INITIAL INFORMATION - 참조 함수

```
# !pip install scikit-surprise
```

```
import pandas as pd
import numpy as np

def load_movies_dataset() -> pd.DataFrame:
    """영화에 대한 정보 불러오기"""
    movie_data_columns = [
    'movie_id', 'title', 'release_date', 'video_release_date', 'url',
    'unknown', 'Action', 'Adventure', 'Animation', "Children's",
    'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy', 'Film-Noir',
    'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller',
    'War', 'Western'
    ]

    movie_data = pd.read_csv(
        'datasets/ml-100k/u.item',
        sep = '|',
        encoding = "ISO-8859-1",
        header = None,
        names = movie_data_columns,
        index_col = 'movie_id'
    )
    movie_data['release_date'] = pd.to_datetime(movie_data['release_date'])
    return movie_data

def load_ratings() -> pd.DataFrame:
    ratings_data = pd.read_csv(
        'datasets/ml-100k/u.data',
        sep = '\t',
        names=['user_id', 'movie_id', 'rating', 'timestamp']
    )
    return ratings_data
```



```
movie_data = load_movies_dataset()
ratings_data = load_ratings()
```

```
movie_data.head()
```

```
ratings_data.head(10)
```

	user_id	movie_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596
5	298	474	4	884182806
6	115	265	2	881171488
7	253	465	5	891628467
8	305	451	3	886324817
9	6	86	3	883603013

```
ratings_data['user_id'].max()
```

```
943
```

Ratings dataset

Contains the **interactions** between users and movies

- User **196** rated movie **242** with a score of **3**
- User **186** rated movie **302** with a score of **3**
- User **22** rated movie **377** with a score of **3**

```
ratings_data[ratings_data['movie_id'] == 1]['rating'].describe()
```

```
count    452.000000
mean       3.878319
std        0.927897
min        1.000000
25%        3.000000
50%        4.000000
75%        5.000000
max        5.000000
Name: rating, dtype: float64
```

Double-click (or enter) to edit

> NOW SOLVE!!!!

해답) 문제 풀이

```
from surprise import SVD, NMF, accuracy
from surprise import Dataset, Reader
from surprise.model_selection import cross_validate, train_test_split

# Surprise has some preset datasets, including ml-100k!
# data = Dataset.load_builtin('ml-100k')

reader = Reader(rating_scale=(1, 5))
data = Dataset.load_from_df(ratings_data[['user_id', 'movie_id', 'rating']], reader)

trainset, testset = train_test_split(data, test_size=.25)

# Let's train a new Nonnegative SVD
model = SVD(n_factors=100, biased=False)
model.fit(trainset)

# In reality, we should perform a train/test split and check RMSE to see if our model is trained
# but today, for simplicity, I'm skipping this step
predictions = model.test(testset)
accuracy.rmse(predictions)
```

```
RMSE: 0.9580
0.957965924295794
```

To undo cell deletion use ⌘/Ctrl+M Z or the Undo option in the Edit menu

Surprise SVD stores the product matrix under the `model.qi` attribute.

```
pd.DataFrame(model.qi).head(10)
```

	0	1	2	3	4	5	6	7	8	9	...	90	91	92	93	94
0	-0.028085	0.007326	0.267067	-0.236421	-0.077943	-0.019932	-0.133990	0.285557	0.273829	0.001874	...	0.081802	0.219284	0.120746	-0.097746	0.467054
1	0.106267	0.063955	0.362999	-0.038772	0.106562	0.193185	-0.174445	0.112519	0.027699	0.314792	...	0.093350	0.138825	-0.055726	0.119043	0.368421
2	0.113641	0.480834	0.050770	0.140915	0.299281	0.073561	-0.192111	0.067109	-0.019272	-0.087939	...	-0.072078	0.024927	0.147691	-0.174189	0.169043
3	-0.018501	0.081170	0.315538	-0.381532	0.158467	0.059910	0.114721	0.254688	0.157660	0.057107	...	0.014172	0.148316	-0.035810	0.017940	0.218948
4	-0.039118	0.117172	0.251221	-0.165814	-0.054695	0.022311	-0.138875	-0.018530	0.164057	0.120751	...	0.206625	0.013239	0.039077	-0.148116	0.366774
5	0.447446	0.185904	0.451171	-0.394825	0.145879	0.090853	0.073207	0.055838	0.254270	0.092589	...	0.093432	-0.112373	-0.182172	0.098663	0.085598
6	0.196233	0.145739	0.209179	0.148519	-0.231206	-0.062533	-0.255633	0.002110	0.241216	-0.068118	...	0.134930	0.015341	0.026284	-0.107322	0.265486
7	-0.082921	0.187218	0.253966	0.043173	0.140965	-0.159558	0.208955	0.067746	0.060982	-0.199792	...	0.205104	0.241081	0.068046	-0.044265	0.355204
8	-0.079435	0.260487	0.054842	-0.438233	0.070607	0.118612	0.002515	-0.067611	0.497394	-0.344685	...	-0.086256	-0.230251	-0.221332	-0.062740	0.067165
9	0.079875	0.086168	0.118833	0.002549	-0.237364	0.157296	-0.248504	0.123689	0.259569	0.154115	...	0.465869	0.121347	0.015153	-0.090638	0.393071

10 rows × 100 columns

Exploring the product matrix

The matrix has `n_factors` columns (we chose 10). Every row represents a movie

```
print(f"The shape of our product matrix is {model.qi.shape}.")
print(f"There are {ratings_data['movie_id'].unique().shape[0]} unique movies movies")
```

```
The shape of our product matrix is (1638, 100).
There are 1682 unique movies movies
```

Generating predictions with simplicity

Before looking into the latent features of our movies, let's use the API provided by Surprise. More specifically, Surprise provides us 1 API

- `model.predict` computes the rating prediction for given user and movie

Let's look at how we can use this API to generate movies that a given user may like

```
>>> model.predict('302', '1')
Prediction(uid=302, iid=1, r_ui=None, est=3.5327866666666665, details={'was_impossible': False})
```

NOTE: User ID and Movie ID are **strings**

```
# The prediction for user 196 to like movie#1 (Toy Story)
print(movie_data.loc[1])
print()
```

```
user_score_prediction = model.predict(196, 1)
print(user_score_prediction)
print(f"\n\nUSER 196 gives Toy Story: {user_score_prediction.est}")
```

```
    title                                                    Toy Story (1995)
    release_date                                          1995-01-01 00:00:00
    video_release_date                                                    NaN
    url                   http://us.imdb.com/M/title-exact?Toy%20Story%2...
    unknown                                                                 0
    Action                                                                  0
    Adventure                                                               0
    Animation                                                               1
    Children's                                                              1
    Comedy                                                                  1
    Crime                                                                   0
    Documentary                                                             0
    Drama                                                                   0
    Fantasy                                                                 0
    Film-Noir                                                               0
    Horror                                                                  0
    Musical                                                                 0
    Mystery                                                                 0
    Romance                                                                 0
    Sci-Fi                                                                  0
    Thriller                                                                0
    War                                                                     0
    Western                                                                 0
    Name: 1, dtype: object

    user: 196        item: 1          r_ui = None   est = 3.66   {'was_impossible': False}


    USER 196 gives Toy Story: 3.657722819161514
```

