Presentation

Encar

- Python으로 구현하는 추천시스템

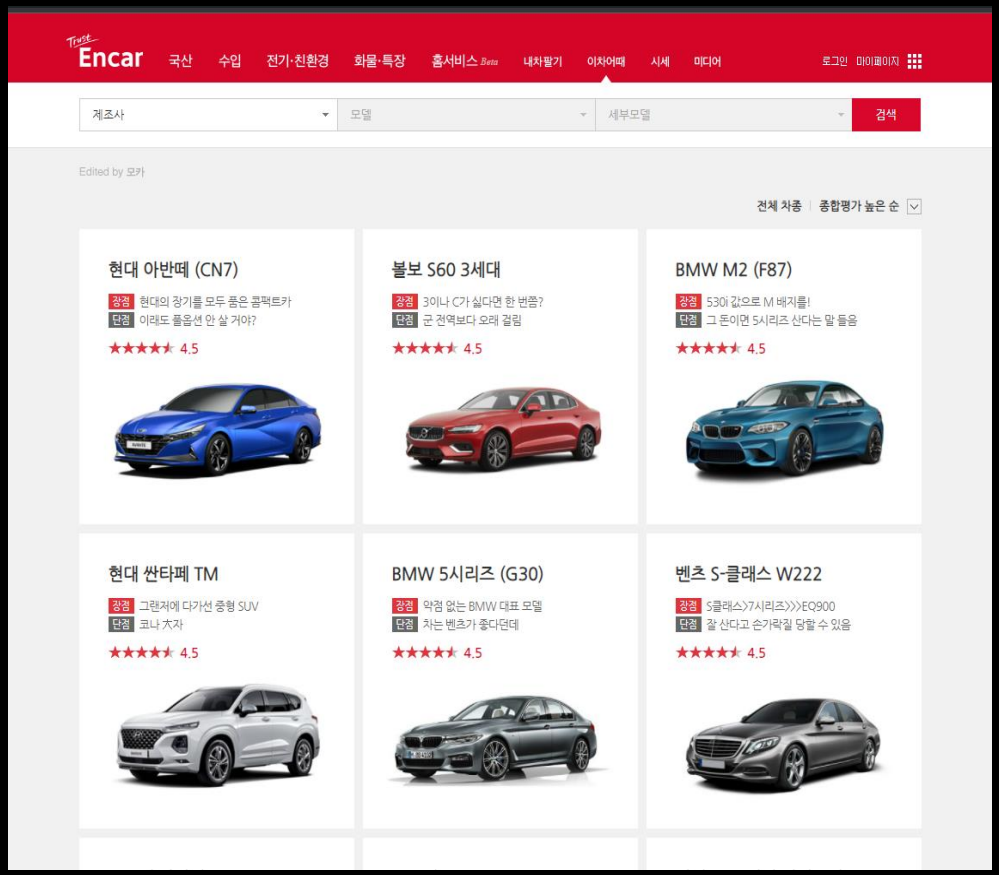
INDEX

01 개요 및 크롤링

개요

목적 : 중고차 사이트에서 사람들이 남긴 평점을 바탕으로 비슷한 취향의 다양한 중고차를 추천 (ex: 아반떼 -> 쏘나타 / 기아 K3 ...)

크롤링 한 사이트 : Encar (전 SK엔카)의 이차어때
http://www.encar.com/mocha.do?WT.hit=index_contents_title



크롤링

데이터 수집(크롤링) : SK엔카 중고차 평점 및 추천

In [1]:
```
import pandas as pd
import numpy as np
from selenium import webdriver
import time
from tqdm import tqdm_notebook
import chromedriver_autoinstaller
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
chrome_path = chromedriver_autoinstaller.install()
driver = webdriver.Chrome(chrome_path)

driver.get('http://www.encar.com/mocha.do?WT.hit=index_contents_title')
time.sleep(1)
```

In [ ]:
```
# 더보기 클릭
try :
    for more in range(10):
        more_page = driver.find_element_by_css_selector('.btn_info_more')
        more_page.click()
except:
    pass
```

In [ ]:
```
# 크롤링한 url 정제 시작
for i in product_raw:
    url = i.get_attribute('href')
    product_url_list.append(url)
time.sleep(1)

# 제목 크롤링 시작
for j in product_raw2:
    title = j.text
    product_title_list.append(title)

    print(title)

print("")
print('url갯수 : ', len(product_url_list))
print('title갯수 : ', len(product_title_list))
```

기존 사용했던 방식
그대로 사용

In [ ]:
```
df = pd.DataFrame({'url':product_url_list, 'title':product_title_list})
df
```

In [ ]:
```
# csv파일로 저장
df.to_csv("sk_encar_url.csv", encoding = 'utf-8-sig')
```

In [2]:
```
# "url_list.csv" 불러오기
url_load = pd.read_csv("sk_encar_url.csv")
url_load = url_load.drop("Unnamed: 0", axis=1)
url_load
```

Out[2]:

	url	title
0	http://www.encar.com/mocha/mochacontents.do?me...	현대 아반떼 (CN7)
1	http://www.encar.com/mocha/mochacontents.do?me...	볼보 S60 3세대
2	http://www.encar.com/mocha/mochacontents.do?me...	BMW M2 (F87)
3	http://www.encar.com/mocha/mochacontents.do?me...	현대 싼타페 TM
4	http://www.encar.com/mocha/mochacontents.do?me...	BMW 5시리즈 (G30)
...	...	...
176	http://www.encar.com/mocha/mochacontents.do?me...	현대 뉴 아반떼 XD
177	http://www.encar.com/mocha/mochacontents.do?me...	르노삼성 뉴SM3
178	http://www.encar.com/mocha/mochacontents.do?me...	쌍용 코란도 스포츠
179	http://www.encar.com/mocha/mochacontents.do?me...	현대 뉴 그랜저 XG
180	http://www.encar.com/mocha/mochacontents.do?me...	쌍용 렉스턴 W

