# Recommend Practice

# with Kaggle Dataset

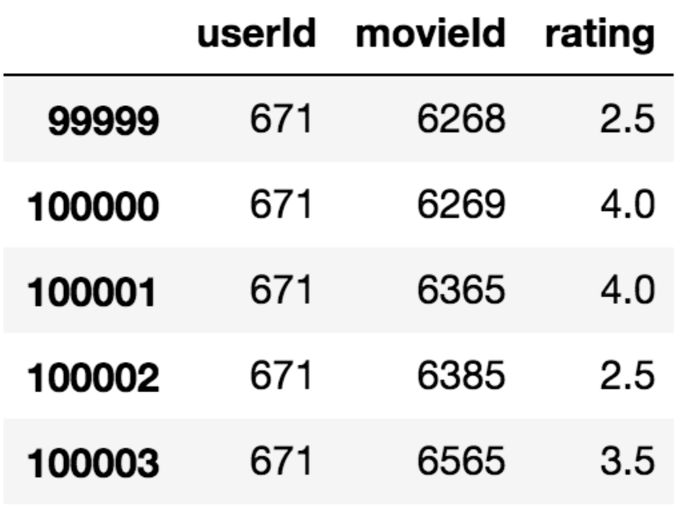
## 1. Dataset

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

------------------------------------------------------------------------------------------------------------

```
import pandas as pd
import numpy as np
from scipy import spatial

rating_df = pd.read_csv("ratings_small.csv")
rating_df.drop('timestamp', axis=1, inplace=True)
rating_df.tail()
```

------------------------------------------------------------------------------------------------------------



| | userId | movieId | rating |
|---|---|---|---|
| **99999** | 671 | 6268 | 2.5 |
| **100000** | 671 | 6269 | 4.0 |
| **100001** | 671 | 6365 | 4.0 |
| **100002** | 671 | 6385 | 2.5 |
| **100003** | 671 | 6565 | 3.5 |

**2. Check Dataset**

```
----------------------------------------------------------------------------------------------------------
# 데이터 셋에서 유니크 데이터 확인
unique_user = rating_df["userId"].unique()
unique_movie = rating_df["movieId"].unique()
unique_rating = rating_df["rating"].unique()
unique_rating = sorted(unique_rating)
print("sorted rating : {}".format(unique_rating))
len(unique_user), len(unique_movie), len(unique_rating)
----------------------------------------------------------------------------------------------------------
sorted rating : [0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0]
(671, 9066, 10)
```

Rating - 데이터 분포 확인

```
----------------------------------------------------------------------------------------------------------
rating_df.groupby("rating").size().reset_index(name='rating_counts')
----------------------------------------------------------------------------------------------------------
```

User - 데이터 분포 확인

```
----------------------------------------------------------------------------------------------------------
user_counts_df = rating_df.groupby("userId").size().reset_index(name='user_rating_count')
user_counts_df = user_counts_df.sort_values(by=['user_rating_count'], ascending=False)
user_counts_df.tail()
----------------------------------------------------------------------------------------------------------
```

Movie - 데이터 분포 확인

```
----------------------------------------------------------------------------------------------------------
movie_counts_df = rating_df.groupby("movieId").size().reset_index(name='movie_rating_count')
movie_counts_df = movie_counts_df.sort_values(by=['movie_rating_count'], ascending=False)
movie_counts_df.tail()
---------------------------------------------------------------------------------------------------------
```

### 3. Preprocessing

reduce dataset

```
# user의 최소 평가수, 영화의 최소 평가수
user_limit, movie_limit = 100, 100
```

```
# user_limit번 이상 평가한 UserId
filtered_userId = list(user_counts_df[user_counts_df["user_rating_count"] > user_limit]["userId"])
print(len(filtered_userId))
```

