ITWILL BIG DATA

Wine Recommender

Project Summary:

Machine learning recommendation system using wine information and review data from Vivino website

Techniques used:

- Web Scraping
- Collaborative Filtering
- Content-based Filtering
- Web developing

Contents

01

Introduction

Bacground of Project

Guideline

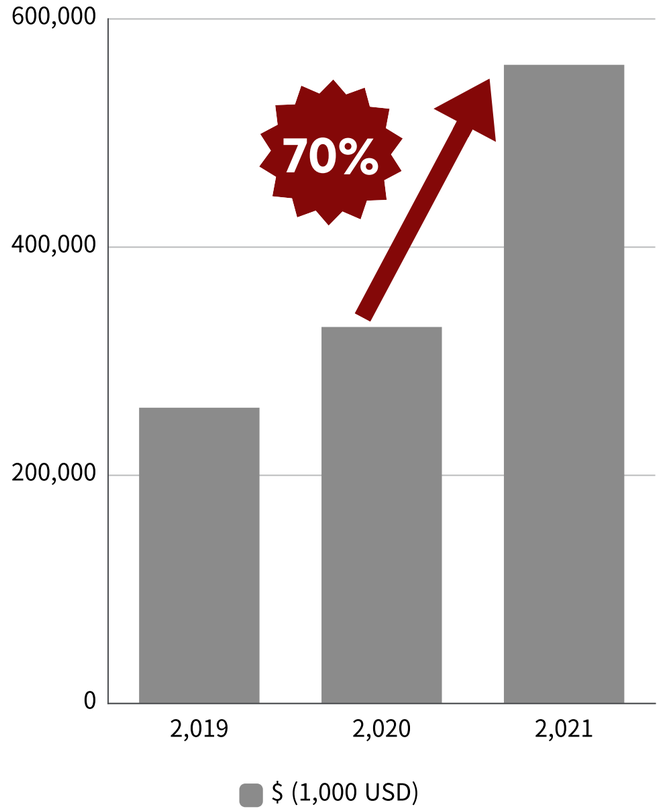
01

Background of Project

Amount of Imports by Liquor Type ($ billions USD)

	2020 JAN~JUL	2021 JAN~JUL
Wine	1.6	3.25
Beer	1.4	1.3
Spirits	0.6	1.0
Etc.	2.7	2.5

Amount of Imported Wine by Year



Source: Korea Customs Service

02

Background of Project

nd Most Attractive Wine Market Globally

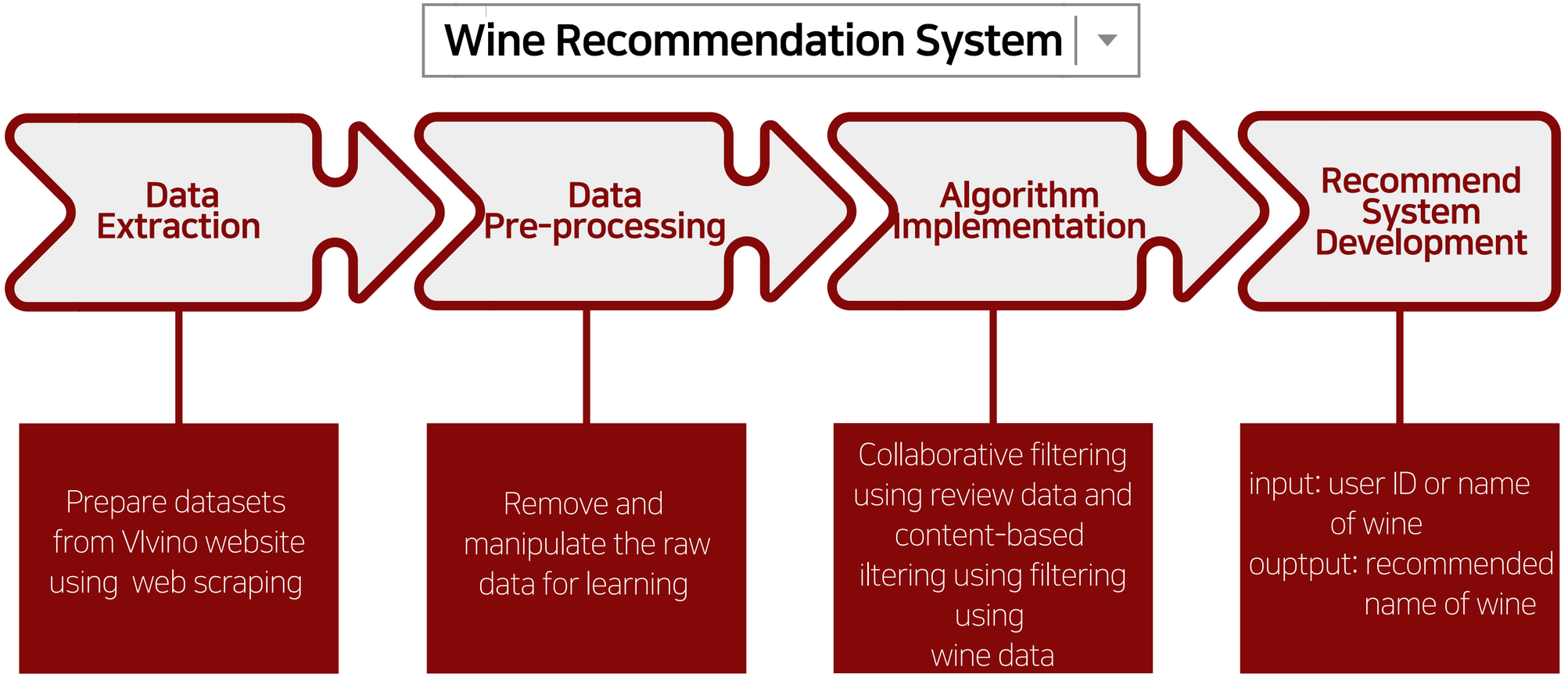
Due to COVID-19,

Koreans drinking alone at home increased significantly

Source : Wine Intelligence

03

Guideline



02

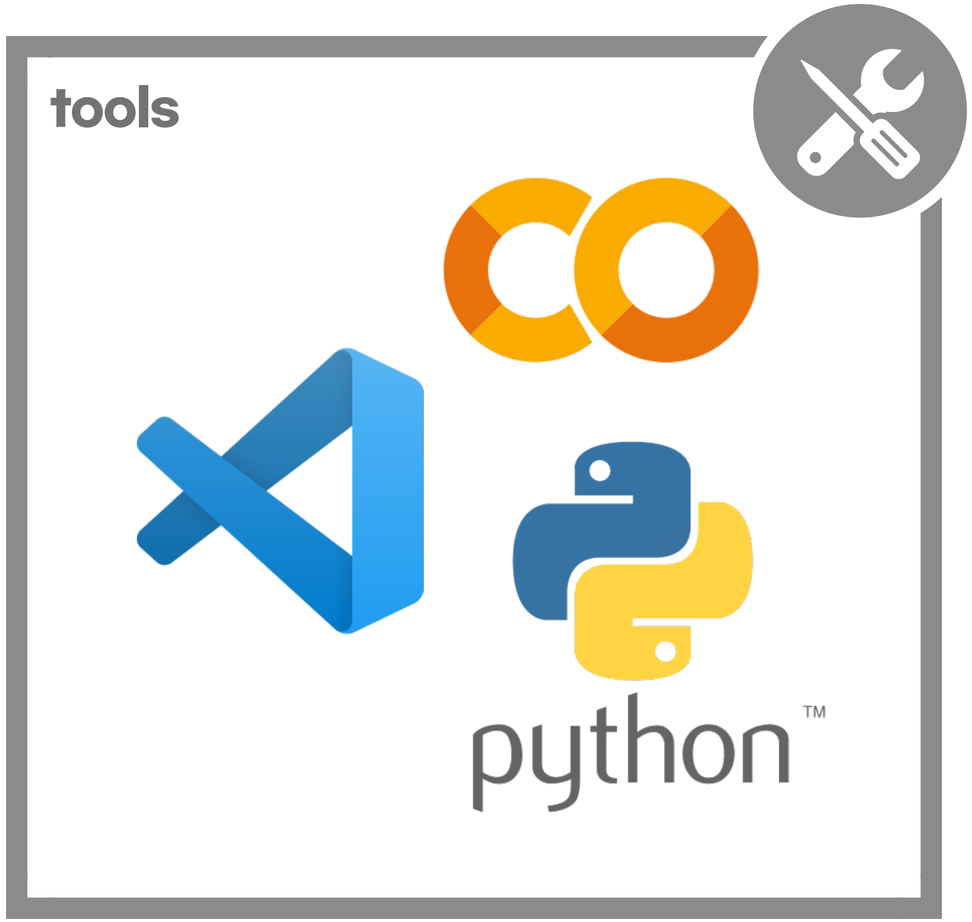
Data Extraction

Working Environment

Data Extraction

04

Working Environment



05

Data Extraction

15M wines

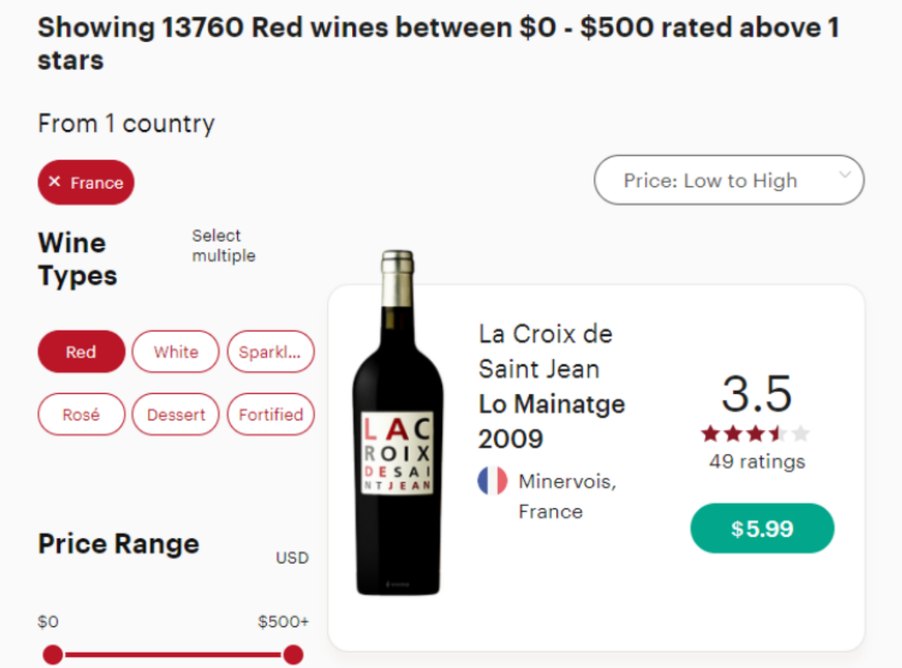
59M users

241M ratings

- World's biggest online wine community
- Provides services such as wine information retrieval, price comparison, and purchase
- Recognize labels with smartphone cameras and provide information