181 rows × 2 columns

181개의 차량 추출

크롤링

크롤링 시작 (for문)

In [ ]:
```
%%time

title_list = []
score_list = []
nick_id_list = []

number = 181

for i in tqdm_notebook(range(0, number)):
    url = url_load['url'][i]
    chrome_path = chromedriver_autoinstaller.install()
    driver = webdriver.Chrome(chrome_path)
    driver.get(url)

    # 댓글 클릭
    driver.find_element_by_link_text('댓글').click()
    time.sleep(1)

    try :
        for more in range(10):
            more_page = driver.find_element_by_css_selector('.btn_info_more')
            more_page.click()
    except:
        pass

    time.sleep(1)

    nick = driver.find_elements_by_css_selector('.list_comment .txt_user')

    for k in range(len(nick)):

        #title
        title = url_load['title'][i]

        # 별점 크롤링
        try:
            result_1 = driver.find_elements_by_css_selector('.list_comment .img_mocha')
            stars = result_1[ 2*k +1 ].get_attribute('style')
            score = int(stars.split(' ')[1].replace('%;' , '')) * 0.05
        except :
            pass

        # 닉네임 크롤링
        try:
            nick_id = nick[k].text.split(' ')[0]

        except:
            pass


        #딕셔너리 저장
        title_list.append(title)
        nick_id_list.append(nick_id)
        score_list.append(score)

    driver.close()
```

- 크롤링 과정에서 별이 텍스트가 아닌 이미지.
- style에 있는 'width = 80%;'를 이용해 별점 추출 : get_attribute('style')

- split과 인덱싱을 활용
- 80을 추출했으면 0.05를 곱해줘서 5점만점으로 만들기

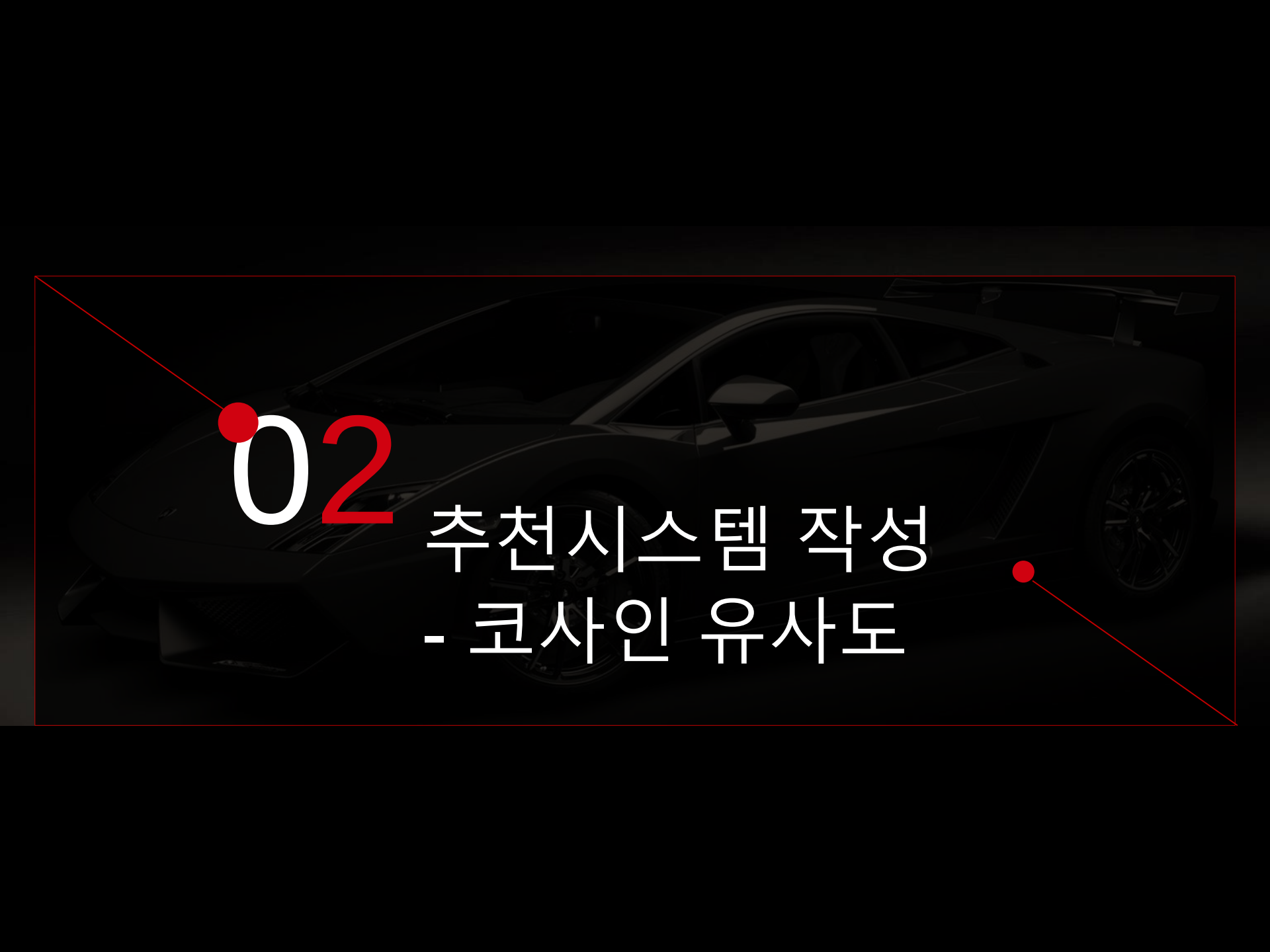
In [ ]:
```
sk_encar_df = pd.DataFrame()

sk_encar_df['title'] = title_list
sk_encar_df['id'] = nick_id_list
sk_encar_df['rating'] = score_list

print(sk_encar_df.shape)
sk_encar_df.head(5)
```

In [ ]:
```
sk_encar_df.to_csv("encar.csv", encoding='utf-8-sig')
```

23	현대 아반떼 (CN7)	rkfka3030	5
24	현대 아반떼 (CN7)	kti0317	5
25	현대 아반떼 (CN7)	ghd5029	5
26	현대 아반떼 (CN7)	human115	5
27	현대 아반떼 (CN7)	rkfka3030	5
28	현대 아반떼 (CN7)	yscsj	5
29	현대 아반떼 (CN7)	soltyrei	5
30	볼보 S60 3세대	skydiscus	1.5
31	볼보 S60 3세대	dlwldus27	0
32	볼보 S60 3세대	eos30d	5
33	볼보 S60 3세대	kd8366	3
34	볼보 S60 3세대	marty25	3
35	볼보 S60 3세대	sofi0604	5
36	볼보 S60 3세대	lgy828282	5
37	BMW M2 (F87)	insolbi	5
38	BMW M2 (F87)	onesun20(	5
39	현대 싼타페 TM	eld8899	0
40	현대 싼타페 TM	kjhgytr	3
41	현대 싼타페 TM	ssh2525	5
42	현대 싼타페 TM	a71028621	0.5



02 추천시스템 작성
- 코사인 유사도

추천 시스템 작성 – 코사인 유사도

```
In [121]: data = pd.read_csv('encar.csv')
          data = data.drop('Unnamed: 0', axis = 1)
          data.head()
```

Out [121]:

	title	id	rating
0	현대 아반떼 (CN7)	alexius	4.5
1	현대 아반떼 (CN7)	bluebird7788	4.5
2	현대 아반떼 (CN7)	asdfasdf724	4.0
3	현대 아반떼 (CN7)	dudwo6085	2.0
4	현대 아반떼 (CN7)	yod3456	5.0

```
In [122]: data.title.nunique()  # 홈페이지에서 제공하는 차량의 수는 181대이지만 리뷰가 없는 차량이 5대 존재하여 176대만 크롤링.
```

Out [122]: 176

사용자-아이템 평점 행렬로 변환

```
In [123]: # columns='title' 로 title 컬럼으로 pivot 수행.
          ratings_matrix = data.pivot_table('rating', index='id', columns='title')

          # NaN 값을 모두 0 으로 변환
          ratings_matrix = ratings_matrix.fillna(0)
          ratings_matrix
```

Out [123]:

title	BMW 1 시리즈 (F20)	BMW 3 시리즈 (F30)	BMW 5 시리즈 (F10)	BMW 5 시리즈 (G30)	BMW M2 (F87)	BMW Z4 (E89)	기아 K3	기아 K5	기아 K5 2 세대	기아 K5 3 세대	...	현대 올뉴 투싼	현대 제네시스	현대 제네시스 DH	현대 제네시스 쿠페	현대 코나	현대 모나 하이브리드	현대 투스카니	현대 투싼 ix	현대 팰리세이드	혼다 New CR-V
id																					
Baestyle	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
Bestsoul4	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
Bosshuga	0.0	0.0	0.0	3.0	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
Button0308	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
Chdnet	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
zksdk	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
zksvk	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
zpsemf925	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
zq3824	0.0	0.0	3.0	3.0	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
zsw6670	0.0	0.0	0.0	3.0	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

1034 rows × 176 columns

아이템-사용자 평점 행렬로 변환

```
In [124]: # 아이템-사용자 행렬로 transpose 한다.
          ratings_matrix_T = ratings_matrix.transpose()  # 전치 행렬

          print(ratings_matrix_T.shape)
          ratings_matrix_T.head(5)
```

```
(176, 1034)
```

Out [124]:

id	Baestyle	Bestsoul4	Bosshuga	Button0308	Chdnet	Choivs0	DreamLover	GONZA_admin	Giddens	HMCTJ	...	zephyr97	zerocool4u	zizilub	zizi
title															
BMW 1시리즈 (F20)	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	
BMW 3시리즈 (F30)	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	
BMW 5시리즈 (F10)	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	