Recommend 출력 함수 만들기


```
def generate_recommended_movies_for_user(model, user_id):
    """Return a DataFrame containing recommendations for the user, and the
    associated score
    """
    results = []
    for movie_id, movie_title in movie_id_to_title_map.items():

        # For each movie, calculate score prediction
        prediction = model.predict(user_id, movie_id)
        results.append((movie_id, prediction.est, movie_title))

    return pd.DataFrame(results, columns=['movie_id', 'Estimated Prediction', 'Movie Title']).set_index('movie_id')


def display_best_and_worse_recommendations(recommendations: pd.DataFrame):
    recommendations.sort_values('Estimated Prediction', ascending=False, inplace=True)

    top_recommendations = recommendations.iloc[:10]
    top_recommendations.columns = ['Prediction (sorted by best)', 'Movie Title']
    # worse_recommendations = recommendations.iloc[-10:]
    # worse_recommendations.columns = ['Prediction (sorted by worse)', 'Movie Title']

    return top_recommendations


# Let's generate some recommendations for a user 302
recommendations = generate_recommended_movies_for_user(model, 302)
display_best_and_worse_recommendations(recommendations)
```

movie_id	Prediction (sorted by best)	Movie Title
1570	3.529347	Quartier Mozart (1992)
1505	3.529347	Killer: A Journal of Murder (1995)
1533	3.529347	I Don't Want to Talk About It (De eso no se ha...
1619	3.529347	All Things Fair (1996)
1520	3.529347	Fear, The (1995)
1515	3.529347	Wings of Courage (1995)
1507	3.529347	Three Lives and Only One Death (1996)
1631	3.529347	Slingshot, The (1993)
1343	3.529347	Lotto Land (1995)
1659	3.529347	Getting Away With Murder (1996)

내가 좋아하는 영화 고르고, 데이터에 추가해서 추천 영화 뽑기

```
# 나는 최근 영화만 알기 때문에 최근 영화만 살펴보기
movie_data.sort_values('release_date', ascending=False).iloc[:100]


movie_data.sort_values('release_date', ascending=False).iloc[:200].to_clipboard(sep='\t')
# 엑셀에서 내가 좋아하는 영화 선택
```

```
#선택한 내가 좋아하는 영화
my_movie_lst = pd.Series([916, 355,350,258,298,252,987,250], name='movie_id')
movie_data.loc[my_movie_lst, ['title', 'release_date']]
```

movie_id	title	release_date
916	Lost in Space (1998)	1998-03-27
355	Sphere (1998)	1998-02-13
350	Fallen (1998)	1998-01-16
258	Contact (1997)	1997-07-11
298	Face/Off (1997)	1997-06-27
252	Lost World: Jurassic Park, The (1997)	1997-05-23
987	Underworld (1997)	1997-05-09
250	Fifth Element, The (1997)	1997-05-09

```
ratings_attach = my_movie_lst.to_frame().assign(rating=5)
ratings_attach.insert(0, 'user_id', 1000)


ratings_data_ = pd.concat([ratings_data, ratings_attach], axis=0).reset_index(drop=True)


reader = Reader(rating_scale=(1, 5))
data = Dataset.load_from_df(ratings_data_[['user_id', 'movie_id', 'rating']], reader)

trainset, testset = train_test_split(data, test_size=.25)

# Let's train a new Nonnegative SVD
model = SVD(n_factors=100, biased=False)
```

To undo cell deletion use ⌘/Ctrl+M Z or the Undo option in the Edit menu

```
predictions = model.test(testset)
accuracy.rmse(predictions)
```

```
RMSE: 0.9524
0.9523943284708634
```

```
# Let's generate some recommendations for a myself - user_id(1000)
recommendations = generate_recommended_movies_for_user(model, 1000)
display_best_and_worse_recommendations(recommendations)
```

movie_id	Prediction (sorted by best)	Movie Title
64	5.0	Shawshank Redemption, The (1994)
12	5.0	Usual Suspects, The (1995)
258	5.0	Contact (1997)
251	5.0	Shall We Dance? (1996)
285	5.0	Secrets & Lies (1996)
515	5.0	Boot, Das (1981)
169	5.0	Wrong Trousers, The (1993)
357	5.0	One Flew Over the Cuckoo's Nest (1975)
408	5.0	Close Shave, A (1995)
272	5.0	Good Will Hunting (1997)