무슨 와인2조

와인 추천 시스템

Like Wine Project

강상범, 김예린, 박경희,
박수진, 서태원, 조희정

멋쟁이 사자처럼 K-Digital Training AI스쿨

CONTENTS

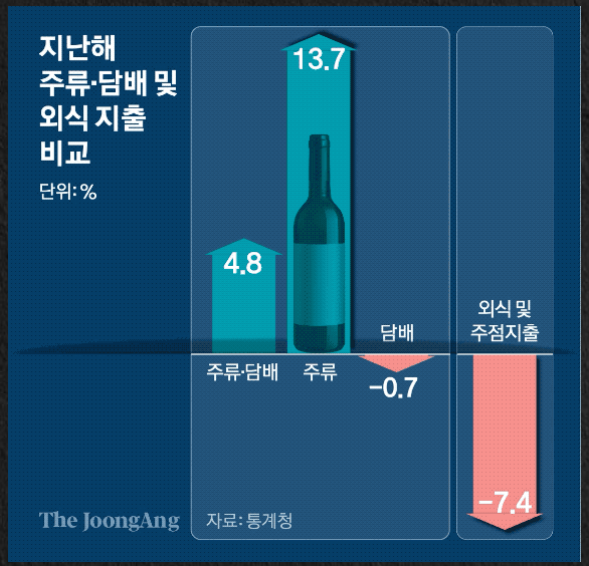
목차

01

프로젝트 소개

Overview

프로젝트 소개



코로나 19 상황 속 다른 분야와 달리 **성장세 유지**,

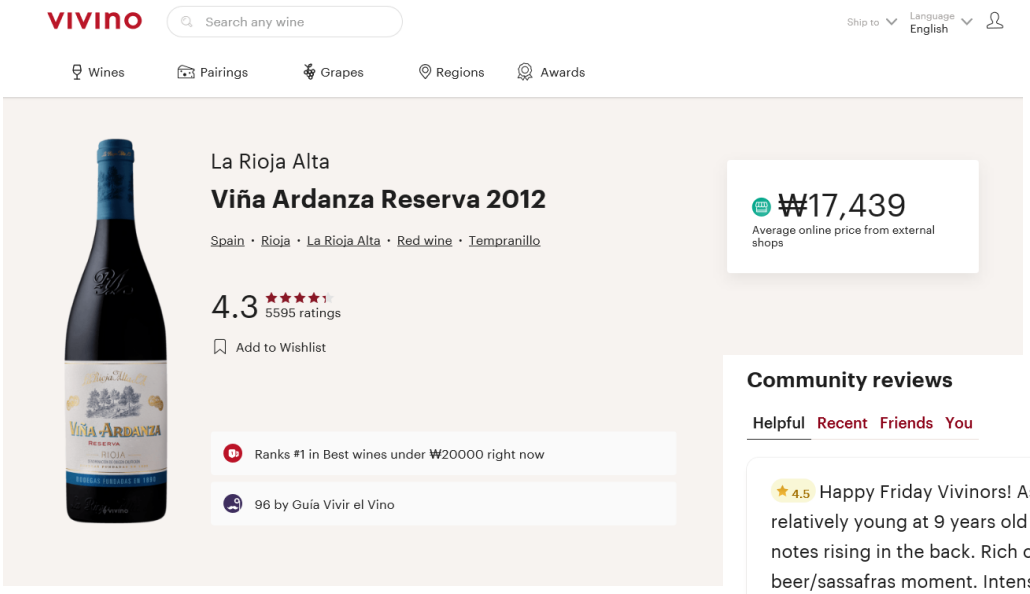
집콕족 등의 요인으로 **비대면 소비 증대**

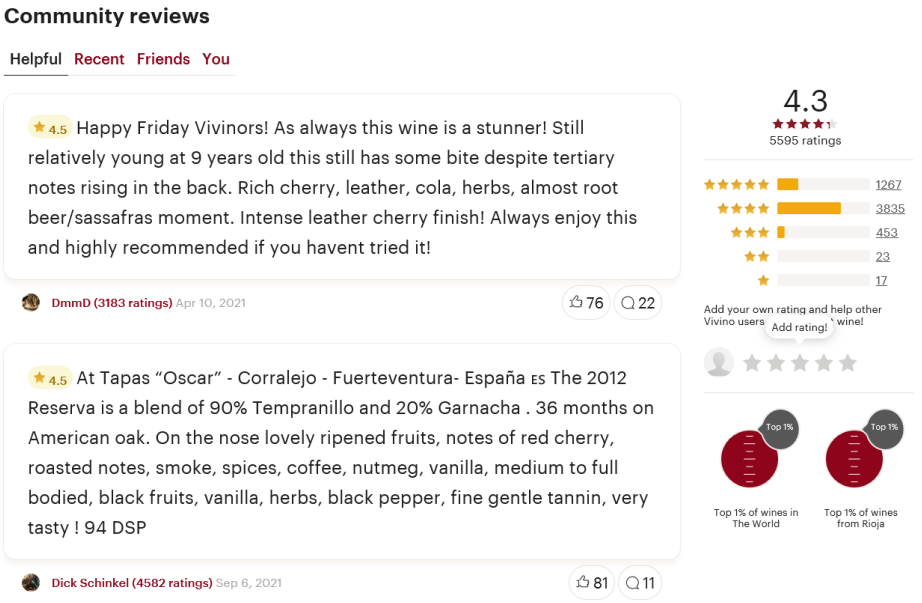
02

데이터 소개/탐색/시각화

Overview

데이터 소개_VIVINO web





✓ 다수의 와인 정보를 얻을 수 있는
VIVINO web을 선택

데이터 수집

VIVINO web 크롤링

```
web = driver.page_source
source = BeautifulSoup(web, 'html.parser')
```

```
url_list = source.find_all('a', {'class':'anchor__anchor--3DOSm wineCard__c
```

```
len(url_list)
```

```
url_list[0]['href']
```

```
'/gitana-lupi-rezerva/w/1468452?year=2016&price_id=23484005'
```

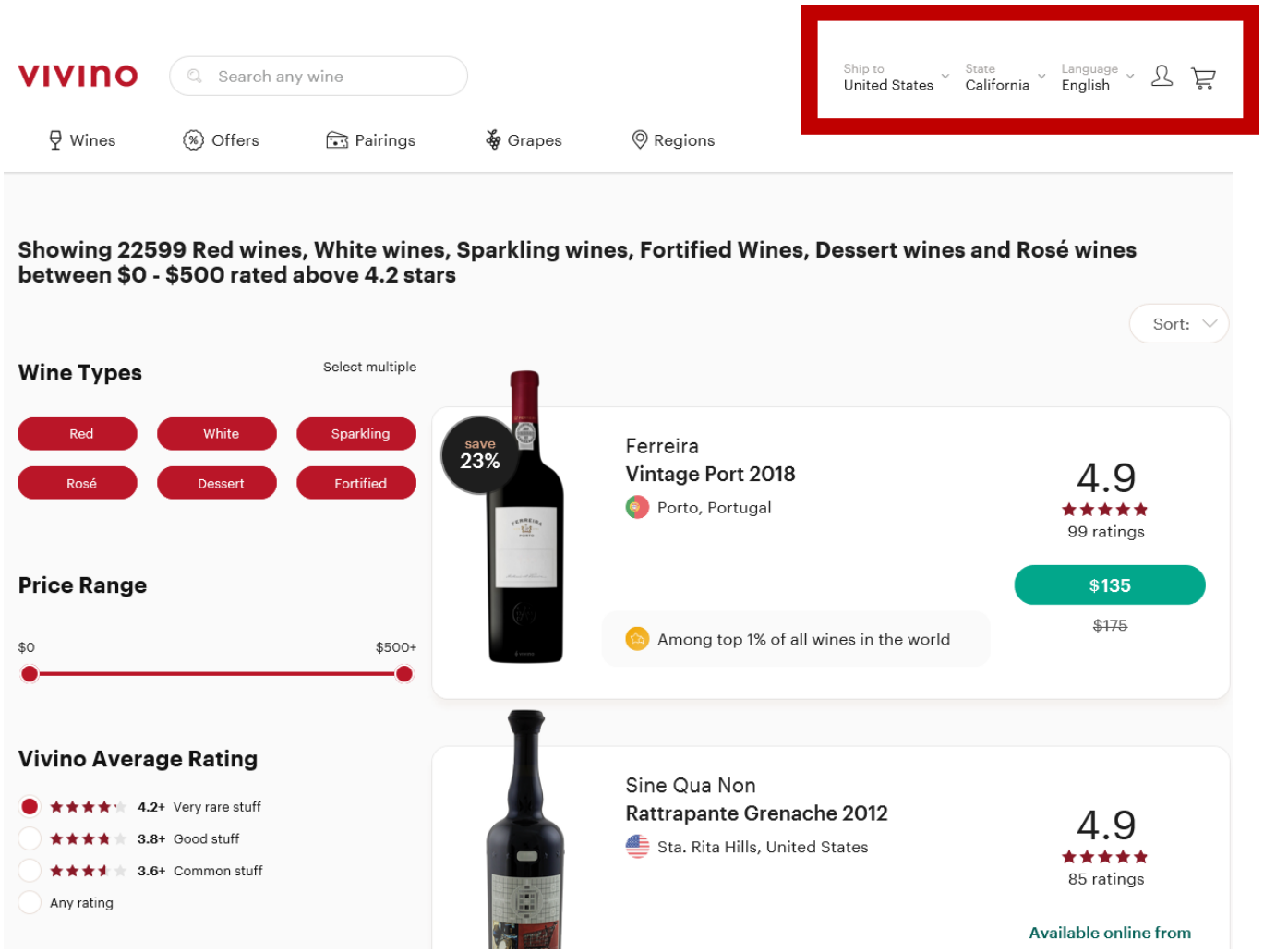
```
wine_url=[]
for i in range(len(url_list)):
    wine_url.append('https://www.vivino.com/US-CA/en' + url_list[i]['href'])
#wine_url
```

```
driver.close()
driver.quit()
```