차량과 차량들 간 코사인 유사도 산출

```
In [125]: from sklearn.metrics.pairwise import cosine_similarity

          car_cos_sim = cosine_similarity(ratings_matrix_T, ratings_matrix_T)
          car_cos_sim
```

```
Out [125]: array([[1.        , 0.13305139, 0.04393368, ..., 0.14405508, 0.        ,
                   0.28077997],
                  [0.13305139, 1.        , 0.06131864, ..., 0.10052949, 0.        ,
                   0.19594357],
                  [0.04393368, 0.06131864, 1.        , ..., 0.06638985, 0.        ,
                   0.16637333],
                  ...,
                  [0.14405508, 0.10052949, 0.06638985, ..., 1.        , 0.        ,
                   0.24245555],
                  [0.        , 0.        , 0.        , ..., 0.        , 1.        ,
                   0.01484798],
                  [0.28077997, 0.19594357, 0.16637333, ..., 0.24245555, 0.01484798,
                   1.        ]])
```

```
# cosine_similarity() 로 반환된 넘파이 행렬을 차량명과 매핑하여 DataFrame으로 변환
# array -> dataframe
car_cos_sim_df = pd.DataFrame(data=car_cos_sim, index=ratings_matrix.columns,
                              columns=ratings_matrix.columns)

print(car_cos_sim_df.shape)
car_cos_sim_df.head(5)
```

title	BMW 1 시리즈 (F20)	BMW 3 시리즈 (F30)	BMW 5 시리즈 (F10)	BMW 5 시리즈 (G30)	BMW M2 (F87)	BMW Z4 (E89)	기아 K3	기아 K5	기아 K5 2세대	기아 K5 3 세대	...	현대 올뉴 투싼	현대 제네시스	현대 제네시스 DH	현대 제네시스 쿠페	현대 코나	현
title																	
BMW 1시리즈 (F20)	1.000000	0.133051	0.043934	0.160594	0.0	0.079064	0.231340	0.130330	0.174178	0.0	...	0.080694	0.117741	0.129002	0.000000	0.095928	0.
BMW 3시리즈 (F30)	0.133051	1.000000	0.061319	0.116147	0.0	0.073567	0.161441	0.090951	0.121551	0.0	...	0.056313	0.082166	0.090024	0.087890	0.066944	0.
BMW																	

```
# 포르쉐 911 와(과) 유사한 차량 6대 확인해보기 / 내림차순
car_cos_sim_df["포르쉐 911"].sort_values(ascending=False)[:6]
```

```
title
포르쉐 911                1.000000
포르쉐 마칸                0.343256
벤츠 GLC-클래스 X253       0.286714
볼보 V40 크로스컨트리      0.279618
르노삼성 QM5               0.279618
혼다 New CR-V              0.254284
Name: 포르쉐 911, dtype: float64
```


title
포르쉐 911 1.000000
포르쉐 마칸 0.343256
벤츠 GLC-클래스 X253 0.286714
볼보 V40 크로스컨트리 0.279618
르노삼성 QM5 0.279618
혼다 New CR-V 0.254284
Name: 포르쉐 911, dtype: float64

03 CF-KNN / CF-MF

CF-KNN : 아이템 기반 인접 이웃 협업 필터링

CF-KNN : 아이템 기반 인접 이웃 협업 필터링으로 개인화된 차량 추천 (가중 예측평점)

```
# 평점 벡터(행 벡터)와 유사도 벡터(열 벡터)를 내적(dot)해서 예측 평점을 계산하는 함수 정의
def predict_rating(ratings_arr, car_cos_sim_arr):
    ratings_pred = ratings_arr.dot(car_cos_sim_arr)/ np.array([np.abs(car_cos_sim_arr).sum(axis=1)])
    return ratings_pred
```

```
# ratings_matrix.values : 사용자-아이템 평점 행렬
# car_cos_sim_df.values : 코사인 유사도 데이터프레임
ratings_pred = predict_rating(ratings_matrix.values , car_cos_sim_df.values) # 사용자-아이템 평점 행렬 * 코사인유사도값
ratings_pred
```

```
array([[0.04215487, 0.05886277, 0.02421391, ..., 0.02800368, 0.        ,
        0.04306893],
       [0.02498555, 0.02325898, 0.        , ..., 0.        , 0.        ,
        0.02188056],
       [0.02741489, 0.02644845, 0.02576816, ..., 0.00993373, 0.        ,
        0.01527782],
       ...,
       [0.03613206, 0.02501476, 0.01567265, ..., 0.01812561, 0.        ,
        0.02751467],
       [0.11177925, 0.13751922, 0.51121826, ..., 0.05272336, 0.        ,
        0.08546571],
       [0.02741489, 0.02644845, 0.02576816, ..., 0.00993373, 0.        ,
        0.01527782]])
```

```
# 데이터프레임으로 변환
ratings_pred_df = pd.DataFrame(data=ratings_pred, index= ratings_matrix.index,
                               columns = ratings_matrix.columns)
print(ratings_pred_df.shape)
ratings_pred_df.head(5)
```

```
(1034, 176)
```

title	BMW 1 시리즈 (F20)	BMW 3 시리즈 (F30)	BMW 5 시리즈 (F10)	BMW 5 시리즈 (G30)	BMW M2 (F87)	BMW Z4 (E89)	기아 K3	기아 K5	기아 K5 2세대	기아 K5 3세대	...	현대 올 뉴 투싼	현대 제네시스	현대 제네시스 DH	현대 제네시스 쿠페	현
id																
Baestyle	0.042155	0.058863	0.024214	0.046945	0.0	0.045993	0.032969	0.063489	0.034072	0.000000	...	0.025843	0.024985	0.023912	0.000000	0.01
Bestsoul4	0.024986	0.023259	0.000000	0.017574	0.0	0.000000	0.000000	0.000000	0.018849	0.000000	...	0.030635	0.000000	0.000000	0.000000	0.00
Bosshuga	0.027415	0.026448	0.025768	0.261334	0.0	0.014502	0.011695	0.010510	0.017727	0.000000	...	0.018335	0.008863	0.008482	0.063746	0.01
Button0308	0.030056	0.027979	0.017264	0.031709	0.0	0.021861	0.023506	0.021124	0.051825	0.027847	...	0.036851	0.017814	0.017049	0.000000	0.02
Chdnet	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.00

