

Wine classifier

PinkLAB Edu

3 November, 2025

Table of Contents

| | | |
|------|---|----|
| 1 | Wine | 4 |
| 1.1 | Wine 데이터 | 4 |
| 1.2 | 기원전 7000년 무렵 조지아-아르메이나-터키 동북부 (코카서스)에서 흔적이 발견 | 4 |
| 1.3 | 신이 인간에게 내려준 선물 중 포도주만큼 위대한 가치를 지닌 것은 없다 - 플라톤 | 5 |
| 1.4 | 그런데 와인 맛은 어떻게 분류 하지?..... | 5 |
| 1.5 | 와인 데이터 받기 | 7 |
| 1.6 | PinkWink의 github에서 받을 수 있다..... | 8 |
| 1.7 | 데이터를 읽어보자..... | 8 |
| 1.8 | 두 데이터의 구조는 동일하다..... | 9 |
| 1.9 | 컬럼의 종류는? | 9 |
| 1.10 | 두 데이터를 하나로 합쳐보자 | 10 |
| 1.11 | quality 컬럼은 3부터 9등급까지 | 10 |
| 1.12 | histogram을 그려보자 | 11 |
| 1.13 | 레드/화이트 와인별로 등급 Histogram은? | 12 |
| 1.14 | 잠깐, plotly.express는 아주 간편하고 강력한 기능을 제공한다 | 13 |
| 2 | 레드 와인 / 화이트 와인 분류기 | 14 |
| 2.1 | 먼저 라벨을 분리하고 | 14 |
| 2.2 | 데이터를 훈련용과 테스트용으로 나눠주자 | 14 |
| 2.3 | 훈련용과 테스트용이 레드/화이트 와인에 따라 어느정도 구분되었을까..... | 15 |
| 2.4 | 결정나무 훈련 | 15 |
| 2.5 | 학습 결과는? | 16 |
| 3 | 데이터 전처리 - MinMaxScaler와 StandardScaler | 17 |
| 3.1 | 와인 데이터의 몇 개 항목의 Boxplot을 그려보자..... | 17 |
| 3.2 | 이럴 때 쓰는 것이 MinMaxScaler와 StandardScaler이다 | 18 |
| 3.3 | MinMaxScaler는 뭘까? | 19 |
| 3.4 | StandardScaler는? | 20 |
| 3.5 | MinMaxScaler를 적용해서 다시 학습 | 21 |
| 3.6 | StandardScaler를 적용 | 21 |
| 3.7 | 아무튼, 결정나무는 화이트와인과 레드와인을 어떻게 구분할까?..... | 22 |

| | |
|------------------------------------|----|
| 3.8 레드와인과 화이트와인을 구분하는 중요 특성? | 23 |
| 4 와인 맛에 대한 분류 - 이진 분류 | 24 |
| 4.1 quality 컬럼을 이진화하자..... | 24 |
| 4.2 레드/화이트 와인 분류와 동일 과정을 거치자..... | 24 |
| 4.3 응? 100프로? 가능한가? | 25 |
| 4.4 왜 이런 일이 생겼는지 확인해보자 | 25 |
| 4.5 다시..... | 26 |
| 4.6 결과..... | 26 |
| 4.7 어떤 와인을 “맛있다”고 할 수 있나? | 27 |

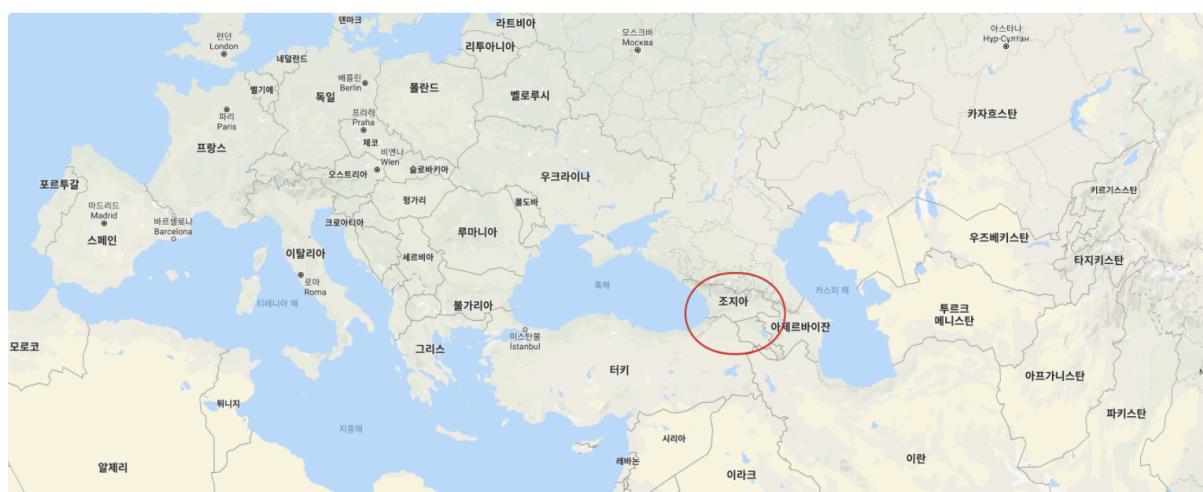
1 Wine

1.1 Wine 데이터

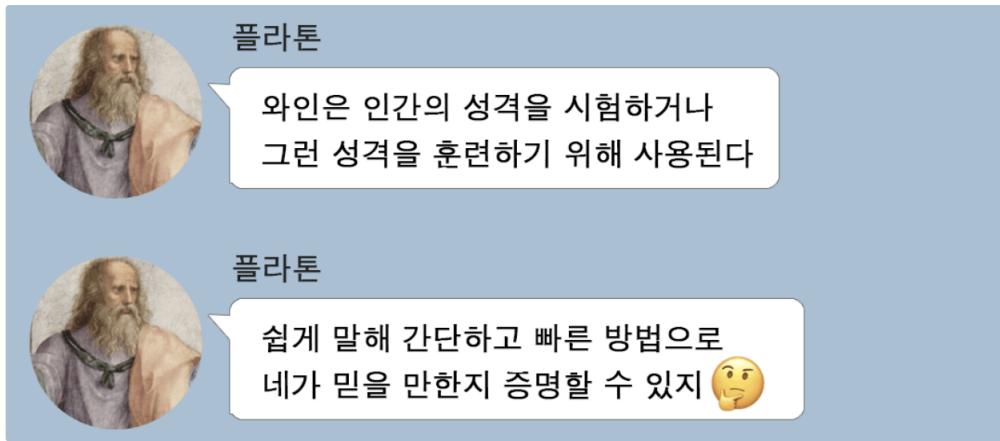


- 분류 문제에서 많이 사용하는 Iris 꽃 데이터만큼 알려지진 않았지만, 와인데이터도 많이 사용한다.
- 인류 역사에서 최초의 술로 알려져 있다.

1.2 기원전 7000년 무렵 조지아-아르메이나-터키 동북부 (코카서스)에서 흔적이 발견



1.3 신이 인간에게 내려준 선물 중 포도주만큼 위대한 가치를 지닌 것은 없다 - 플라톤



1.4 그런데 와인 맛은 어떻게 분류 하지?

| 구분 | 표현 | 내용 |
|----|--------------------|--|
| 당도 | dry(드라이) | 단맛이 느껴지지 않는다. |
| | sweet(스위트한) | 당도가 높을 뿐 아니라 감미롭고 농익은 과일향이 난다. |
| 탄닌 | astringent(떫은) | 입 안이 찍 달라붙을 만큼 탄닌 맛이 강하다. |
| | hard(강한) | 레드와인은 탄닌 맛이 강하고, 화이트 와인은 신맛이 강해 몸이 쭈뼛거릴 정도 |
| | fat(매끄러운) | 풀 바디하고 입 안을 매끄럽게 감싼다. |
| | firm(견고한) | 조화롭고 확실할 때 쓴다. 악하다는 말과 반대되는 표현이다. |
| | chewy(씹히는 듯한) | 탄닌이 많고 맛이 강하지만 억세지 않다. |
| | soft(부드러운) | 거친 탄닌이나 강한 신맛이 없어 부담 없이 즐길 수 있다. |
| | aggressive(억센) | 잇몸이 아릴 정도로 신맛. 또는 탄닌이 너무 많아 목구멍 뒷부분이 바싹 마를 정도의 신맛이 난다. |
| 산도 | piercing(콕콕찌르는 듯한) | 산도가 아주 높을 때나 과일향이 진동할 때 느낄 수 있다. |
| | crisp(상쾌한) | 신맛이 적당히 들어 있어 상쾌한 기분이 든다. |
| | prickly(알싸한) | 잔류 이산화 탄소 가스로 인해 거품이 약간 일어난다. 깔끔한 화이트 와인에서는 무척 산뜻한 느낌을 준다. |
| | fresh(신선한) | 싱싱한 과일맛과 신맛이 조화를 이루고 있다. |
| | flabby(맛이 연약한) | 신맛이 부족해서 맛이 분명하지 않다. |
| | tart(시큼한) | 덜 익은 사과처럼 매우 톡 쏘면서 신맛이 난다. 기세가 약하고 과일향이 적으면서 산도가 높아 무척 시다. |
| 알콜 | powerful(향이 강렬한) | 다양한 맛과 향이 담겨진 와인을 표현하지만, 특히 알코올 함량이 높은 경우 |
| | stony(돌처럼 단단한) | 드라이하고 미네랄 냄새처럼 분필향이 나지만 활기는 떨어진다. |
| | rich(감칠맛이 나는) | 맛이 무겁고 진하면서도 향이 적당하고 알콜이 가득하다. |
| | light(라이트한) | 알코올이나 바디가 적어 깔끔한 맛이 난다. |

| | | |
|---------|---|--|
| 향기 | deep(깊이 있는) | 향이 풍부하다. |
| | aromatic(아로마가 그윽한) | 모든 와인에는 아로마가 있다. 그러나 아로마가 그윽한 와인은 특히 톡 쏘거나 향기가 진하다. |
| | rounded(향이 조화로운) | 향이 지나치게 자극적이지 않고 만족스럽다. |
| | neutral(중립적인) | 향이 뚜렷하지 않다. |
| | complex(복잡 미묘한) | 여러 가지 향이 함께 느껴진다. |
| 풍미 | 식물 향 grassy(풀냄새가 나는) green(풋풋한) | 갓 베어 낸 풀냄새가 난다. 고추 열매, 구스베리 또는 라임향이라고 하는것이 더 정확하다. 제대로 숙성되지 않아 맛이 기대에 미치지 못하거나, 풀잎 냄새 ^[35] 일부 화이트 와인에서는 구스베리나 사과향이 어울려 후레쉬하고 톡 쏘는 맛을 낸다. |
| | 과일 향 ripe(농익은) jammy(잼 같은) | 잘 익은 포도로 만든 와인에서 나는 맛 좋은 과일향이다. 졸인 과일향이 난다. 주로 레드 와인에서도 풍긴다. |
| | 향신료 향 spicy(향긋한 또는 매콤한) | 후추, 계피 등의 향 ^[36] 이다. |
| | meaty(육질의) | 진한 레드 와인에서 느껴지는 강하고 씹히는 맛으로 고기의 육즙이 연상된다. |
| | 오크숙성 향 oaky(오크향을 풍기는) toasty(토스트 냄새가 나는) buttery(버터 냄새가 나는) | 새 오크통에서 숙성된 와인은 약간 스위트한 바닐라 향, 토스트 냄새 버터 냄새가 난다. 오크숙성으로 생긴 버터 바른 토스트 냄새이다. 오크 숙성을 통해 버터 냄새가 난다. |
| | 광물질 mineral(미네랄 냄새가 나는) dusty(더스티) earthy(흙 냄새가 나는) petrolly(휘발유 냄새를 풍기는) | 독일 와인과 프랑스 루이르밸리 와인에서 자주 나는 냄새로 부싯돌이나 분필 냄새가 난다. 드라이하면서 흙 냄새가 약간 나며, 멋진 과일향과 어울리면 아주 매력적인 와인이 될 수 있다. 축축한 흙 냄새와 향을 풍긴다. 깔끔한 와인에서 아주 좋다. 리슬링으로 만든 숙성된 와인에서는 향그려운 휘발유 냄새가 난다. |
| 바디 감 | full(향이 무겁고 진한) steely(쇠같이 단단한) big(바디가 가득한) structured(맛이 짜여진) | 입 안에서 무게가 느껴진다. 강한 신맛과 과일향은 적지만 바디가 약하지 않을 때 사용한다. 과일향, 신맛, 탄닌, 알코올 등 여러 가지 맛과 향이 어울린 상태. 신맛과 탄닌이 기본을 이루면서 과일향이 적당히 감싸고 있다. |
| | 맛 bold(현저한) upfront(솔직한) clean(깔끔한) supple(순한) dull(맛이없는) | 산도, 당분, 탄닌, 알코올이 균형을 이뤄 향이 뚜렷하고 쉽게 감별할 수 있다. 와인의 맛을 있는 그대로 보여준다. 맛이 애매하지 않고 분명하다. 박테리이나 화학 불순물이 느껴지지 않아 깔끔하고 산뜻하다. 활기차고 연한 느낌으로, 향보다는 와인의 식감을 표현한 말이다. 딱히 무슨 맛이라 말할 수 없이 유쾌하지 않은 맛을 말한다. 숙성도중 공기 노출이 지나쳤다는 증거. |