더 많은 데이터 확보를 위해
특정 나라의 URL 사용

데이터 수집

VIVINO web 크롤링



데이터 수집

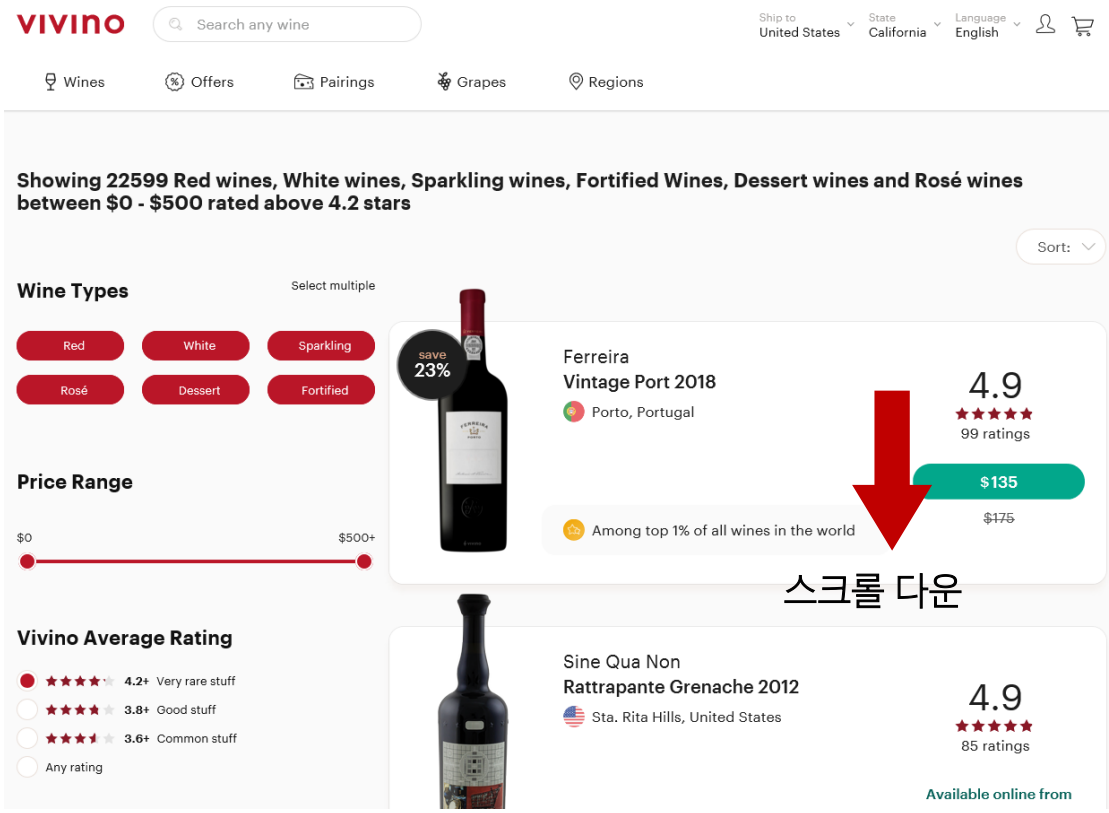
VIVINO web 크롤링

```
scroll_pause_time = 2
```

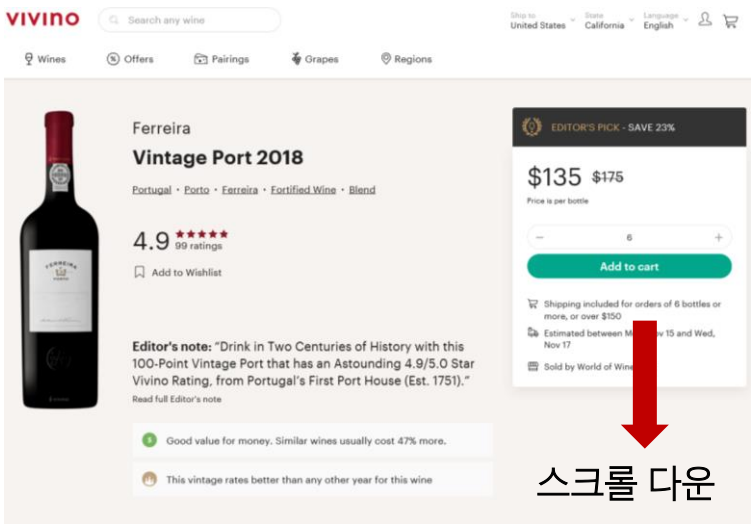
```
last_height = driver.execute_script("return document.body.scrollHeight")

while True:
    # 끝까지 스크롤 다운
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")

    # 1초 대기
    time.sleep(scroll_pause_time )

    # 스크롤 다운 후 스크롤 높이 다시 가져옴
    new_height = driver.execute_script("return document.body.scrollHeight")
    if new_height == last_height:
        break
    last_height = new_height

```

데이터 수집

VIVINO web 크롤링



조건에 맞는 와인 URL 확보



확보된 URL들을 통해 각 와인의 정보 확보

데이터 수집

VIVINO web 크롤링

```
count = 1

for url in red_url[6000:]: #너무 많아서 1000개씩 쪼개주면서 함
      web = requests.get(url, headers=headers).content
      source = BeautifulSoup(web, 'html.parser')
      try:
        for idx, script in enumerate(source.find_all('script')):
          if 'alcohol' in str(script):
            num=idx
        data = source.find_all('script')[num]
        data = data.get_text()
        data = data.split('vintagePageInformation = ')[1].split(';\n  window.__PRE
        data = json.loads(data)

      except:
        continue

      wine_id.append(url.split('id=')[1])
      name.append(source.find('span',{'class':'vintage'}).get_text().replace("\n",
      try:
        if 'price' in data:
            price.append(data['price']['amount'])
        else:
            price.append(data['prices_and_availability']['availability']['median']
      except:
        price.append("")



      score.append(data['vintage']['statistics']['ratings_average'])
      winery.append(data['vintage']['wine']['winery']['name'])

      if data['vintage']['wine']['grapes']!=[]:
          grapes.append(data['vintage']['wine']['grapes'][0]['name'])
      else:
          grapes.append("")
      try:
        country.append(data['vintage']['wine']['region']['country']['name'])
        region.append(data['vintage']['wine']['region']['name'])
      except:
        country.append("")
        region.append("")
          try:
      #if ('alcohol' in str(data['vintage']['wine_facts'])) & (data['vintage']['w
          alcohol.append(data['vintage']['wine_facts']['alcohol'])
      except:
      #else:
          alcohol.append("")

      food = []
      if 'foods' in str(data['vintage']['wine']):
          for i in range(len(data['vintage']['wine']['foods'])):
              food.append(data['vintage']['wine']['foods'][i]['name'])
          foods.append(food)
      else:
          foods.append("")
      print(count)
      count+=1
```

데이터 수집

전체 데이터

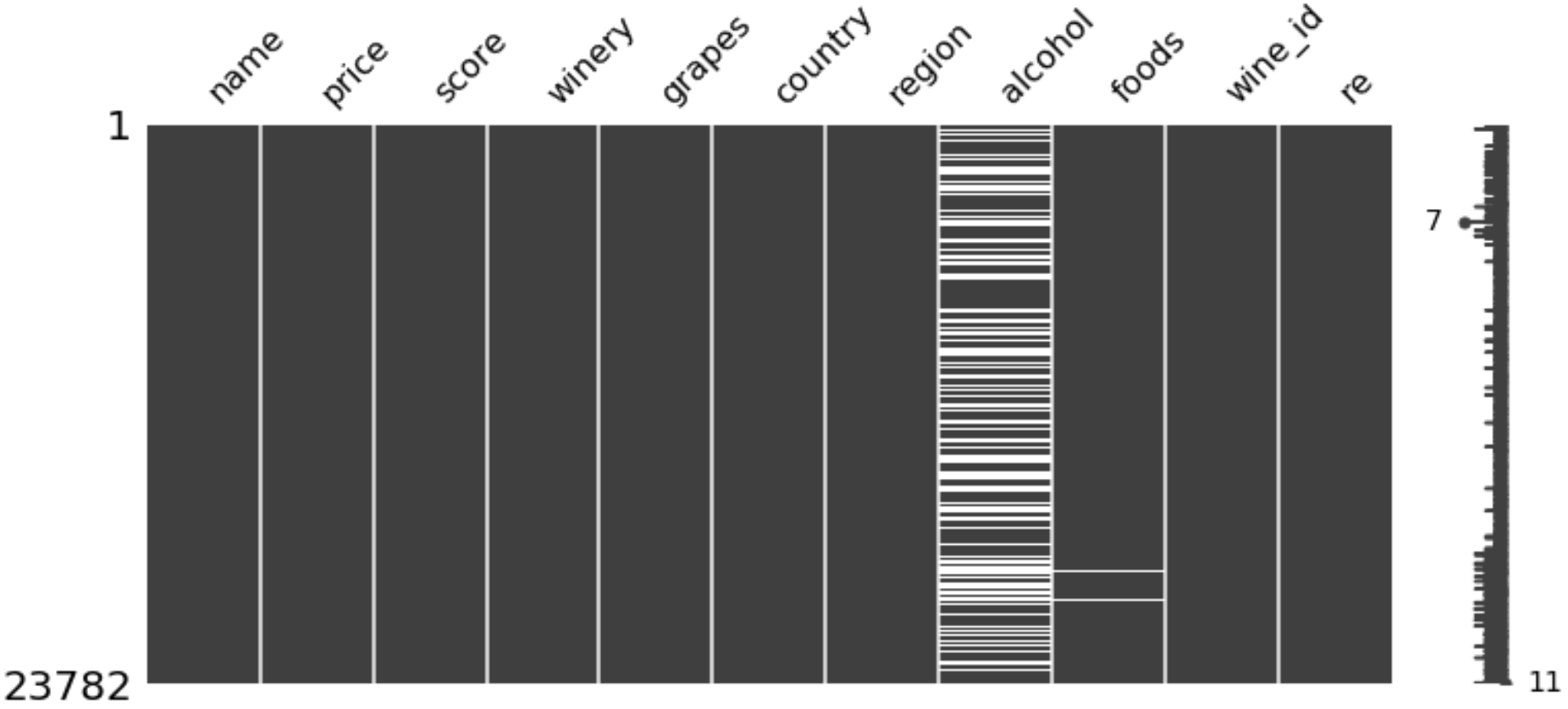
```
In [4]: 1 wine_df
```

	name	price	score	winery	grapes	country	region	alcohol	foods	wine_id	review
	Rim Rock Vineyard Syrah 2017	52.99	4.3	Piedrasassi	Shiraz/Syrah	United States	Arroyo Grande Valley	13.5	['Beef', 'Lamb', 'Game (deer, venison)', 'Poul...	19603399	['Dry tingly cool bottle pairs with seafood ho...
	Tradition Ch창 teauneuf-du-Pape 2015	54.99	4.2	Domaine Giraud	Grenache	France	Ch창 teauneuf-du-Pape	NaN	['Lamb', 'Pork', 'Poultry']	18191466	['Black cherry, oak. Leather, earthy. Great b...
	Estate Cabernet Sauvignon 2013	73.95	4.3	Brandlin	Cabernet Sauvignon	United States	Mount Veeder	NaN	['Beef', 'Lamb', 'Game (deer, venison)',	13886692	['Wine walk favorite! This estate cab is fant...