06

Data Extraction




```python
r1 = requests.get(
    "https://www.vivino.com/api/explore/explore?per_page=50",
    params={
        "country_code": 'us',
        "state": 'ca',
        "currency_code": "USD",
        "country_codes[]": ['fr'],
        "grape_filter": 'varietal',
        "min_rating": 1,
        "order_by": 'price',
        "order": 'asc',
        "price_range_max": '500',
        "price_range_min": '0',
        "wine_type_ids[]": ['1'],
    },
    headers={
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
    },
)
```

Data collection proceeds using the API provided by Vivino

03

Data EDA & Pre-processing

EDA

Pre-processing

07 EDA Review Data

```
review.head()
```

	year	wine_id	user_rating	user_id
0	2017	4664487	4.0	1367356
1	2017	4664487	4.0	22285577
2	2017	4664487	3.5	18742969
3	2017	4664487	3.5	37874675
4	2017	4664487	3.5	19888747

Columns :
year
wine_id
user_rating
user_id

```
review.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1215808 entries, 0 to 1215807
Data columns (total 4 columns):
 #   Column       Non-Null Count    Dtype  
---  ------       --------------    -----  
 0   year         1196416 non-null  object 
 1   wine_id      1215808 non-null  int64  
 2   user_rating  1215808 non-null  float64
 3   user_id      1215808 non-null  int64  
dtypes: float64(1), int64(2), object(1)
```

08 EDA

Review Data

```
for col in review.columns:
    print(review[col].describe())
```

```
count     1196416
unique         85
top          2018
freq       205602
Name: year, dtype: object

count    1.215808e+06
mean     3.966055e+00
std      5.927958e-01
min      1.000000e+00
25%      3.500000e+00
50%      4.000000e+00
75%      4.500000e+00
max      5.000000e+00
Name: user_rating, dtype: float64
```

```
review.year.unique()
```

```
array(['2017', '2018', '2020', 'N.V.', '2019', '2021', '2016', '2014',
       '2012', '2015', '2013', nan, '2010', '2011', '2005', '2009',
       '2008', '1999', '2007', '2004', '1994', '2002', '2003', '1998',
       '1996', '2001', '2006', '1988', '2000', '1990', '1997', '2018.0',
       '2013.0', '2019.0', '2017.0', '2014.0', '2016.0', '2020.0',
       '2008.0', '2015.0', '2012.0', '1981', '1995', '1982', '1970',
       '1986', '1978', '1984', '1965', '1929', '1931', '1955', '1991',
       '1987', '1946', '1954', '1989', '1993', '1985', '1964', '1966',
       '1983', '1975', '1992', '1979', '1976', '1974', '1969', '1961',
       '1967', '1968', '1971', '1977', '1980', '1973', '1933', '1908',
       '1963', '1972', '1957', '1937', '1960', '2021.0', '2010.0',
       '2011.0', '2009.0'], dtype=object)
```

09
EDA Wine Data

```
wine.head()
```

	winery	year	wine_id	title	acidity	intensity	sweetness	tannin	price	type_id	country
0	New Age	N.V.	7704484	Sweet White N.V.	-1.000000	-1.000000	-1.000000	-1.000000	8.990000	2	Argentina
1	Angulo Innocenti	2017	4664487	Nonni Malbec 2017	2.605630	3.406412	2.059130	2.221030	9.490833	1	Argentina
2	Nieto Senetiner	2018	82429	Cabernet Sauvignon 2018	3.061122	3.816159	2.006314	3.500747	9.890000	1	Argentina
3	Alamos	2020	1879	Malbec 2020	2.719179	3.786719	2.034467	2.399564	9.950000	1	Argentina
4	Terrazas de los Andes	2018	1181213	Altos del Plata - Cabernet Sauvignon 2018	3.098722	3.840139	1.568474	3.423486	9.980000	1	Argentina

Columns :
winery
year
wine_id
title
acidity
intensity
sweetness
tannin
price
type_id
country

```
wine.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29890 entries, 0 to 29889
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   winery      29890 non-null  object
 1   year        29837 non-null  object
 2   wine_id     29890 non-null  int64
 3   title       29890 non-null  object
 4   acidity     29890 non-null  float64
 5   intensity   29890 non-null  float64
 6   sweetness   27818 non-null  float64
 7   tannin      17797 non-null  float64
 8   price       29890 non-null  float64
 9   type_id     29890 non-null  int64
 10  country     29890 non-null  object
dtypes: float64(5), int64(2), object(4)
```

10 EDA Wine Data

```
for col in wine.columns:
    print(wine[col].describe())
```

```
count                       29890
unique                      22524
top       Cabernet Sauvignon 2018
freq                          180
Name: title, dtype: object
count      29890
unique        28
top       France
freq        6494
Name: country, dtype: object
count       29890
unique       8400
top       Penfolds
freq           98
Name: winery, dtype: object
count     29837
unique       84
top        2018
freq       4917
Name: year, dtype: object
```

```
wine.describe()
```

	wine_id	acidity	intensity	sweetness	tannin	price	type_id
count	2.989000e+04	29890.000000	29890.000000	27818.000000	17797.000000	29890.000000	29890.000000
mean	2.119428e+06	2.755941	2.795596	1.329250	1.919474	46.510680	2.654433
std	2.113044e+06	1.818980	1.880243	1.248798	1.948760	59.703720	4.331374
min	4.600000e+01	-1.000000	-1.000000	-1.000000	-1.000000	1.915833	1.000000
25%	1.129395e+06	2.755015	2.509066	1.324724	-1.000000	16.990000	1.000000
50%	1.410056e+06	3.344235	3.448307	1.607825	2.846014	26.666667	2.000000
75%	2.469712e+06	3.907779	3.985017	1.923046	3.420193	49.990000	2.000000
max	1.085184e+07	5.000000	5.000000	5.000000	4.930000	500.000000	24.000000

11

Pre-processing

Review Data

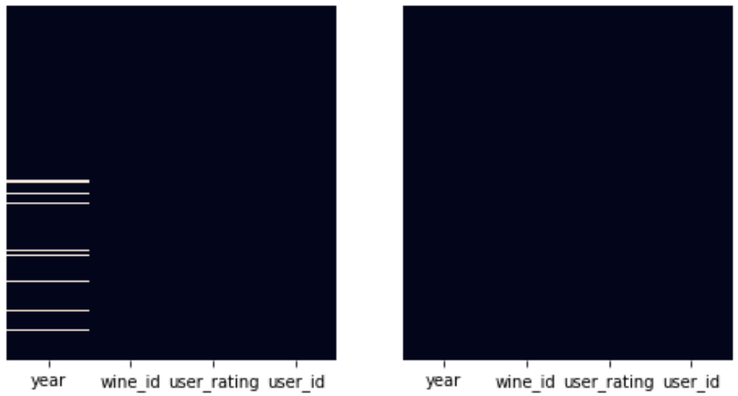
Missing Values

If year = NA,

-> remove data

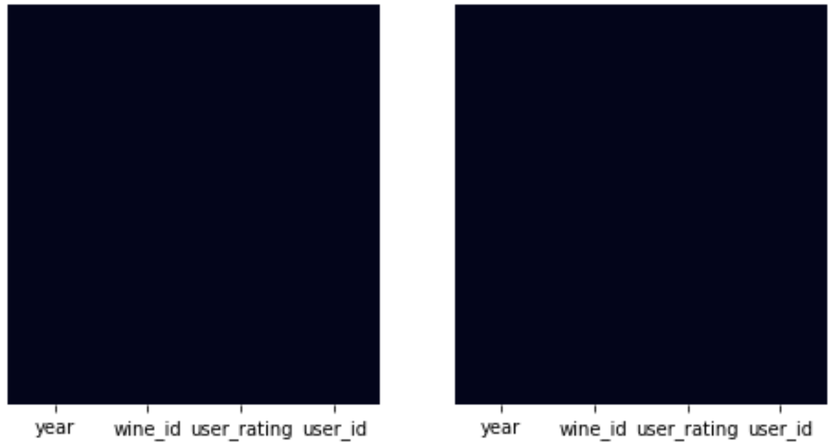
CODE:

```
fig, ax = plt.subplots(1, 2, figsize=(8, 4))
sns.heatmap(review.isna(), yticklabels=False, cbar=False, ax=ax[0])
sns.heatmap(review==-1, yticklabels=False, cbar=False, ax=ax[1])
```



OUTPUT:

```
review = review.dropna()
```



12

Pre-processing

Review Data

Create wine codes

When the title of wine is the same but the year is different,

-> Combining year + wine ID to create wine_code

(if year = NA, then save as year = 9999)

CODE:

```
y = review.year
y[y.values=='N.V.'] = 9999
y = y.astype(float).astype('int64')
review.year = y
review['wine_code'] = y.astype(str) + review.wine_id.astype(str)
review['wine_code'] = review['wine_code'].astype('int64')
review = review[['year', 'wine_id', 'wine_code', 'user_rating', 'user_id']]
review = review.reset_index(drop=True)
```

OUTPUT:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1196416 entries, 0 to 1196415
Data columns (total 5 columns):
 #   Column       Non-Null Count    Dtype
---  ------       --------------    -----
 0   year         1196416 non-null  int64
 1   wine_id      1196416 non-null  int64
 2   wine_code    1196416 non-null  int64
 3   user_rating  1196416 non-null  float64
 4   user_id      1196416 non-null  int64
dtypes: float64(1), int64(4)
```

13

Pre-processing

Wine Data

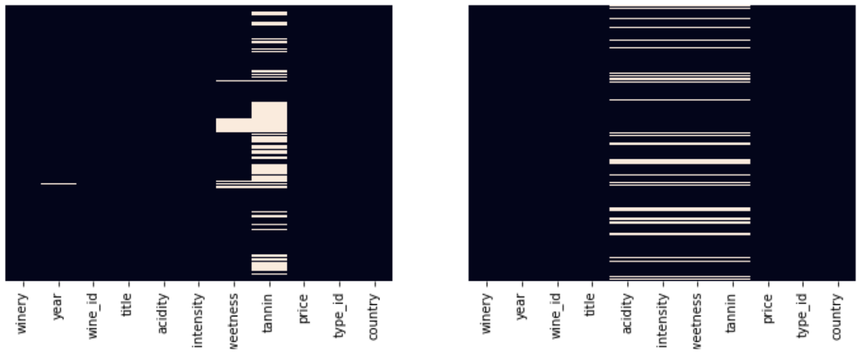
Missing Values (delete)

If acidity =/= -1, characteristic data are also missing

-> remove data

CODE:

```
fig, ax = plt.subplots(1, 2, figsize=(12, 4))
sns.heatmap(wine.isna(), yticklabels=False, cbar=False, ax=ax[0])
sns.heatmap(wine==-1, yticklabels=False, cbar=False, ax=ax[1])
```




```
wine = wine.iloc[wine.year.dropna().index]
wine = wine[wine.acidity!=-1]
```

Missing Values (average)

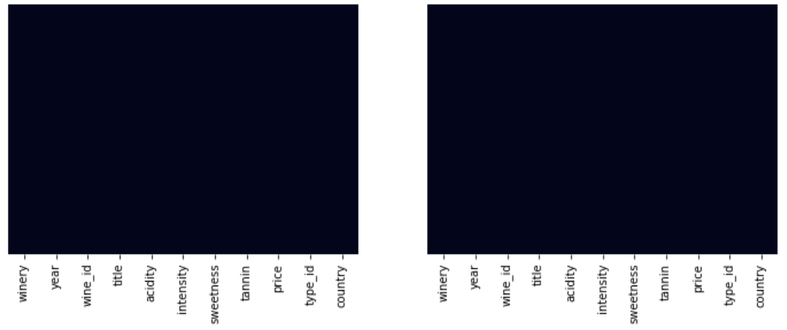
Missing values in sweetness and tannin

-> replace with average by wine type

CODE:

```
for t in wine.type_id.unique():
    tmean = wine[wine.type_id==t].tannin.mean()
    smean = wine[wine.type_id==t].sweetness.mean()
    if np.isnan(tmean):
        wine.loc[(wine.tannin.isna())&(wine.type_id==t), 'tannin'] = 0
    else:
        wine.loc[(wine.tannin.isna())&(wine.type_id==t), 'tannin'] = tmean

    if np.isnan(smean):
        wine.loc[(wine.sweetness.isna())&(wine.type_id==t), 'sweetness'] = 0
    else:
        wine.loc[(wine.sweetness.isna())&(wine.type_id==t), 'sweetness'] = smean
```



14

Pre-processing

Wine Data

Create wine codes

When the title of wine is the same but the year is different,

-> Combining year + wine ID to create wine_code

(if year = NA, then save as year = 9999)

CODE:

```
y2 = wine.year
y2[y2.values=='N.V.'] = 9999
y2 = y2.astype(float).astype('int64')
wine.year = y2
wine['wine_code'] = y2.astype(str) + wine.wine_id.astype(str)
wine['wine_code'] = wine['wine_code'].astype('int64')
wine = wine[['winery', 'year', 'wine_id', 'wine_code', 'title',
 'acidity', 'intensity', 'sweetness', 'tannin', 'price', 'type_id', 'country']]
```

OUTPUT:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24653 entries, 1 to 29889
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   winery      24653 non-null  object
 1   year        24653 non-null  int64
 2   wine_id     24653 non-null  int64
 3   wine_code   24653 non-null  int64
 4   title       24653 non-null  object
 5   acidity     24653 non-null  float64
 6   intensity   24653 non-null  float64
 7   sweetness   24653 non-null  float64
 8   tannin      24653 non-null  float64
 9   price       24653 non-null  float64
 10  type_id     24653 non-null  int64
 11  country     24653 non-null  object
dtypes: float64(5), int64(4), object(3)
```

15

Pre-processing

Wine Data

Combine wine + review datasets

- Filter only wines that exist in both wine and review data sets
- Add a review rating average column for each wine to the wine data (user_rating)
- Sort index numbers

CODE:

```
wine = wine[wine.wine_code.isin(review.wine_code.unique())]
review = review[review.wine_code.isin(wine.wine_code.unique())]

wr = review.groupby('wine_code').user_rating.mean()
wine = wine.sort_values('wine_code')
wine['user_rating'] = wr.values

review = review.reset_index(drop=True)
wine = wine.reset_index(drop=True)
```

OUTPUT:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21925 entries, 0 to 21924
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   winery       21925 non-null  object
 1   year         21925 non-null  int64
 2   wine_id      21925 non-null  int64
 3   wine_code    21925 non-null  int64
 4   title        21925 non-null  object
 5   acidity      21925 non-null  float64
 6   intensity    21925 non-null  float64
 7   sweetness    21925 non-null  float64
 8   tannin       21925 non-null  float64
 9   price        21925 non-null  float64
 10  type_id      21925 non-null  int64
 11  country      21925 non-null  object
 12  user_rating  21925 non-null  float64
dtypes: float64(6), int64(4), object(3)
```

16

Pre-processing

Wine Data

One-Hot Encoding & Scaling

Numerical

The range of characteristic values (acidity, intensity, sweetness, tannin) and price are different
-> Normalization Scaling

Categorical

Wine type (type_id)
-> One-Hot Encoding

CODE:

```
def clean_and_scale(df):
    type_ohe = pd.get_dummies(df['type_id'])
    df = wine.join(type_ohe)
    df = df.drop(['title', 'type_id', 'country'], axis=1).set_index('wine_code')
    scaled = MinMaxScaler().fit_transform(df)
    scaled_df = pd.DataFrame(data=scaled, index=df.index)
    return scaled_df
scaled_df = clean_and_scale(wine)
scaled_df
```

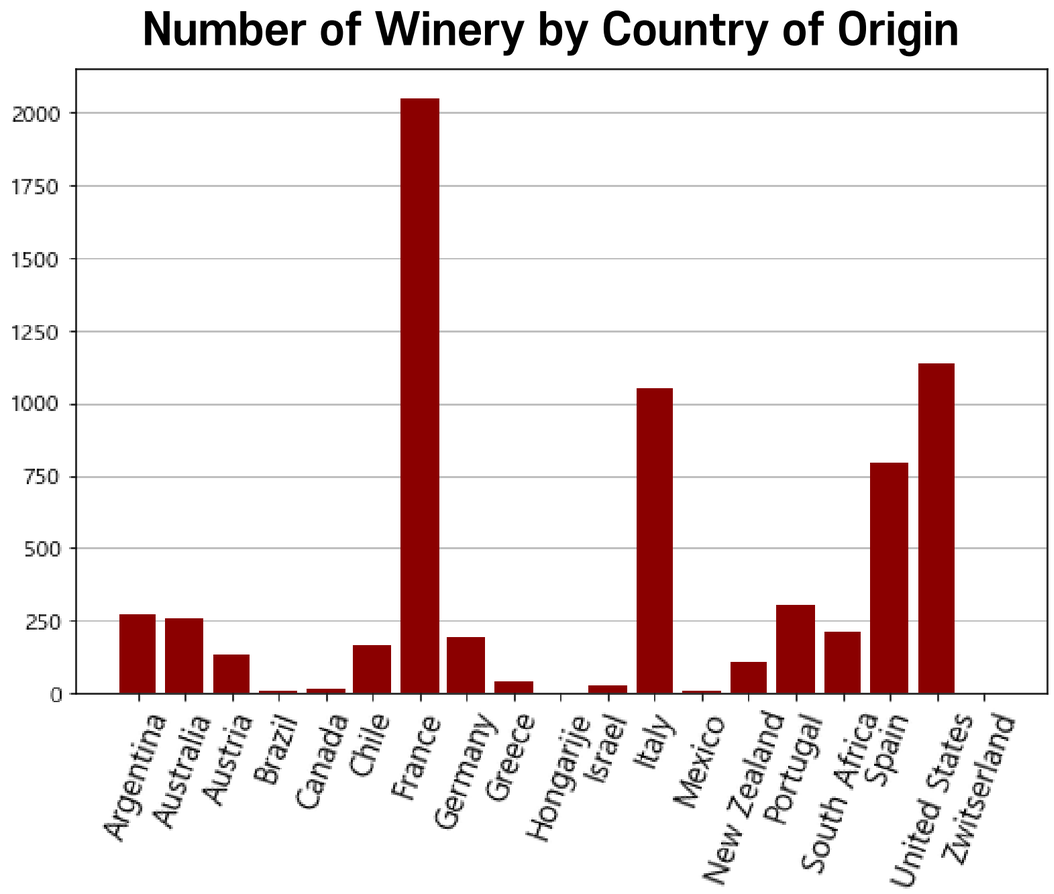
04

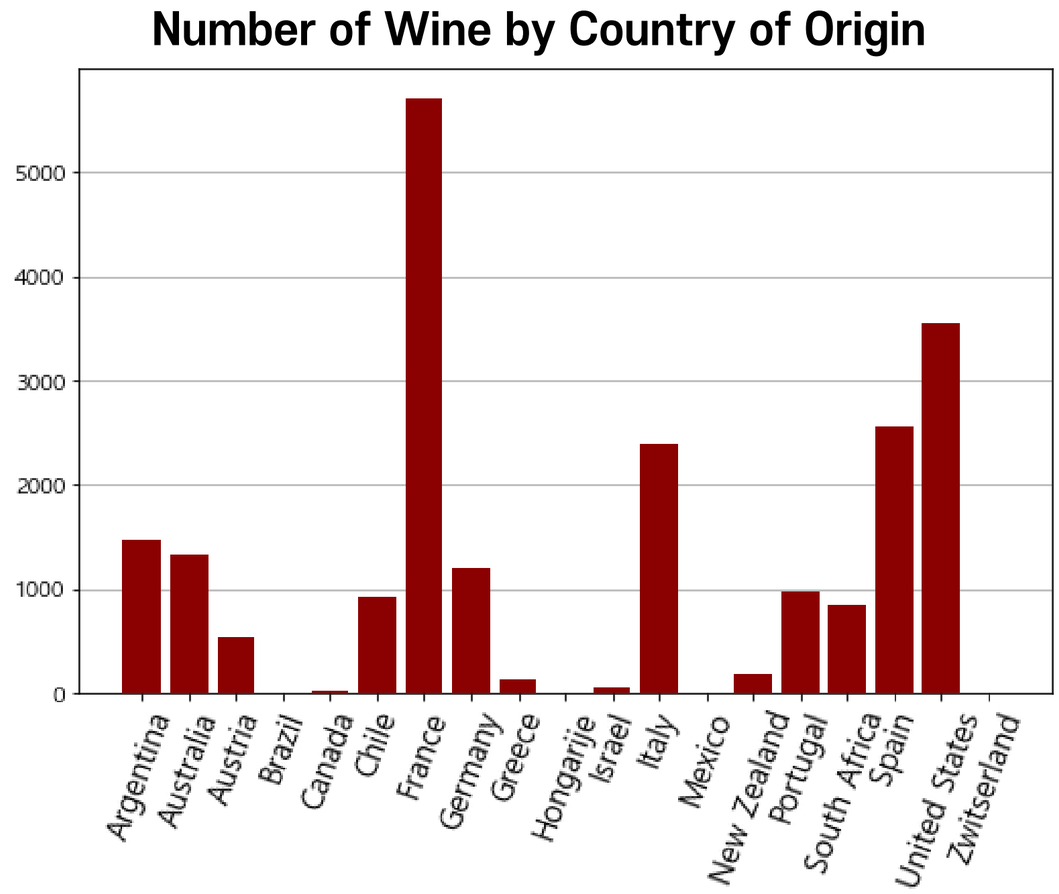
Data Visualization

Wine Data

Review Data

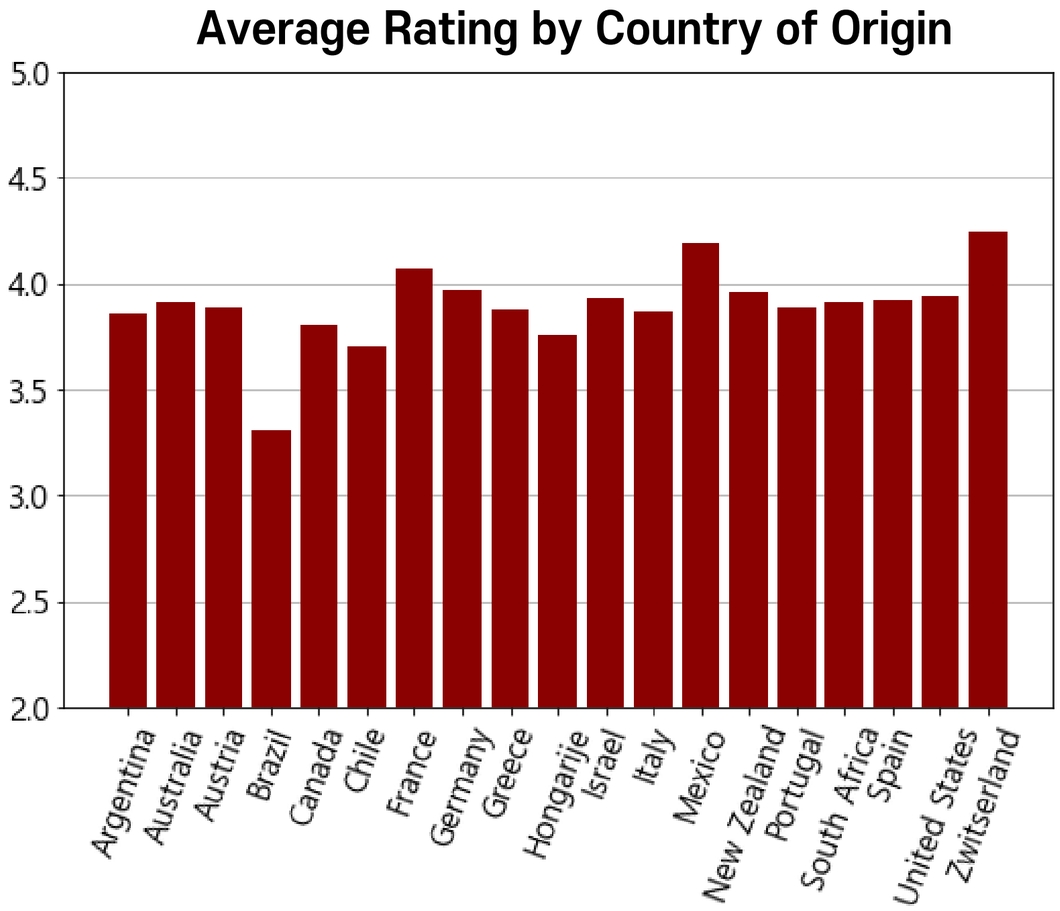
Data Visualization

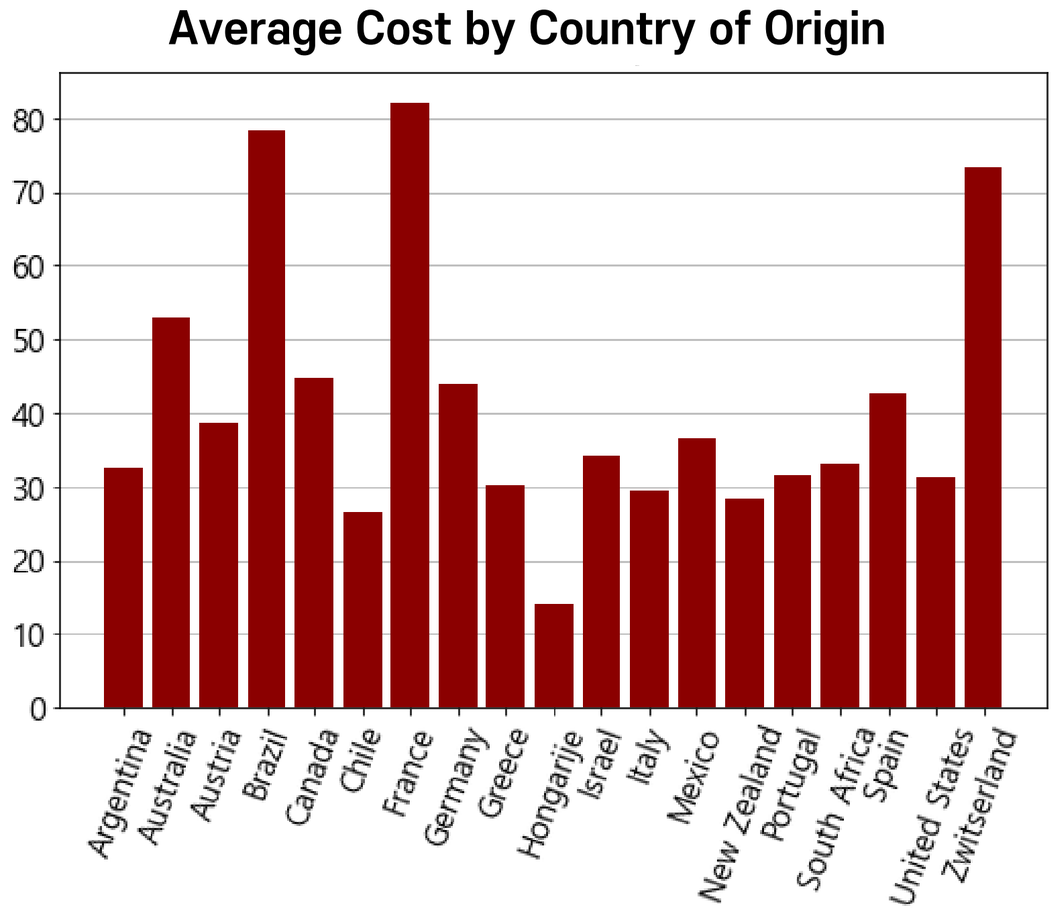




The number of winery and wine by country of origin are proportional.

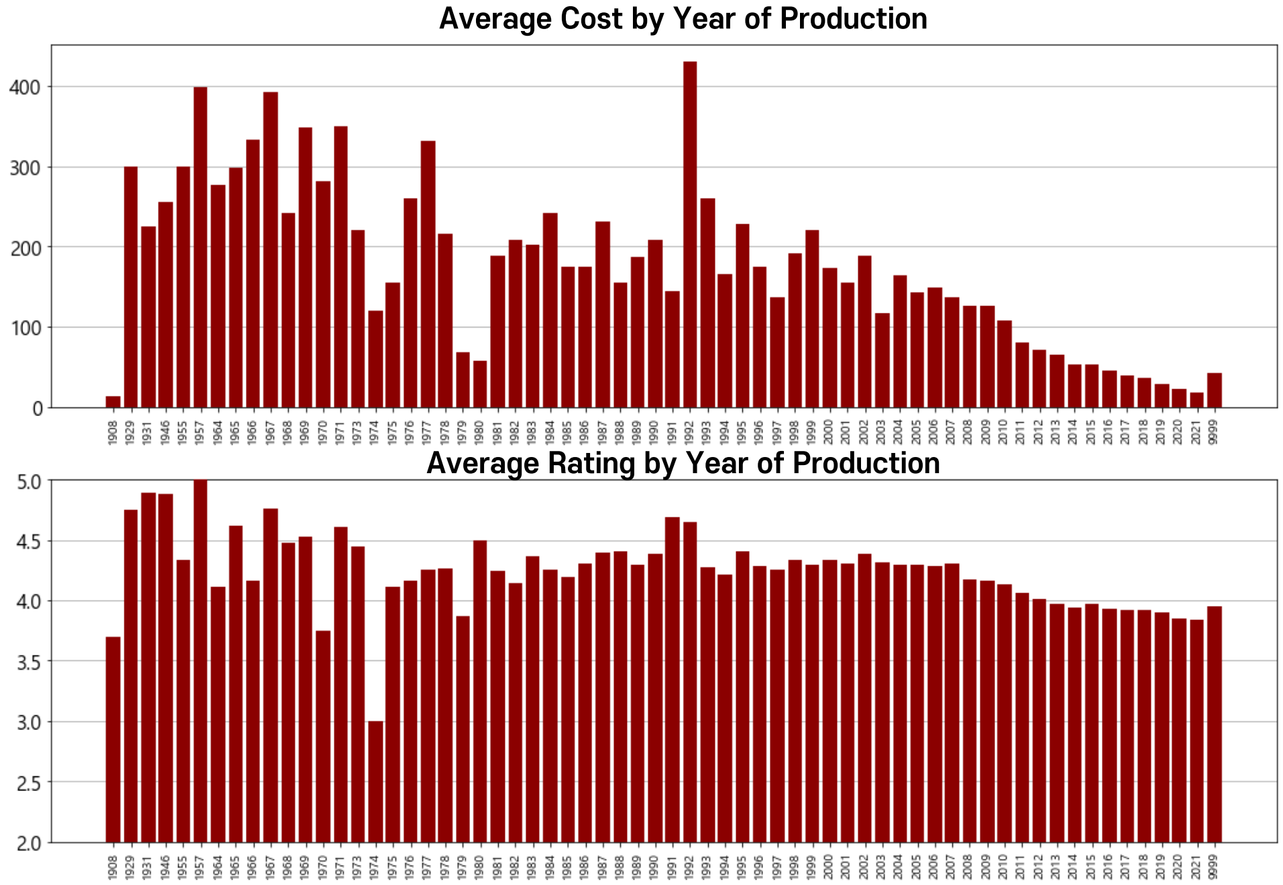
Data Visualization





Average rating and average cost by country of origin are not proportional (ex. Brazil).

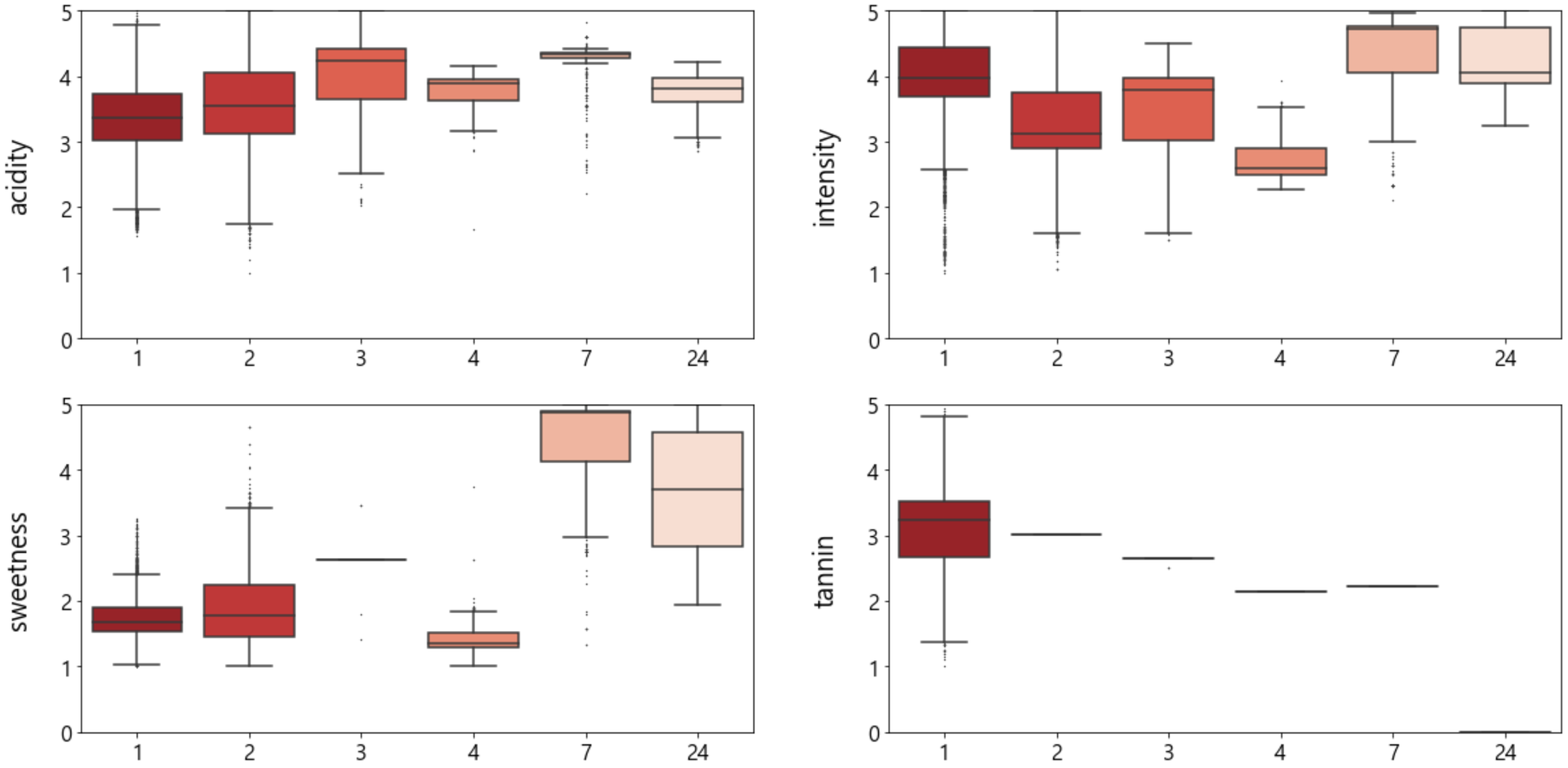
Data Visualization



- The closer the year of production, the lower the price and rating.
- Average rating varies slightly compared to price.