- 아래서 소믈리에도 있나 보다
- 와인 맛의 분류는 아주 엄청나다



1.5 와인 데이터 받기

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Welcome to the UC Irvine Machine Learning Repository!

We currently maintain 487 data sets as a service to the machine learning community. You may [view all data sets](#) through our searchable interface. For a general overview of the Repository, please visit our [About](#) page. For information about citing data sets in publications, please read our [citation policy](#). If you wish to donate a data set, please consult our [donation policy](#). For any other questions, feel free to contact the Repository librarians.

Supported By: In Collaboration With:

| Latest News: | | Newest Data Sets: | | Most Popular Data Sets (hits since 2007): | |
|--|--|---|--|--|--|
| 09-24-2018: Welcome to the new Repository admins Dheeru Dua and Efi Karra Taniskidou! | | 09-30-2019: Hepatitis C Virus (HCV) for Egyptian patients | | 2870366: Iris | |
| 04-04-2013: Welcome to the new Repository admins Kevin Bache and Moshe Lichman! | | 09-23-2019: QSAR fish toxicity | | 1595991: Adult | |
| 03-01-2010: Note from donor regarding Netflix data | | 09-23-2019: QSAR aquatic toxicity | | 1237675: Wine | |
| 10-16-2009: Two new data sets have been added. | | 09-21-2019: Online Retail II | | 1044914: Car Evaluation | |
| 09-14-2009: Several data sets have been added. | | 09-20-2019: Human Activity Recognition from Continuous Ambient Sensor Data | | 1030941: Wine Quality | |
| 03-24-2008: New data sets have been added! | | 09-20-2019: Beijing Multi-Site Air-Quality Data | | 1021976: Heart Disease | |
| 06-25-2007: Two new data sets have been added: UJI Pen Characters, MAGIC Gamma Telescope | | 09-20-2019: MEx | | 1008996: Breast Cancer Wisconsin (Diagnostic) | |

Featured Data Set: Molecular Biology (Splice-Junction Gene Sequences)

Task: Classification
Data Type: Sequential, Domain-Theory
Attributes: 61
Instances: 3190

Primate splice-junction gene sequences (DNA) with associated imperfect domain theory

- 레드와인 품질 데이터

- <https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv>

- 화이트와인 품질 데이터
 - <https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv>
- 위 경로의 데이터를 github에 옮겨 두었다.

1.6 PinkWink의 github에서 받을 수 있다

The screenshot shows a GitHub repository interface for the 'ML_tutorial' project. At the top, there are buttons for 'Unwatch' (1), 'Star' (0), and 'Fork' (0). Below the header, there are tabs for 'Code', 'Issues (0)', 'Pull requests (0)', 'Projects (0)', 'Wiki', 'Security', 'Insights', and 'Settings'. The 'Code' tab is selected. A dropdown menu shows 'Branch: master'. The main area displays three files in the 'dataset' directory:

| File | Type | Last Commit |
|-----------------------|-----------|----------------|
| wine.csv | wine_data | 1 minute ago |
| winequality-red.csv | wine_data | 17 minutes ago |
| winequality-white.csv | wine_data | 17 minutes ago |

1.7 데이터를 읽어보자

```
import pandas as pd

red_url = 'https://raw.githubusercontent.com/PinkWink/ML_tutorial' +\
           '/master/dataset/winequality-red.csv'
white_url = 'https://raw.githubusercontent.com/PinkWink/ML_tutorial' +\
            '/master/dataset/winequality-white.csv'

red_wine = pd.read_csv(red_url, sep = ';')
white_wine = pd.read_csv(white_url, sep = ";")
```

✓ 0.0s

Python

1.8 두 데이터의 구조는 동일하다

`red_wine.head()`

✓ 0.0s Python

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates |
|---|---------------|------------------|-------------|----------------|-----------|---------------------|----------------------|---------|------|-----------|
| 0 | 7.4 | 0.70 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.9978 | 3.51 | 0.56 |
| 1 | 7.8 | 0.88 | 0.00 | 2.6 | 0.098 | 25.0 | 67.0 | 0.9968 | 3.20 | 0.68 |
| 2 | 7.8 | 0.76 | 0.04 | 2.3 | 0.092 | 15.0 | 54.0 | 0.9970 | 3.26 | 0.65 |
| 3 | 11.2 | 0.28 | 0.56 | 1.9 | 0.075 | 17.0 | 60.0 | 0.9980 | 3.16 | 0.58 |
| 4 | 7.4 | 0.70 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.9978 | 3.51 | 0.56 |

`white_wine.head()`

✓ 0.0s Python

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates |
|---|---------------|------------------|-------------|----------------|-----------|---------------------|----------------------|---------|------|-----------|
| 0 | 7.0 | 0.27 | 0.36 | 20.7 | 0.045 | 45.0 | 170.0 | 1.0010 | 3.00 | 0.45 |
| 1 | 6.3 | 0.30 | 0.34 | 1.6 | 0.049 | 14.0 | 132.0 | 0.9940 | 3.30 | 0.49 |
| 2 | 8.1 | 0.28 | 0.40 | 6.9 | 0.050 | 30.0 | 97.0 | 0.9951 | 3.26 | 0.44 |
| 3 | 7.2 | 0.23 | 0.32 | 8.5 | 0.058 | 47.0 | 186.0 | 0.9956 | 3.19 | 0.40 |
| 4 | 7.2 | 0.23 | 0.32 | 8.5 | 0.058 | 47.0 | 186.0 | 0.9956 | 3.19 | 0.40 |

1.9 컬럼의 종류는?

