분류 - 영화추천

#01. 패키지 참조

```
import warnings
warnings.filterwarnings('ignore')

import os
import numpy as np
import seaborn as sb
from matplotlib import pyplot as plt
from pandas import read_csv, pivot_table
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
```

#02. 데이터 가져오기

jupyter가 참조하고 있는 현재 디렉토리 확인

```
print(os.getcwd())
```

```
c:\Users\leekh\J&Y Dropbox\메가IT수업자료\G. 머신러닝\02.Sklearn
```

영화 데이터 가져오기

실 분석용은 아니다. 분석 후 결과값을 맵핑시키기 위한 데이터이다.

```
origin_mv = read_csv("netflix/Netflix_Dataset_Movie.csv", encoding='utf-8')
origin_mv.head()
```

	Movie_ID	Year	Name
0	1	2003	Dinosaur Planet
1	2	2004	Isle of Man TT 2004 Review
2	3	1997	Character
3	4	1994	Paula Abdul's Get Up & Dance
4	5	2004	The Rise and Fall of ECW

별점 데이터 가져오기

```
origin_rating = read_csv("netflix/Netflix_Dataset_Rating.csv", encoding='utf-8')
print(origin_rating.info())
```

```
origin_rating.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17337458 entries, 0 to 17337457
Data columns (total 3 columns):
 #   Column    Dtype
---  ------    -----
 0   User_ID   int64
 1   Rating    int64
 2   Movie_ID  int64
dtypes: int64(3)
memory usage: 396.8 MB
None
```

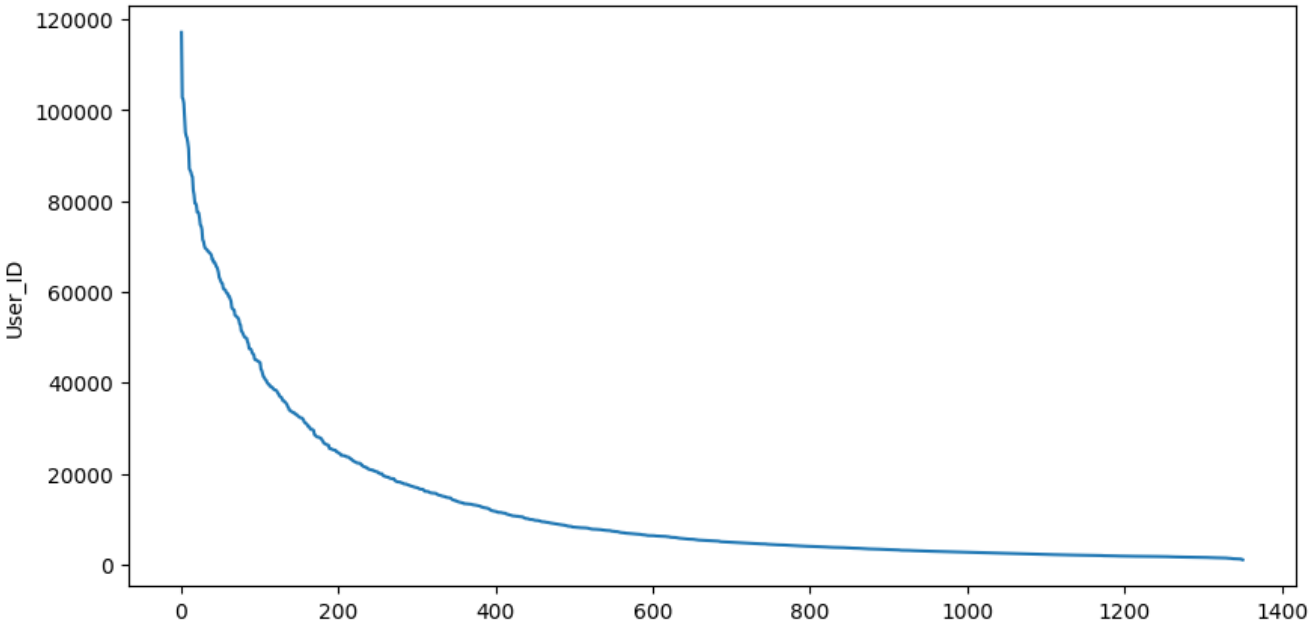
	User_ID	Rating	Movie_ID
0	712664	5	3
1	1331154	4	3
2	2632461	3	3
3	44937	5	3
4	656399	4	3

```
gdf = origin_rating.filter([
    'Movie_ID', 'User_ID'
]).groupby('Movie_ID').count().sort_values(
    'User_ID', ascending=False)
gdf
```

	User_ID
Movie_ID	
1905	117075
2452	102721
4306	102376
571	101450
3860	98545
...	...
4161	1215
1375	1213
717	1212
2870	1092
4238	1042

1350 rows × 1 columns

```
plt.figure(figsize=(10,5))
sb.lineplot(data=gdf, y='User_ID', x=range(1, len(gdf.index)+1))
plt.show()
plt.close()
```