5 rows × 176 columns

예측 평점 정확도를 판단하기 위해 오차 함수인 **RMSE**를 이용

```
from sklearn.metrics import mean_squared_error

# 사용자가 평점을 부여한 차량에 대해서만 예측 성능 평가 MSE 를 구함.
def get_mse(pred, actual):
    # Ignore nonzero terms.
    pred = pred[actual.nonzero()].flatten()
    actual = actual[actual.nonzero()].flatten()
    return mean_squared_error(pred, actual)

print('아이템 기반 모든 인접 이웃 MSE: ', get_mse(ratings_pred, ratings_matrix.values ))
```

```
아이템 기반 모든 인접 이웃 MSE:  10.10572699658128
```

차량 추천

```
# 임의의 사람(encar.csv 속 id)에게 차량을 추천해보자  ( 임의의 사람 : kjhgytr )
# 추천에 앞서 임의의 사람이 높은 평점을 준 차량을 확인해보면
user_rating_id = ratings_matrix.loc['kjhgytr', :]     # kjhgytr <- 이 사람이 남긴 평점 확인
user_rating_id[ user_rating_id > 0].sort_values(ascending=False)[:10]    # 0보다 크다는 건 평점을 줬다는 것(고의로 0점 준 것 제외

#무려히 kjhgytr <- 이 사람은 리뷰를 많이 작성했으므로 10개의 결과가 나오지만, 1~2개의 리뷰를 남긴 사람은 결과가 적다
```

```
title
현대 제네시스 DH          5.0
쉐보레(GM대우) 트레일블레이저    4.5
현대 투스카니             4.0
현대 베라크루즈            4.0
현대 아슬란              4.0
현대 쏘나타 더 브릴리언트       3.0
현대 싼타페 DM            3.0
현대 싼타페 TM            3.0
현대 싼타페 더 프라임         3.0
현대 쏘나타 (DN8)         3.0
Name: kjhgytr, dtype: float64
```

⇨ 'kjhgytr' 라는 사람이 평점을 남긴 차량 중 top10을 보여준다.

CF-KNN : 아이템 기반 인접 이웃 협업 필터링

```
def get_unview_car(ratings_matrix, userId):
    # userId로 입력받은 사용자의 모든 차량정보 추출하여 Series로 반환함.
    # 반환된 user_rating 은 차량명(title)을 index로 가지는 Series 객체임.
    user_rating = ratings_matrix.loc[userId,:]

    # user_rating이 0보다 크면 기존에 리뷰한 차량임. 대상 index를 추출하여 list 객체로 만듬
    already_viewed = user_rating[ user_rating > 0].index.tolist()

    # 모든 차량명을 list 객체로 만듬.
    car_list = ratings_matrix.columns.tolist()

    # list comprehension으로 already_viewed에 해당하는 차량은 car_list에서 제외함.
    unview_list = [ car for car in car_list if car not in already_viewed]

    return unview_list
```

```
# pred_df : 앞서 계산된 차량별 예측 평점
# unview_list : 사용자가 보지 않은 차량들
# top_n : 상위 n개를 가져온다.

def recomm_car_by_userid(pred_df, userId, unview_list, top_n=10):
    # 예측 평점 DataFrame에서 사용자id index와 unview_list로 들어온 차량명 컬럼을 추출하여
    # 가장 예측 평점이 높은 순으로 정렬함.
    recomm_car = pred_df.loc[userId, unview_list].sort_values(ascending=False)[:top_n]
    return recomm_car
```

```
# 사용자가 보지 않은 차량명 추출
unview_list = get_unview_car(ratings_matrix, 'kjhgytr' )

# 아이템 기반의 인접 이웃 협업 필터링으로 차량 추천
recomm_car = recomm_car_by_userid(ratings_pred_df, 'kjhgytr' , unview_list, top_n=10)

# 평점 데이터를 DataFrame으로 생성.
recomm_car_df = pd.DataFrame(data=recomm_car.values,index=recomm_car.index, columns=['pred_score'])
recomm_car_df
```

title	pred_score
기아 스포티지 더 볼드	0.898578
르노삼성 더 뉴 SM6	0.822791
쉐보레(GM대우) 콜로라도	0.801462
쉐보레(GM대우) 트래버스	0.764515
현대 더 뉴 아반떼 AD	0.759467
쉐보레(GM대우) 올 뉴 크루즈	0.757596
현대 그랜저 뉴 럭셔리	0.752694
제네시스 G80	0.731140
현대 아반떼 (CN7)	0.728666
르노삼성 SM5	0.716319