`white_wine.columns`

✓ 0.0s Python

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

- fixed acidity : 고정 산도
- volatile acidity : 휘발성 산도
- citric acid : 시트르산
- residual sugar : 잔류 당분
- chlorides : 염화물

- free sulfur dioxide : 자유 이산화황
- total sulfur dioxide : 총 이산화황
- density : 밀도
- pH
- sulphates : 황산염
- alcohol
- quality : 0 ~ 10 (높을 수록 좋은 품질)

1.10 두 데이터를 하나로 합쳐보자

```
red_wine['color'] = 1.
white_wine['color']= 0.

wine = pd.concat([red_wine, white_wine])
wine.info()
```

✓ 0.0s

Python

```
<class 'pandas.core.frame.DataFrame'>
Index: 6497 entries, 0 to 4897
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   fixed acidity    6497 non-null   float64 
 1   volatile acidity 6497 non-null   float64 
 2   citric acid     6497 non-null   float64 
 3   residual sugar   6497 non-null   float64 
 4   chlorides        6497 non-null   float64 
 5   free sulfur dioxide 6497 non-null   float64 
 6   total sulfur dioxide 6497 non-null   float64 
 7   density          6497 non-null   float64 
 8   pH               6497 non-null   float64 
 9   sulphates        6497 non-null   float64 
 10  alcohol          6497 non-null   float64 
 11  quality          6497 non-null   int64  
 12  color            6497 non-null   float64 
dtypes: float64(12), int64(1)
memory usage: 710.6 KB
```

1.11 quality 컬럼은 3부터 9등급까지

```
wine['quality'].unique()
```

✓ 0.0s

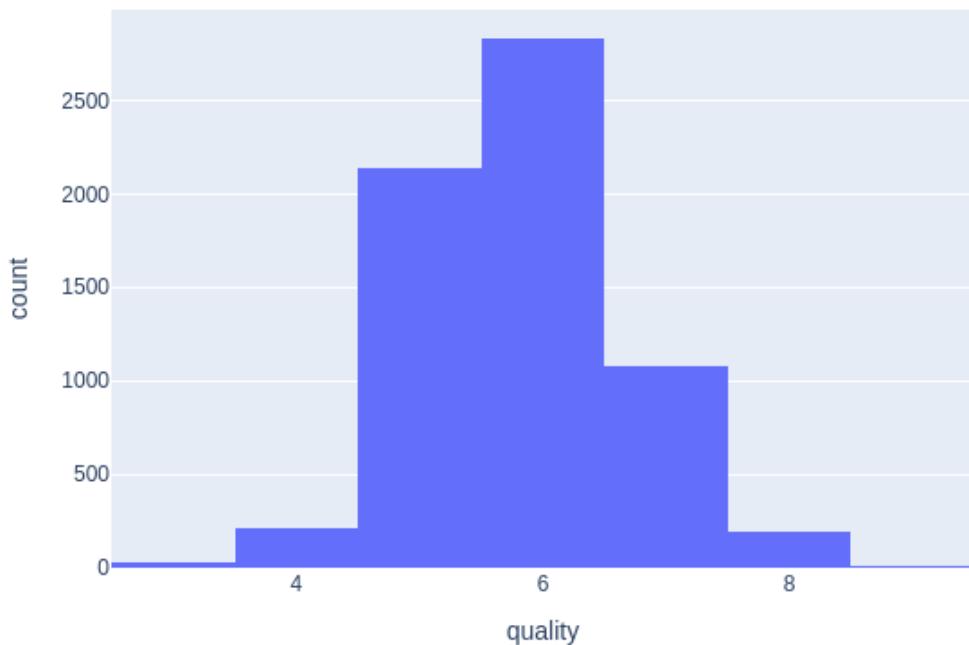
Python

```
array([5, 6, 7, 4, 8, 3, 9])
```

1.12 histogram을 그려보자

```
import plotly.express as px  
  
fig = px.histogram(wine, x = 'quality')  
fig.show()
```

Python

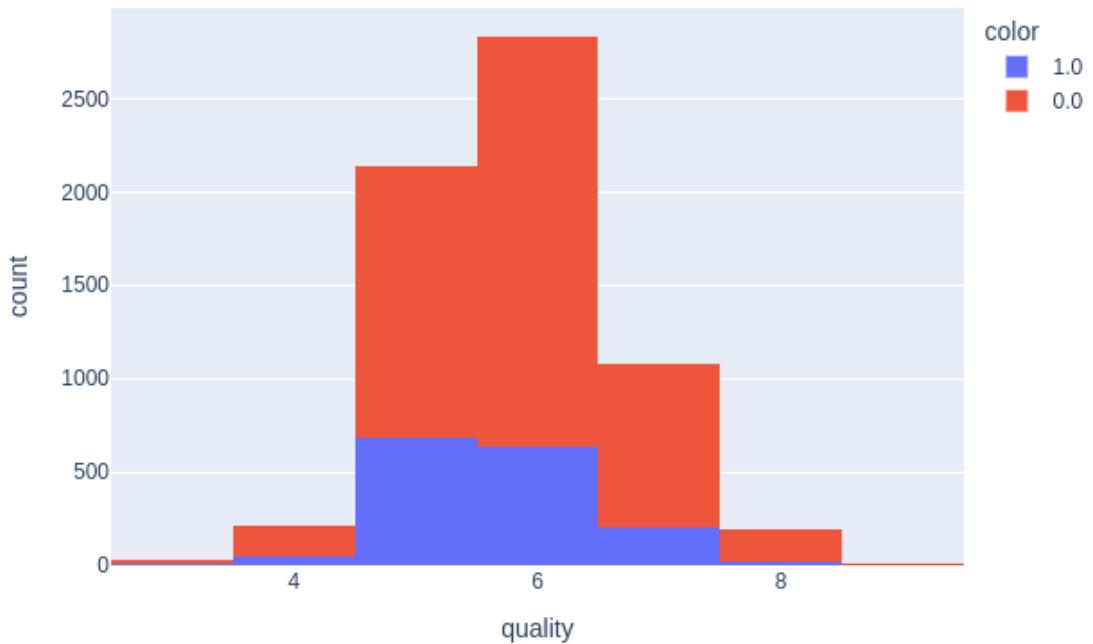


1.13 레드/화이트 와인별로 등급 Histogram은?