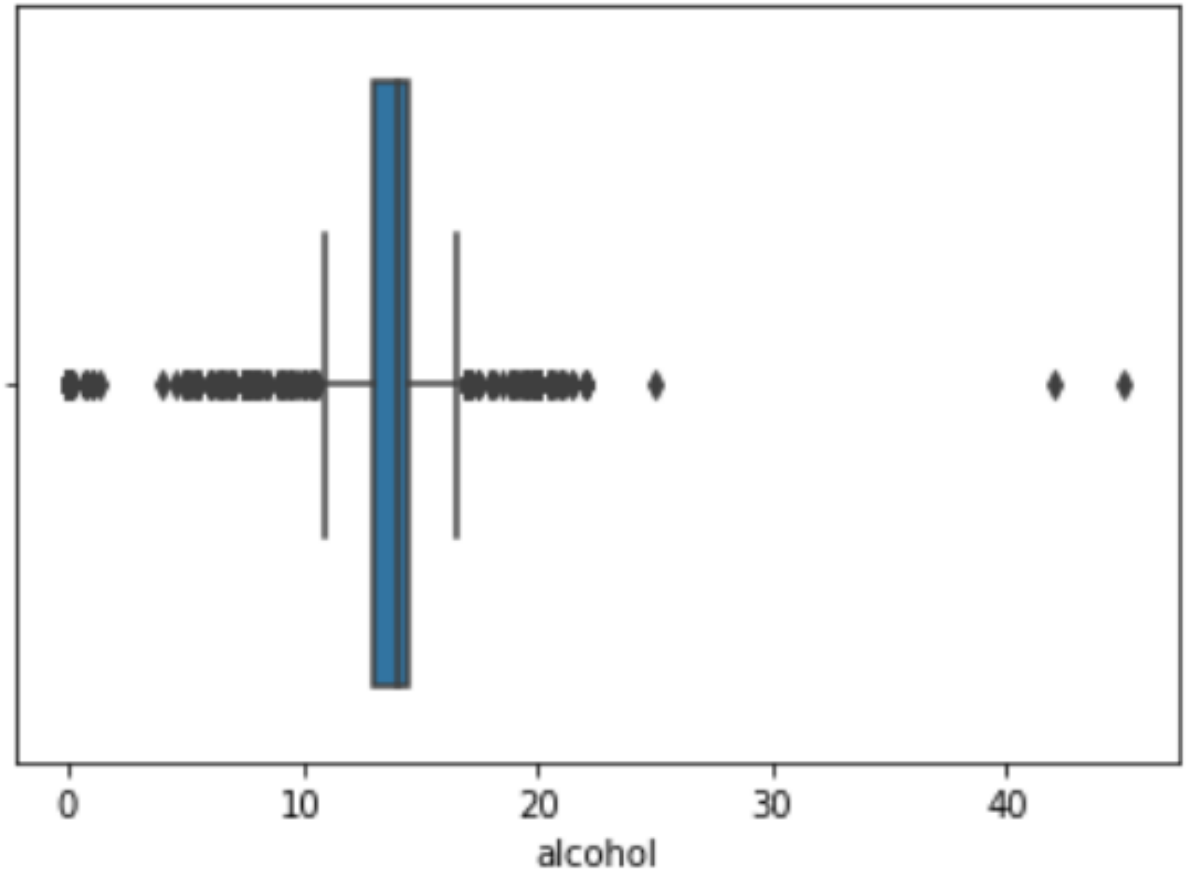
데이터 시각화

```
1  msno.matrix(wine_df, figsize=(12,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1b6d1d01400>
```



데이터 시각화



데이터 분석

데이터 시각화

```
1 wine_df[(wine_df['alcohol']<5) & (wine_df['alcohol']!=0)]
```

	name	price	score	winery	grapes	country	region	alcohol	foods	wine_id	re
1717	Shiraz 2003	59.99	4.3	Barossa Old Vine Company	Shiraz/Syrah	Australia	Barossa Valley	0.1	['Beef', 'Lamb', 'Game (deer, venison)', 'Poul...	1510660	['Full of cherry with creamy finish..', 'Wow. ...
3129	Lee Vineyard Pinot Noir 2017	64.99	4.3	Saintsbury	Pinot Noir	United States	Los Carneros	1.4	['Beef', 'Veal', 'Game (deer, venison)', 'Poul...	22490883	['more acidic than most recent saintsbury wine...
10784	Gigondas Le Claux 2015	119.95	4.3	Château de Saint Cosme	Grenache	France	Gigondas	0.8	['Beef', 'Lamb', 'Game (deer, venison)']	13701963	['Good wine. Smooth ', 'This is now drinking w...
10994	Bin 707 Cabernet Sauvignon 2004	699.00	4.5	Penfolds	Cabernet Sauvignon	Australia	South Australia	0.1	['Beef', 'Lamb', 'Poultry']	22606940	['Amazing wine. Let it breathe. After promotio...
11964	Panek Vineyard Cabernet Sauvignon 2014	224.99	4.6	Pulido-Walker	Cabernet Sauvignon	United States	Napa Valley	4.6	['Beef', 'Lamb', 'Game (deer, venison)', 'Matu...	24840468	['A real sexy, smooth stunner. The middle of a...
12209	Meshach 2009	155.00	4.5	Grant Burge	Shiraz/Syrah	Australia	Barossa Valley	0.1	['Beef', 'Lamb', 'Game (deer, venison)', 'Poul...	25566002	['Burge's flagship red stands as one of the ve...
12999	Châteauneuf-du-Pape Vieilles Vignes 2010	99.99	4.4	Domaine de la Janasse	Garnacha	France	Châteauneuf-du-Pape	0.8	['Lamb', 'Pork', 'Poultry']	20267194	['Tough luck for this 100 Parker points wine f...

데이터 분석

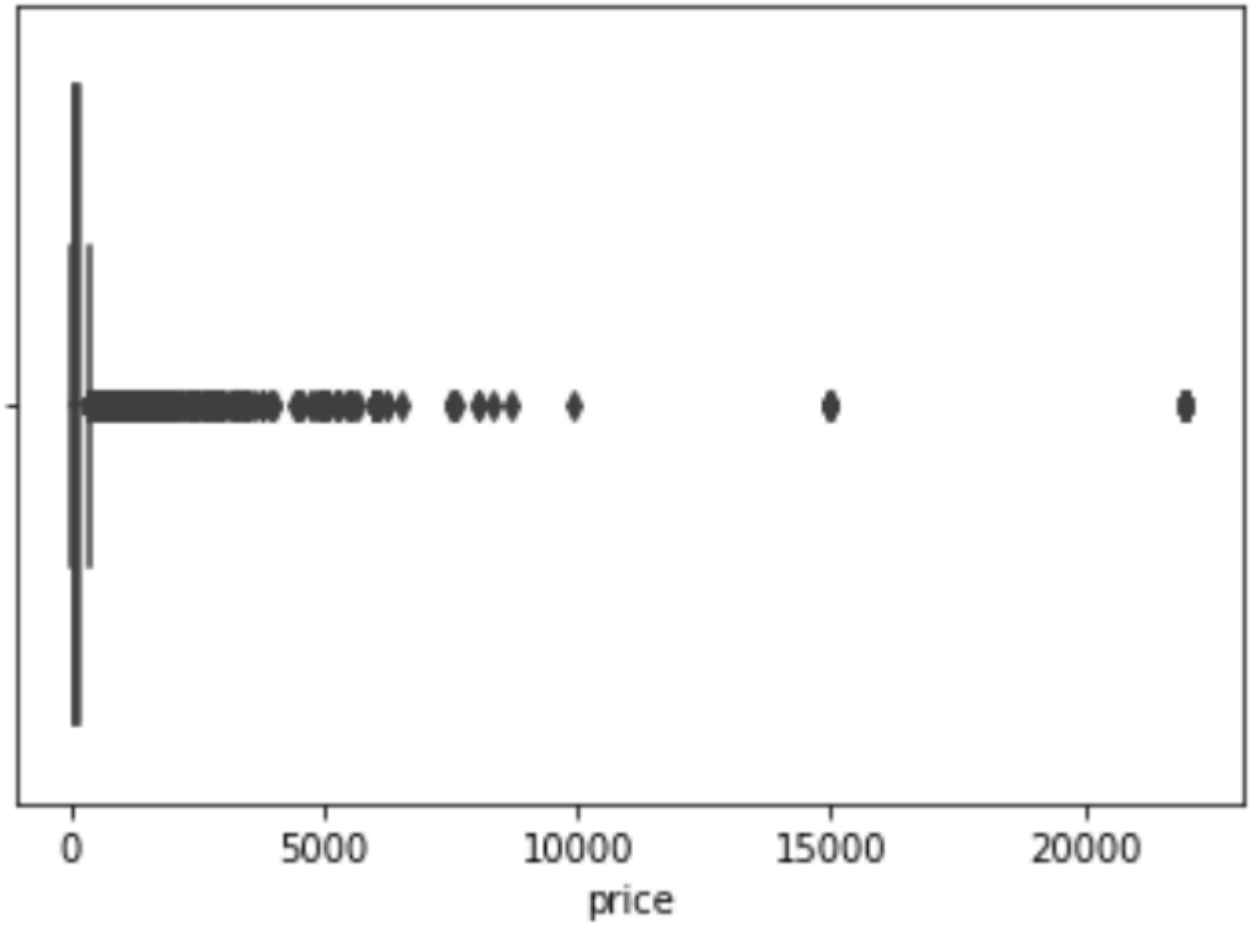
데이터 시각화

```
1 wine_df[(wine_df['alcohol']> 40)]
```

	name	price	score	winery	grapes	country	region	alcohol	foods	wine_id	re
11847	Amarone della Valpolicella Classico 2001	87.99	4.4	Allegrini	Corvina	Italy	Amarone della Valpolicella Classico	45.0	['Beef', 'Lamb', 'Game (deer, venison)', 'Blue...	15243072	['2001 but really something. Still nice acidit...
18520	Grappa Tignanello N.V.	59.99	4.3	Antinori	Cabernet Sauvignon	Italy	Chianti Classico	42.0	['Beef', 'Lamb', 'Veal', 'Pork', 'Game (deer, ...	24849065	['Powerful yet refined and smooth\n', 'Quite s...