평점을 남기지 않은
차량 중에서 추천

CF-MF : 행렬 분해 기반의 잠재 요인 협업 필터링

CF-MF : 행렬 분해 기반의 잠재 요인 협업필터링

```
import numpy as np
from sklearn.metrics import mean_squared_error
from tqdm import tqdm_notebook

def get_rmse(R, P, Q, non_zeros):
    error = 0
    # 두개의 분해된 행렬 P와 Q.T의 내적 곱으로 예측 R 행렬 생성
    full_pred_matrix = np.dot(P, Q.T)

    # 실제 R 행렬에서 널이 아닌 값의 위치 인덱스 추출하여 실제 R 행렬과 예측 행렬의 RMSE 추출
    x_non_zero_ind = [non_zero[0] for non_zero in non_zeros]
    y_non_zero_ind = [non_zero[1] for non_zero in non_zeros]
    R_non_zeros = R[x_non_zero_ind, y_non_zero_ind]

    full_pred_matrix_non_zeros = full_pred_matrix[x_non_zero_ind, y_non_zero_ind]

    mse = mean_squared_error(R_non_zeros, full_pred_matrix_non_zeros)
    rmse = np.sqrt(mse)

    return rmse
```

```
def matrix_factorization(R, K, steps=200, learning_rate=0.01, r_lambda = 0.01):
    num_users, num_items = R.shape
    # P와 Q 매트릭스의 크기를 지정하고 정규분포를 가진 랜덤한 값으로 입력합니다.
    np.random.seed(1)
    P = np.random.normal(scale=1./K, size=(num_users, K))
    Q = np.random.normal(scale=1./K, size=(num_items, K))

    break_count = 0

    # R > 0 인 행 위치, 열 위치, 값을 non_zeros 리스트 객체에 저장.
    non_zeros = [ (i, j, R[i,j]) for i in range(num_users) for j in range(num_items) if R[i,j] > 0 ]

    # P와 Q 매트릭스를 계속 업데이트(확률적 경사하강법)
    for step in tqdm_notebook(range(steps)):
        for i, j, r in non_zeros:
            # 실제 값과 예측 값의 차이인 오류 값 구함
            eij = r - np.dot(P[i, :], Q[j, :].T)

            # Regularization을 반영한 SGD 업데이트 공식 적용
            P[i,:] = P[i,:] + learning_rate*(eij * Q[j, :] - r_lambda*P[i,:])
            Q[j,:] = Q[j,:] + learning_rate*(eij * P[i, :] - r_lambda*Q[j,:])

        rmse = get_rmse(R, P, Q, non_zeros)
        if (step % 10) == 0 :
            print("### iteration step : ", step," rmse : ", rmse)

    return P, Q
```

```
%%time
# 경사하강법을 이용한 행렬 분해
P, Q = matrix_factorization(ratings_matrix.values, K=50, steps=200, learning_rate=0.01, r_lambda = 0.01)

pred_matrix = np.dot(P, Q.T)
```

```
100%|##########| 200/200 [00:04<00:00, 45.42it/s]

### iteration step :  0  rmse :  3.9549852302060553
### iteration step :  10  rmse :  3.3762035690444345
### iteration step :  20  rmse :  1.8725140240742757
### iteration step :  30  rmse :  0.9106390512711298
### iteration step :  40  rmse :  0.4679322765540731
### iteration step :  50  rmse :  0.2570237878910771
### iteration step :  60  rmse :  0.15165518809117562
### iteration step :  70  rmse :  0.10208556738856547
### iteration step :  80  rmse :  0.07513773561489367
### iteration step :  90  rmse :  0.058497724692134115
### iteration step :  100  rmse :  0.04758957680840349
### iteration step :  110  rmse :  0.040241689101181 8
### iteration step :  120  rmse :  0.035216380147600 4
### iteration step :  130  rmse :  0.031737863953460 31
### iteration step :  140  rmse :  0.02929926420999 14
### iteration step :  150  rmse :  0.027564334273450012
### iteration step :  160  rmse :  0.026308972884995372
### iteration step :  170  rmse :  0.02538343763718825
### iteration step :  180  rmse :  0.024687251989397998
### iteration step :  190  rmse :  0.024152534258532355
Wall time: 4.49 s
```

CF-MF : 행렬 분해 기반의 잠재 요인 협업 필터링 / 추천 비교

```
ratings_pred_MF_df = pd.DataFrame(data=pred_matrix, index= ratings_matrix.index,
                                  columns = ratings_matrix.columns)
# 예측 평점 행렬 확인
print(ratings_pred_MF_df.shape)
ratings_pred_MF_df.head(15)
```

Chdnet	1.487630	1.312089	0.886532	0.159268	-0.093391	0.055387	0.742498	0.212957	1.090198	-0.084502	...	0.936868	0.801212	0.732619	1.2
Choivs0	4.502943	3.134478	2.138671	2.362597	0.439741	2.107464	2.916117	2.878755	3.482449	2.199450	...	3.252067	3.770623	3.749677	3.0
DreamLover	2.213136	1.734225	0.574689	1.212889	0.124135	1.144842	1.203524	1.622762	1.782559	1.013565	...	1.606706	1.941069	1.801760	1.4
GONZA_admin	4.502847	3.783890	3.123512	3.611836	-0.678811	2.180256	3.338948	2.631550	3.697018	2.856598	...	3.257231	3.826563	3.639568	3.7
Giddens	4.086842	4.446160	3.503779	2.264983	0.345781	2.961221	3.256926	3.454190	3.210301	3.635958	...	3.519057	4.184052	4.420560	3.0
HMCTJ	3.575205	3.616642	1.874744	1.696827	0.672836	2.136681	2.638162	2.950730	3.015107	2.578921	...	2.991474	3.637644	3.427373	2.7
Khy112973	2.993095	2.721505	1.628797	1.101579	0.649195	2.250164	2.499041	2.708782	3.515107	3.210147	...	3.261970	3.582719	3.048945	2.2
Kijek1972	2.003605	1.613195	1.316250	1.400797	0.278416	0.935075	1.421731	1.517082	1.844720	1.235090	...	1.910015	1.848653	1.815848	1.8
Lee201612	0.286504	0.241211	0.241486	0.198353	0.176444	0.245760	0.221073	0.243878	0.336292	0.212086	...	0.402624	0.307591	0.260023	0.2
Real3oN	3.290848	2.916895	2.533341	0.732538	0.768182	1.728889	2.641723	2.272412	3.521771	4.978442	...	1.542912	3.391525	3.375716	2.0
Sch7799	0.970031	0.888654	0.420847	0.486704	-0.029221	0.596733	0.697454	0.755073	0.826532	0.704179	...	0.878869	0.995268	0.898078	0.6