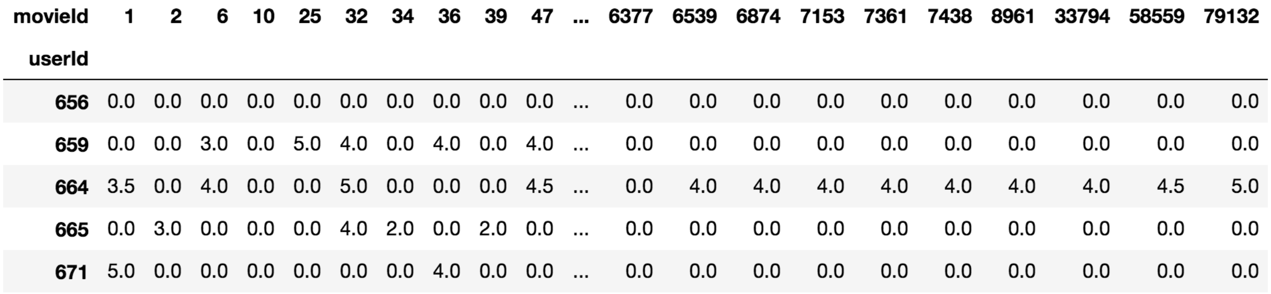
```
# movie_limit개 이상 평가 받은 movieId
filtered_movieId = list(movie_counts_df[movie_counts_df["movie_rating_count"] > movie_limit]["movieId"])
print(len(filtered_movieId))
```

```
# filtering - 100000 -> 15567
filterd_df = rating_df[rating_df['userId'].isin(filtered_userId)]
filterd_df = filterd_df[filterd_df['movieId'].isin(filtered_movieId)]
print(len(filterd_df))
filterd_df.tail()
```

```
15567
```

## 4. Pivot

```
----------------------------------------------------------------------------------------------------------------------------

user_df = filterd_df.pivot_table(values="rating", index=["userId"], columns=["movieId"],\
                        aggfunc=np.average, fill_value=0, dropna=False)

user_df.tail()

----------------------------------------------------------------------------------------------------------------------------
```



| movieId | 1 | 2 | 6 | 10 | 25 | 32 | 34 | 36 | 39 | 47 | ... | 6377 | 6539 | 6874 | 7153 | 7361 | 7438 | 8961 | 33794 | 58559 | 79132 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| userId | | | | | | | | | | | | | | | | | | | | | |
| 656 | 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 | 0.0 | 0.0 | 0.0 | 0.0 |
| 659 | 0.0 | 0.0 | 3.0 | 0.0 | 5.0 | 4.0 | 0.0 | 4.0 | 0.0 | 4.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 664 | 3.5 | 0.0 | 4.0 | 0.0 | 0.0 | 5.0 | 0.0 | 0.0 | 0.0 | 4.5 | ... | 0.0 | 4.0 | 4.0 | 4.0 | 4.0 | 4.0 | 4.0 | 4.0 | 4.5 | 5.0 |
| 665 | 0.0 | 3.0 | 0.0 | 0.0 | 0.0 | 4.0 | 2.0 | 0.0 | 2.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 671 | 5.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 4.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

## 5. Functions

### (1) Euclidean Distance Similarity

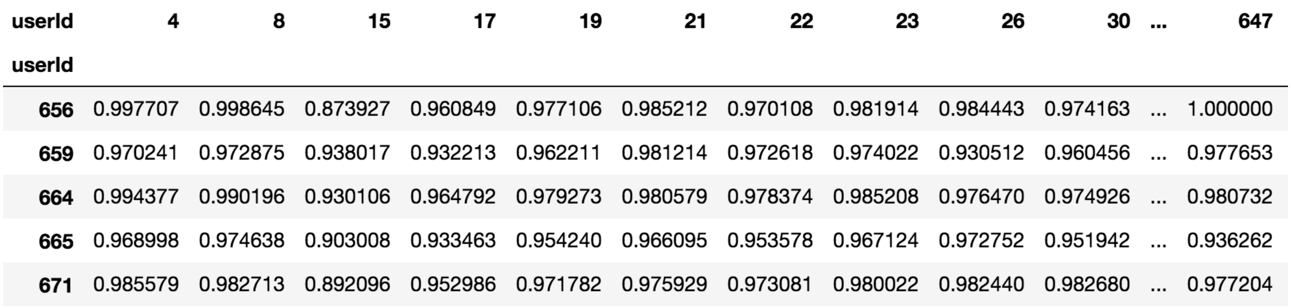
```
----------------------------------------------------------------------------------------------------------------------------

def euclidean_similarity(vector_1, vector_2):


  idx = vector_1.nonzero()[0]

  if len(idx) == 0:

    return 0

  vector_1, vector_2 = np.array(vector_1)[idx], np.array(vector_2)[idx]


  idx = vector_2.nonzero()[0]

  if len(idx) == 0:

    return 0

  vector_1, vector_2 = np.array(vector_1)[idx], np.array(vector_2)[idx]


  return np.linalg.norm(vector_1 - vector_2)

----------------------------------------------------------------------------------------------------------------------------
```

