

THE NETFLIX PRIZE

Recommender System

- Lim Jungmin

NETFLIX

Netflix Prize

Home

Rules

Leaderboard

Update

Leaderboard

Showing Test Score. [Click here to show quiz score](#)

Display top leaders.

Rank	Team Name	Best Test Score	% Im
------	-----------	-----------------	------

Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic

1	BellKor's Pragmatic Chaos	0.8567	
2	The Ensemble	0.8567	
3	Grand Prize Team	0.8582	
4	Opera Solutions and Vandelay United	0.8588	
5	Vandelay Industries !	0.8591	
6	PragmaticTheory	0.8594	
7	BellKor in BiqChaos	0.8601	
8	Dace	0.8612	
9	Feeds2	0.8622	
10	BiqChaos	0.8623	
11	Opera Solutions	0.8623	
12	BellKor	0.8624	

Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in

13	xiangliang	0.8642	
----	----------------------------	--------	--

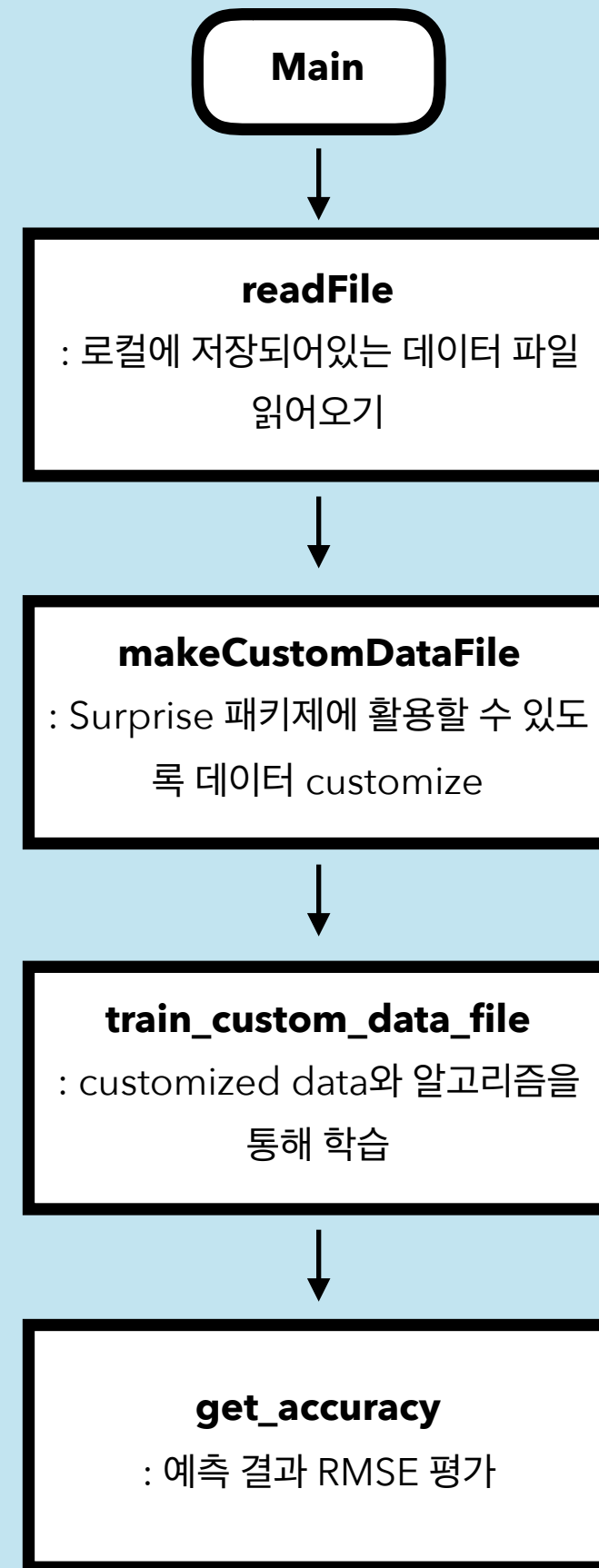

```
if __name__ == '__main__':
```

```
    # 1. 데이터 파일 읽어오기  
    ✓ start_time = time.time()  
    data_file = readFile("/Users/limjungmin/  
    Netflix_Recommender/netflix-prize-data/  
    combined_data_1.txt")  
    run_time = time.time() - start_time  
    print ( " Run time for readFile : %.4f (sec)" %  
    (run_time) )
```

```
    # 2. Surprise 패키지에 활용할 수 있도록 데이터 전처리  
    ✓ start_time = time.time()  
    custom_data_file=makeCustomDataFile(data_file)  
    run_time = time.time() - start_time  
    print ( " Run time for makeCustomDataFile : %.  
    4f (sec)" % (run_time) )
```

```
    # 3. 사용할 알고리즘(SVD)를 통한 학습 진행  
    ✓ start_time = time.time()  
    predictions =  
    train_custom_data_file(custom_data_file,  
    algo = SVD())  
    run_time = time.time() - start_time  
    print ( " Run time for train_custom_data_file :  
    %.4f (sec)" % (run_time) )
```

```
    # 4. 예측 결과를 가지고 RMSE 측정값 구하기  
    ✓ get_accuracy(predictions)
```



