Report

Class 데이터 과학

Title | 영화 평점 예측(추천)

학 과 명	컴퓨터공학과
교 수 명	이영석 교수님
학 번	201601989
이 름	김 진 섭
제 출 일	2019. 06. 09

충남대학교
Chungnam National University

목 차

과제 목표

1. 네이버 평점 크롤링

- 조교님이 제공해준 100명이 작성한 리뷰 데이터를 크롤링을 활용하여 수집한다.
- 사용자의 정보, 평가정보, 영화 정보를 가지고 있는 csv 파일을 생성한다.
- user.csv(100명의 사용자)
- rating.csv(5,124개의 리뷰)

2. 추천 시스템

■ 유사 사용자 탐색

- 유사 사용자를 탐색하고 평점을 예측한다.
- 수집된 데이터에 대한 유사도 계산을 수행하여 준다.
- 이웃을 탐색하는데, 이웃을 탐색할 대상은 가장 많은 평점을 남긴 사용자 Top10을 선정하여 이웃을 탐색한다.
- 유사도 계산 방법1 : Cosine (상품간의 각도)
- 유사도 계산 방법2 : Euclidean (상품간의 거리)
- 유사도 계산 방법3 : Correlation (변량 상관도)

■ 영화 평점 예측

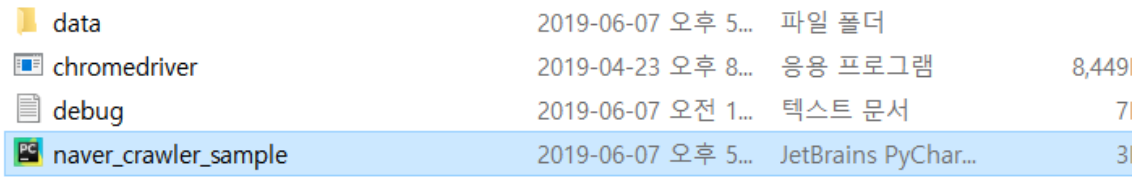
- 리뷰를 가장 많이 남긴 사용자와 그 사용자의 가장 최신 리뷰의 두 번째 영화의 평점 예측(최근에 남긴 두 번째 리뷰)
- 유사 사용자 탐색에서 사용한 데이터 활용
- 근접 유저가 한명이라도 있다면 평점을 계산하게됨
- 평점에 대한 Error 확인
- 에러는 MAE, MSE로 측정한다.

[과제1] 네이버 평점 데이터 수집

1. user.csv
- 100명의 사용자 데이터
- github에 제공된 데이터를 사용

2. rating.csv
- 총 5,123개의 리뷰
- 크롤링을 활용하여 데이터 생성

(환경 세팅)



Chromedriver를 설치하고 위처럼 실행할 python 파일과 같은 디렉토리에 위치시킨다. pip install을 활용하여 사용해야하는 모듈(Selenium, BeautifulSoup)을 설치한다.

(진행 과정)

```
from selenium import webdriver
from bs4 import BeautifulSoup
import os
import re
import random
import time
import pandas as pd
```

크롤링을 위해 필요한 python 모듈과 기타 처리를 위해 필요한 모듈을 import한다.

```
driver = webdriver.Chrome('./chromedriver')
driver.implicitly_wait(3)
```

driver를 사용하기 위해 세팅한다.

```
#영화 번호 가져오기
def get_movie_link(soup):
    movie_links = soup.select('a[href]')

    movie_links_list = []
    for link in movie_links:
        if re.search(r'st=mcode&sword' and r'&target=after$', link['href']):
            target_url = 'https://movie.naver.com/movie/point/af/list.nhn' + str(link['href'])
            movie_links_list.append(target_url)

    return movie_links_list[1:]
```

영화의 번호를 가져오는 것이다.

select로 가져와서 그 영화의 번호를 가져오기 위하여 url 정보를 가져온다.