**(2) Cosine Similarity**

```
----------------------------------------------------------------------------------------------------------
# 둘다 본 데이터만 조회
def cosine_smimilarity(vector_1, vector_2):

    # vector 1 zero data filtering
    idx = vector_1.nonzero()[0]
    if len(idx) == 0:
        return 0
    vector_1, vector_2 = np.array(vector_1)[idx], np.array(vector_2)[idx]

    # vector 2 zero data filtering
    idx = vector_2.nonzero()[0]
    if len(idx) == 0:
        return 0
    vector_1, vector_2 = np.array(vector_1)[idx], np.array(vector_2)[idx]

    return 1 - spatial.distance.cosine(vector_1, vector_2)
----------------------------------------------------------------------------------------------------------
```

**(3) Similarity Matrix**

```
----------------------------------------------------------------------------------------------------------
def similarity_matrix(df, similarity_func):

    # index 데이터 저장
    index = df.index

    # 데이터 프레임 전치 (index - article, columns - user)
    df = df.T

    # 모든 user 데이터 사이의 유사도를 구해 행렬 생성
    matrix = []
    for idx_1, value_1 in df.items():
        # row 데이터 저장
        row = []
        for idx_2, value_2 in df.items():

            # 두 user 사이의 유사도 구함
            row.append(similarity_func(value_1, value_2))
        matrix.append(row)

    return pd.DataFrame(matrix, columns=index, index=index)
------------------------------------------------------------------------------------------------------
# test code - Similarity Matrix
------------------------------------------------------------------------------------------------------
# similarity matrix
sm_df = similarity_matrix(user_df, cosine_smimilarity)
sm_df.tail()
------------------------------------------------------------------------------------------------------
```



| userId | 4 | 8 | 15 | 17 | 19 | 21 | 22 | 23 | 26 | 30 | ... | 647 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| userId | | | | | | | | | | | | |
| **656** | 0.997707 | 0.998645 | 0.873927 | 0.960849 | 0.977106 | 0.985212 | 0.970108 | 0.981914 | 0.984443 | 0.974163 | ... | 1.000000 |
| **659** | 0.970241 | 0.972875 | 0.938017 | 0.932213 | 0.962211 | 0.981214 | 0.972618 | 0.974022 | 0.930512 | 0.960456 | ... | 0.977653 |
| **664** | 0.994377 | 0.990196 | 0.930106 | 0.964792 | 0.979273 | 0.980579 | 0.978374 | 0.985208 | 0.976470 | 0.974926 | ... | 0.980732 |
| **665** | 0.968998 | 0.974638 | 0.903008 | 0.933463 | 0.954240 | 0.966095 | 0.953578 | 0.967124 | 0.972752 | 0.951942 | ... | 0.936262 |
| **671** | 0.985579 | 0.982713 | 0.892096 | 0.952986 | 0.971782 | 0.975929 | 0.973081 | 0.980022 | 0.982440 | 0.982680 | ... | 0.977204 |

### (4) Mean Score

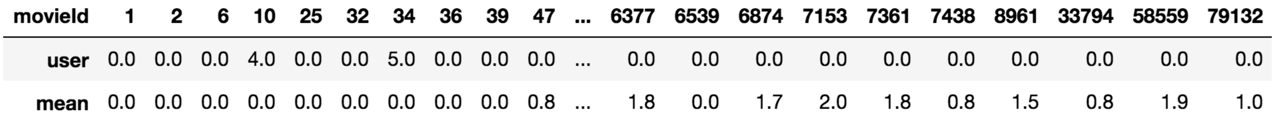
```
------------------------------------------------------------------------------------------------------
# 유사도가 높은 user에 대한 평균값 구하는 함수
def mean_score(df, sm_df, target, closer_count):

    # 유사도 행렬에서 추천 user와 가까운 user의 유사도 데이터 프레임
    sms_df = sm_df.drop(target)
    sms_df = sms_df.sort_values(target, ascending=False)
    sms_df = sms_df[target][:closer_count]
    # 유사도가 높은 user를 나타내는 데이터 프레임
    smsw_df = df.loc[sms_df.index]

    # 결과 데이터 프레임 생성
    ms_df = pd.DataFrame(columns=df.columns)
    ms_df.loc["user"] = df.loc[target]
    ms_df.loc["mean"] = smsw_df.mean()

    return ms_df
```

----------------------------------------------------------------------------------------------------------

```
# test code - Mean Score
```