* Netflix Prize #1 : 0.8567

* Surprise 모듈 사용 결과 : 0.9028
낮을 수록 오차 범위가 줄어든다.

```
def readFile(path):

    data_file = open(path)
    return data_file
```

readFile


: 전달된 path를 통해 file을 열고, 해당 데이터 파일을 반환.

Netflix-prize-data INFO

CustomerID,Rating,Date

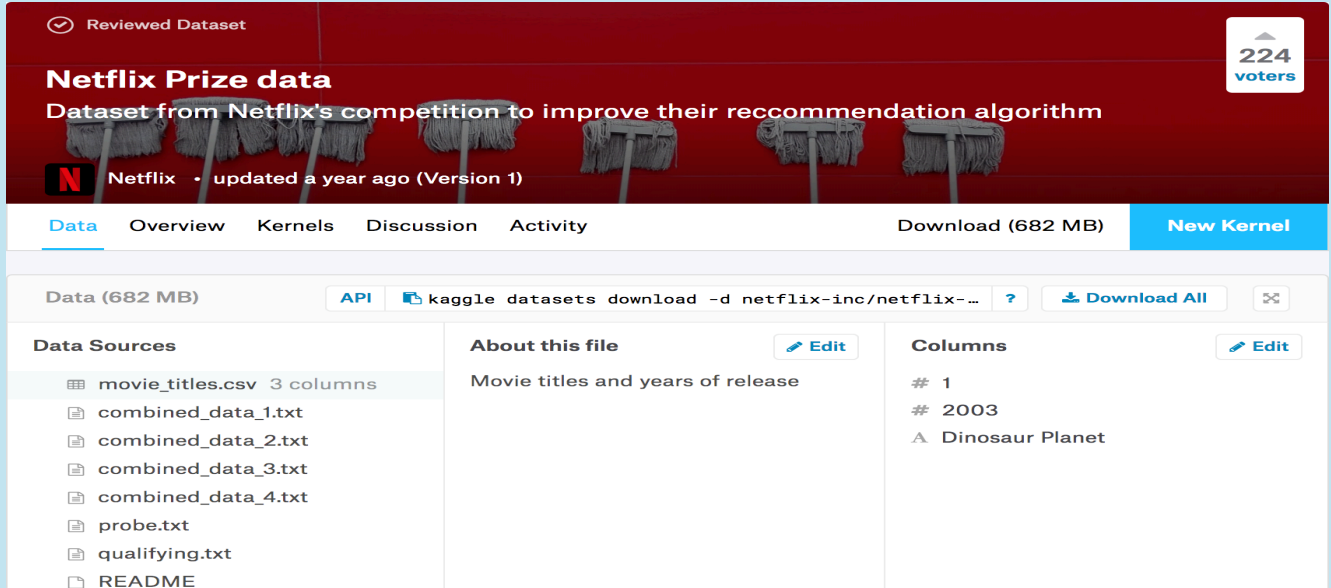
- MovieIDs range from 1 to 17770 sequentially.
- CustomerIDs range from 1 to 2649429, with gaps. There are 480189 users.
- Ratings are on a five star (integral) scale from 1 to 5.
- Dates have the format YYYY-MM-DD.