#03. 데이터 전처리

별점 테이블 재배치(피벗테이블)

```
fav_movie_df = gdf.where(gdf['User_ID'] > 80000).dropna()
fav_movie_df
```

	User_ID
Movie_ID	
1905	117075.0
2452	102721.0
4306	102376.0
571	101450.0
3860	98545.0
2862	95053.0
3962	94235.0
4432	93886.0
3938	92893.0
2782	91502.0
3106	87082.0
2152	86756.0

	User_ID
Movie_ID	
3624	86354.0
1962	85605.0
2372	85247.0
1180	82347.0
3427	81371.0

```
rating_df = origin_rating[origin_rating['Movie_ID'].isin(fav_movie_df.index)]
rating_df
```

	User_ID	Rating	Movie_ID
2184894	1734805	3	571
2184895	364518	4	571
2184896	1392773	5	571
2184897	716091	5	571
2184898	1527030	5	571
...	...	...	...
17163696	480064	3	4432
17163697	666284	5	4432
17163698	1462398	5	4432
17163699	158902	5	4432
17163700	1595055	5	4432

1584498 rows × 3 columns

```
pivot_rating = pivot_table(rating_df, index='User_ID', columns='Movie_ID', values='Rat
pivot_rating
```

Movie_ID	571	1180	1905	1962	2152	2372	2452	2782	2862	3106	3427	3624
User_ID												
6	NaN	3.0	4.0	4.0	3.0	5.0	5.0	5.0	4.0	NaN	4.0	4.0
7	4.0	5.0	5.0	5.0	3.0	4.0	5.0	5.0	5.0	4.0	3.0	5.0
79	NaN	4.0	4.0	5.0	3.0	4.0	4.0	4.0	4.0	NaN	4.0	1.0
97	3.0	5.0	4.0	NaN	3.0	NaN	5.0	NaN	NaN	5.0	2.0	NaN
134	4.0	NaN	5.0	4.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...
2649370	5.0	4.0	4.0	4.0	NaN	3.0	5.0	5.0	NaN	NaN	NaN	4.0

Movie_ID	571	1180	1905	1962	2152	2372	2452	2782	2862	3106	3427	3624
User_ID												
2649378	5.0	5.0	4.0	NaN	4.0	4.0	3.0	4.0	5.0	4.0	NaN	3.0
2649388	3.0	4.0	2.0	4.0	4.0	NaN	NaN	NaN	3.0	4.0	NaN	NaN
2649426	NaN	NaN	5.0	4.0	4.0	4.0	NaN	5.0	NaN	4.0	NaN	5.0
2649429	4.0	NaN	NaN	5.0	5.0	NaN	5.0	5.0	5.0	5.0	4.0	5.0

143163 rows × 17 columns

결측치 처리

```
pivot_rating.fillna(0, inplace=True)
pivot_rating
```

Movie_ID	571	1180	1905	1962	2152	2372	2452	2782	2862	3106	3427	3624
User_ID												
6	0.0	3.0	4.0	4.0	3.0	5.0	5.0	5.0	4.0	0.0	4.0	4.0
7	4.0	5.0	5.0	5.0	3.0	4.0	5.0	5.0	5.0	4.0	3.0	5.0
79	0.0	4.0	4.0	5.0	3.0	4.0	4.0	4.0	4.0	0.0	4.0	1.0
97	3.0	5.0	4.0	0.0	3.0	0.0	5.0	0.0	0.0	5.0	2.0	0.0
134	4.0	0.0	5.0	4.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...
2649370	5.0	4.0	4.0	4.0	0.0	3.0	5.0	5.0	0.0	0.0	0.0	4.0
2649378	5.0	5.0	4.0	0.0	4.0	4.0	3.0	4.0	5.0	4.0	0.0	3.0
2649388	3.0	4.0	2.0	4.0	4.0	0.0	0.0	0.0	3.0	4.0	0.0	0.0
2649426	0.0	0.0	5.0	4.0	4.0	4.0	0.0	5.0	0.0	4.0	0.0	5.0
2649429	4.0	0.0	0.0	5.0	5.0	0.0	5.0	5.0	5.0	5.0	4.0	5.0

143163 rows × 17 columns

#04. 분류 모델 구축

단일 수행

```
k = 5
knn_classifier = KNeighborsClassifier(n_neighbors=k)
knn_classifier.fit(pivot_rating, pivot_rating.index)
y_pred = knn_classifier.predict(pivot_rating)
y_pred
```

```
array([     6,      7,     79, ..., 268484, 346689, 207790], dtype=int64)
```

```
score = accuracy_score(pivot_rating.index, y_pred)
score
```

```
0.1802351166153266
```

이후 진행해야 할 부분

1. 임의의 사용자 데이터 구성
    - 선정된 영화들 중 무작위로 별점을 부여한 사용자 데이터
2. 임의의 사용자 데이터에 대한 분류값 얻기
3. 생성된 분류를 기반으로 근접한 이웃 데이터 얻기
4. 근접 이웃이 별점을 부여한 영화 목록을 조회하여 추천 데이터로 제시

```

```