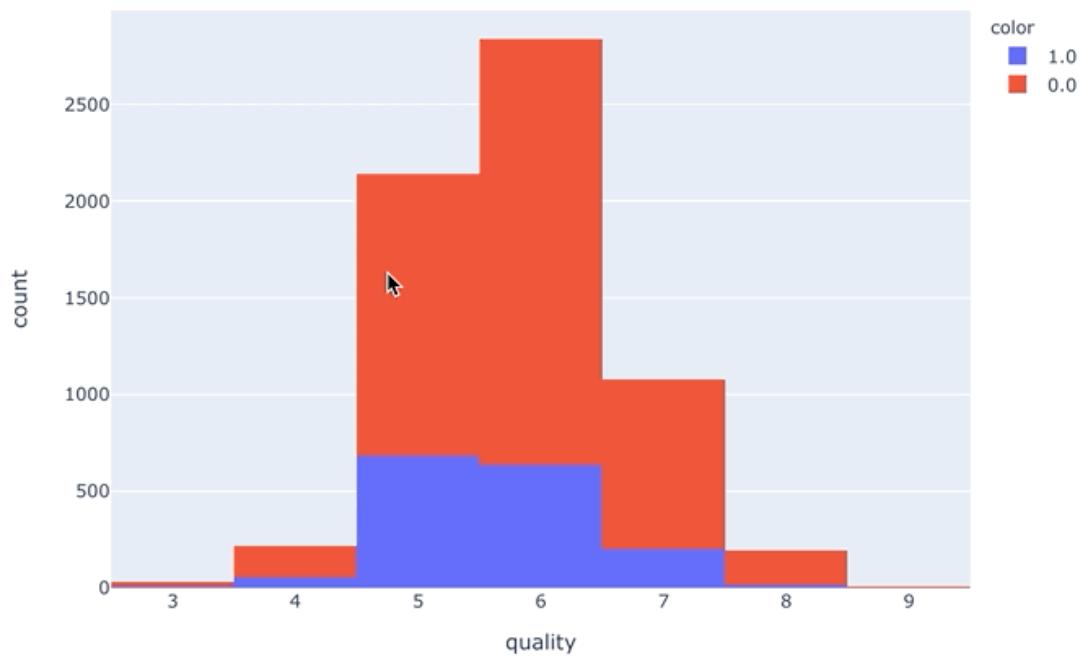
```
fig = px.histogram(wine, x = 'quality' , color = 'color')
fig.show()
```

✓ 0.0s

Python



1.14 잠깐, plotly.express는 아주 간편하고 강력한 기능을 제공한다



2 레드 와인 / 화이트 와인 분류기

2.1 먼저 라벨을 분리하고

```
X = wine.drop(['color'], axis = 1)  
y = wine['color']
```

✓ 0.0s

Python

2.2 데이터를 훈련용과 테스트용으로 나눠주자

```
from sklearn.model_selection import train_test_split  
import numpy as np  
  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,  
np.unique(y_train, return_counts = True) random_state = 13)
```

✓ 0.5s

Python

```
(array([0., 1.]), array([3913, 1284]))
```

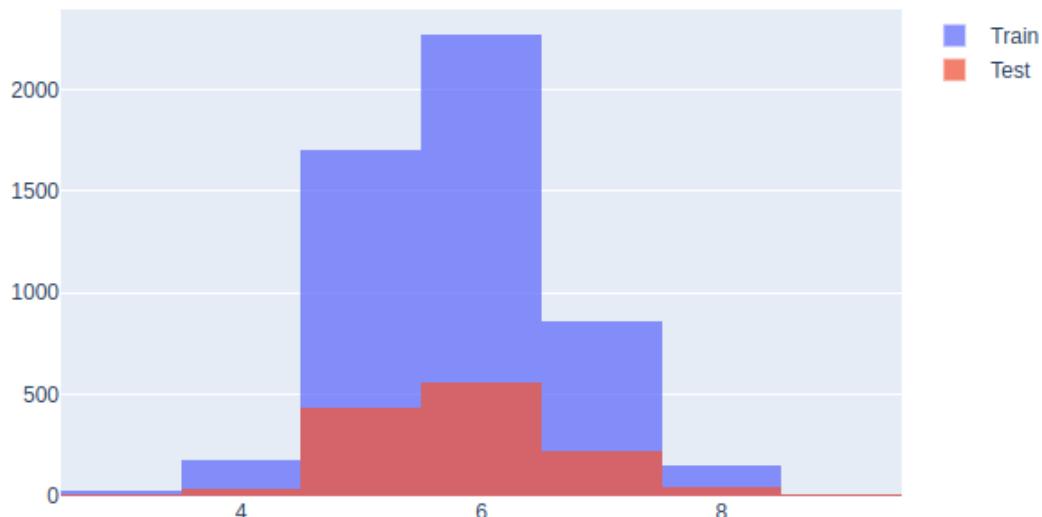
2.3 훈련용과 테스트용이 레드/화이트 와인에 따라 어느정도 구분되었을까

```
import plotly.graph_objects as go

fig = go.Figure()
fig.add_trace(go.Histogram(x = X_train['quality'] , name = "Train"))
fig.add_trace(go.Histogram(x = X_test['quality'] , name = "Test"))

fig.update_layout(barmode = 'overlay')
fig.update_traces(opacity = 0.75)
fig.show()
```

✓ 0.0s Python



2.4 결정나무 훈련

```
from sklearn.tree import DecisionTreeClassifier

wine_tree = DecisionTreeClassifier(max_depth = 2 , random_state = 13)
wine_tree.fit(X_train, y_train)
```

✓ 0.0s Python

DecisionTreeClassifier
DecisionTreeClassifier(max_depth=2, random_state=13)

2.5 학습 결과는?

```
from sklearn.metrics import accuracy_score  
  
y_pred_tr = wine_tree.predict(X_train)  
y_pred_test = wine_tree.predict(X_test)  
  
print('Train Acc : ', accuracy_score(y_train, y_pred_tr))  
print("Test Acc : ", accuracy_score(y_test, y_pred_test))
```

✓ 0.0s

Python

Train Acc : 0.9553588608812776

Test Acc : 0.9569230769230769

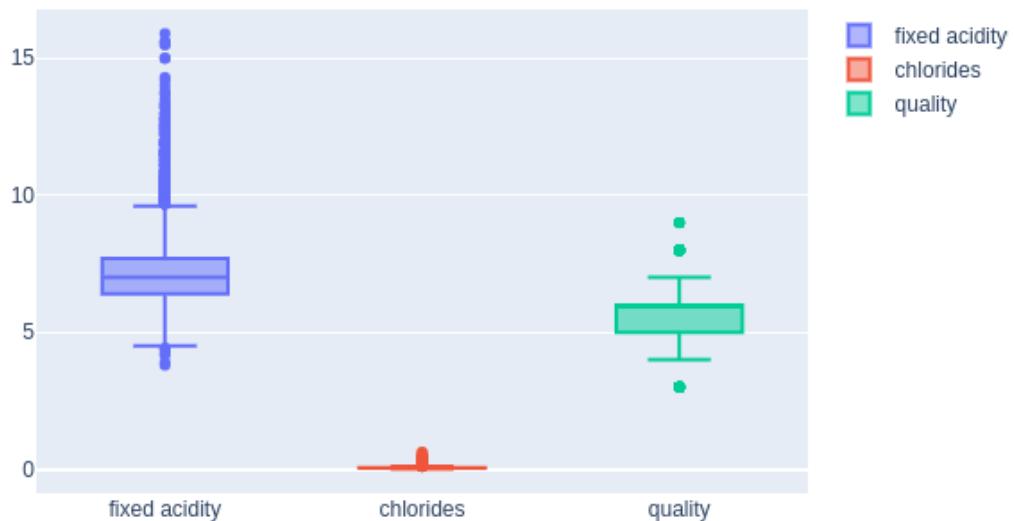
3 데이터 전처리 - MinMaxScaler와 StandardScaler

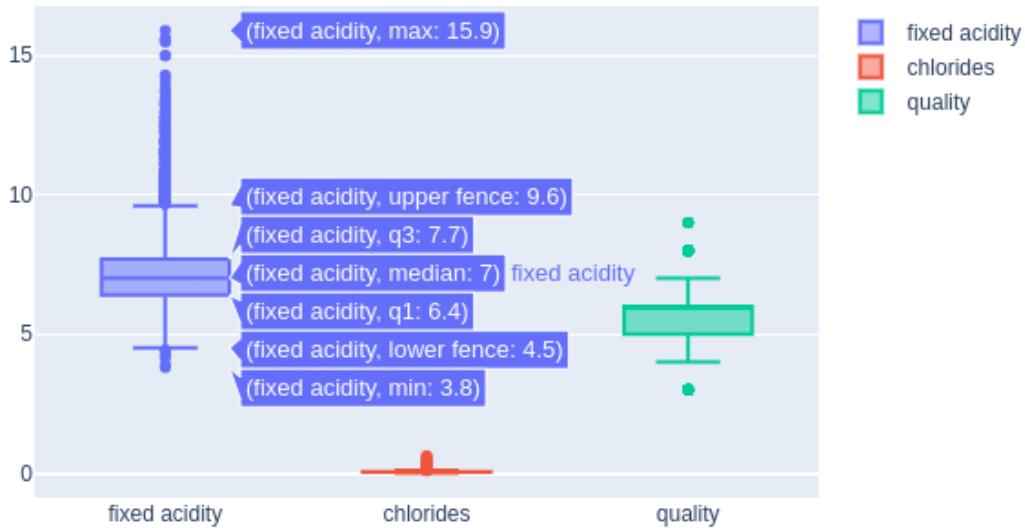
3.1 와인 데이터의 몇 개 항목의 Boxplot을 그려보자

```
fig = go.Figure()  
fig.add_trace(go.Box(y = X['fixed acidity'], name = 'fixed acidity'))  
fig.add_trace(go.Box(y = X['chlorides'], name = 'chlorides'))  
fig.add_trace(go.Box(y = X['quality'], name = 'quality'))  
  
fig.show()
```

✓ 0.0s

Python





- 컬럼들의 최대/최소 범위가 각각 다르고, 평균과 분산이 각각 다르다.
- 특성(feature)의 편향 문제는 최적의 모델을 찾는데 방해가 될 수도 있다.

3.2 이럴 때 쓰는 것이 MinMaxScaler와 StandardScaler이다

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler

MMS = MinMaxScaler()
SS = StandardScaler()

SS.fit(X)
MMS.fit(X)

X_ss = SS.transform(X)
X_mms = MMS.transform(X)
```

Python

```
X_ss_pd = pd.DataFrame(X_ss, columns = X.columns)
X_mms_pd = pd.DataFrame(X_mms, columns = X.columns)
```

Python

- 결정나무에서는 이런 전처리는 의미를 가지지 않는다.
- 주로 Cost Function을 최적화할 때 유용할 때가 있다
- MinMaxScaler와 StandardScaler 중 어떤 것이 좋을지는 해봐야 안다

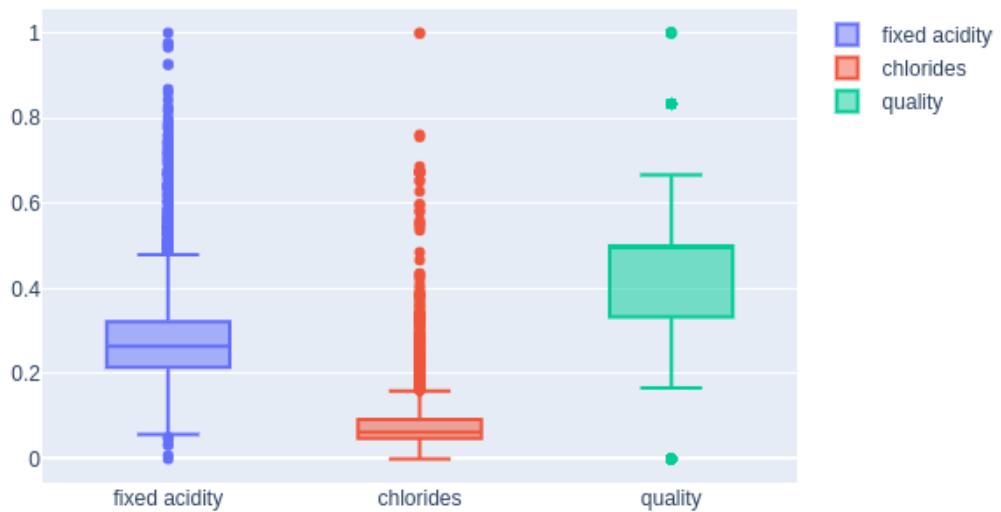
3.3 MinMaxScaler는 뭘까?

```
fig = go.Figure()
fig.add_trace(go.Box(y = X_mms_pd['fixed acidity'], name = 'fixed acidity'))
fig.add_trace(go.Box(y = X_mms_pd['chlorides'], name = 'chlorides'))
fig.add_trace(go.Box(y = X_mms_pd['quality'], name = 'quality'))

fig.show()
```

✓ 0.0s

Python



- 최대 최소값을 1과 0으로 강제로 맞추는 것

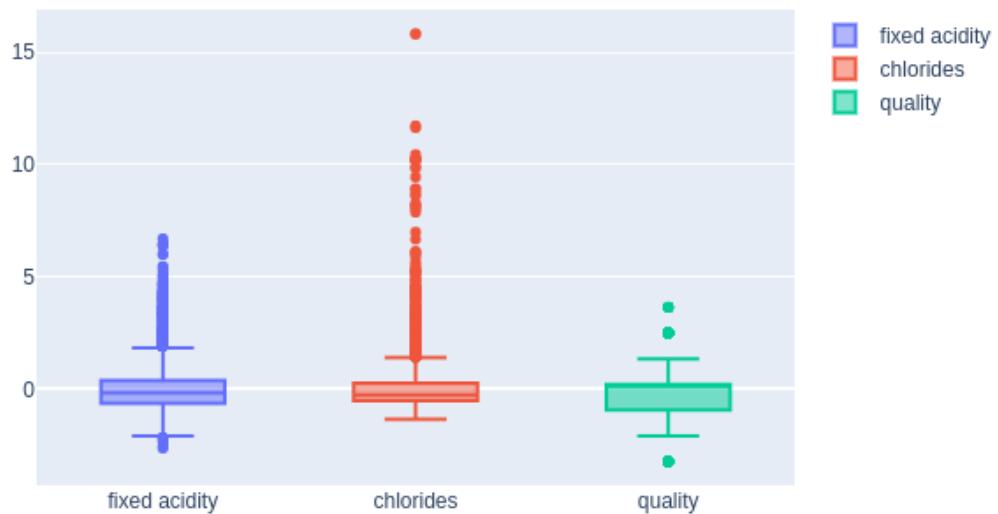
3.4 StandardScaler는?

```
fig = go.Figure()
fig.add_trace(go.Box(y = X_ss_pd['fixed acidity'], name = 'fixed acidity'))
fig.add_trace(go.Box(y = X_ss_pd['chlorides'], name = 'chlorides'))
fig.add_trace(go.Box(y = X_ss_pd['quality'], name = 'quality'))

fig.show()
```

✓ 0.0s

Python



- 평균을 0으로 표준편차를 1로 맞추는 것

3.5 MinMaxScaler를 적용해서 다시 학습

```
X_train, X_test, y_train, y_test = train_test_split(X_mms_pd, y,
                                                test_size = 0.2,
                                                random_state = 13)

wine_tree = DecisionTreeClassifier(max_depth = 2, random_state = 13)
wine_tree.fit(X_train, y_train)

y_pred_tr = wine_tree.predict(X_train)
y_pred_test = wine_tree.predict(X_test)

print('Train Acc : ', accuracy_score(y_train, y_pred_tr))
print("Test Acc : " , accuracy_score(y_test, y_pred_test))
```

✓ 0.0s

Python

Train Acc : 0.9553588608812776
 Test Acc : 0.9569230769230769

- 다시 이야기하지만 결정나무에서는 이런 전처리는 거의 효과가 없다.

3.6 StandardScaler를 적용

```
X_train, X_test, y_train, y_test = train_test_split(X_ss_pd, y,
                                                test_size = 0.2,
                                                random_state = 13)

wine_tree = DecisionTreeClassifier(max_depth = 2, random_state = 13)
wine_tree.fit(X_train, y_train)

y_pred_tr = wine_tree.predict(X_train)
y_pred_test = wine_tree.predict(X_test)

print('Train Acc : ', accuracy_score(y_train, y_pred_tr))
print("Test Acc : " , accuracy_score(y_test, y_pred_test))
```

✓ 0.0s

Python

Train Acc : 0.9553588608812776
 Test Acc : 0.9569230769230769

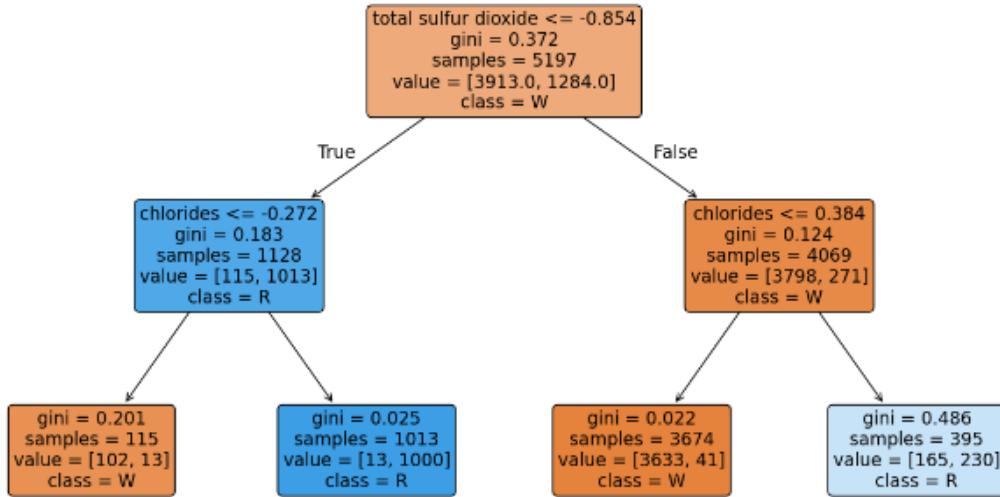
3.7 아무튼, 결정나무는 화이트와인과 레드와인을 어떻게 구분할까?

```
import matplotlib.pyplot as plt
from sklearn import tree

fig = plt.figure(figsize=(15, 8))
_ = tree.plot_tree(wine_tree,
                   feature_names = list(X_train.columns),
                   class_names = ["W", "R"],
                   rounded = True,
                   filled=True)
```

✓ 0.6s

Python



- 뭔 지 몰라도 total sulfur dioxide가 중요한 역할을 하나보다
- total sulfur dioxide → 총 이산화황~ TSO2

3.8 레드와인과 화이트와인을 구분하는 중요 특성?