추후 url정보를 replace로 제거하여 사용하면 그것이 영화의 번호가 된다.

```
#해당 reviewNo의 리뷰를 쓴 사람의 모든 리뷰 정보(reviewNo 이하) 가져오기
def get_review(reviewNo):
    user_id = []
    user_rating = []
    user_reviewnum = []
    user_movieId = []
    page = 1
    FirstReviewnum = 0
```

review를 가져오는 코드이다. 우선, 리뷰에 대한 정보 (user id, 평점, 리뷰 번호, 영화 번호)를 저장하기 위하여 리스트를 생성한다. 항상 reviewNo이 하나 주어지면 그것을 쓴 사람의 review를 가져오게 되는 방식이며, 1 page부터 시작된다.

FirstReviewnum은 어떤 page의 첫 번째 리뷰 넘버값으로 추후에 마지막 페이지인지 확인하는 용도로 사용한다.

```
    while True:
        driver.get('https://movie.naver.com/movie/point/af/list.nhn?st=nickname&sword={0}&target=after&page={1}'.format(reviewNo,page))
        page +=1
```

무한루프를 돌며, 해당 review를 올린 사람의 모든 review정보를 가져온다. 무한루프를 도는 이유는 page가 여러장 일 수 있기 때문이다.

```
        html = driver.page_source
        soup = BeautifulSoup(html, 'html.parser')
        userId = soup.select('table.list_netizen > tbody > tr > td > a.author')
        reviewRating = soup.select('#old_content > table > tbody > tr > td.point')
        reviewNum = soup.select('#old_content > table > tbody > tr > td.ac.num')
        if len(reviewNum) == 0:
            break
        if (page != 2) and (FirstReviewnum == int(str(reviewNum[0]).replace('<td class="ac num">', '').replace('</td>', ''))):
            break
        FirstReviewnum = int(str(reviewNum[0]).replace('<td class="ac num">', '').replace('</td>', ''))
```

html을 가져오고, BeautifulSoup을 활용해 파싱을 하여준다.

soup에서 select들을 가져온다. user의 id와 rating 그리고 review num에 대한 select 정보를 가져오도록 한다. (그렇게 되면 한 페이지에 있는 영화의 정보를 가져올 수 있다.)

만약 가져온 값이 없다면 break로 종료를 시킨다,(삭제되었는데, 하나만 쓴 경우)

그 외의 경우에는 매 페이지의 첫 번째 Review num 값이 이전과 동일한지 확인하여 동일하다면 모두 확인하여쓰므로 break를 하여준다.

FirstReviewnum을 만들어준다.(첫번째 Review의 number 추후 끝 페이지 확인용도)

```
for i in range(len(reviewRating)):
    user = userId[i].text.replace('*', '')
    rating = str(reviewRating[i]).replace('<td class="point">', '').replace('</td>', '')
    num = str(reviewNum[i]).replace('<td class="ac num">', '').replace('</td>', '')
    num = int(num)
    user_id.append(user)
    user_rating.append(rating)
    user_reviewnum.append(num)
movie_links = get_movie_link(soup)
movieId = [link.replace('https://movie.naver.com/movie/point/af/list.nhn?st=mcode&sword=', '').replace('&target=after', '') for link in movie_links]
user_movieId = user_movieId + movieId
time.sleep(random.randrange(2, 5))
```

for문을 돌며 위에서 가져온 select 태그들의 값들을 가져와 넣어준다.
replace를 통하여 html의 태그들을 지워주고 값들만 가져오도록한다.
그 값들을 모두 리스트에 넣어준다.(그렇게 되면 한 페이지에 대한 평점 정보를 가져온 것과 동일하다.) 마지막에 movie Id를 가져오기 위하여 get_movie_link를 수행하고 그 결과 역시 replace로 적절하게 바꾸어준다.
time.sleep을 하는 이유는 빠르게 크롤링을 해주면 ip의 벤을 당할 수 있는데 이를 방지하기 위함이다.

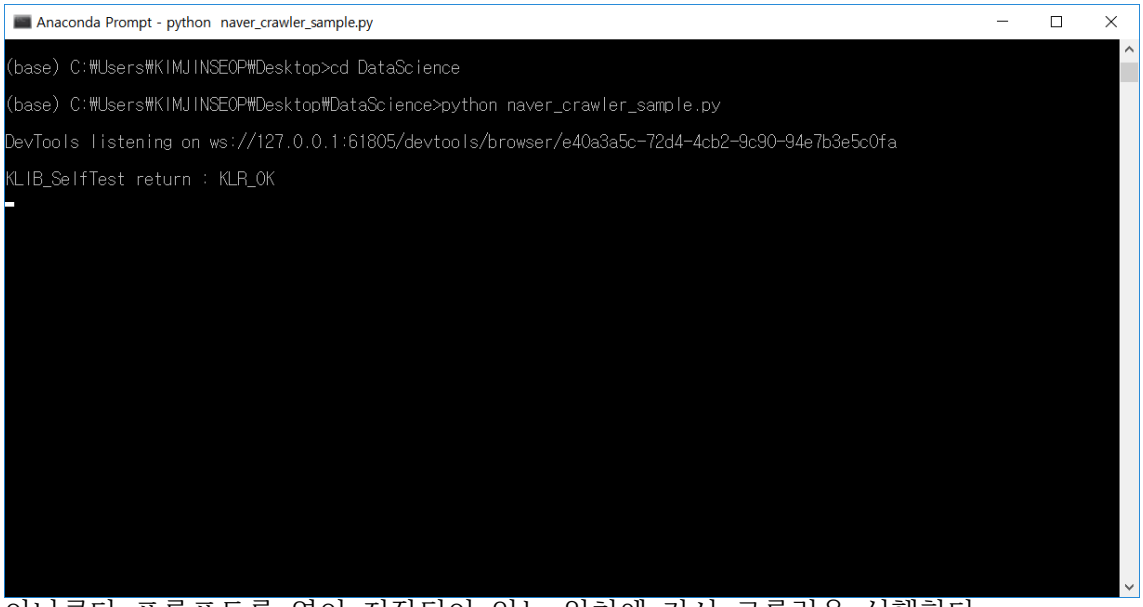
```
userData = pd.DataFrame()
userData['userId'] = user_id
userData['rating'] = user_rating
userData['movieId'] = user_movieId
userData['reviewNo'] = user_reviewnum
userData = userData[userData['reviewNo'] <= int(reviewNo)]
del userData['reviewNo'] ##
return userData
```

이제 최종적으로 데이터프레임으로 return을 시켜준다.
위에서 생성한 값들을 활용하여 데이터 프레임을 만들어준다. 이 때, 조교님이 생성한 데이터와 동일해지기 위하여 조교님이 제공하여준 100명의 reviewNo보다 작거나 같은 reviewNo을 가진 리뷰로 줄여주고, 마지막으로 reviewNo 변수를 제거하여준다.

```
def main():
    data = pd.read_csv('./data/naver_user.csv')
    df = pd.DataFrame()
    for reviewNum in data['reviewNo']:
        ReviewData = get_review(reviewNum)
        df = pd.concat([df,ReviewData])
    df = df[False == df.duplicated(['userId', 'rating', 'movieId'], keep='first')]
    df.to_csv('./data/rating.csv', index = False)
    driver.close()
```

이것은 main을 실행하는 과정이다.
우선 조교님이 주신 100명의 데이터를 가져오고 그 데이터들 100개의 reviewNum을 활용하여 Review들을 크롤링을 수행한다.
그 크롤링한 데이터를 하나의 프레임으로 모아준다. 이 때, duplicated를 활용하여 중복되는 데이터가 있다면 처리하여준다.
데이터프레임의 to_csv를 활용하여 저장한다.

(최종 확인)



아나콘다 프롬프트를 열어 저장되어 있는 위치에 가서 크롤링을 실행한다.
(추가 삭제된 리뷰는 csv파일에 직접추가해 준다)

[과제1] 네이버 평점 데이터 수

```
In [2]: df = pd.read_csv('./data/rating.csv')
        df.tail()
```

Out[2]:

	userId	rating	movieId
5119	huya	10	73394
5120	huya	10	70773
5121	huya	8	16220
5122	huya	10	36666
5123	huya	10	37235

데이터 프레임을 가져오기 위하여 패키지를 가져오고 read_csv를 활용하여 실행한다.

```
user_num = df.userId.unique().shape[0]
movie_num = df.movieId.unique().shape[0]
print('유저 수는 {0}명 이고, ₩n영화의 수는 {1}개 입니다'.format(user_num,movie_num))
```

```
유저 수는 100명 이고,
영화의 수는 2697개 입니다
```

user의 수와 영화의 수에 대한 결과이다.
유저의 수는 100명이며 영화의 수는 2697임을 알 수 있다.

[과제2-a] 유사 사용자 탐색

```
import pandas as pd
import csv
from collections import defaultdict
from datetime import datetime
import matplotlib.patches as mpatches
import matplotlib
import time
import math
from operator import itemgetter
from scipy.spatial import distance
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

과제를 수행하기 위해 필요한 모듈을 import한다.

```
df = pd.read_csv('./data/rating.csv')
df.tail()
```

	userId	rating	movieId
5119	huya	10	73394
5120	huya	10	70773
5121	huya	8	16220
5122	huya	10	36666
5123	huya	10	37235

```
df_user = pd.read_csv('./data/naver_user.csv')
df_user.head()
```

	reviewNo	userId
0	15772038	airf
1	15772037	nanw
2	15772036	zxcv
3	15772035	sdh1
4	15772032	guan

과제 수행을 위해 필요한 두 개의 csv파일을 가져온다.

[과제 2-a] 유사 사용자 탐색

```
data = pd.merge(df, df_user, on = ['userId'],how = 'left')
data['Count'] = 1
```

merge를 활용하여 위 두 개의 데이터를 userId로 합해주는데, rating을 왼쪽에 두고 how를 left로 설정해 merge해 준다. 그렇게 되면 사용자의 가장 최신 리뷰번호가 reviewNo로 rating.csv 데이터 옆에 붙은 dataframe이 만들어 진다.

그리고 Count를 추가하여 top10을 선정할 준비를 한다.

가장 많은 평점을 남긴 사용자 10명의 2번째 리뷰

```
Top10 = data.groupby(['userId'])['Count'].sum().nlargest(10).reset_index()
data = data[True == data.duplicated(['reviewNo'], keep='first')]
data = data[False == data.duplicated(['reviewNo'], keep='first')]
del Top10['Count']
del data['Count']
Top10 = pd.merge(Top10, data, on = ['userId'],how = 'left')
Top10.head(10)
```

	userId	rating	movieId	reviewNo
0	ykm3	3	145162	15771936
1	sang	10	161967	15771961
2	tsp0	7	163788	15771934
3	hosu	7	180399	15771998
4	zxcv	10	86507	15772036
5	zard	10	158653	15772012
6	artn	10	172174	15771948
7	suha	7	180399	15771976
8	ldsl	1	157297	15771977
9	imag	9	181409	15771940

가장 많은 리뷰를 남긴 사용자 10명의 두 번째 리뷰에 대한 데이터를 가져온다. Groupby와 nlagest를 활용하여 Top10을 선정한다.

이 때 가장 많은 평점을 남긴 사용자의 두 번째 리뷰를 가져와야하므로, 우선 duplicated를 활용하여 True일 때 걸러주어 첫 번째 리뷰를 제외한 나머지 리뷰를 가져온다. (이 때 reviewNo이 동일한 것은 같은 사용자가 작성한 것이며 keep값이 first이므로 첫 번째 리뷰가 지워지게된다.) 한번 더 duplicated를 사용해 false를 가져오며 두 번째 리뷰를 가져온다. 그렇게 되면 data에 들어가 있는 데이터는 모든 사용자의 두 번째 리뷰가 된다.

이제 Top10은 모두 선정 되었으므로, Count변수가 의미 없으므로 지워준다.

Top10에 대하여 merge를 활용하여 두 번째 리뷰를 붙여준다.

