.movieId
```

```
0      862
1     8844
2    15602
3    31357
4    11862
...
45459  222848
45460   30840
45463   67758
45464  227506
45465  461257
Name: movieId, Length: 32269, dtype: int64
```

Pandas로 Metadata 읽고 정제하기

- genre 정제하기: json string을 python set으로 변환하는 함수 구현

```
def str_to_set(x):  
    genre_set = set()  
    for item in eval(x):  
        genre_set.add(item['name'])  
    return genre_set
```

Pandas로 Metadata 읽고 정제하기

- genre 정제하기: meta.genres의 모든 값에 str_to_set 적용

```
meta.genres = meta.genres.apply(str_to_set)
meta
```

	movieId	title	language	genres
0	862	Toy Story	en	{Comedy, Animation, Family}
1	8844	Jumanji	en	{Family, Fantasy, Adventure}
2	15602	Grumpier Old Men	en	{Comedy, Romance}
3	31357	Waiting to Exhale	en	{Comedy, Romance, Drama}
4	11862	Father of the Bride Part II	en	{Comedy}
...
45459	222848	Caged Heat 3000	en	{Science Fiction}
45460	30840	Robin Hood	en	{Romance, Action, Drama}

Pandas로 Keywords 읽고 정제하기

- keywords 데이터 읽기

```
keywords = pd.read_csv( '/kaggle/input/the-movies-dataset/keywords.csv' )  
keywords
```

	id	keywords
0	862	[{'id': 931, 'name': 'jealousy'}, {'id': 4290, ...
1	8844	[{'id': 10090, 'name': 'board game'}, {'id': 1...
2	15602	[{'id': 1495, 'name': 'fishing'}, {'id': 12392...
3	31357	[{'id': 818, 'name': 'based on novel'}, {'id': ...
4	11862	[{'id': 1009, 'name': 'baby'}, {'id': 1599, 'n...
...

Pandas로 Keywords 읽고 정제하기

- keywords 데이터의 keywords를 집합으로 변환하기

```
keywords.keywords = keywords.keywords.apply(str_to_set)
keywords
```

	id	keywords
0	862	{friendship, rivalry, boy, boy next door, toy,...
1	8844	{new home, board game, recluse, disappearance,...
2	15602	{duringcreditsstinger, best friend, old men, f...
3	31357	{chick flick, single mother, interracial relat...
4	11862	{pregnancy, mother daughter relationship, cont...
...

Pandas로 Keywords 읽고 정제하기

- id 열을 movieId로 이름변경하고 데이터타입을 숫자로 변환하기

```
keywords = keywords.rename(columns={'id': 'movieId'})
keywords.movieId = pd.to_numeric(keywords.movieId)
keywords
```

	movieId	keywords
0	862	{friendship, rivalry, boy, boy next door, toy,...
1	8844	{new home, board game, recluse, disappearance,...
2	15602	{duringcreditsstinger, best friend, old men, f...
3	31357	{chick flick, single mother, interracial relat...
4	11862	{pregnancy, mother daughter relationship, cont...
...

Keywords와 Metadata 합치기

- 같은 movieId를 갖는 데이터를 합치기

같은 movieId를 갖는 데이터

```
meta = pd.merge(meta, keywords, on='movieId', how='inner')
```

meta

outer = 어떻게 연결된 데이터 (NULL)

	movieId	title	language	genres	keywords
0	862	Toy Story	en	{Comedy, Animation, Family}	{friendship, rivalry, boy, boy next door, toy,...
1	8844	Jumanji	en	{Family, Fantasy, Adventure}	{new home, board game, recluse, disappearance,...
2	15602	Grumpier Old Men	en	{Comedy, Romance}	{duringcreditsstinger, best friend, old men, f...
3	31357	Waiting to Exhale	en	{Comedy, Romance, Drama}	{chick flick, single mother, interracial relat...
4	11862	Father of the Bride Part II	en	{Comedy}	{pregnancy, mother daughter relationship, cont...
...

meta = 쿼리 결과만 남긴다!

Jaccard Similarity

- 영화 찾아보고 합쳐서 출력해보기

```
dk = meta.loc[meta.title == 'The Dark Knight'].iloc[0]
dkr = meta.loc[meta.title == 'The Dark Knight Rises'].iloc[0]
pd.concat([dk, dkr], axis=1).T
```

moviedb		title	language	genres	keywords
10278	155	The Dark Knight	en	{Crime, Action, Thriller, Drama}	{chaos, organized crime, based on comic, vigil...
14315	49026	The Dark Knight Rises	en	{Crime, Action, Thriller, Drama}	{terrorism, catwoman, batman, terrorist, gotha...

Jaccard Similarity

- Jaccard 유사도 함수 구현 및 실행

```
def jaccard_similarity(s1, s2):  
    if len(s1|s2) == 0:  
        return 0  
    return len(s1&s2)/len(s1|s2)
```

```
jaccard_similarity(dk.genres|dk.keywords,  
dkr.genres|dkr.keywords)
```

0.37142857142857144

Rating 데이터 읽고 정제하기

- Rating 데이터 읽기

```
ratings = pd.read_csv( '/kaggle/input/the-movies-dataset/ratings_small.csv' )  
ratings
```

	userId	movieId	rating	timestamp
0	1	31	2.5	1260759144
1	1	1029	3.0	1260759179
2	1	1061	3.0	1260759182
3	1	1129	2.0	1260759185
4	1	1172	4.0	1260759205
...

Rating 데이터 읽고 정제하기

- Rating의 movieId를 숫자로 변환