15 rows × 176 columns

```
# 사용자가 보지 않은 차량명 추출
unview_list = get_unview_car(ratings_matrix, 'kjhgytr' )

# 아이템 기반의 인접 이웃 협업 필터링으로 차량 추천
recomm_car = recomm_car_by_userid(ratings_pred_MF_df, 'kjhgytr' , unview_list, top_n=10)

# 평점 데이터를 DataFrame으로 생성.
recomm_car_df = pd.DataFrame(data=recomm_car.values,index=recomm_car.index, columns=['pred_score'])
recomm_car_df
```

title	pred_score
인피니티 Q50	3.334255
BMW 1시리즈 (F20)	3.295433
현대 LF 쏘나타 하이브리드	3.133025
기아 K7	3.093997
벤츠 C-클래스 W205	3.081868
기아 K5 2세대	3.053732
현대 NF 쏘나타 트랜스폼	3.030800
현대 투싼 ix	2.957858
기아 니로	2.947205
쌍용 G4 렉스턴	2.934259

CF-KNN 기반 차량 추천

title	pred_score
기아 스포티지 더 볼드	0.898578
르노삼성 더 뉴 SM6	0.822791
쉐보레(GM대우) 콜로라도	0.801462
쉐보레(GM대우) 트래버스	0.764515
현대 더 뉴 아반떼 AD	0.759467
쉐보레(GM대우) 올 뉴 크루즈	0.757596
현대 그랜저 뉴 럭셔리	0.752694
제네시스 G80	0.731140
현대 아반떼 (CN7)	0.728666
르노삼성 SM5	0.716319

VS

CF-MF 기반 차량 추천

title	pred_score
인피니티 Q50	3.334255
BMW 1시리즈 (F20)	3.295433
현대 LF 쏘나타 하이브리드	3.133025
기아 K7	3.093997
벤츠 C-클래스 W205	3.081868
기아 K5 2세대	3.053732
현대 NF 쏘나타 트랜스폼	3.030800
현대 투싼 ix	2.957858
기아 니로	2.947205
쌍용 G4 렉스턴	2.934259

04 surprise 패키지

surprise 패키지를 사용한 추천

surprise 패키지 : SVD()

```
import surprise
from surprise import SVD       # 행렬 분해 알고리즘
from surprise import Dataset
from surprise import Reader
from surprise import accuracy
from surprise.model_selection import train_test_split
from surprise.dataset import DatasetAutoFolds
```

```
# column 순서 맞추기
data2 = data[['id','title','rating']]
```

```
data2.to_csv('encar2.csv', encoding = 'utf-8-sig')
```

```
data2 = pd.read_csv('encar2.csv')

data2 = data2.drop('Unnamed: 0', axis=1)

data2
```

	id	title	rating
0	alexius	현대 아반떼 (CN7)	4.5
1	bluebird7788	현대 아반떼 (CN7)	4.5
2	asdfasdf724	현대 아반떼 (CN7)	4.0
3	dudwo6085	현대 아반떼 (CN7)	2.0
4	ycd3456	현대 아반떼 (CN7)	5.0
...	...	...	...
1898	ktty982	쌍용 렉스턴 W	3.0
1899	dewung	쌍용 렉스턴 W	3.5
1900	drth0070	쌍용 렉스턴 W	0.5
1901	dudgksx1	쌍용 렉스턴 W	0.5
1902	yahoo32	쌍용 렉스턴 W	0.5

1903 rows × 3 columns

```
%%time
# 데이터 로드하기
reader = Reader(rating_scale=(0.0, 5.0))
# ratings DataFrame 에서 컬럼은 userId, itemId, rating 순서를 지켜야 합니다.
data2_reader = Dataset.load_from_df(data2[['id', 'title', 'rating']], reader)

# 데이터를 학습셋, 테스트셋으로 나누기
trainset, testset = train_test_split(data2_reader, test_size=.25, random_state=0)

# 행렬 분해 알고리즘 학습 및 예측, 평가(RMSE)
model_svd = SVD(n_factors=50, random_state=0)
model_svd.fit(trainset)
predictions = model_svd.test( testset )
accuracy.rmse(predictions)
```

```
RMSE: 1.3015
Wall time: 41.5 ms

1.3014755410407182
```

Column의 순서를 맞추고 header = False / Index = False의 옵션을 줘야 하지만, 제거를 하지 않았을 때의 결과가 유의미한 차이가 없어 제거하지 않았음.