```
df = pd.merge(df, df_user, on = ['userId'],how = 'left')
df.head()
```

	userId	rating	movieId	reviewNo
0	airf	2	136900	15772038
1	airf	10	163788	15772038
2	airf	10	174065	15772038
3	nanw	10	154667	15772037
4	nanw	10	136900	15772037

df를 위 초기 data처럼 reviewNo을 붙여준다.

```
UM_matrix_ds = df.pivot(index ='reviewNo', columns = 'movieId',values = 'rating')
UM_matrix_ds.head(5)
```

movieId	10002	10003	10004	10005	10006	10008	10009	10012	10016	10018	...	181409	181410	181411	181414	181419	181711	182348	182360	1
reviewNo																				
15771922	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
15771925	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	8.0	NaN	NaN	NaN	NaN	NaN	NaN	
15771926	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
15771927	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
15771929	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

5 rows × 2697 columns

UM_matrix_ds는 x축은 영화이고, y축은 사용자이다. x축과 y축이 교차하는 지점의 값은 평점이다. 즉 100명의 사용자의 평점에 대한 정보이다. 여기서 보면 15771925번 사람이 181410 영화에 대하여 8점의 평점을 주었음을 알 수 있다.

유사 사용자 탐색을 위해 선언

```
def distance_cosine(a,b):
    return 1-distance.cosine(a,b)
```

```
def distance_correlation(a,b):
    return 1-distance.correlation(a,b)
```

```
def distance_euclidean(a,b):
    return 1/(1+distance.euclidean(a,b))
```

유사 사용자를 탐색하기 위하여 세 개의 함수를 선언한다.
distance_cosine은 두 개의 데이터 사이의 각도를 측정하는 것이다.
distance_correlation은 두 개의 변량에 대한 상관도를 측정하는 것이다.
distance_euclidean은 두 개의 변량에 대한 유클리디안 거리를 나타낸다. 1/(1+x)로 (단 x 는 두 개의 유클리디안 값) 만들어 가까울 때 1 멀어질수록 0에 가깝게 만들어주었다.

```
def nearest_neighbor_user(user, topN, simFunc):
    u1 = UM_matrix_ds.loc[user].dropna()
    ratedIndex = u1.index
    nn = {}

    for uid, row in UM_matrix_ds.iterrows():
        interSectionU1 = []
        interSectionU2 = []
        if uid == user : continue
        for i in ratedIndex:
            if not math.isnan(row[i]):
                interSectionU1.append(u1[i])
                interSectionU2.append(row[i])
        interSectionLen = len(interSectionU1)

        if interSectionLen < 3: continue
        sim = simFunc(interSectionU1, interSectionU2)

        if not math.isnan(sim): nn[uid] = sim
    return sorted(nn.items(), key=itemgetter(1))[:-(topN+1):-1]
```

사용자의 이웃을 찾는 함수이다. 해당 user의 topN명 만큼 가까운 유저를 simFunc 방식을 활용하여 측정해 뽑아낸다. 마지막에 sorted를 활용하여 오른차 순으로 정렬한 뒤 뒤에서 topN개 만큼 뽑아내어 준다.

[과제 2-a] 결과 - Cosine

```
print('Cosine 결과')
topN = 3
for user in Top10['reviewNo']:
    neighbor = []
    nearest = nearest_neighbor_user(int(user),3,distance_cosine)
    for i in range(0,topN):
        neighbor.append([nearest[i][0], round(nearest[i][1],2)])
    print('User {0} neighbors : {1}'.format(user, neighbor))
```