데이터 분석

데이터 시각화



데이터 시각화

```
1  wine_df['price'].describe()
```

```
count     23713.000000
mean        222.036865
std         815.167280
min           5.490000
25%          54.990000
50%          89.990000
75%         175.000000
max       21999.990000
Name: price, dtype: float64
```

03

데이터 전처리과정

데이터 전처리

결측치

```
1 wine_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23782 entries, 0 to 23781
Data columns (total 11 columns):
name        23782 non-null object
price       23713 non-null float64
score       23782 non-null float64
winery      23782 non-null object
grapes      23722 non-null object
country     23775 non-null object
region      23775 non-null object
alcohol     14523 non-null float64
foods       23501 non-null object
wine_id     23782 non-null int64
re          23782 non-null object
dtypes: float64(3), int64(1), object(7)
memory usage: 2.0+ MB
```

```
1 wine_df.isnull().sum()
```

```
name          0
price        69
score         0
winery        0
grapes       60
country       7
region        7
alcohol    9259
foods       281
wine_id       0
re            0
dtype: int64
```

Price, grapes, country, region, alcohol, foods에 Null값 존재

데이터 전처리

결측치_alcohol

```
wine_mean = wine_df['alcohol'].mean() # 전체 알콜 평균
```

```
all_grapes = wine_df['grapes'].unique() # 포도 품종 리스트
```

```
# 포도 품종 별로 알콜 평균으로 채워줌
for i in range(len(wine_df)):
    for grape in all_grapes:
        try:
            if wine_df['grapes'][i] == grape:
                if (math.isnan(wine_df['alcohol'][i]) == True) | (wine_df['alcohol'][i] == 0.0):
                    wine_price.at[i,'alcohol'] = wine_df[wine_df[ 'grapes'] == grape]['alcohol'].mean()
        except:
            continue
```

```
# 포도 품종이 1개 밖에 없거나 다 결측치인 경우도 있기 때문에 그런 경우에는 전체 을
wine_df['alcohol'].fillna(wine_mean,inplace = True)
```

```
wine_df['alcohol'].replace(0.0,wine_mean,inplace = True)
```

```
wine_df.isnull().sum()
```

데이터 전처리

결측치 및 데이터 보완

```
1 wine_df['wine_id'].value_counts()
```

```
24469086    2
23382023    2
25199482    2
24943458    2
3267125     2
```

```
1 #id열 중복제거
2 wine_df['wine_id'] = wine_df['wine_id'].drop_duplicates( keep ='first', inplace =
```

```
1 wine_df = wine_df.dropna(axis=0)
```

```
1 wine_df.isnull().sum()
```

```
name        0
price       0
score       0
winery      0
grapes      0
country     0
region      0
alcohol     0
foods       0
wine_id     0
re          0
dtype: int64
```

데이터 전처리

Min/Max Scaling

✓ 평점(score), 도수(alcohol),가격(price)Min/Max로 정규화

```
1  from sklearn.preprocessing import MinMaxScaler
2
3  score_scaler = MinMaxScaler()
4  score_scaler
```

```
MinMaxScaler()
```

```
1  # 모델을 돌리기 위해 array화
2
3  score_data = mima_wine.values
4  score_data
```

```
array([[ 52.99     ,   4.3      ,  13.5      ],
       [ 54.99     ,   4.2      ,  13.4916305],
       [ 73.95     ,   4.3      ,  13.4916305],
       ...,
       [ 79.99     ,   4.4      ,  14.4      ],
       [369.99     ,   4.5      ,  13.4916305],
       [ 33.99     ,   4.3      ,  13.       ]])
```

```
1  score_scaler.fit(score_data)
```

```
MinMaxScaler()
```

```
1  result= score_scaler.transform(score_data)
```

```
1  result_df = pd.DataFrame(result,columns=['s_price','s_score','s_alcohol'])
2  result_df.head()
```

	s_price	s_score	s_alcohol
0	0.004801	0.333333	0.237968
1	0.005003	0.222222	0.237744
2	0.006919	0.333333	0.237744
3	0.001465	0.222222	0.264706
4	0.005358	0.333333	0.237744

데이터 전처리

Min/Max Scaling

✓ 평점(score), 도수(alcohol),가격(price)Min/Max로 정규화
✓ rev_alcohol, rev_ alcohol열 추가

```
#데이터 합치기

df = pd.concat([wine_df,result_df],axis=1)
df.head()
```

country	region	alcohol	foods	wine_id	re	s_price	s_score	s_alcohol
United States	Arroyo Grande Valley	13.500000	['Beef', 'Lamb', 'Game (deer, venison)', 'Poul...	19603399.0	['Dry tingly cool bottle pairs with seafood ho...	0.004801	0.333333	0.237968
France	Ch창teauneuf-du-Pape	13.491631	['Lamb', 'Pork', 'Poultry']	18191466.0	['Black cherry, oak. Leather, earthy. Great b...	0.005003	0.222222	0.237744

```
df['rev_s_alcohol'] = 1.0 - df['s_alcohol']
df['rev_s_alcohol'].head()
```

```
0    0.762032
1    0.762256
2    0.762256
3    0.735294
4    0.762256
Name: rev_s_alcohol, dtype: float64
```

```
df['rev_s_price'] = 1.0 - df['s_price']
df['rev_s_price'].head()
```

```
0    0.995199
1    0.994997
2    0.993081
3    0.998535
4    0.994642
Name: rev_s_price, dtype: float64
```

04

적용한 분석기법 및 모델소개

와인이름 추천 시스템

와인이름 추천 시스템_이름입력

```
1 wine_name = input("와인 이름을 입력해주세요: ")
```

```
와인 이름을 입력해주세요: Rim Rock
```

```
1 new_name = string.capwords(wine_name)
2 new_name
```

```
'Rim Rock'
```

```
1 df[df['name'].str.contains(new_name)==True]
```

	name	price	score	winery	grapes	country	region	alcohol	foods	wine_id	(review) re	rev_alcohol
0	Rim Rock Vineyard Syrah 2017	0.00216	0.333333	Piedrasassi	Shiraz/Syrah	United States	Arroyo Grande Valley	0.298441	['Beef', 'Lamb', 'Game (deer, venison)', 'Poul...	19603399	['Dry tingly cool bottle pairs with seafood ho...	0.701559

```
1 # 인덱스 찾기
2 wine_index =df[df['name'].str.contains(new_name)==True].index[0]
3 wine_index
```

```
0
```

와인이름 추천 시스템_유사도분석

```python
def cosine_func(col_name,wine_index):

    #정규 표현식으로 영문만 남음
    only_english  = [ re.sub('[^a-zA-Z]', ' ',sentence).lower() for sentence in df[col_name]]
    #토큰만들기
    col_tokenized = [ nltk.word_tokenize(item) for item in only_english ]
    #불용어 빼기
    no_stopwords = [ i for i in col_tokenized if i not in stopwords]
    #다시 합쳐주기
    final_review = [ ' '.join(item) for item in no_stopwords ]

    #Tfidf
    tfidf_vect = TfidfVectorizer()
    feature_vect = tfidf_vect.fit_transform(final_review)

    #코사인 유사도 분석 => 사용자가 입력한 와인과 전체
    similarity_simple_pair = cosine_similarity(feature_vect[wine_index], feature_vect)
    result_list = similarity_simple_pair.tolist()[0]
    df[f'{col_name}_result'] = result_list

    return df
```

와인이름 추천 시스템_유사도 추가

```
# 데이터프레임에 유사도 추가
my_col = ['foods','re'](review)
for target_col in my_col:
    cosine_func(target_col,wine_index)
```

```
df.head()
```

	s_score	s_alcohol	rev_s_alcohol	rev_s_price	name	winery	foods_result	re_result
	0.333333	0.237968	0.762032	0.995199	Rim Rock Vineyard Syrah 2017	Piedrasassi	1.000000	1.000000
	0.222222	0.237744	0.762256	0.994997	Tradition Châteauneuf-du-Pape 2015	Domaine Giraud	0.334767	0.203903

와인이름 추천 시스템

```
result_arr = []
for column in df.columns.values:
    if (df[column].dtype == 'float64') |  (df[column].dtype == 'int64'):
        if(column ! = 'wine_id'):
            result_arr.append(column)
```

```
result_arr = result_arr[3:]
result_arr
```

```
['s_price',
 's_score',
 's_alcohol',
 'rev_s_alcohol',
 'rev_s_price',
 'foods_result',
 're_result']
```

```
df_result_weight = df[result_arr].copy()
```

```
df_result_weight.head()
```

	s_price	s_score	s_alcohol	rev_s_alcohol	rev_s_price	foods_result	re_result
0	0.004801	0.333333	0.237968	0.762032	0.995199	1.000000	1.000000
1	0.005003	0.222222	0.237744	0.762256	0.994997	0.334767	0.203903
2	0.006919	0.333333	0.237744	0.762256	0.993081	0.561923	0.167851
3	0.001465	0.222222	0.264706	0.735294	0.998535	0.437460	0.196907
4	0.005358	0.333333	0.237744	0.762256	0.994642	1.000000	0.142127