Data Visualization

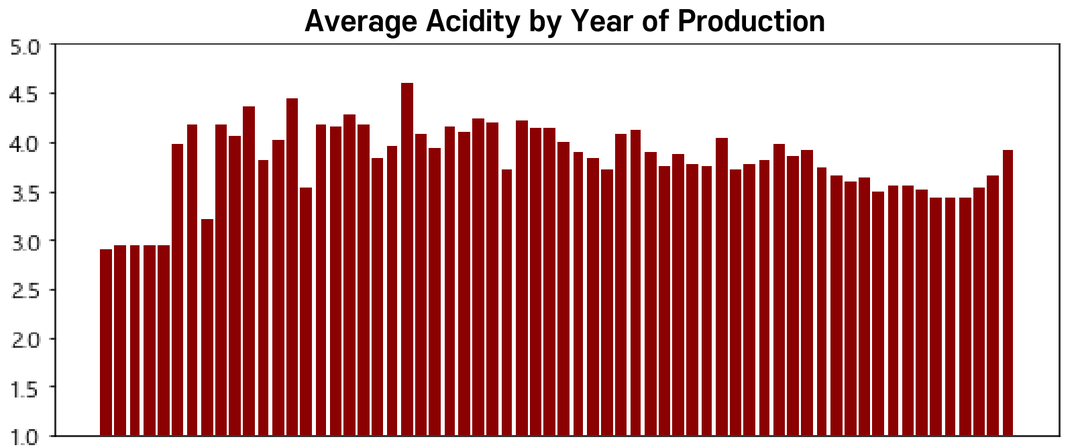
Wine Characteristic Statistics by Type

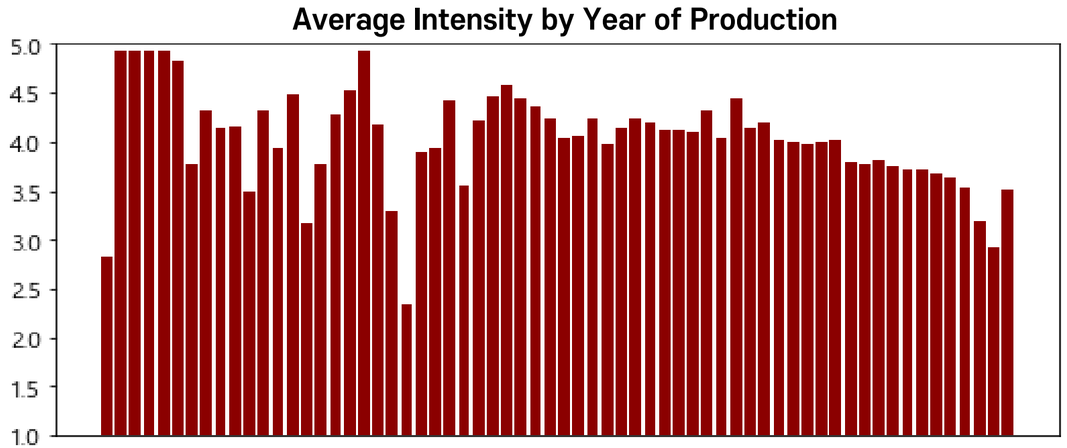


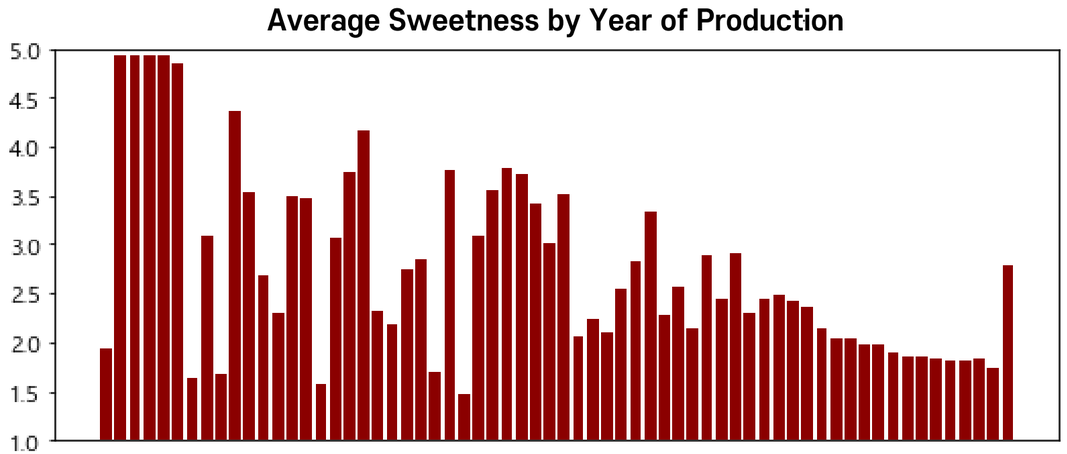
1 – red
2 – white
3 – sparkling
4 – rose
7 – dessert
24 – fortified

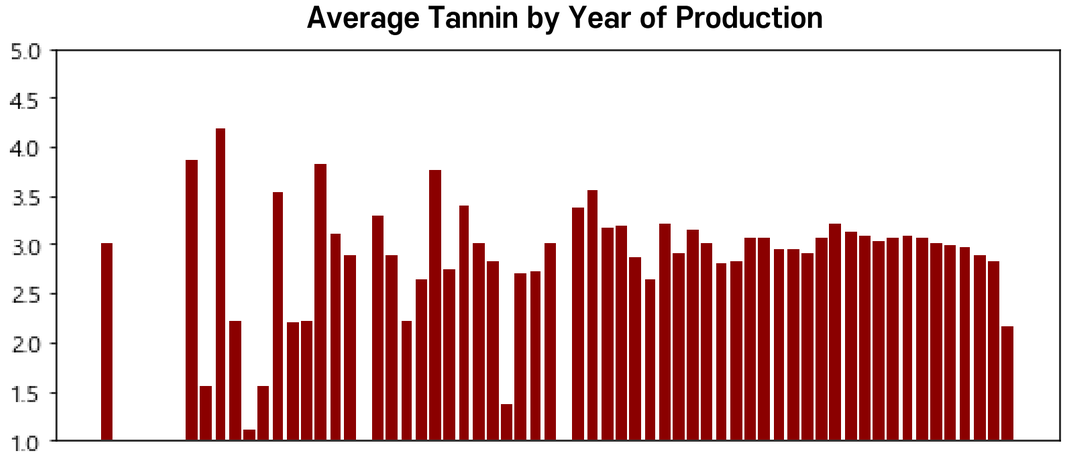
21

Data Visualization



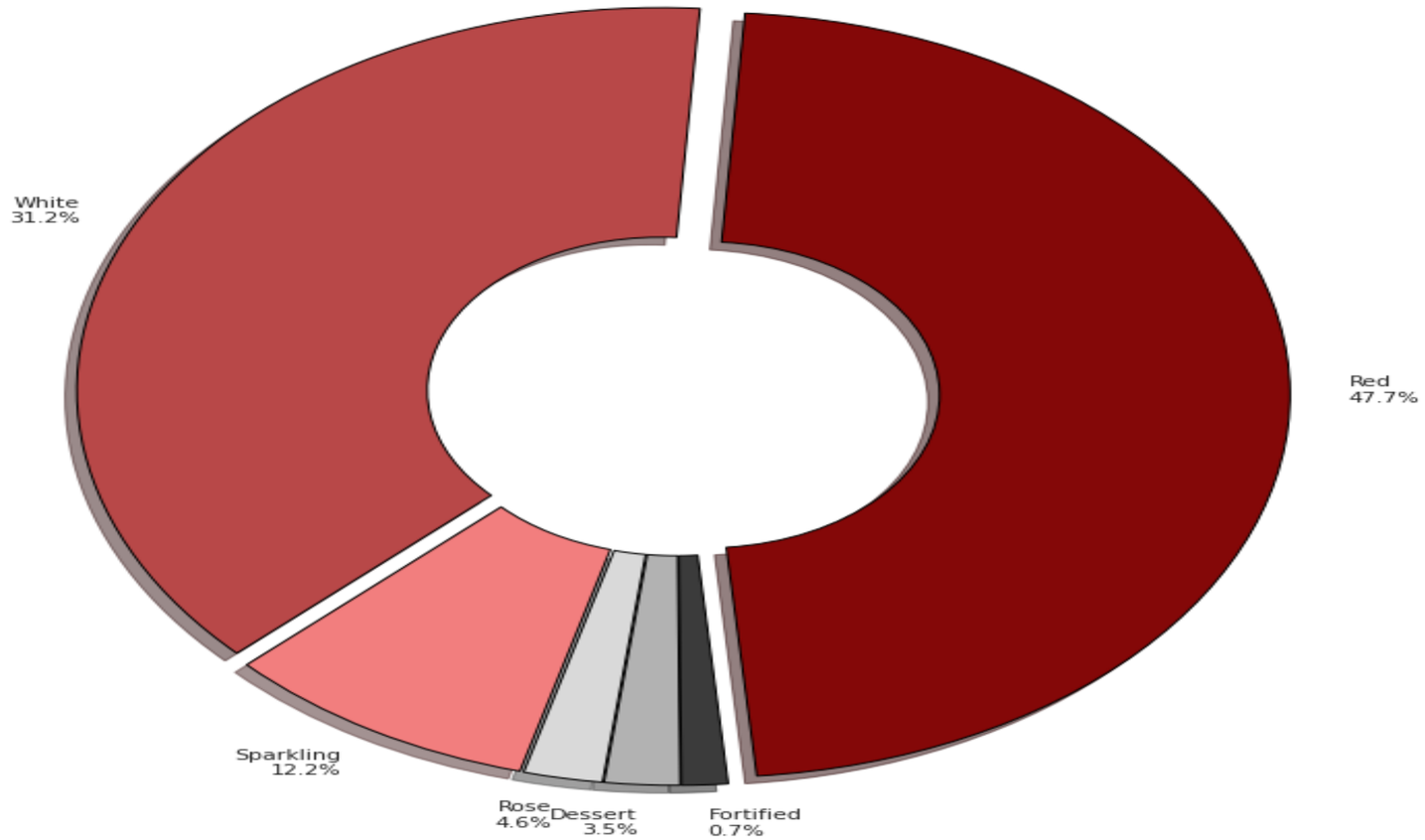






Data Visualization

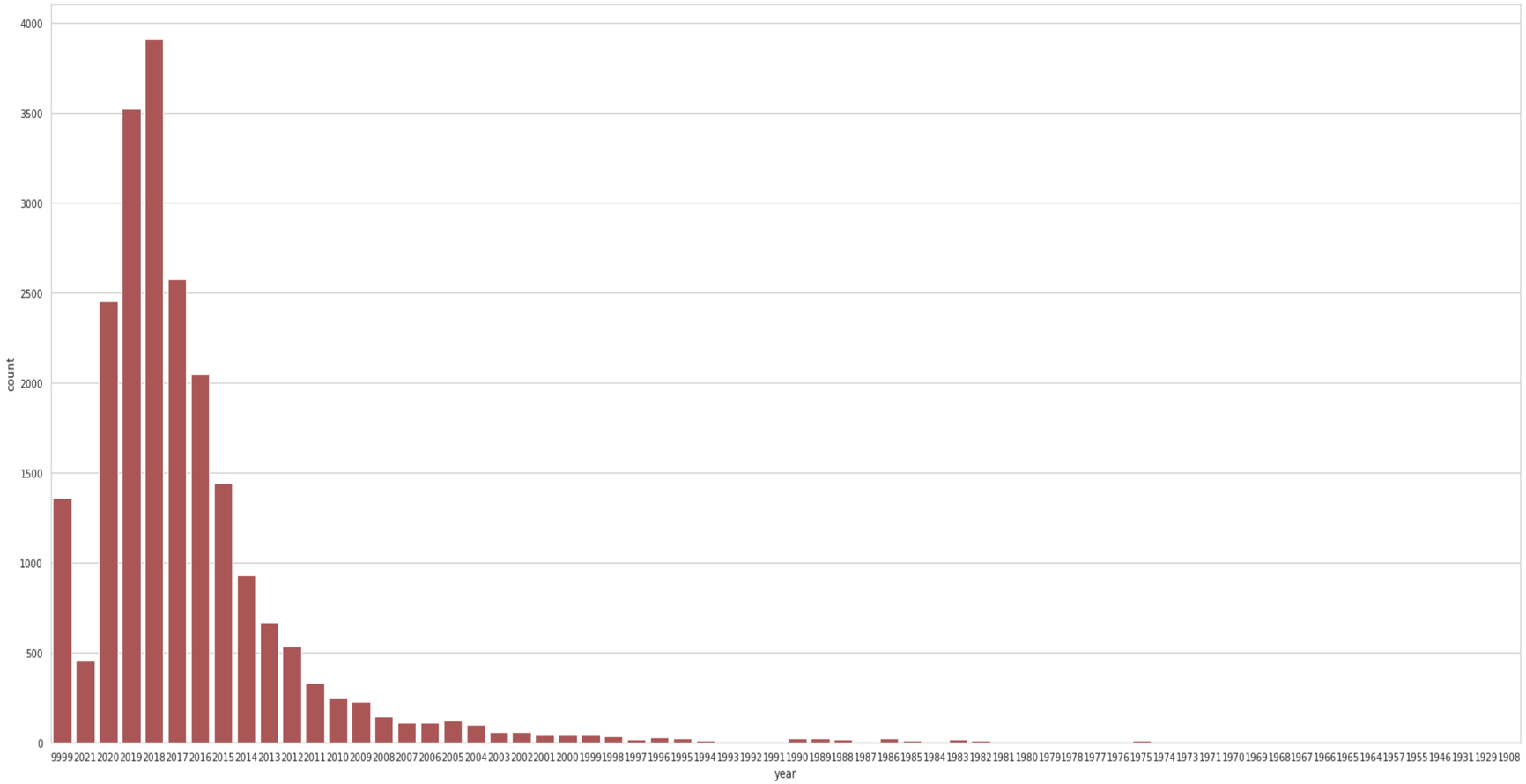
Wine Type Distribution Rate



23

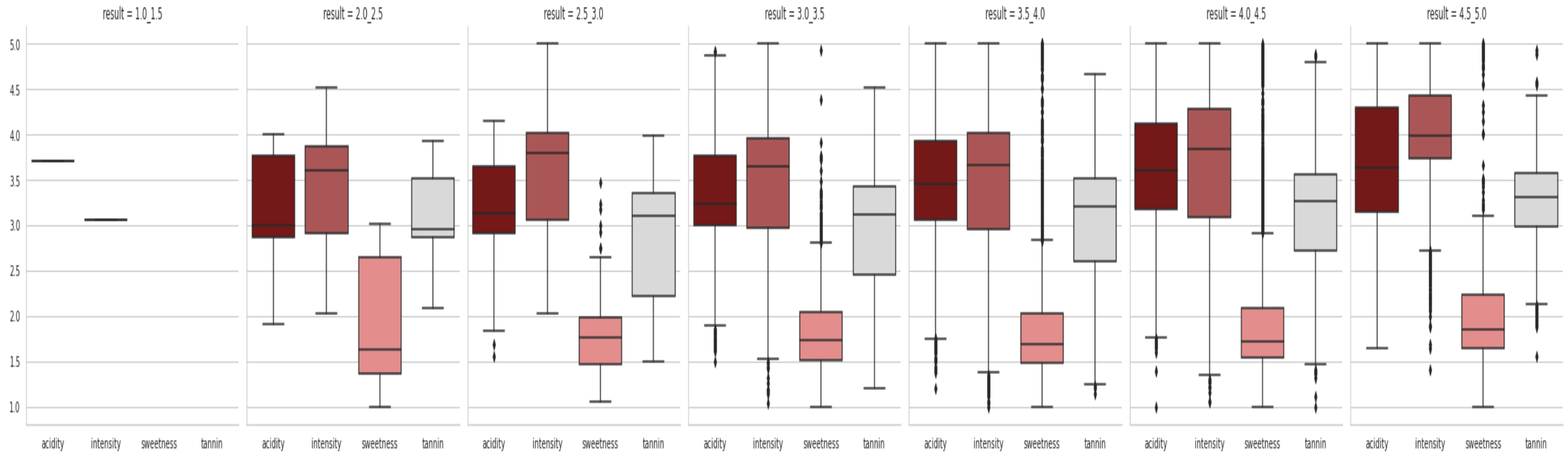
Data Visualization

Wine Vintage



24

Data Visualization



acidity **intensity**

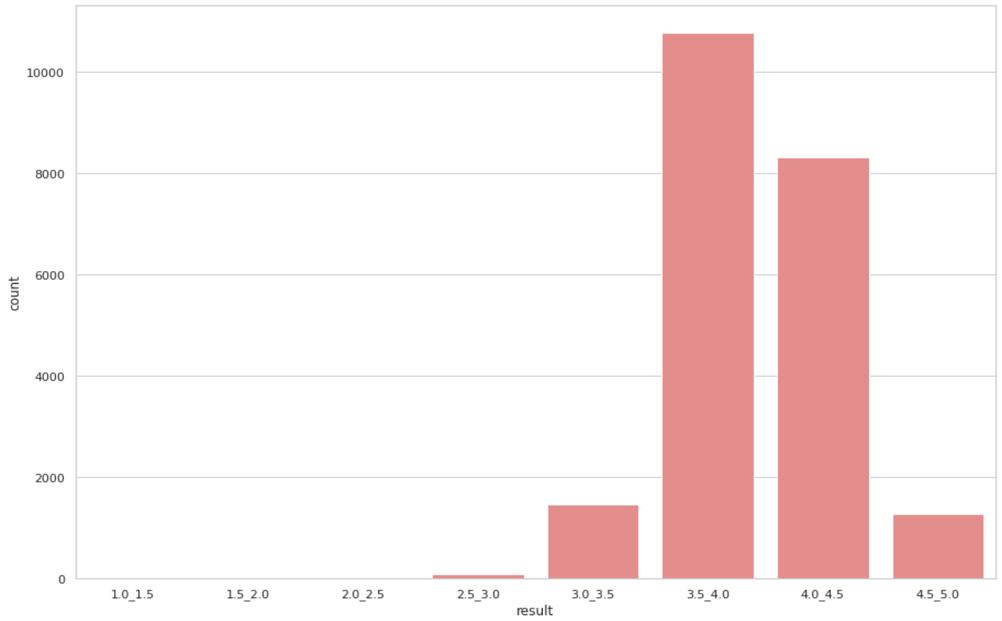
tannin **sweetness**

Characteristic Distribution by Rating

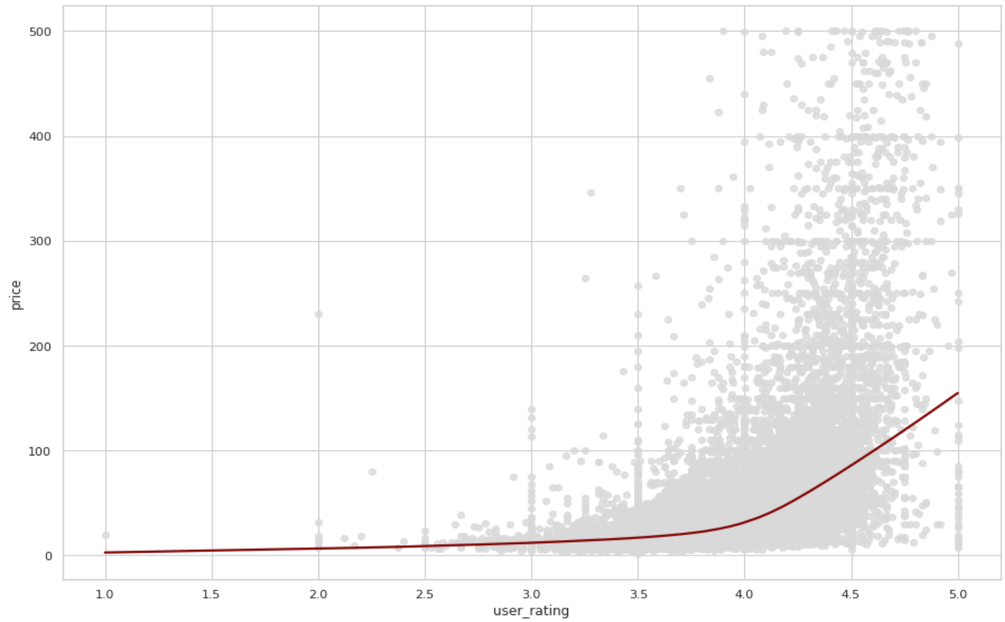
- ✓ **Characteristic distribution shows generally similar tendencies.**
- ✓ **However, wine with a slightly higher characteristic distribution received a good rating.**

25

Data Visualization



Number of Wine by Rating



Wine Price by Average Rating