```
Cosine 결과
User 15771936 neighbors : [[15771972, 1.0], [15771993, 0.97], [15772003, 0.95]]
User 15771961 neighbors : [[15771947, 1.0], [15771970, 1.0], [15771944, 1.0]]
User 15771934 neighbors : [[15771942, 1.0], [15772009, 1.0], [15771970, 0.99]]
User 15771998 neighbors : [[15771970, 1.0], [15771965, 1.0], [15771947, 1.0]]
User 15772036 neighbors : [[15771959, 1.0], [15771943, 0.98], [15771966, 0.98]]
User 15772012 neighbors : [[15772003, 1.0], [15772016, 0.95], [15772000, 0.92]]
User 15771948 neighbors : [[15771966, 1.0], [15771945, 1.0], [15771943, 0.98]]
User 15771976 neighbors : [[15771966, 1.0], [15771947, 1.0], [15771926, 1.0]]
User 15771977 neighbors : [[15771971, 1.0], [15771959, 1.0], [15771938, 0.99]]
User 15771940 neighbors : [[15771993, 1.0], [15771947, 1.0], [15771981, 1.0]]
```

Cosine값으로 이웃을 찾은 결과 값이다. 소수점 둘째자리까지 표현하기 위하 top10의 use에 대하여 nearest_neighbor_user를 수행한 그 결과를 이용하여 neighbor의 정보를 round를 통해 반올림하여 새로 나타내어 보여주었다. 1에 가까울수록 가까운 이웃이다.

[과제 2-a] 결과 - Correlation

```
print('Correlation 결과')
topN = 3
for user in Top10['reviewNo']:
    neighbor = []
    nearest = nearest_neighbor_user(int(user),3,distance_correlation)
    for i in range(0,topN):
        neighbor.append([nearest[i][0], round(nearest[i][1],2)])
    print('User {0} neighbors : {1}'.format(user, neighbor))
```

```
Correlation 결과
C:₩Users₩KIMJINSEOP₩Anaconda3₩lib₩site-packages₩scipy₩spatial₩distance.py:702: RuntimeWarning: invalid
  dist = 1.0 - uv / np.sqrt(uu * vv)
User 15771936 neighbors : [[15771993, 1.0], [15772022, 0.69], [15772019, 0.5]]
User 15771961 neighbors : [[15771972, 1.0], [15772015, 0.94], [15771974, 0.93]]
User 15771934 neighbors : [[15771942, 1.0], [15772030, 0.98], [15771947, 0.96]]
User 15771998 neighbors : [[15772005, 0.89], [15771922, 0.88], [15771977, 0.66]]
User 15772036 neighbors : [[15771966, 0.97], [15771943, 0.94], [15771971, 0.62]]
User 15772012 neighbors : [[15772016, 0.94], [15771974, 0.58], [15771969, 0.58]]
User 15771948 neighbors : [[15771943, 0.94], [15771938, 0.73], [15771926, 0.59]]
User 15771976 neighbors : [[15771940, 0.85], [15771974, 0.58], [15771927, 0.53]]
User 15771977 neighbors : [[15771971, 1.0], [15771959, 1.0], [15771998, 0.66]]
User 15771940 neighbors : [[15771981, 0.98], [15771995, 0.97], [15771974, 0.94]]
```

Correlation을 이용해 이웃을 찾은 결과이다. 위와 동일하게 소수점 둘째 자리로 표현하였고, top 3명의 이웃을 뽑아 보여주었다. 1에 가까울수록 가까운 이웃이다.

[과제 2-a] 결과 - Euclidean

```
print('Euclidean 결과')
topN = 3
for user in Top10['reviewNo']:
    neighbor = []
    nearest = nearest_neighbor_user(int(user),3,distance_euclidean)
    for i in range(0,topN):
        neighbor.append([nearest[i][0], round(nearest[i][1],2)])
    print('User {0} neighbors : {1}'.format(user, neighbor))
```