와인이름 추천 시스템_가중치

```
price_w = float(input("가중치를 입력해주세요.(합 1) Wn가격 : "))
score_w = float(input("평점 : "))
alcohol_w = float(input("도수 : "))
food_w = float(input("어울리는 음식: "))
review_w = float(input("리뷰: "))
```

```
가중치를 입력해주세요.(합 1)
가격 : 0.2
평점 : 0.2
도수 : 0.2
어울리는 음식: 0.2
리뷰: 0.2
```

```
# how == 1, low == 2

how_alcohol = int(input("도수 선호도(high:1 low:2) : "))
how_price = int(input("가격 선호도(high:1 low:2) : "))
```

```
도수 선호도(high:1 low:2) : 1
가격 선호도(high:1 low:2) : 1
```

와인이름 추천 시스템_가중치 계산

```
# 가중치 추가
if how_price == 1:
    df['weight'] = df_result_weight['s_price'] * price_w
else:
    df['weight'] = df_result_weight['rev_s_price'] * price_w
df['weight'] += df_result_weight['s_score'] * score_w
if how_alcohol == 1:
    df['weight'] += df_result_weight['s_alcohol'] * alcohol_w
else:
    df['weight'] += df_result_weight['rev_s_alcohol'] * alcohol_w
df['weight'] += df_result_weight['foods_result'] * food_w
df['weight'] += df_result_weight['re_result'] * review_w
```

```
df['weight']
```

```
df.head()
```

s_alcohol	rev_s_alcohol	rev_s_price	name	winery	foods_result	re_result	weight
0.237968	0.762032	0.995199	Rim Rock Vineyard Syrah 2017	Piedrasassi	1.000000	1.000000	0.257610

와인이름 추천 시스템_결과값 출력

```
1 df_sorted_by_values = df.sort_values(by='weight' ,ascending=False)
```

```
1 result = df_sorted_by_values[:6]
```

```
1 result
```

	url	wine_id	price	score	grapes	country	region	alcohol
10542	https://www.vivino.com/US-CA/en/petrus-pomerol...	4216005.0	5999.95	4.8	Cabernet Franc	France	Pomerol	13.5

04

적용한 분석기법 및 모델소개

항목별 와인 추천 시스템

항목별 와인추천 시스템_항목입력

```
In [3]: alcohol_min = float(input("알코올 도수 최솟값을 입력해주세요!! : "))
        알코올 도수 최솟값을 입력해주세요!! : 10

In [4]: alcohol_max = float(input("알코올 도수 최댓값을 입력해주세요!! : "))
        알코올 도수 최댓값을 입력해주세요!! : 17

In [5]: price_min = float(input("가격 최솟값을 입력해주세요!! : "))
        가격 최솟값을 입력해주세요!! : 30

In [6]: price_max = float(input("가격 최댓값을 입력해주세요!! : "))
        가격 최댓값을 입력해주세요!! : 150

In [7]: food_name = input("같이 먹을 음식을 입력해주세요!! : ")
        같이 먹을 음식을 입력해주세요!! : shellfish

In [8]: review_tagname = input("이번 파티의 핵심 단어를 입력해주세요!! : ")
        이번 파티의 핵심 단어를 입력해주세요!! : I wanna be real

In [9]: grape_name = input("포도품종을 입력해주세요!! : ")
        포도품종을 입력해주세요!! :

In [10]: country_name = input("선호하는 나라가 있나요 : ")
         선호하는 나라가 있나요 :
```

항목별 와인추천 시스템_우선순위 값 입력

In [33]:
```
num1 = int(input("평점 : "))
num2 = int(input("같이 먹을 음식 : "))
num3 = int(input("리뷰 : "))
```

```
평점 : 3
같이 먹을 음식 : 1
리뷰 : 2
```

In [48]:
```
def weight_cal(num):
    if num == 1:
        w = 5.0
    elif num == 2 :
        w = 3.5
    else:
        w = 2
    return w
```

항목별 와인추천 시스템_포도품종/나라

In [52]:
```
grape_select = df['grapes'].value_counts().head(15).keys().to_list()

vivino_grapes = ['Cabernet Sauvignon', 'Merlot', 'Chardonnay', 'Pinot Noir', 'Malbec',
'Sauvignon Blanc', 'Shiraz/Syrah', 'Zinfandel', 'Nebbiolo', 'Sangiovese',
'Pinot Grigio', 'Riesling', 'Chenin Blanc', 'Moscato', 'Albarino']
for i in range(len(vivino_grapes)):
    if vivino_grapes[i] not in grape_select:
        grape_select.append(vivino_grapes[i])
grape_select
```

```
['Cabernet Sauvignon',
 'Pinot Noir',
 'Chardonnay',
 'Shiraz/Syrah',
 'Nebbiolo',
 'Sangiovese',
```

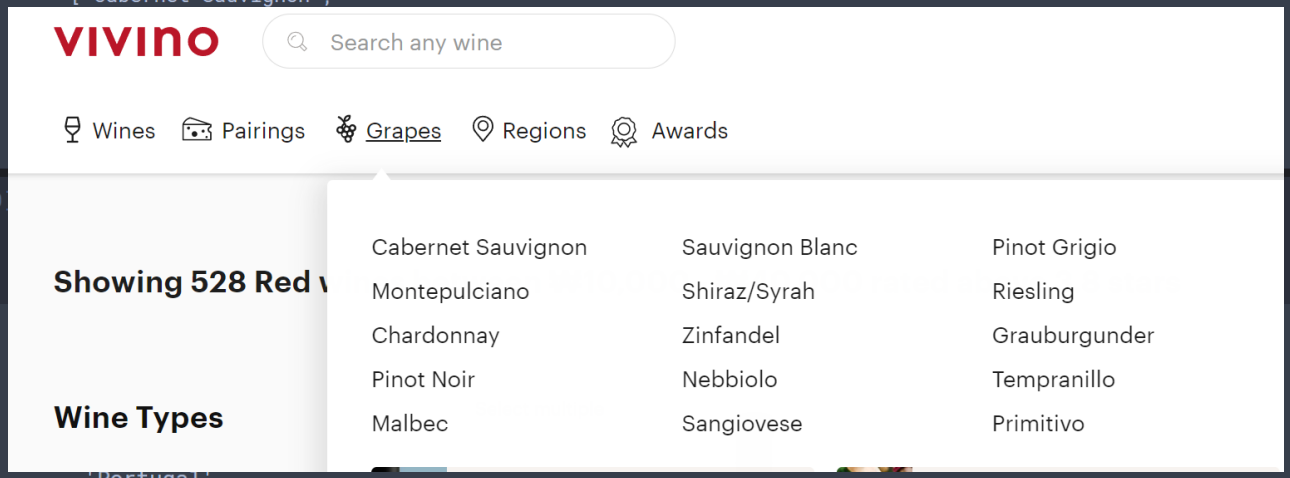
In [40]:
```
country_select = df['country'].value_counts().head(15).keys().to_list()
country_select
```

```
['United States',
 'France',
 'Italy',
 'Spain',
 'Portugal',
 'Australia',
 'Argentina',
```

항목별 와인추천 시스템_포도품종/나라

```
In [52]:
grape_select = df['grapes'].value_counts().head(15).keys().to_list()

vivino_grapes = ['Cabernet Sauvignon', 'Merlot', 'Chardonnay', 'Pinot Noir', 'Malbec',
'Sauvignon Blanc', 'Shiraz/Syrah', 'Zinfandel', 'Nebbiolo', 'Sangiovese',
'Pinot Grigio', 'Riesling', 'Chenin Blanc', 'Moscato', 'Albarino']
for i in range(len(vivino_grapes)):
    if vivino_grapes[i] not in grape_select:
        grape_select.append(vivino_grapes[i])
grape_select
```

```
['Cabernet Sauvignon',
```




```
In [40
'Portugal',
'Australia',
'Argentina',
```

항목별 와인추천 시스템_포도품종/나라

```
In [52]:
grape_select = df['grapes'].value_counts().head(15).keys().to_list()

vivino_grapes = ['Cabernet Sauvignon', 'Merlot', 'Chardonnay', 'Pinot Noir', 'Malbec',
'Sauvignon Blanc', 'Shiraz/Syrah', 'Zinfandel', 'Nebbiolo', 'Sangiovese',
'Pinot Grigio', 'Riesling', 'Chenin Blanc', 'Moscato', 'Albarino']
for i in range(len(vivino_grapes)):
    if vivino_grapes[i] not in grape_select:
        grape_select.append(vivino_grapes[i])
grape_select
```

```
['Cabernet Sauvignon',
```

vivino

Search any wine

Wines | Pairings | Grapes | Regions | Awards

Cabernet Sauvignon	Sauvignon Blanc	Pinot Grigio
Montepulciano	Shiraz/Syrah	Riesling
Chardonnay	Zinfandel	Grauburgunder
Pinot Noir	Nebbiolo	Tempranillo
Malbec	Sangiovese	Primitivo

Showing 528 Red

Wine Types

```
In [40
```

```
'Portugal',
'Australia',
'Argentina',
```

항목별 와인추천 시스템_

In [21]:

```
def customer_data(df):
    df = df[(df['alcohol'] >= alcohol_min) & (df['alcohol'] <= alcohol_max)]
    df = df[(df['price'] >= price_min) & (df['price'] <= price_max) ]

    if grape_name != '':
        if grape_name in grape_select:
            df = df[df['grapes'] == grape_name]

    if country_name != '':
        if country_name in country_select:
            df = df[df['country'] == country_name]
        elif country_name == "etc":
            for i in country_select:
                df = df[df['country'] != i]


    return df
```

항목별 와인추천 시스템_유사도 분석

1.

```
def cosine_func(col_name,wine_index):

    #정규 표현식으로 영문만 남음
    only_english  = [ re.sub('[^a-zA-Z]', ' ',str(sentence)).lower() for sentence
    #토큰만들기
    col_tokenized = [ nltk.word_tokenize(item) for item in only_english ]
    #불용어 빼기
    no_stopwords = [ i for i in col_tokenized if i not in stopwords]
    #다시 합쳐주기
    final_review = [ ' '.join(item) for item in no_stopwords ]

    #Tfidf
    tfidf_vect = TfidfVectorizer()
    feature_vect = tfidf_vect.fit_transform(final_review)

    #코사인 유사도 분석 => 사용자가 입력한 와인과 전체
    similarity_simple_pair = cosine_similarity(feature_vect[wine_index], feature_v
    result_list = similarity_simple_pair.tolist()[0]
    df[f'{col_name}_result'] = result_list

    return df
```

2.

```
# 데이터프레임에 유사도 추가
my_col = ['foods','re']
for target_col in my_col:
    cosine_func(target_col,wine_index)
```

```
df.head()
```

:ry	region	alcohol	foods	wine_id	review	rev_alcohol	foods_result	re_result
ed es	Arroyo Grande Valley	0.300000	['Beef', 'Lamb', 'Game (deer, venison)', 'Poul...	19603399	['Dry tingly cool bottle pairs with seafood ho...	0.700000	1.000000	1.000000
ce	Châteauneuf-du-Pape	0.311111	['Lamb', 'Pork', 'Poultry']	18191466	['Black cherry, oak. Leather, earthy. Great h	0.688889	0.334653	0.203839

3.

```
result_arr=["s_score","s_alcohol","s_price","foods_result","re_result"]
```

```
df_result_weight = df[result_arr].copy()
df_result_weight.head()
```

	s_score	s_alcohol	s_price	foods_result	re_result
343	0.222222	0.237968	0.002981	0.422903	0.077732
670	0.222222	0.237744	0.008440	0.422903	0.045953
1493	0.222222	0.237968	0.007529	0.422903	0.030910
1573	0.222222	0.237744	0.003477	0.422903	0.026085
4911	0.333333	0.237744	0.008440	0.422903	0.051161

항목별 와인추천 시스템_가중치 및 결과값

In [54]:

```
# 가중치 추가

df['weight'] = df_result_weight['s_score']* 0.01 *weight_cal(num1)
df['weight'] += df_result_weight['s_alcohol'] * 0.1
df['weight'] += df_result_weight['s_price'] * 1
df['weight'] += df_result_weight['foods_result'] * 0.01 * weight_cal(num2)
df['weight'] += df_result_weight['re_result'] * 0.1 * weight_cal(num3)
```

항목별 와인추천 시스템_가중치 및 결과값

In [54]:
```
# 가중치 추가

df['weight'] = df_result_weight['s_score']* 0.01 *weight_cal(num1)
df['weight'] += df_result_weight['s_alcohol'] * 0.1
df['weight'] += df_result_weight['s_price'] * 1
df['weight'] += df_result_weight['foods_resul
df['weight'] += df_result_weight['re_result']
```

In [48]:
```
def weight_cal(num):
    if num == 1:
        w = 5.0
    elif num == 2 :
        w = 3.5
    else:
        w = 2
    return w
```

04

적용한 분석기법 및 모델소개

이미지기반으로 한 추천 시스템

이미지를 기반으로 한 추천 시스템-(Prepare the data)

AI Hub

개방 데이터 | 외부 데이터 | 활용 사례 | 개발 지원 | 경진대회 | 게시판 | 마이페이지 | 로그아

개방 데이터

- 비전
- 음성/자연어
- 교육
- 국토환경
- 농축수산
- 안전
- 자율주행
- 헬스케어

인공지능 학습용 데이터 다운로드 프로그램 설치

Windows & Mac용

> 간단 사용설명서
> 맥용 설치&삭제 가이드

> 개방 데이터 > 비전 > 한국 이미지

한국 이미지(음식)

소개 | 다운로드

데이터셋명	한국 이미지(음식)			
데이터 분야	비전	데이터 유형		이미지
구축기관	한국과학기술연구원	데이터 관련 문의처	담당자명	조정현(한국과학기술연구원)
가공기관			전화번호	02-958-6650
검수기관			이메일	jhcho@kist.re.kr
구축 데이터량	15만	구축년도		2018년
버전	1.0	최종수정일자		2019.12.31
소개	한국 음식 150종(종별 약 1천 장)의 데이터를 구축한 이미지 데이터 제공			
주요 키워드	한식 이미지, 한국 음식, 구이, 김치, 떡, 면, 국			
저작권 및 이용정책	본 데이터는 과학기술정보통신부가 주관하고 한국지능정보사회진흥원이 지원하는 '인공지능 학습용 데이터 구축사업'으로 구축된 데이터입니다. [데이터 이용정책 상세보기]			

이미지를 기반으로 한 추천 시스템-(Prepare the data)

AI Hub 사이트에 있는 데이터셋 활용

AI Hub

개방 데이터 | 외부 데이터 | 활용 사례 | 개발 지원 | 경진대회 | 게시판 | 마이페이지 | 로그아

개방 데이터

- 비전
- 음성/자연어
- 교육
- 국토환경
- 농축수산
- 안전
- 자율주행
- 헬스케어

인공지능 학습용 데이터 다운로드 프로그램 설치

Windows & Mac용

> 간단 사용설명서
> 맥용 설치&삭제 가이드

개방 데이터 > 비전 > 한국 이미지

한국 이미지(음식)

소개 | 다운로드

데이터셋명	한국 이미지(음식)			
데이터 분야	비전	데이터 유형		이미지
구축기관	한국과학기술연구원	데이터 관련 문의처	담당자명	조정현(한국과학기술연구원)
가공기관			전화번호	02-958-6650
검수기관			이메일	jhcho@kist.re.kr
구축 데이터량	15만	구축년도		2018년
버전	1.0	최종수정일자		2019.12.31
소개	한국 음식 150종(종별 약 1천 장)의 데이터를 구축한 이미지 데이터 제공			
주요 키워드	한식 이미지, 한국 음식, 구이, 김치, 떡, 면, 국			
저작권 및 이용정책	본 데이터는 과학기술정보통신부가 주관하고 한국지능정보사회진흥원이 지원하는 '인공지능 학습용 데이터 구축사업'으로 구축된 데이터입니다. [데이터 이용정책 상세보기]			

이미지를 기반으로 한 추천 시스템-(Prepare the data)

```
[ ]  import splitfolders

     splitfolders.ratio('/content/gdrive/MyDrive/Colab Notebooks/data/food,
                        output="/content/gdrive/MyDrive/Colab Notebooks/data",
                        seed=1337, ratio=(0.75, 0.15, 0.1))
```

```
[22] train_path = '/content/drive/MyDrive/Colab Notebooks/data/train'
     valid_path = '/content/drive/MyDrive/Colab Notebooks/data/valid'
     test_path = '/content/drive/MyDrive/Colab Notebooks/data/test'



     train_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input).flow_from_directory(
         directory=train_path, target_size=(224,224), batch_size=70, color_mode='rgb')
     valid_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input).flow_from_directory(
         directory=valid_path, target_size=(224,224), batch_size=70, color_mode='rgb')
     test_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input).flow_from_directory(
         directory=test_path, target_size=(224,224), batch_size=70, color_mode='rgb', shuffle=False)
```

```
Found 23270 images belonging to 31 classes.
Found 4644 images belonging to 31 classes.
Found 3106 images belonging to 31 classes.
```

이미지를 기반으로 한 추천 시스템-(Prepare the data)

```
[ ]  import splitfolders

     splitfolders.ratio('/content/gdrive/MyDrive/Colab Notebooks/data/food,
                        output="/content/gdrive/MyDrive/Colab Notebooks/data",
                        seed=1337, ratio=(0.75, 0.15, 0.1))
```

```
[22] train_path = '/content/drive/MyDrive/Colab Notebooks/data/train'
     valid_path = '/content/drive/MyDrive/Colab Notebooks/data/valid'
     test_path = '/content/drive/MyDrive/Colab Notebooks/data/test'



     train_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input).flow_from_directory(
         directory=train_path, target_size=(224,224), batch_size=70, color_mode='rgb')
     valid_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input).flow_from_directory(
         directory=valid_path, target_size=(224,224), batch_size=70, color_mode='rgb')
     test_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input).flow_from_directory(
         directory=test_path, target_size=(224,224), batch_size=70, color_mode='rgb', shuffle=False)
```

```
Found 23270 images belonging to 31 classes.
Found 4644 images belonging to 31 classes.
Found 3106 images belonging to 31 classes.
```

이미지를 기반으로 한 추천 시스템-(Prepare the data)

```
[ ]  import splitfolders

     splitfolders.ratio('/content/gdrive/MyDrive/Colab Notebooks/data/food,
                        output="/content/gdrive/MyDrive/Colab Notebooks/data",
                        seed=1337, ratio=(0.75, 0.15, 0.1))
```

```
[22] train_path = '/content/drive/MyDrive/Colab Notebooks/data/train'
     valid_path = '/content/drive/MyDrive/Colab Notebooks/data/valid'
     test_path = '/content/drive/MyDrive/Colab Notebooks/data/test'



     train_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input).flow_from_directory(
         directory=train_path, target_size=(224,224), batch_size=70, color_mode='rgb')
     valid_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input).flow_from_directory(
         directory=valid_path, target_size=(224,224), batch_size=70, color_mode='rgb')
     test_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input).flow_from_directory(
         directory=test_path, target_size=(224,224), batch_size=70, color_mode='rgb', shuffle=False)
```

```
Found 23270 images belonging  31 classes.
Found 4644 images belonging t 31 classes.
Found 3106 images belonging t 31 classes.
```

이미지를 기반으로 한 추천 시스템-(Prepare the data)

```
[ ]  import splitfolders

     splitfolders.ratio('/content/gdrive/MyDrive/Colab Notebo
                        output="/content/gdrive/MyDrive/Colab
                        seed=1337, ratio=(0.75, 0.15, 0.1))
```