```
ratings.movieId = pd.to_numeric(ratings.movieId)
```

	userId	movieId	rating	timestamp
0	1	31	2.5	1260759144
1	1	1029	3.0	1260759179
2	1	1061	3.0	1260759182
3	1	1129	2.0	1260759185
4	1	1172	4.0	1260759205
...

Rating 데이터 읽고 정제하기

- metadata로부터 title 정보 가져와 합치기

```
ratings = pd.merge(ratings, meta[['movieId', 'title']], on='movieId', how='inner')
```

	userId	movieId	rating	timestamp	title
0	1	862.0	2.5	1260759144	Toy Story
1	1	8844.0	3.0	1260759179	Jumanji
2	1	15602.0	3.0	1260759182	Grumpier Old Men
3	1	31357.0	2.0	1260759185	Waiting to Exhale
4	1	11862.0	4.0	1260759205	Father of the Bride Part II
...

Rating 데이터 읽고 정제하기

- pivot table 만들기

```
matrix = ratings.pivot_table(index= 'userId', columns='title',
values='rating')
matrix
```

[illegible]

Pearson Correlation Coefficient

- Pearson 유사도 함수 구현

```
def pearson_similarity(u1, u2):  
    u1_c = u1 - u1.mean()  
    u2_c = u2 - u2.mean()  
    denom = np.sqrt(np.sum(u1_c ** 2) * np.sum(u2_c ** 2))  
    if denom != 0:  
        return np.sum(u1_c * u2_c) / denom  
    else:  
        return 0
```


Pearson Correlation Coefficient

- The Dark Knight 와 Prom Night의 Pearson 유사도 구해보기

```
dk_rating = matrix['The Dark Knight']  
pk_rating = matrix['Prom Night']  
pearson_similarity(dk_rating, pk_rating)
```

```
0.773565934694095
```

비슷한 영화 추천 기능 구현

```
def find_similar_movies(input_title, matrix, n, alpha):
    input_meta = meta.loc[ meta[ 'title' ] == input_title ].iloc[ 0 ]
    input_set = input_meta.genres | input_meta.keywords

    result = []

    for this_title in matrix.columns:
        if this_title == input_title:
            continue

        this_meta = meta.loc[ meta[ 'title' ] == this_title ].iloc[ 0 ]
        this_set = this_meta.genres | this_meta.keywords

        pearson = pearson_similarity(matrix[this_title], matrix[input_title])
        jaccard = jaccard_similarity(this_set, input_set)

        score = alpha * pearson + ( 1-alpha ) * jaccard
        result.append( (this_title, pearson, jaccard, score) )

    result.sort(key=lambda r: r[3], reverse=True)

    return result[:n]
```

비슷한 영화 추천 기능 구현

- input_meta: 입력된 영화의 metadata
- input_set: 입력된 영화의 genres와 keyword의 합집합

```
def find_similar_movies(input_title, matrix, n, alpha):  
    input_meta = meta.loc[ meta[ 'title' ] == input_title ].iloc[ 0 ]  
    input_set = input_meta.genres | input_meta.keywords
```

비슷한 영화 추천 기능 구현

- result: 모든 영화마다 유사도를 계산하여 저장
- 입력된 영화는 유사도 계산을 하지 않고 Pass~

```
result = []  
  
for this_title in matrix.columns:  
    if this_title == input_title:  
        continue
```

비슷한 영화 추천 기능 구현

- this_meta: 유사도를 계산하려는 영화의 metadata
- this_set: 이 영화의 genres와 keywords의 합집합

```
this_meta = meta.loc[ meta['title'] == this_title].iloc[0]  
this_set = this_meta.genres | this_meta.keywords
```

비슷한 영화 추천 기능 구현

- pearson: 입력 영화와 이번 영화의 pearson 유사도 결과
- jaccard: 입력 영화와 이번 영화의 jaccard 유사도 결과

```
pearson = pearson_similarity(matrix[this_title], matrix[input_title])  
jaccard = jaccard_similarity(this_set, input_set)
```

비슷한 영화 추천 기능 구현

- score: pearson 점수와 jaccard 점수의 가중치 합
- result에 계산 결과 추가

```
score = alpha * pearson + (1-alpha) * jaccard  
result.append( (this_title, pearson, jaccard, score) )
```

비슷한 영화 추천 기능 구현

- 모든 영화에 대해 유사도 계산이 끝나면 result를 정렬
- 상위 n개를 return

```
result.sort(key=lambda r: r[3], reverse=True)
```

```
return result[:n]
```


비슷한 영화 추천

- The Dark Knight와 비슷한 영화 추천해보기

```
result = find_similar_movies('The Dark Knight', matrix, 10, 0.3)
pd.DataFrame(result, columns = ['title', 'pearson', 'jaccard', 'score'])
```

	title	pearson	jaccard	score
0	Wild Wild West	0.773566	0.032258	0.254650
1	Prom Night	0.773566	0.022222	0.247625
2	Batman Begins	0.005015	0.292683	0.206383
3	Yamakasi - Les samouraïs des temps modernes	0.377145	0.125000	0.200643
4	Blue Thunder	0.326617	0.133333	0.191318
5	Midnight in the Garden of Good and Evil	0.373841	0.111111	0.189930
6	Topaz	0.377145	0.103448	0.185557
7	Big Bad Mama	0.344649	0.107143	0.178395
8	Sneakers	0.415830	0.068966	0.173025
9	The Enforcer	0.326617	0.103448	0.170399

Questions?