✓ **Reviews are concentrated on medium-priced wines rather than on high-priced wines.**

05

ML Algorithm

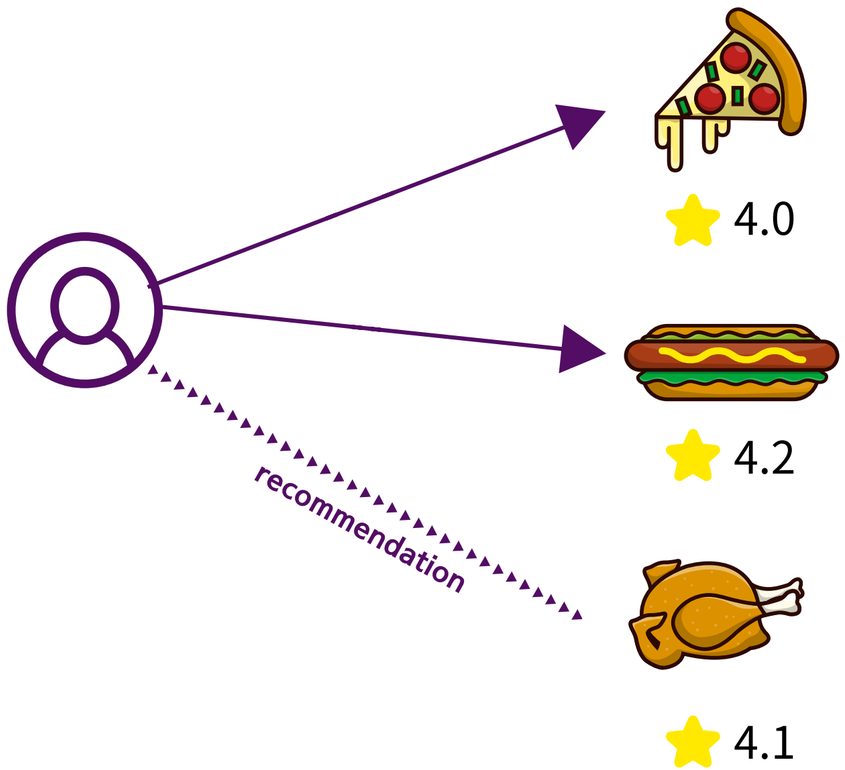
Introduction to Algorithm

Collaborative Filtering

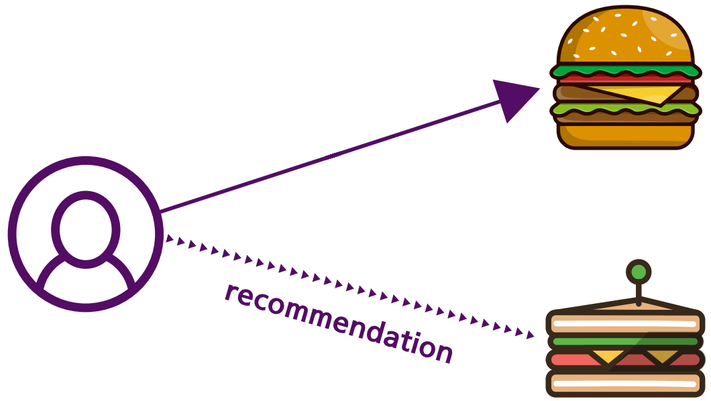
Content-based Filtering

Introduction to Algorithm

Collaborative Filtering (Item-based)



Content-based Filtering



27

Collaborative Filtering

01 Minimum number of reviews per wine and user applied:

user_cnt_min = 10

wine_cnt_min = 10

02 Convert to pivot table:

rmat = cnt_df.pivot_table('user_rating', index='user_id', columns='wine_code')

03 Fit the model, calculate cosine similarity:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

04 Apply weight:

Weighted wine with at least five common reviewers

-> Improved accuracy

05 Calculate the predicted rating:

Use similarities and weights to calculate predicted ratings for wines not evaluated by the user

05 OUTPUT:

Based on the user's reviews, recommend the top 3 wines they'll like the most

28
Collaborative Filtering

CODE:

```python
def fit(dataframe):
    df = dataframe.copy()

    # 와인 간 공통 작성 유저 수 index, columns:와인
    b1 = np.array((df.T>0).astype(float))
    b2 = b1.T
    binary = np.dot(b1, b2)
    binary_df = pd.DataFrame(data=binary)

    # 결측치 0으로 대체
    df = df.fillna(0)

    # 평점:평점 - 유저 별 평균 평점 연산 정확도 향상 기능1
    mean_by_user = df.mean(axis=1)
    df_bias = (df.T - mean_by_user).T
    df_bias_T = df_bias.transpose()

    # 코사인 유사도 계산
    isim = cosine_similarity(df_bias_T, df_bias_T)
    isim_df = pd.DataFrame(data=isim, index=df_bias_T.index, columns=df_bias_T.index)

    return df, binary_df, df_bias, mean_by_user, isim_df
```

```python
# 최근접 이웃 협업 필터링을 통한 예측 평점 계산

def predict(dataframe, item_sim_arr, mbu, bdf, n=3):
    # 사용자-아이템 평점 행렬 크기만큼 0으로 채운 예측 행렬 초기화
    ratings_arr = dataframe.values
    pred = np.zeros(ratings_arr.shape)

    # 사용자-아이템 평점 행렬의 열 크기만큼 Loop 수행.
    for col in range(ratings_arr.shape[1]):
        # 유사도 행렬에서 유사도가 큰 순으로 n개 데이터 행렬의 index 반환
        print(col, ratings_arr.shape[1])
        top_n_items = [np.argsort(item_sim_arr[:, col])[:-n-1:-1]]
        # b : col과 공통적으로 평가를 남긴 유저가 5명 이상 있는 와인의 index
        b = bdf.iloc[bdf.iloc[:, col].values >= binary_cnt, col].index
        # col과 공통적으로 평가를 남긴 유저가 있는 와인의 개수가 0보다 크다면 top_n_items 리스트 변경
        if len(top_n_items[0][np.isin(top_n_items[0], b)]) > bwine_cnt:
            top_n_items[0] = top_n_items[0][np.isin(top_n_items[0], b)] # 정확도 향상 기능2