데이터(combined_data_1~4.txt 존재, probe set 존재)



```
1:
1488844,3,2005-09-06
822109,5,2005-05-13
885013,4,2005-10-19
30878,4,2005-12-26
823519,3,2004-05-03
893988,3,2005-11-17
124105,4,2004-08-05
1248029,3,2004-04-22
1842128,4,2004-05-09
2238063,3,2005-05-11
1503895,4,2005-05-19
2207774,5,2005-06-06
2590061,3,2004-08-12
2442,3,2004-04-14
543865,4,2004-05-28
1209119,4,2004-03-23
804919,4,2004-06-10
1086807,3,2004-12-28
1711859,4,2005-05-08
372233,5,2005-11-23
1080361,3,2005-03-28
1245640,3,2005-12-19
558634,4,2004-12-14
2165002,4,2004-04-06
1181550,3,2004-02-01
1227322,4,2004-02-06
427928,4,2004-02-26
814701,5,2005-09-29
808731,4,2005-10-31
```

* <https://www.kaggle.com/netflix-inc/netflix-prize-data>



Reviewed Dataset

Netflix Prize data 224 voters

Dataset from Netflix's competition to improve their recommendation algorithm

Netflix · updated a year ago (Version 1)

Data Overview Kernels Discussion Activity Download (682 MB) New Kernel

Data (682 MB) API kaggle datasets download -d netflix-inc/netflix-... Download All

Data Sources	About this file	Columns
<ul style="list-style-type: none"> movie_titles.csv 3 columns combined_data_1.txt combined_data_2.txt combined_data_3.txt combined_data_4.txt probe.txt qualifying.txt README 	<p>Movie titles and years of release</p>	<ul style="list-style-type: none"> # 1 # 2003 A Dinosaur Planet

```
def makeCustomDataFile(data_file):

# combined_data_1.txt에는 4499개의 movieID가 저장.
    custom_data_file = open("/Users/limjungmin/
    Netflix_Recommender/u.data", 'w')

    #cnt = 0 : 디버깅용 Count 계수
    for line in data_file:

        if ":" in line:
            movieID = line.split(":")[0]
            #print(movieID)
            #cnt+=1
        else :

            info = line.split(",")

            userID = info[0]
            rating = info[1]
            date = info[2].split('\n')[0]

            str = userID + ";" + movieID + ";" +
            rating + "\r\n"
            custom_data_file.write(str)

            #if cnt > 50 : break

    print("make Custom Data File Done")

    reader = surprise.Reader(
    line_format='user item rating', sep=';')

    data = surprise.Dataset.load_from_file('/Users/
    limjungmin/Netflix_Recommender/u.data',
    reader=reader)

    df = pd.DataFrame(data.raw_ratings,
    columns=["user", "item", "rate", "id"])

    del df["id"]

    print(df.head(10))

    return data
```

makeCustomDataFile

: 기존의 데이터를 **surprise** 패키지에 사용 가능하도록 전처리.

* https://surprise.readthedocs.io/en/latest/getting_started.html?highlight=custom#use-a-custom-dataset

원본 데이터의 형식

[movieID: userID, rating, date]

1:

1488844, 3, 2005-09-06

822109, 5, 2005-05-13

...



custom 데이터의 형식

[userID; movieID; rating]

1488844; 1; 3

822109, 1, 5

...

	user	item	rate
0	217506	1	2.0
1	1488844	1	3.0
2	822109	1	5.0
3	885013	1	4.0
4	30878	1	4.0
5	823519	1	3.0
6	893988	1	3.0
7	124105	1	4.0
8	1248029	1	3.0
9	1842128	1	4.0

* 출력결과

```
def train_custom_data_file(data, algo):

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

    algo.fit(trainset)

    predictions = algo.test(testset)

    return predictions
```

✓ # 3. 사용할 알고리즘(SVD)을 통한 학습 진행

```
start_time = time.time()
predictions =
train_custom_data_file(custom_data_file,
                        algo = SVD())
```

* 프로그램 Main에서 사용할 알고리즘 전달

- SVD() 알고리즘 사용 방법

https://surprise.readthedocs.io/en/latest/matrix_factorization.html?highlight=svd

- 사용가능한 알고리즘 목록

random_pred.NormalPredictor
baseline_only.BaselineOnly
knns.KNNBasic
knns.KNNWithMeans
knns.KNNWithZScore
knns.KNNBaseline
matrix_factorization.SVD
matrix_factorization.SVDpp
matrix_factorization.NMF
slope_one.SlopeOne
co_clustering.CoClustering