제거를 하면 0행이 column으로 올라가서 행이 하나 줄어든다.

surprise 패키지를 사용한 추천

Cross Validation(교차 검증)과 GridSearchCV(하이퍼 파라미터 튜닝)

```
%%time
from surprise.model_selection import cross_validate

reader = Reader(rating_scale=(0.0, 5.0))
data2_reader = Dataset.load_from_df(data2[['id', 'title', 'rating']], reader)
model_svd = SVD(n_factors=50, random_state=0)

# 교차 검증
cross_validate(model_svd, data2_reader, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.2513  1.2444  1.3009  1.3032  1.1871  1.2574  0.0427
MAE (testset)     0.9515  0.9632  1.0144  0.9766  0.9387  0.9689  0.0260
Fit time          0.05    0.05    0.05    0.05    0.05    0.05    0.00
Test time         0.00    0.00    0.00    0.00    0.00    0.00    0.00
Wall time: 259 ms

{'test_rmse': array([1.25127689, 1.24444027, 1.3008664 , 1.30320272, 1.1871402 ]),
 'test_mae': array([0.9515351 , 0.96318195, 1.01436888, 0.97664917, 0.93874352]),
 'fit_time': (0.05053771151489258,
  0.04824352264404297,
  0.046602725982666016,
  0.04524374008178711,
  0.04622960090637207),
 'test_time': (0.0010013580322265625,
  0.0010843276977539062,
  0.0009670257568359375,
  0.001916646957397461,
  0.0019667148590087891)}
```

그리드 서치 CV 이용

```
%%time
from surprise.model_selection import GridSearchCV

# 최적화할 파라미터들을 딕셔너리 형태로 지정.
param_grid = {'n_epochs': [20, 40, 60, 70, 80], 'n_factors': [50, 100, 200] }

# GridSearchCV 세팅 : CV를 3개 폴드 세트로 지정, 성능 평가는 rmse, mse 로 수행 하도록 GridSearchCV 구성
gs = GridSearchCV(SVD, param_grid, measures=['rmse', 'mae'], cv=5)
gs

gs.fit(data2_reader)

# 최고 RMSE Evaluation 점수와 그때의 하이퍼 파라미터
print(gs.best_score['rmse'])
print(gs.best_params['rmse'])
```

```
1.2381513966012978
{'n_epochs': 60, 'n_factors': 50}
Wall time: 17.1 s
```

Surprise 를 이용한 개인화 차량 추천 시스템

전체 데이터로 학습 진행(학습셋, 테스트셋 나누지 않고)

```
%%time
from surprise.dataset import DatasetAutoFolds

reader = Reader(line_format='user item rating', sep=',', rating_scale=(0.0, 5.0))

data_folds = DatasetAutoFolds(df = data2, reader=reader)   #df를 줘도 됨.

# 전체 데이터를 학습데이터로 생성함.
trainset = data_folds.build_full_trainset()

# SVD 클래스타입으로 추천모델 학습(하이퍼 파라미터는 앞서 그리드서치로 구한 것들)
model_svd = SVD(n_epochs=60, n_factors=50, random_state=0)

model_svd.fit(trainset)
```

```
Wall time: 159 ms

<surprise.prediction_algorithms.matrix_factorization.SVD at 0x1cf6cf5e9d0>
```

```
# predict 메소드를 사용해서 예측 평점 구하기

#사용자 아이디(닉네임)
uid = 'kjhgytr'

# 차량 이름
iid = '현대 투스카니'

recom_result = model_svd.predict(uid, iid, verbose=True)        #r_ui : 실제 평점
```

```
user: kjhgytr    item: 현대 투스카니    r_ui = None    est = 3.60    {'was_impossible': False}
```

다른 아이디 / 차량 입력

```
#사용자 아이디(닉네임)
uid = 'GONZA_admin'

# 차량 이름
iid = '포르쉐 718 박스터'

recom_result = model_svd.predict(uid, iid, verbose=True)        #r_ui : 실제 평점
```

```
user: GONZA_admin item: 포르쉐 718 박스터 r_ui = None    est = 4.93    {'was_impossible': False}
```

설계 및 진행과정, 고민 과정

추천시스템 만들기 - Encar 차량(중고차) 추천

22.02.09

1. 데이터 수집(크롤링) : Encar 중고차 평점 및 닉네임 수집
 - 유저 아이디
 - 차량 아이디 (차량이름)
 - 평점 (별점 - 숫자로 환산필요)

2. 코사인 유사도 + CF-KNN (아이템 기반 인접 이웃 협업 필터링)

3. 유사도 높은 순으로 출력

22.02.10

1. CF-MF 적용 - 경사하강법 / 추천

2. Surprise 패키지 - RMSE / CV / GridSearch / 닉네임과 차량명을 기입하면 예측 평점 출력

3. ppt 제작

※ 중간중간 막혔던 부분

1. 크롤링 - 별점
 - 크롤링 과정에서 별이 텍스트가 아닌 이미지.
 - style에 있는 'width = 80%;'를 이용해 별점 추출 : get_attribute('style')
 - split과 인덱싱을 활용
 - 80을 추출했으면 0.05를 곱해줘서 5점만점으로 만들기

2. 셀레늄 - 별점
 - 10개의 별점을 크롤링하고 싶어서 셀레늄을 활용하면 20개의 셀레늄 값이 나왔다.
 - 하나씩 text해보니 0,2,4,6 ... 번째는 '' / 1,3,5,7 번째가 별점
 - 홀수번째 값만 추출

3. surprise 패키지
 - header와 index를 지우고 csv를 저장하면 행 하나가 header로 바뀌므로 행 하나 사라짐
 - 하지만 예측력은 올라감


Thank you
코드 분석 : https://github.com/SubinKim22/project