1. 구글드라이브에 데이터 업로드 후 Mount해주기
2. Splitfolders를 이용해 각 클래스의 사진들을 3등분 (/content/gdrive/MyDrive/Colab Notebooks/data/food -> food폴더 안에는 31개의 클래스별 폴더가 있는 상태)
3. Data폴더 안에 생성된 3개의 폴더를 확인후 각 폴더를 경로로 잡아주기
4. 각 폴더를 ImageGenerator를 통해 사이즈를 조정하며 불러오기

cf. 구글드라이브의 경우 서로 다른 유저가 업로드한 파일을 공유가 가능. Colab 내에서 모든 데이터를 활용해 분석

```
[22] train_path = '/content/drive/MyDrive/Colab Notebooks/data/train'
     valid_path = '/content/drive/MyDrive/Colab Notebooks/data/valid'
     test_path = '/content/drive/MyDrive/Colab Notebooks/data/test'



     train_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input).flow_from_directory(
         directory=train_path, target_size=(224,224), batch_size=70, color_mode='rgb')
     valid_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input).flow_from_directory(
         directory=valid_path, target_size=(224,224), batch_size=70, color_mode='rgb')
     test_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input).flow_from_directory(
         directory=test_path, target_size=(224,224), batch_size=70, color_mode='rgb', shuffle=False)
```

```
Found 23270 images belonging to 31 classes.
Found 4644 images belonging to 31 classes.
Found 3106 images belonging to 31 classes.
```

이미지를 기반으로 한 추천 시스템

```
[13]  IMG_SHAPE = (IMG_SIZE, IMG_SIZE, 3) # (224, 224, 3)

      # 사전 훈련된 모델 MobileNet V2에서 기본 모델을 생성합니다.
      base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                                      include_top=False,
                                                      weights='imagenet')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobile
9412608/9406464 [==============================] - 0s 0us/step
9420800/9406464 [==============================] - 0s 0us/step
```

```
[14]  base_model.trainable = False
```

이미지를 기반으로 한 추천 시스템

```
[13]  IMG_SHAPE = (IMG_SIZE, IMG_SIZE, 3) # (224, 224, 3)

      # 사전 훈련된 모델 MobileNet V2에서 기본 모델을 생성합니다.
      base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                                      include_top=False,
                                                      weights='imagenet')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobile
9412608/9406464 [==============================] - 0s 0us/step
9420800/9406464 [==============================] - 0s 0us/step
```

```
[14]  base_model.trainable = False
```

Layer.trainable = False를
설정하면 훈련 중 지정된 층의
가중치가 업데이트 되지 않음

이미지를 기반으로 한 추천 시스템

분류 층 추가

```
[15] # 특징을 이미지 한개 당 1280개의 요소 벡터로 변환하여 5x5 공간 위치에 대한 평균
     global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
     # tf.keras.layers.Dense층을 사용하여 특징을 이미지당 20개 예측으로 변환
     # feature_batch_average = global_average_layer(feature_batch)
     # print(feature_batch_average.shape)
```

```
[16] prediction_layer = tf.keras.layers.Dense(units=31, activation='softmax')
     # prediction_batch = prediction_layer(feature_batch_average)
     # print(prediction_batch.shape)
```

```
[17] model = tf.keras.Sequential([
       base_model,
       global_average_layer,
       prediction_layer
     ])
```

이미지를 기반으로 한 추천 시스템

```
[19] base_learning_rate = 0.01
     model.compile(optimizer=tf.keras.optimizers.Adam(lr=base_learning_rate),
                   loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
                   metrics=['accuracy','TopKCategoricalAccuracy'])
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/adam.py:105: UserWarning: The `lr` argum
  super(Adam, self).__init__(name, **kwargs)
```

```
# initial_epochs = 40
history = model.fit(train_batches,
                    epochs=15,
                    validation_data=valid_batches)
```

이미지를 기반으로 한 추천 시스템

In [3]:
```
model = tf.keras.models.load_model('mobileNetV2_wine.h5')
```

```
def predict_food(ingredient_name):

    img_path = 'data/'+ ingredient_name
    new_img = image.load_img(img_path, target_size=(224,224))
    arr_img = image.img_to_array(new_img)
    img = arr_img.reshape((1,) + arr_img.shape)
    img = img.astype('float32')/255

    num_code = model.predict(img, batch_size=70, verbose=1).argmax()  # 왜 프로바가 없다고 나오는 걸까
    proba = model.predict(img).max() * 100

    for key, value in name_code.items():
        if num_code == value: name = key

    name = name.replace('_', ' ')
    result = name
    print('\nPredicted Image: ')

    show_img = imread_hangul_path(img_path)
    plt.imshow(cv2.cvtColor(show_img, cv2.COLOR_BGR2RGB))
    plt.axis("off")
    plt.show()

    print(f'This is: {name} \nWith a probability of: {proba}')
    return result
```

이미지를 기반으로 한 추천 시스템

[21]:
```
plt.figure(figsize=(10, 10))

predicted_food = predict_food(img)

plt.show()
```

```
1/1 [==============================] - 0s 43ms/step

Predicted Image:
```

```
This is: 꼬막찜
With a probability of: 94.87444758415222
```

이미지를 기반으로 한 추천 시스템

In [15]:
```
plt.figure(figsize=(10, 10))

predict_food = predict_food(img)

plt.show()
```

```
1/1 [==============================] - 1s 856ms/step

Predicted Image:
```

```
This is: 찜닭
With a probability of: 99.96998310089111
```

In [16]:
```
predict_food
```

```
'찜닭'
```

이미지를 기반으로 한 추천 시스템

In [3]:

```
food_catagories = {'갈비찜':'Beef','계란찜':'Pork', '김치찜':'Spicy food', '꼬
                   '닭볶음탕':['Spicy fodd', 'Poultry'], '수육':'Pork', '순대
                   '찜닭':'Poultry', '해물찜':'Spicy food'}


# Beef
# Lamb
# Veal
# Pork
# Game (deer, venison)
# Poultry
# Mushrooms
# Cured meat
# Goat Cheese
# Mature and hard cheese
# Mild and soft cheese
# Pasta
# Spicy food
# Aperitif
# Appetizers and snacks
# Lean fish
# Rich fish(Salmon, tuna etc)
# Shellfish
# Vegetarian
```

이미지를 기반으로 한 추천 시스템

In [5]:
```
predict_food = "닭볶음탕"
```

In [6]:
```
food_name = food_catagories.get(predict_food)
if type(food_name) == str:
    food_name = [food_name]


food_name
```

```
['Spicy fodd', 'Poultry']
```

In [129]:
```
df['weight'] = df_result_weight['s_score']*0.1
df['weight'] += df_result_weight['s_alcohol']*0.1
df['weight'] += df_result_weight['s_price']*10
df['weight'] += df_result_weight['foods_result']*0.1*10
df['weight'] += df_result_weight['re_result']*1
```

이미지를 기반으로 한 추천 시스템

In [5]:
```
predict_food = "닭볶음탕"
```

In [6]:
```
food_name = food_catagories.get(predict_food)
if type(food_name) == str:
    food_name = [food_name]


food_name
```

```
['Spicy fodd', 'Poultry']
```

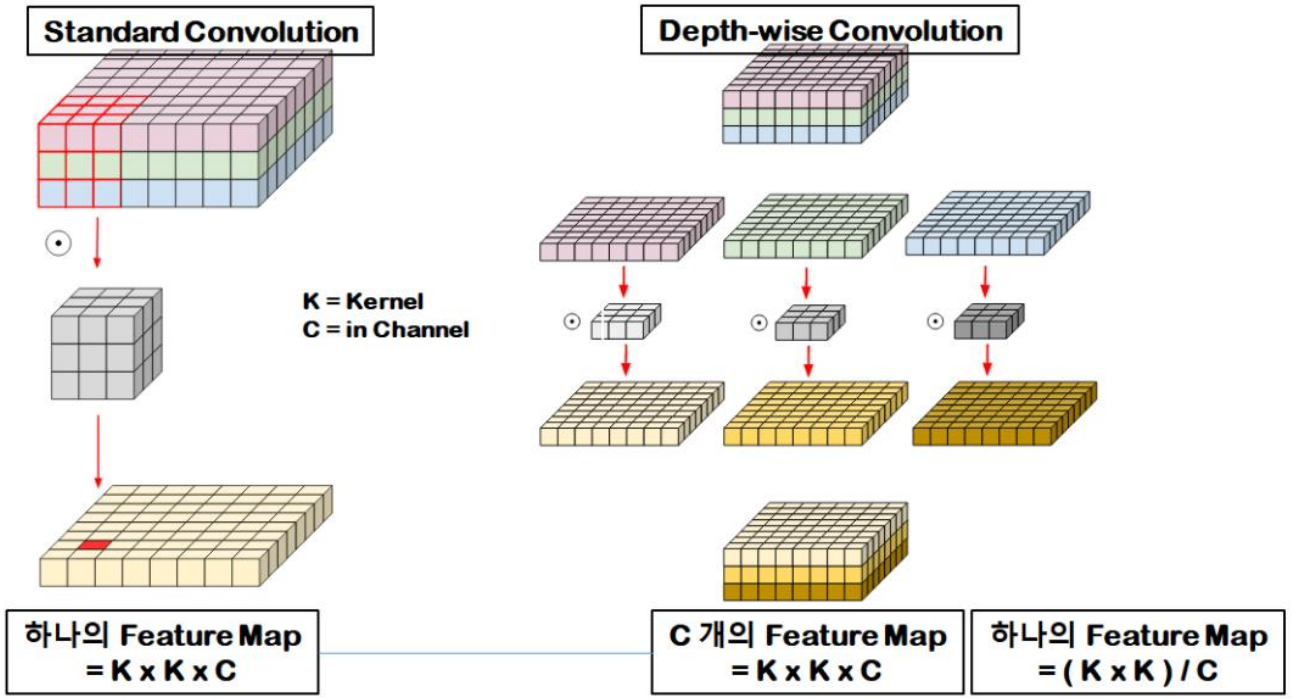
In [129]:
```
df['weight'] = df_result_weight['s_score']*0.1
df['weight'] += df_result_weight['s_alcohol']*0.1
df['weight'] += df_result_weight['s_price']*10
df['weight'] += df_result_weight['foods_result']*0.1*10
df['weight'] += df_result_weight['re_result']*1
```

05

분석 및 모델링 결과

분석 및 모델링 결과

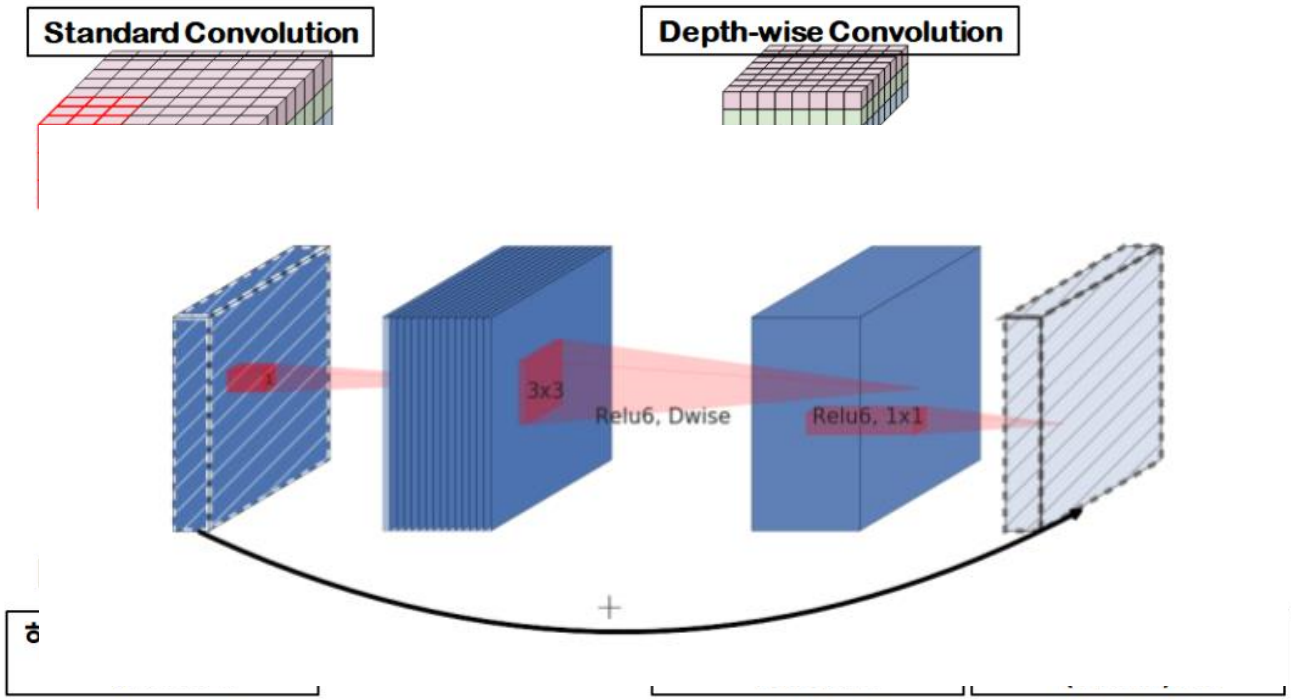
Mobilenet



기존 Convolution 방식과 달리 Mobilenet은 Depth-wise Convolution방식을 활용
효율적인 계산을 통해 더 빠른 시간 내에 원하는 결과값을 도출해낼 수 있음

분석 및 모델링 결과

Mobilenet

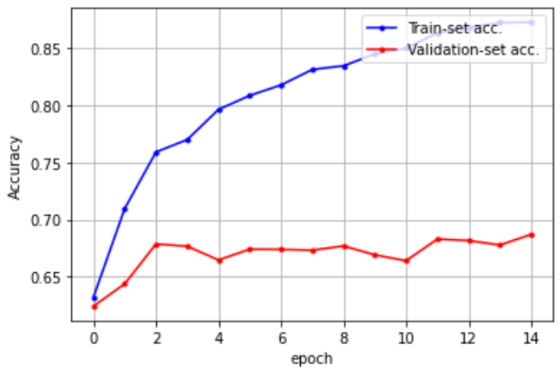


기존 Convolution 방식과 달리 Mobilenet은 Depth-wise Convolution방식을 활용
효율적인 계산을 통해 더 빠른 시간 내에 원하는 결과값을 도출해낼 수 있음

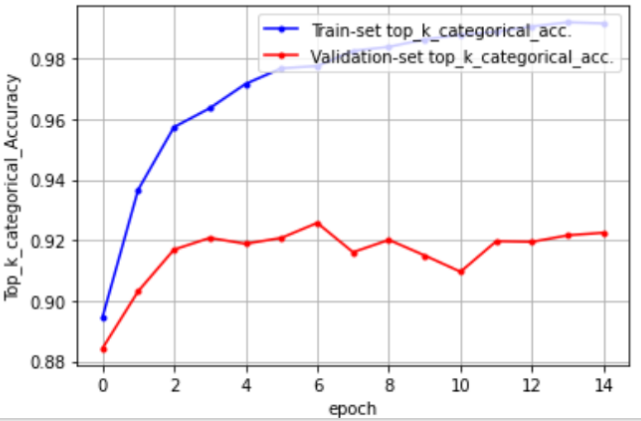
분석 및 모델링 결과

Mobilenet

```
[ ]  import numpy as np
     import matplotlib.pyplot as plt

     acc = history.history['accuracy']
     val_acc = history.history['val_accuracy']

     x_len = np.arange(len(acc))

     plt.plot(x_len, acc, marker='.', c='blue', label="Train-set acc.")
     plt.plot(x_len, val_acc, marker='.', c='red', label="Validation-set acc.")

     plt.legend(loc='upper right')
     plt.grid()
     plt.xlabel('epoch')
     plt.ylabel('Accuracy')
     plt.show()
```



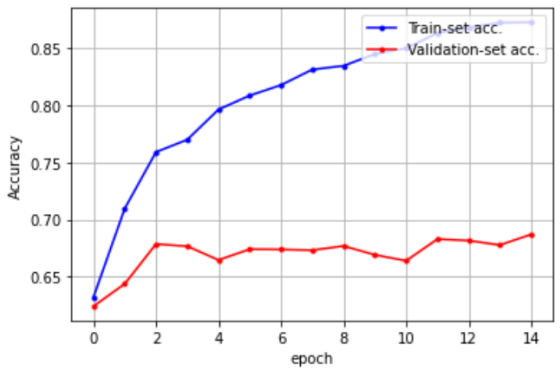

```
[ ]  top_k_acc = history.history['top_k_categorical_accuracy']
     val_top_k_acc = history.history['val_top_k_categorical_accuracy']

     x_len = np.arange(len(top_k_acc))

     plt.plot(x_len, top_k_acc, marker='.', c='blue', label="Train-set top_k_c
     plt.plot(x_len, val_top_k_acc, marker='.', c='red', label="Validation-set

     plt.legend(loc='upper right')
     plt.grid()
     plt.xlabel('epoch')
     plt.ylabel('Top_k_categorical_Accuracy')
     plt.show()
```



분석 및 모델링 결과

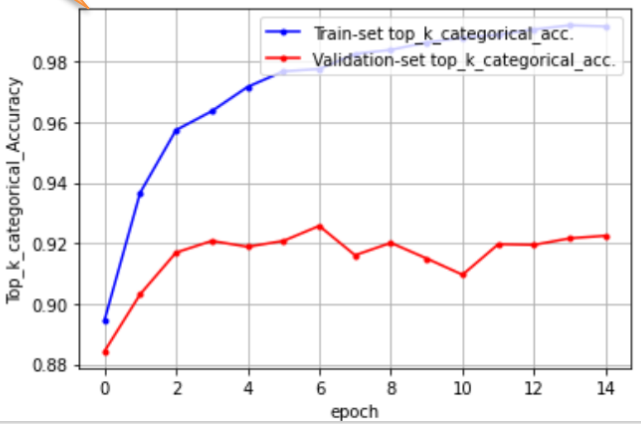
Mobilenet

```
[ ]  import numpy as np
     import matplotlib.pyplot as plt

     acc = history.history['
     val_acc = history.histo

     x_len = np.arange(len(a

     plt.plot(x_len, acc, marker='.', c='blue', label="Train-set ac
     plt.plot(x_len, val_acc, marker='.', c='red', label="Validation-se

     plt.legend(loc='upper right')
     plt.grid()
     plt.xlabel('epoch')
     plt.ylabel('Accuracy')
     plt.show()
```



Top K Categorical Accuracy는
90%이상의 결과 도출!!!

```
[ ]  top_k_acc = history.history['top_k_categorical_accuracy']
     val_top_k_acc = history.history['val_top_k_categorical_accuracy']

     x_len = np.arange(len(top_k_acc))

     plt.plot(x_len, top_k_acc, marker='.', c='blue', label="Train-set top_k_c
     plt.plot(x_len, val_top_k_acc, marker='.', c='red', label="Validation-set

     plt.legend(loc='upper right')
     plt.grid()
     plt.xlabel('epoch')
     plt.ylabel('Top_k_categorical_Accuracy')
     plt.show()
```



06

모델을 활용한
웹 서비스에 대한 소개

07

팀 구성 및 역할,
후속과제

무슨 와인2조

팀 구성 및 역할

팀장

박경희

데이터 탐색
데이터 전처리
중간발표 및 PPT
모델 3 코드구현
최종발표 및 PPT

팀원

강상범

데이터 탐색
데이터 웹크롤링
모델 3 데이터 웹 크롤링
모델 3 코드구현
전체 모델 수정 및 보완

팀원

김예린

데이터 분석
데이터 시각화
모델 1 코드구현
모델 2 코드구현
Django 백엔드

무슨 와인2조

팀 구성 및 역할

팀원

박수진

데이터탐색
데이터 웹 크롤링
모델 1 코드 구현
모델 2 코드 구현
Django 백엔드

팀원

서태원

데이터분석
데이터 웹 크롤링
모델 2 코드 구현
모델 3코드 구현
최종 발표 및 PPT

팀원

조희정

데이터 시각화
데이터 전처리
모델 1 코드구현
모델 2 코드구현
Django 프론트 엔드

continued

후속과제

1. 바디감, 아로마등 와인의 특성도 크롤링 하여 더 구체적인 와인 정보를 기반한 추천시스템으로 보완

2. 이미지 기반 추천 시스템 내 사진 class를 더욱 확대해 다양한 음식에 맞게 추천 시스템 보완

3. Django 관련 보완 (와인 입력창 자동완성 기능 구현)