```
dict(zip(X_train.columns, wine_tree.feature_importances_))  
✓ 0.0s  
{'fixed acidity': np.float64(0.0),  
 'volatile acidity': np.float64(0.0),  
 'citric acid': np.float64(0.0),  
 'residual sugar': np.float64(0.0),  
 'chlorides': np.float64(0.24230360549660776),  
 'free sulfur dioxide': np.float64(0.0),  
 'total sulfur dioxide': np.float64(0.7576963945033922),  
 'density': np.float64(0.0),  
 'pH': np.float64(0.0),  
 'sulphates': np.float64(0.0),  
 'alcohol': np.float64(0.0),  
 'quality': np.float64(0.0)}
```

Python

- MaxDepth를 높이면 저 수치에도 변화가 온다.

4 와인 맛에 대한 분류 - 이진 분류

4.1 quality 컬럼을 이진화하자

```
wine['taste'] = [1. if grade > 5 else 0. for grade in wine['quality']]
wine.info()

✓ 0.0s
```

Python

```
<class 'pandas.core.frame.DataFrame'>
Index: 6497 entries, 0 to 4897
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   fixed acidity    6497 non-null   float64 
 1   volatile acidity 6497 non-null   float64 
 2   citric acid      6497 non-null   float64 
 3   residual sugar   6497 non-null   float64 
 4   chlorides        6497 non-null   float64 
 5   free sulfur dioxide 6497 non-null   float64 
 6   total sulfur dioxide 6497 non-null   float64 
 7   density          6497 non-null   float64 
 8   pH               6497 non-null   float64 
 9   sulphates        6497 non-null   float64 
 10  alcohol          6497 non-null   float64 
 11  quality          6497 non-null   int64  
 12  color            6497 non-null   float64 
 13  taste             6497 non-null   float64 
dtypes: float64(13), int64(1)
memory usage: 761.4 KB
```

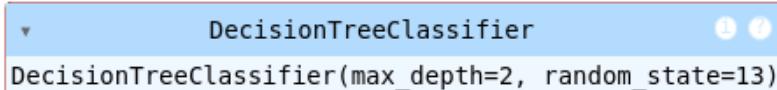
4.2 레드/화이트 와인 분류와 동일 과정을 거치자

```
X = wine.drop(['taste'], axis = 1)
y = wine['taste']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
                                                    random_state = 13)
wine_tree = DecisionTreeClassifier(max_depth = 2, random_state = 13)
wine_tree.fit(X_train, y_train)

✓ 0.0s
```

Python



```
DecisionTreeClassifier(max_depth=2, random_state=13)
```

4.3 응? 100프로? 가능한가?

```
y_pred_tr = wine_tree.predict(X_train)
y_pred_test = wine_tree.predict(X_test)

print('Train Acc : ', accuracy_score(y_train, y_pred_tr))
print("Test Acc : " , accuracy_score(y_test, y_pred_test))
```

✓ 0.0s

Python

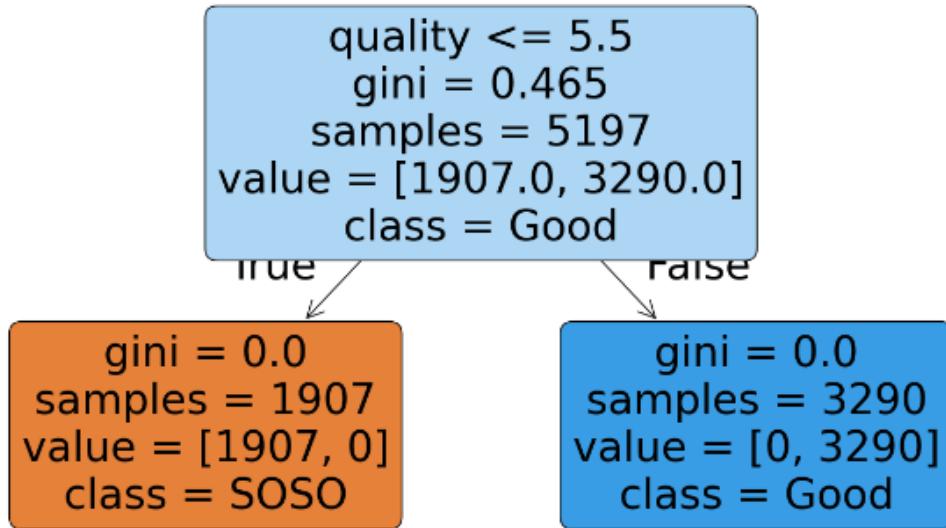
Train Acc : 1.0
 Test Acc : 1.0

4.4 왜 이런 일이 생겼는지 확인해보자

```
fig = plt.figure(figsize=(15, 8))
= tree.plot_tree(wine_tree,
                  feature_names =list(X_train.columns),
                  class_names = ["SOSO" , "Good"],
                  rounded = True,
                  filled=True)
```

✓ 0.1s

Python



- 앗 . Quality... quality 컬럼으로 taste 컬럼을 만들었으니 quality 컬럼은 제거했어야 했다.

4.5 다시

```
X = wine.drop(['taste', 'quality'], axis = 1)
y = wine['taste']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
                                                    random_state = 13)
wine_tree = DecisionTreeClassifier(max_depth = 2, random_state = 13)
wine_tree.fit(X_train, y_train)
```

✓ 0.0s

Python

```
▼      DecisionTreeClassifier
DecisionTreeClassifier(max_depth=2, random_state=13)
```

4.6 결과

```
y_pred_tr = wine_tree.predict(X_train)
y_pred_test = wine_tree.predict(X_test)

print('Train Acc : ', accuracy_score(y_train, y_pred_tr))
print("Test Acc : " , accuracy_score(y_test, y_pred_test))
```

✓ 0.0s

Python

```
Train Acc :  0.7294593034442948
Test Acc :  0.7161538461538461
```

4.7 어떤 와인을 “맛있다”고 할 수 있나?

```
fig = plt.figure(figsize=(15, 8))
_ = tree.plot_tree(wine_tree,
                   feature_names =list(X_train.columns),
                   class_names = ["SOSO", "Good"],
                   rounded = True,
                   filled=True)
```

✓ 0.2s

Python