```
Euclidean 결과
User 15771936 neighbors : [[15771972, 0.25], [15771993, 0.2], [15772020, 0.12]]
User 15771961 neighbors : [[15772031, 0.33], [15771970, 0.33], [15771954, 0.33]]
User 15771934 neighbors : [[15772030, 0.41], [15771988, 0.29], [15772003, 0.25]]
User 15771998 neighbors : [[15771922, 0.17], [15772030, 0.16], [15772005, 0.15]]
User 15772036 neighbors : [[15771959, 1.0], [15771943, 0.24], [15772022, 0.19]]
User 15772012 neighbors : [[15772016, 0.22], [15772003, 0.15], [15771974, 0.13]]
User 15771948 neighbors : [[15771945, 0.41], [15771943, 0.23], [15771938, 0.15]]
User 15771976 neighbors : [[15771974, 0.31], [15771926, 0.31], [15771966, 0.29]]
User 15771977 neighbors : [[15771971, 1.0], [15771959, 0.5], [15771938, 0.25]]
User 15771940 neighbors : [[15771993, 1.0], [15771947, 1.0], [15771981, 0.33]]
```

Euclidean의 거리를 활용하여 만들어낸 결과이다. 1이 가장 가까운 값이다.

[과제2-b] 영화 평점 예측

영화의 평점을 예측하고, 예측한 평점과 그 예측에 대한 에러값을 MSE와 MAE를 보여준다.

MAE는 Mean Squared Error로 평균 제곱의 오차이다.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

식은 위와 같다. Y햇은 예측값 Y값은 실값으로 두 개의 차이를 제곱하여 평균을 내는 것을 의미한다.

MAE는 Mean Absolute Error로 평균 절대값의 오차이다.

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n}|y_i - x_i|}{n}$$

식은 위와 같다. y값이 예측값 x값이 참값이라면 두 값의 차이를 절대값을 취하여 평균을 내는 것이다.

위 두 값 모두 작을수록 정확히 예측을 했다고 할 수 있다.

[과제 2-b] 영화 평점 예측

```
def predict_rating(userid, nn= 100, simFunc= distance_cosine):
    neighbor = nearest_neighbor_user(userid,nn,simFunc)
    neigbor_id = [id for id, sim in neighbor]

    neighbor_movie = UM_matrix_ds.loc[neigbor_id].dropna(1,how = 'all', thresh = 1)
    neighbor_dict = (dict(neighbor))
    ret = []

    for movieId, row in neighbor_movie.iteritems():
        jsum, wsum = 0, 0
        for v in row.dropna().iteritems():
            sim = neighbor_dict.get(v[0],0)
            jsum += sim
            wsum += (v[1]*sim)
        ret.append([movieId,wsum/jsum])
    return ret
```

영화 평점을 예측하기 위하여 만든 함수이다. thresh값을 1로 줌으로써 가까운 근처 유저가 1명이라도 평점을 남겼다면 평점을 예측하도록 한다.

[과제 2-b] Cosine 예측 결과 ¶

```
result = []
for i in range(10):
    userId = int(Top10.iloc[i].reviewNo)
    movieId = int(Top10.iloc[i].movieId)
    predict = predict_rating(userId, 300, distance_cosine)

    for movie in predict:
        if movieId == movie[0]:
            result.append([int(userId), int(movieId),movie[1]])
resultdf = pd.DataFrame(result, columns=['userId','movieId','rating'])
print('Cosine 결과')
resultdf
```

Cosine 결과

	userId	movieId	rating
0	15771936	145162	5.796201
1	15771961	161967	8.595640
2	15771934	163788	9.150525
3	15771998	180399	7.000000
4	15772036	86507	8.561077
5	15771976	180399	7.000000
6	15771977	157297	5.800920

Cosine으로 값을 예측한 결과이다. 위에서 선언한 predict_rating을 활용하여 예측을 수행한다. 그 결과는 아래의 값과 같다. 예측을 수행한 데이터는 이전에 뽑았던 리뷰를 가장 많이 쓴 10명의 두 번째 리뷰 데이터이다.(예상 평점)

```
realdata_rating = []
for userid in resultdf['userId']:
    realdata_rating.append(float(Top10[Top10['reviewNo'] == userid]['rating']))

resultdata_rating = resultdf.rating.tolist()

error_rate_absol = mean_absolute_error(realdata_rating, resultdata_rating)
error_rate_squared = mean_squared_error(realdata_rating, resultdata_rating)
print('₩nError Rate(Absolute) : ',error_rate_absol)
print('Error Rate(Squared) : ',error_rate_squared)
```

```
Error Rate(Absolute) :  1.7987041227601153
Error Rate(Squared) :  5.647865308638797
```

Cosine을 활용해 예측한 결과의 MAE와 MSE이다. (에러값 측정)

MAE는 약1.8

MSE는 약 5.65이다.

[과제 2-b] Correlation 예측 결과

```
result = []
for i in range(10):
    userId = int(Top10.iloc[i].reviewNo)
    movieId = int(Top10.iloc[i].movieId)
    predict = predict_rating(userId, 300, distance_correlation)

    for movie in predict:
        if movieId == movie[0]:
            result.append([int(userId), int(movieId),movie[1]])
resultdf = pd.DataFrame(result, columns=['userId','movieId','rating'])
print('Correlation 결과')
resultdf
```

```
C:₩Users₩KIMJINSEOP₩Anaconda3₩lib₩site-packages₩scipy₩spatial₩distance.py:702: RuntimeWarning: invalid value encountered in double_scalars
  dist = 1.0 - uv / np.sqrt(uu * vv)
C:₩Users₩KIMJINSEOP₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:15: RuntimeWarning: invalid value encountered in double_scalars
  from ipykernel import kernelapp as app
```

Correlation 결과

	userId	movieId	rating
0	15771936	145162	5.433217
1	15771961	161967	10.715807
2	15771934	163788	15.406481
3	15771998	180399	7.000000
4	15772036	86507	11.304849
5	15771976	180399	7.000000
6	15771977	157297	0.954683

Correlation를 활용한 예측 결과이다. 위 Cosine과 동일한 데이터를 예측하였다. (예상 평점)

```
realdata_rating = []
for userid in resultdf['userId']:
    realdata_rating.append(float(Top10[Top10['reviewNo'] == userid]['rating']))

resultdata_rating = resultdf.rating.tolist()

error_rate_absol = mean_absolute_error(realdata_rating, resultdata_rating)
error_rate_squared = mean_squared_error(realdata_rating, resultdata_rating)
print('₩nError Rate(Absolute) : ',error_rate_absol)
print('Error Rate(Squared) : ',error_rate_squared)
```

```
Error Rate(Absolute) :  1.8436674517673026
Error Rate(Squared) :  11.258077133072339
```

Correlation을 활용해 예측한 결과의 MAE와 MSE이다. (에러값 측정)

MAE는 약 1.84

MSE는 약 11.26이다.

[과제 2-b]Euclidean 예측 결과 ¶

```
result = []
for i in range(10):
    userId = int(Top10.iloc[i].reviewNo)
    movieId = int(Top10.iloc[i].movieId)
    predict = predict_rating(userId, 300, distance_euclidean)

    for movie in predict:
        if movieId == movie[0]:
            result.append([int(userId), int(movieId),movie[1]])
resultdf = pd.DataFrame(result, columns=['userId','movieId','rating'])
print('Euclidean 결과')
resultdf
```

Euclidean 결과

	userId	movieId	rating
0	15771936	145162	4.649008
1	15771961	161967	9.263031
2	15771934	163788	8.956179
3	15771998	180399	7.000000
4	15772036	86507	8.054289
5	15771976	180399	7.000000
6	15771977	157297	6.588583

Euclidean을 활용한 예측 결과이다. 위 Cosine과 동일한 데이터를 예측하였다. (예상 평점)

```
realdata_rating = []
for userid in resultdf['userId']:
    realdata_rating.append(float(Top10[Top10['reviewNo'] == userid]['rating']))

resultdata_rating = resultdf.rating.tolist()

error_rate_absol = mean_absolute_error(realdata_rating, resultdata_rating)
error_rate_squared = mean_squared_error(realdata_rating, resultdata_rating)
print('₩nError Rate(Absolute) : ',error_rate_absol)
print('Error Rate(Squared) : ',error_rate_squared)
```

```
Error Rate(Absolute) :  1.6966357708958737
Error Rate(Squared) :  6.015291725703062 5
```

Euclidean을 활용해 예측한 결과의 MAE와 MSE이다. (에러값 측정)

MAE는 약 1.7

MSE는 약 6.02이다.

위 데이터의 Error를 보면 Euclidean < Correlation < Cosine이므로 해당 데이터에 대하여 Euclidean을 활용하는 것이 가장 정확한 예측을 한 것이라고 할 수 있다.