----------------------------------------------------------------------------------------------------------

```
# mean score df
ms_df = mean_score(user_df, sm_df, 4, 5) # (target:4, closer count:5)
ms_df.tail()
```



| movieId | 1 | 2 | 6 | 10 | 25 | 32 | 34 | 36 | 39 | 47 | ... | 6377 | 6539 | 6874 | 7153 | 7361 | 7438 | 8961 | 33794 | 58559 | 79132 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| user | 0.0 | 0.0 | 0.0 | 4.0 | 0.0 | 0.0 | 5.0 | 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 |
| mean | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.8 | ... | 1.8 | 0.0 | 1.7 | 2.0 | 1.8 | 0.8 | 1.5 | 0.8 | 1.9 | 1.0 |

----------------------------------------------------------------------------------------------------------

**(5) Recommend**

----------------------------------------------------------------------------------------------------------

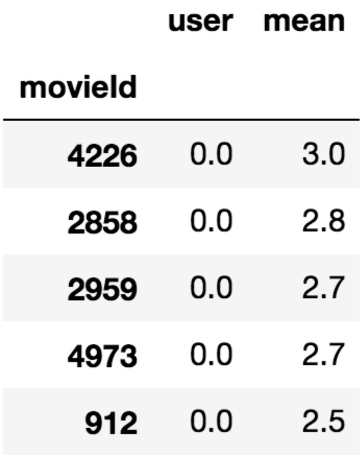
```
def recommend(ms_df):
    recommand_df = ms_df.T
    recommand_df = recommand_df[recommand_df["user"] == 0]
    recommand_df = recommand_df.sort_values("mean", ascending=False)
    return recommand_df, list(recommand_df.index)
```

----------------------------------------------------------------------------------------------------------

# test code - Recommend

```
--------------------------------------------------------------------------------------------------------

# recommend
recommend_df, recommend_list = recommend(ms_df)
print(recommend_list[:10])
recommend_df.head()

--------------------------------------------------------------------------------------------------------
```

```
[4226, 2858, 2959, 4973, 912, 50, 5952, 4306, 3996, 4993]
```



| | user | mean |
|---|---|---|
| **movieId** | | |
| **4226** | 0.0 | 3.0 |
| **2858** | 0.0 | 2.8 |
| **2959** | 0.0 | 2.7 |
| **4973** | 0.0 | 2.7 |
| **912** | 0.0 | 2.5 |

**(6) MAE**

```
--------------------------------------------------------------------------------------------------------

def mae(value, pred):

    # user 데이터에서 0인 데이터 제거
    idx = value.nonzero()[0]
    vector_1, vector_2 = np.array(value)[idx], np.array(pred)[idx]

    # pred 데이터에서 0인 데이터 제거
    idx = pred.nonzero()[0]
    vector_1, vector_2 = np.array(value)[idx], np.array(pred)[idx]

    # 수식 계산후 결과 리턴
    return sum(np.absolute(value - pred)) / len(idx)

--------------------------------------------------------------------------------------------------------
```

# Evaluate Function

----------------------------------------------------------------------------------------------------------------

```
# 전체 user에 대한 mae의 평균
def evaluate(df, sm_df, closer_count, algorithm):

    # user 리스트
    users = df.index
    evaluate_list = []

    # 모든 user에 대해서 mae 값을 구함
    for target in users:
        pred_df = mean_score(df, sm_df, target, closer_count)
        evaluate_list.append(algorithm(pred_df.loc["user"], pred_df.loc["mean"]))

    # 모든 user의 mae값의 평균을 리턴
    return np.average(evaluate_list)
```

----------------------------------------------------------------------------------------------------------------

# test code - evaluate

----------------------------------------------------------------------------------------------------------------

```
# evaluate
evaluate(user_df, sm_df, 5, mae)
```

----------------------------------------------------------------------------------------------------------------

```
1.8915940294133151
```

**6. Find Best Variance**

```
----------------------------------------------------------------------------------------------------
def find_best(user_df, similarity, closer_count):

   # similarity matrix
   sm_df = similarity_matrix(user_df, similarity)

   # evaluate
   return evaluate(user_df, sm_df, closer_count, mae)
----------------------------------------------------------------------------------------------------
```

```
----------------------------------------------------------------------------------------------------
similarity_str = ["euclidean_similarity", "cosine_smimilarity"]
similarity_list = [euclidean_similarity, cosine_smimilarity]
closer_start, closer_end = 10, 20

for idx, similarity in enumerate(similarity_list):
   print("similarity :", similarity_str[idx])
   for closer_count in range(closer_start, closer_end + 1):
      print(closer_count, find_best(user_df, similarity, closer_count))
----------------------------------------------------------------------------------------------------
```