        # 개인화된 예측 평점을 계산
        for row in range(ratings_arr.shape[0]):
            pred[row, col] = item_sim_arr[col, :][top_n_items].dot(ratings_arr[row, :][top_n_items].T)
            pred[row, col] /= np.sum(np.abs(item_sim_arr[col, :][top_n_items]))

    pred_df = pd.DataFrame(data=pred, index=dataframe.index, columns=dataframe.columns)
    pred_df = (pred_df.T + mbu).T # 뺏던 유저별 평균을 다시 더해줌
    return pred_df
```

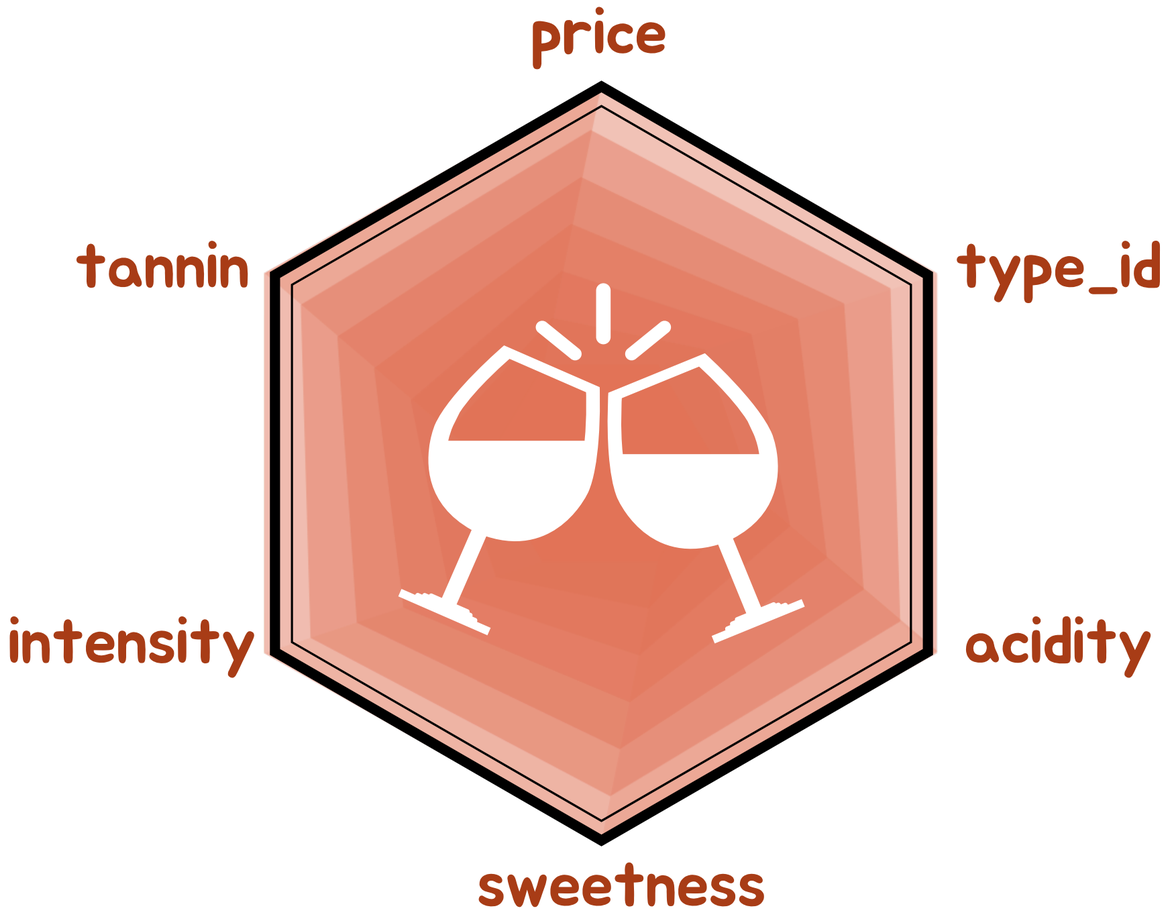
```python
def recommend(dataframe, pred_df, id, top_n=5):
    user_rating = dataframe.loc[id,:]
    already_seen = user_rating[ user_rating > 0].index.tolist()
    wines_list = dataframe.columns.tolist()
    unrat_list = [ wine for wine in wines_list if wine not in already_seen]

    recomm_wines = pred_df.loc[id, unrat_list].sort_values(ascending=False)[:top_n]
    return recomm_wines
```

Content-based Filtering

KNN Algorithm

Similarity Factors of Consideration:



30

Content-based Filtering

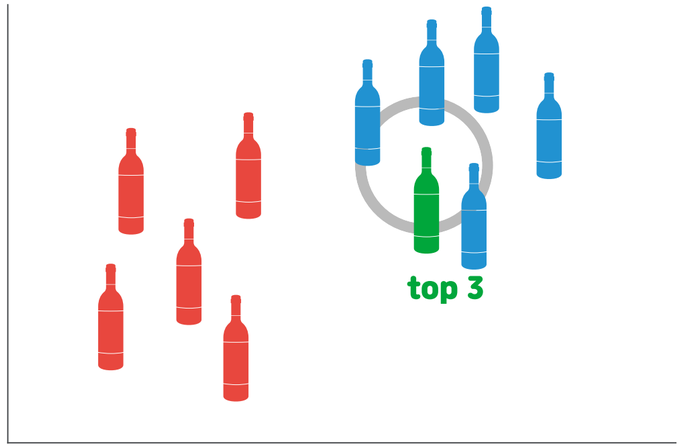
KNN Algorithm

01 Columns used:
acidity, intensity, sweetness, tannin, price, type_id

02 Fit the knn model using the scaled dataset:
knn = NearestNeighbors(metric='manhattan', algorithm='brute', n_neighbors=7)

03 Find the nearest three wines:
knn.kneighbors(scaled_df.loc[[wine_id[i]]], (4), return_distance=False)

04 Recommend top 3 wines:



31

Content-based Filtering

KNN Algorithm

CODE:

```
def knn_recommend():
    # 와인 이름 입력
    wine_input = input('와인 이름을 입력하세요: ')
    wine_input = wine_input.lower().strip()
    winename = list(wine[wine['title'].str.contains(wine_input,
                                                    case=False, regex=False)].index)
    idx = 0
    if not winename:
        print('\n##### 데이터 베이스에 없는 와인입니다. #####')
        return None
    else:
        # 해당 와인의 인덱스 추출
        a = wine['title'].str.findall(wine_input, flags=re.IGNORECASE).to_numpy().nonzero()[0]
        try:
            idx = set(a).pop()
        except KeyError:
            print('\n##### 데이터 베이스에 없는 와인입니다. #####')
            return None
    # 인덱스로 와인 코드 찾기
    wine_id = wine.iloc[idx]['wine_code']
    # KNN 모델 피팅
    knn = NearestNeighbors(metric='manhattan', algorithm='brute', n_neighbors=7)
    knn.fit(scaled_df)
    # 최근접이웃 3개 추출
    indices = knn.kneighbors(scaled_df.loc[[wine_id]], (4), return_distance=False)
    knn_df = pd.DataFrame()
    for i in indices[:,1:]:
        knn_df = knn_df.append(wine.iloc[i])
    # 3가지 추천 와인 이름 프린트
    wine_name = wine.iloc[idx]['title']
    print(f'\nProvided wine: {wine_name}')
    return knn_df
```

06

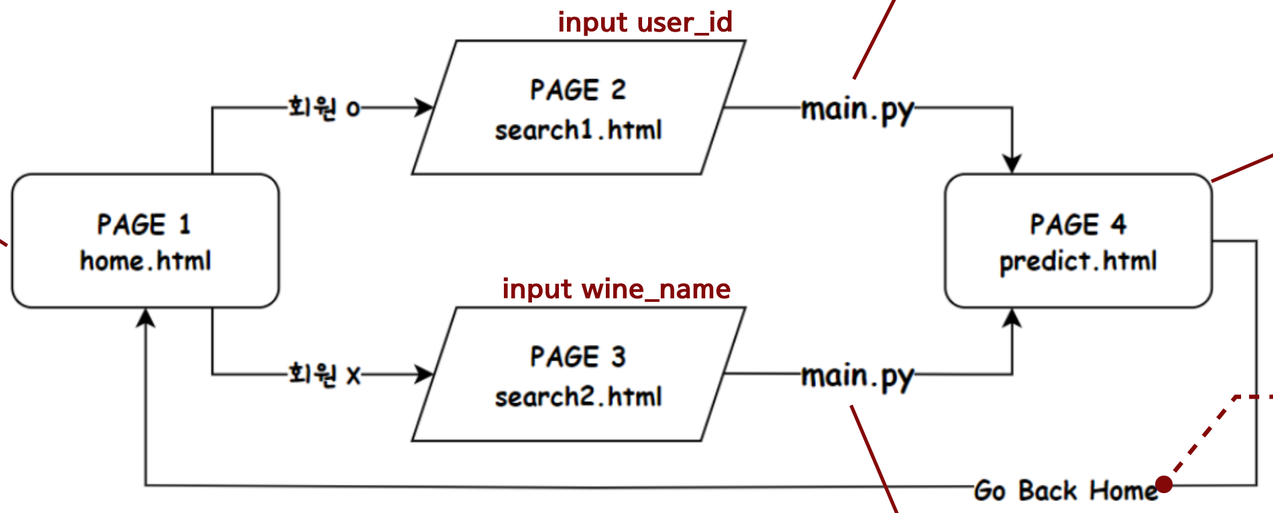
Web Development

Flowchart

Video Demonstration

32 Flowchart

div class : container
Overall home design for css files



package : Flask
request.form('user_id')
→ Collaborative Filtering
→ render_template(predict.html)

div class : card
Used css Effect: Hover

Use button class
Enter home.html address in onclick function
Press button to go to home screen

package : Flask
request.form('wine_name')
→ Content-based Filtering
→ render_template(predict.html)

Video Demonstration


Wine Project
127.0.0.1:5000
Gmail
YouTube
지도
NAVER
한컴스페이스
내 파일 - OneDrive
Wine Recommendation
VIVINO 회원이신가요?
회원
비회원
CHÂTEAU BOISSON
BORDEAUX BLANC
CHÂTEAU BOISSON
BORDEAUX ROSÉ
CHÂTEAU BOISSON
BORDEAUX
CHÂTEAU BOISSON
MOURET
GRAVES
CHÂTEAU
FAYAU
CHÂTEAU
MOURE
GRAVES
CHÂTEAU
DU JUGE
CHÂTEAU
DU JUGE

Thank you!

ITWILL BIG DATA 34

Group Members:

Kunyoon Kim, Eunbin Yoo,

Seungkyo Lee, Woojin Hyun