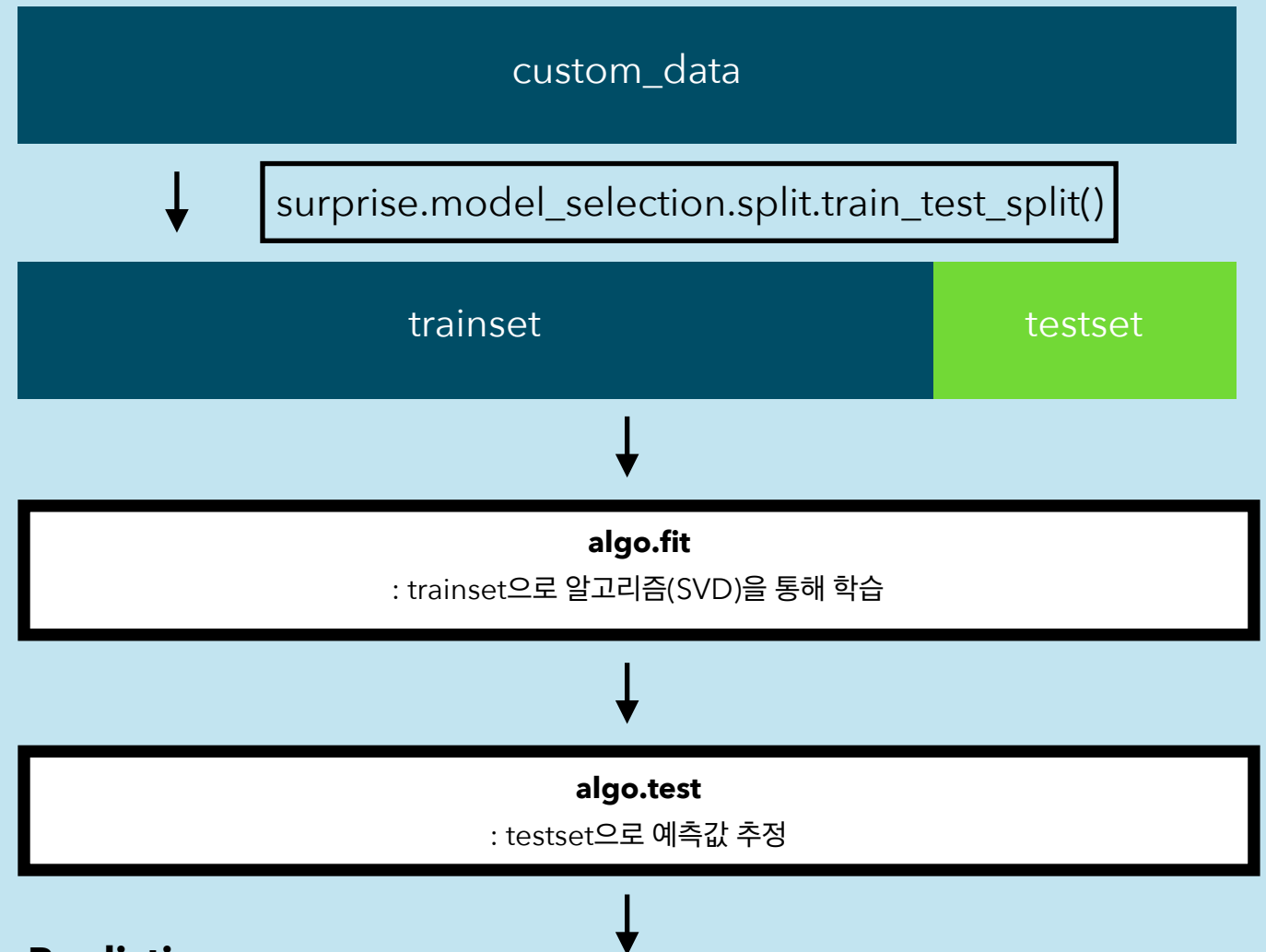
https://surprise.readthedocs.io/en/latest/prediction_algorithms_package.html#

train_custom_data_file

: custom 데이터와 사용할 알고리즘을 전달받아 학습 후 예측 결과물 반환

* [https://surprise.readthedocs.io/en/latest/model_selection.html?](https://surprise.readthedocs.io/en/latest/model_selection.html?highlight=train_test#surprise.model_selection.split.train_test_split)

[highlight=train_test#surprise.model_selection.split.train_test_split](https://surprise.readthedocs.io/en/latest/model_selection.html?highlight=train_test#surprise.model_selection.split.train_test_split)



Prediction

```
predictions[1:10]

[Prediction(uid='1830000', iid='1336', r_ui=3.0, est=2.983732581560073, details={'was_impossible':
False}),
 Prediction(uid='1695685', iid='3893', r_ui=3.0, est=4.006716702498658, details={'was_impossible':
False}),
 Prediction(uid='112076', iid='1608', r_ui=5.0, est=3.804846493184515, details={'was_impossible':
False}),
 Prediction(uid='1699258', iid='312', r_ui=2.0, est=3.2595075184234363, details={'was_impossible':
False}),
```

* uid : userID

r_ui : 사용자가 매긴 평점

est : 알고리즘이 예측한 평점

$$\begin{array}{ccccccc}
 \mathbf{A} & & \mathbf{U} & & \mathbf{\Sigma} & & \mathbf{V}^T \\
 \begin{array}{|c|} \hline \mathbf{A}_k \\ \hline \end{array} & = & \begin{array}{|c|} \hline \mathbf{U}_k \\ \hline \end{array} & \times & \begin{array}{|c|} \hline \mathbf{\Sigma}_k \\ \hline \end{array} & \times & \begin{array}{|c|} \hline \mathbf{V}_k^T \\ \hline \end{array} \\
 m \times n & & m \times m & & m \times n & & n \times n
 \end{array}$$

SVD(Singular Value Decomposition)

SVD(Singular Value Decomposition)의 추가설명 기재 예정

```
def get_accuracy(predictions):
    return accuracy.rmse(predictions)
```

get_accuracy
: prediction 객체를 전달받아 RMSE 평가 점수를 구한 후 반환한다.

```
print(accuracy.rmse(predictions))
```

```
RMSE: 0.9028
0.9027793490017464
```

Rank	Team Name	Best Test Score
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos		
1	BellKor's Pragmatic Chaos	0.8567
2	The Ensemble	0.8567
3	Grand Prize Team	0.8582
4	Opera Solutions and Vandelay United	0.8588
5	Vandelay Industries !	0.8591
6	PragmaticTheory	0.8594
7	BellKor in BigChaos	0.8601
8	Dace	0.8612
9	Feeds2	0.8622
10	BigChaos	0.8623
11	Opera Solutions	0.8623
12	BellKor	0.8624

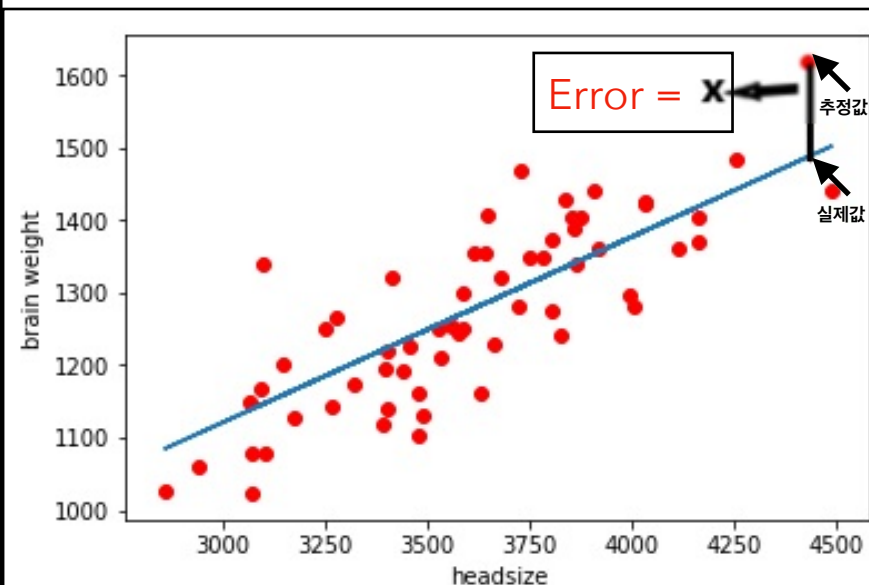
Result

Netfilx Prize에서 1위를 차지한 팀의 RMSE 값은 0.8567이며, surprise 모듈을 사용하여 측정한 RMSE 값은 0.9028이다.

* 추정값이 0에 가까울 수록 실제 값과의 오차가 적음을 나타낸다.

* RMSE

: 평균 제곱근 오차(Root Mean Square Error; RMSE)는 추정 값 또는 모델이 예측한 값과 실제 환경에서 관찰되는 값의 차이를 다룰 때 흔히 사용하는 척도이다.



검증 방법의 개요

RMSE : 오차(Error)를 제곱(Square)해서 평균(Mean)한 값의 제곱근(Root)

즉, 좌측 그림의 전체 Error와 같이, 예상한 평점값과 실제 평점 결과가 **평균적으로** 얼마만큼 차이나는가를 구하는것으로 이해하면된다.

Neflix Prize의 1위 수치인 0.8567의 경우 평균적으로 **0.8567**, Surprise 모듈의 경우 평균적으로 **0.9026**의 수치 정도로 실제 평점과 차이가 난다고 이해하면 된다.