결정 트리

로지스틱 회귀로 와인 분류하기

In [1]:
```
import pandas as pd

wine = pd.read_csv('https://bit.ly/wine_csv_data') # 데이터 csv 파일
```

In [2]:
```
wine.head() # 처음 5개의 샘플 확인 : 0이면 레드와인, 1이면 화이트 와인 -> 이진 분류
```

Out[2]:

	alcohol	sugar	pH	class
0	9.4	1.9	3.51	0.0
1	9.8	2.6	3.20	0.0
2	9.8	2.3	3.26	0.0
3	9.8	1.9	3.16	0.0
4	9.4	1.9	3.51	0.0

In [3]:
```
wine.info() # 총 6497개의 샘플이 있고 4개의 열은 모두 실수값, Non null
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   alcohol   6497 non-null   float64
 1   sugar     6497 non-null   float64
 2   pH        6497 non-null   float64
 3   class     6497 non-null   float64
dtypes: float64(4)
memory usage: 203.2 KB
```

In [5]:
```
wine.describe() # 평균, 표준편차, 최소, 최대, 1사분위, 중간값, 3사분위를 확인
# 알코올 도수와 당도, PH값의 스케일이 다르므로 일 수 있음 -> standardscaler 클래스로 특성을 표준화하자
```

Out[5]:

	alcohol	sugar	pH	class
count	6497.000000	6497.000000	6497.000000	6497.000000
mean	10.491801	5.443235	3.218501	0.753886
std	1.192712	4.757804	0.160787	0.430779
min	8.000000	0.600000	2.720000	0.000000
25%	9.500000	1.800000	3.110000	1.000000
50%	10.300000	3.000000	3.210000	1.000000
75%	11.300000	8.100000	3.320000	1.000000
max	14.900000	65.800000	4.010000	1.000000

In [6]:
```
data = wine[['alcohol', 'sugar', 'pH']].to_numpy() # input
target = wine['class'].to_numpy() # Yilabel)
```

In [7]:
```
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(
    data, target, test_size=0.2, random_state=42) # 8:2 비율로 훈련 세트와 테스트 세트를 나눔
```

In [8]:
```
print(train_input.shape, test_input.shape)
```

```
(5197, 3) (1300, 3)
```

In [12]:
```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()
ss.fit(train_input)

train_scaled = ss.transform(train_input)
test_scaled = ss.transform(test_input)
print(train_scaled.shape)
print(test_scaled.shape)
```

```
(5197, 3)
(1300, 3)
```

In [13]:
```
from sklearn.linear_model import LogisticRegression # LogisticRegression 사용

lr = LogisticRegression()
lr.fit(train_scaled, train_target)

print(lr.score(train_scaled, train_target))
print(lr.score(test_scaled, test_target))
# 훈련 세트와 테스트 세트의 점수가 모두 낮으니 모델이 다소 underfitting되었음 -> 규제 매개변수 c 값을 조정? 다른 알고리즘? 다양한 feature 추가?
```

```
0.7808350971714451
0.7776923076923077
```

설명하기 쉬운 모델과 어려운 모델

In [15]:
```
print(lr.coef_, lr.intercept_)
# 위는 이 Logistic regression 모델을 이해하기 쉽지 않음
# 만약 다항 특성을 추가한다면 설명하기 더 어려울 것
```

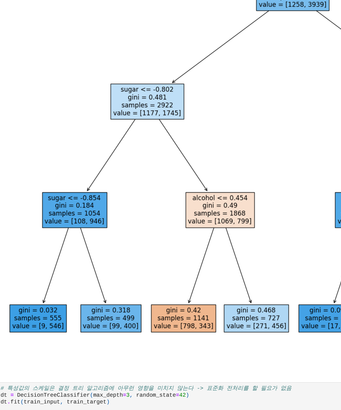
```
[[ 0.51270274  1.6733911  -0.68767781]] [1.81777902]
```

결정 트리

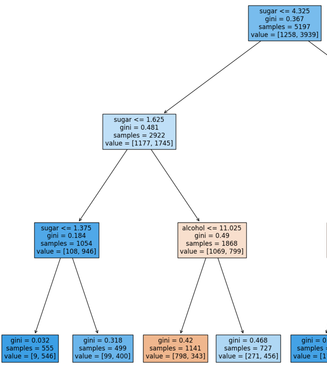
In [16]:
```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(random_state=42)
dt.fit(train_scaled, train_target)

print(dt.score(train_scaled, train_target))
print(dt.score(test_scaled, test_target)) # overfitting
```

```
0.8496213097550433
0.8592307692307692
```

In [17]:
```
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

plt.figure(figsize=(10,7))
plot_tree(dt)
plt.show()
```



In [19]:
```
plt.figure(figsize=(10,7))
plot_tree(dt, max_depth=3, filled=True, feature_names=['alcohol', 'sugar', 'pH'])
# max_depth로 노드 깊이 설정, filled=True로 클래스에 맞게 노드의 색을 채움, feature_name으로 특성의 이름을 전달
plt.show()
```



가지치기

In [20]:
```
dt = DecisionTreeClassifier(max_depth=3, random_state=42)
dt.fit(train_scaled, train_target)

print(dt.score(train_scaled, train_target))
print(dt.score(test_scaled, test_target))
```

```
0.8454877814123533
0.8415384615384616
```

In [21]:
```
plt.figure(figsize=(20,15))
plot_tree(dt, filled=True, feature_names=['alcohol', 'sugar', 'pH'])
plt.show()
```


In [22]:
```
# 특성값의 스케일은 결정 트리 알고리즘에 아무런 영향을 미치지 않는다 -> 표준화 전처리를 할 필요가 없음
dt = DecisionTreeClassifier(max_depth=3, random_state=42)
dt.fit(train_input, train_target)

print(dt.score(train_input, train_target))
print(dt.score(test_input, test_target))
```

```
0.8454877814123533
0.8415384615384616
```

In [23]:
```
plt.figure(figsize=(20,15))
plot_tree(dt, filled=True, feature_names=['alcohol', 'sugar', 'pH'])
plt.show()
```


In [ ]:

In [25]:
```
print(dt.feature_importances_) # 특성 중요도 출력 -> sugar가 가장 중요
```

```
[0.12345626 0.86862934 0.00791144 ]
```

In [ ]: