결정 트리(Decision Tree)

- 분류와 회귀에 사용되는 지도 학습 방법
- 데이터 특성으로 부터 추론된 결정 규칙을 통해 값을 예측
- **if-then-else** 결정 규칙을 통해 데이터 학습
- 트리의 깊이가 깊을 수록 복잡한 모델
- 결정 트리 장점
    - 이해와 해석이 쉽다
    - 시각화가 용이하다
    - 많은 데이터 전처리가 필요하지 않다
    - 수치형과 범주형 데이터 모두를 다룰 수 있다
    - ...

```
import pandas as pd
import numpy as np
import graphviz
import multiprocessing
import matplotlib.pyplot as plt
plt.style.use(['seaborn-whitegrid'])
```

```
from sklearn.datasets import load_iris, load_wine, load_breast_cancer
from sklearn.datasets import load_boston, load_diabetes
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
```

분류를 위한 데이터

붓꽃 데이터

```
iris = load_iris()
```

```
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target
iris_df
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2

와인 데이터

```
wine = load_wine()
```

```
wine_df = pd.DataFrame(data=wine.data, columns=wine.feature_names)
wine_df['target'] = wine.target
wine_df
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavano
0	14.23	1.71	2.43	15.6	127.0	2.80	3
1	13.20	1.78	2.14	11.2	100.0	2.65	2
2	13.16	2.36	2.67	18.6	101.0	2.80	3
3	14.37	1.95	2.50	16.8	113.0	3.85	3
4	13.24	2.59	2.87	21.0	118.0	2.80	2
...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95.0	1.68	0
174	13.40	3.91	2.48	23.0	102.0	1.80	0
175	13.27	4.28	2.26	20.0	120.0	1.59	0
176	13.17	2.59	2.37	20.0	120.0	1.65	0
177	14.13	4.10	2.74	24.5	96.0	2.05	0

178 rows × 14 columns

유방암 데이터

```
cancer = load_breast_cancer()
```

```
cancer_df = pd.DataFrame(data=cancer.data, columns=cancer.feature_names)
cancer_df
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430
...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000

569 rows × 30 columns

회귀를 위한 데이터

보스턴 주택 가격 데이터

```
boston = load_boston()
```

```
boston_df = pd.DataFrame(data=boston.data, columns=boston.feature_names)
boston_df['TARGET'] = boston.target
boston_df
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90
...	...	...	...	...	...	...	...	...	...	...	...	..
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99

당뇨병 데이터

```
diabetes = load_diabetes()
```

505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90

```
diabetes_df = pd.DataFrame(data=diabetes.data, columns=diabetes.feature_names)
diabetes_df['target'] = diabetes.target
diabetes_df
```

	age	sex	bmi	bp	s1	s2	s3	s
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.00259
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.03949
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.00259
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.03430
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.00259
...	...	...	...	...	...	...	...	.
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.00259
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.03430
439	0.041708	0.050680	-0.015906	0.017282	-0.037344	-0.013840	-0.024993	-0.01108
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.02656
441	-0.045472	-0.044642	-0.073030	-0.081414	0.083740	0.027809	0.173816	-0.03949

442 rows × 11 columns

분류 - `DecisionTreeClassifier()`

- `DecisionTreeClassifier`는 분류를 위한 결정트리 모델
- 두개의 배열 X, y를 입력 받음
    - X는 [n_samples, n_features] 크기의 데이터 특성 배열

- y는 [n_samples] 크기의 정답 배열

```
X = [[0, 0], [1, 1]]
y = [0, 1]

model = tree.DecisionTreeClassifier()
model = model.fit(X, y)
```

```
model.predict([[2., 2.]])
```

```
array([1])
```

```
model.predict_proba([[2., 2.]])
```

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

붓꽃 데이터 학습

교차검증

전처리 없이 학습

```
model = DecisionTreeClassifier()
```

```
cross_val_score(
    estimator=model,
    X=iris.data, y=iris.target,
    cv=5,
    n_jobs=multiprocessing.cpu_count()
)
```

```
array([0.96666667, 0.96666667, 0.9       , 0.96666667, 1.        ])
```

전처리 후 학습

- 결정 트리는 규칙을 학습하기 때문에 전처리에 큰 영향을 받지 않는다.

```
model = make_pipeline(
    StandardScaler(),
    DecisionTreeClassifier()
)
```

```
cross_val_score(
    estimator=model,
    X=iris.data, y=iris.target,
```

```
    cv=5,
    n_jobs=multiprocessing.cpu_count()
)
```

```
array([0.96666667, 0.96666667, 0.9       , 0.96666667, 1.        ])
```

학습된 결정 트리 시각화

```
model = DecisionTreeClassifier()
model.fit(iris.data, iris.target)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

텍스트를 통한 시각화

[ ] ↳ 숨겨진 셀 1개

plot_tree를 사용한 시각화

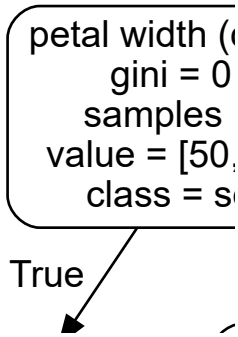
```
tree.plot_tree(model)
```

```
[Text(167.4, 199.32, 'X[3] <= 0.8\ngini = 0.667\nsamples = 150\nvalue = [50, 50, 50]
 Text(141.64615384615385, 163.07999999999998, 'gini = 0.0\nsamples = 50\nvalue = [50
 Text(193.15384615384616, 163.07999999999998, 'X[3] <= 1.75\ngini = 0.5\nsamples = 1
```

graphviz를 사용한 시각화

```
Text(77.26153846153846, 54.359999999999985, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
```

```
dot_data = tree.export_graphviz(decision_tree=model,
                                feature_names=iris.feature_names,
                                class_names=iris.target_names,
                                filled=True, rounded=True,
                                special_characters=True)
graph = graphviz.Source(dot_data)
graph
```



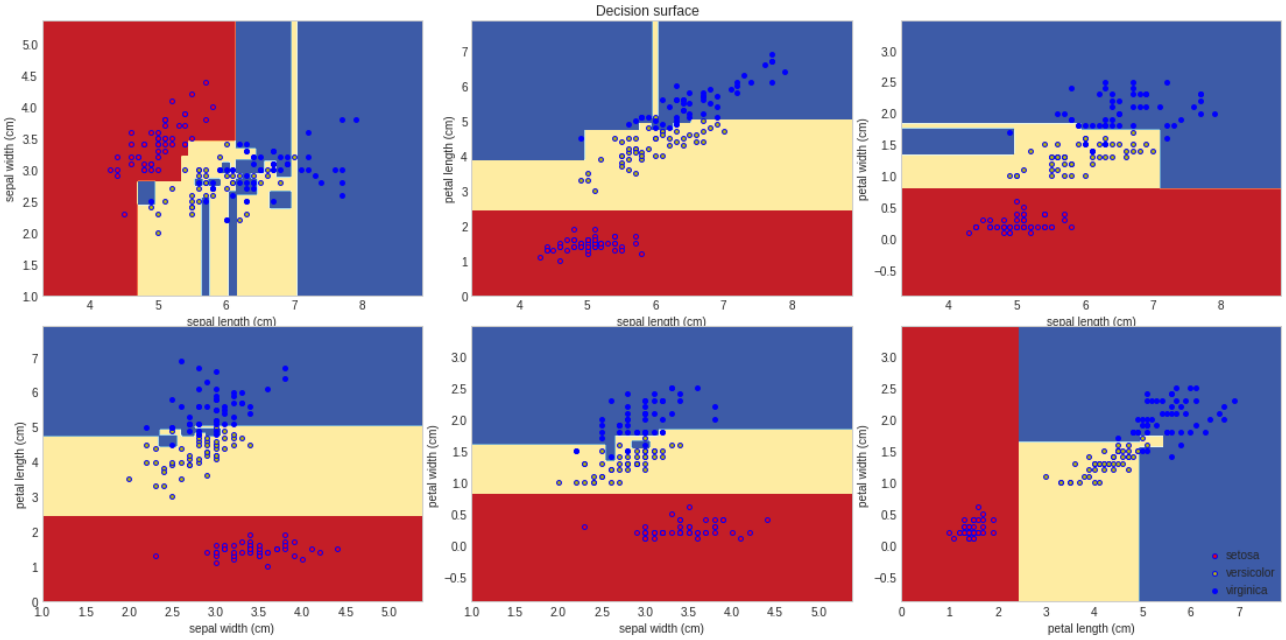
시각화

```
n_classes = 3
plot_colors = 'ryb'
plot_step = 0.02
```

결정 경계 시각화

```
plt.figure(figsize=(16, 8))

for pairidx, pair in enumerate([[0, 1], [0, 2], [0, 3],
                                [1, 2], [1, 3], [2, 3]]):
  X = iris.data[:, pair]
  y = iris.target

  model = DecisionTreeClassifier()
  model = model.fit(X, y)

  plt.subplot(2, 3, pairidx + 1)

  x_min, x_max = X[:, 0].min()-1, X[:, 0].max()+1
  y_min, y_max = X[:, 1].min()-1, X[:, 1].max()+1
  xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step),
                       np.arange(y_min, y_max, plot_step))
  plt.tight_layout(h_pad=-0.5, w_pad=0.5, pad=2.5)

  Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
  Z = Z.reshape(xx.shape)
  cs = plt.contourf(xx, yy, Z, cmap=plt.cm.RdYlBu)

  plt.xlabel(iris.feature_names[pair[0]])
  plt.ylabel(iris.feature_names[pair[1]])

  for i, color in zip(range(n_classes), plot_colors):
    idx = np.where(y == i)
    plt.scatter(X[idx, 0], X[idx, 1], c=color, label=iris.target_names[i],
                cmap=plt.cm.RdYlBu, edgecolor='b', s=15)

plt.suptitle("Decision surface")
plt.legend(loc='lower right', borderpad=0, handletextpad=0)
plt.axis('tight')
```

```
(0.0, 7.88, -0.9, 3.4800000000000044)
```

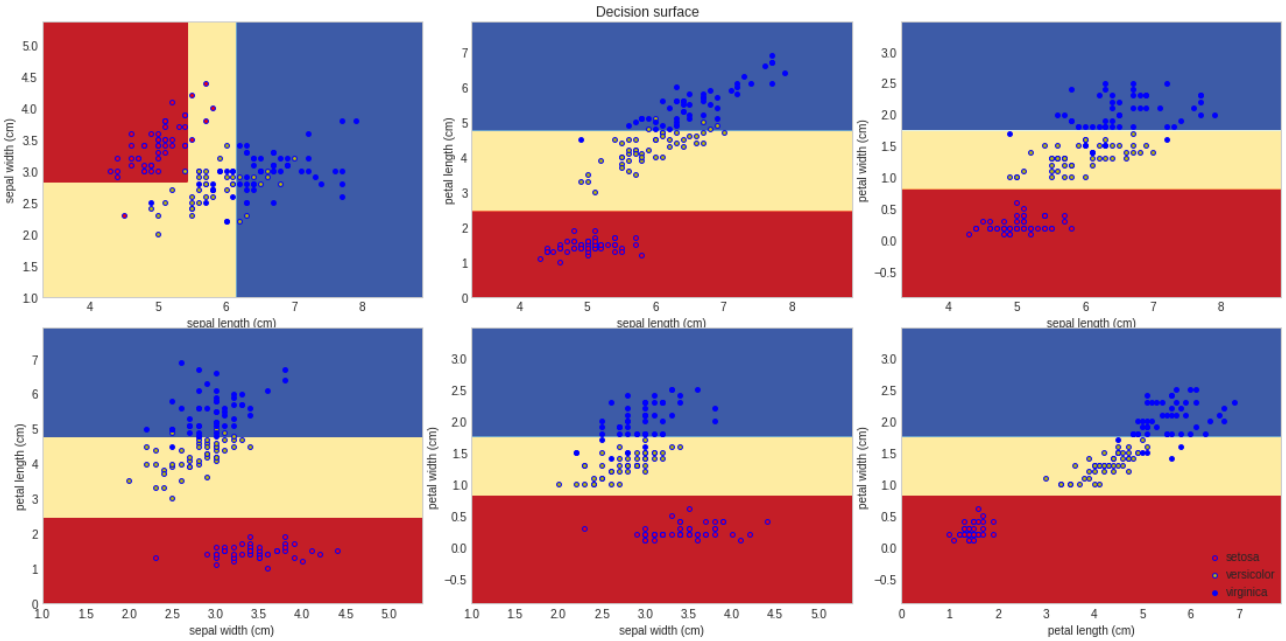


▼ 하이퍼파리미터를 변경해 보면서 결정 경계의 변화 확인

```
plt.figure(figsize=(16, 8))

for pairidx, pair in enumerate([[0, 1], [0, 2], [0, 3],
                                [1, 2], [1, 3], [2, 3]]):
  X = iris.data[:, pair]
  y = iris.target

  model = DecisionTreeClassifier(max_depth=2)
  model = model.fit(X, y)

  plt.subplot(2, 3, pairidx + 1)

  x_min, x_max = X[:, 0].min()-1, X[:, 0].max()+1
  y_min, y_max = X[:, 1].min()-1, X[:, 1].max()+1
  xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step),
                       np.arange(y_min, y_max, plot_step))
  plt.tight_layout(h_pad=-0.5, w_pad=0.5, pad=2.5)

  Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
  Z = Z.reshape(xx.shape)
  cs = plt.contourf(xx, yy, Z, cmap=plt.cm.RdYlBu)

  plt.xlabel(iris.feature_names[pair[0]])
  plt.ylabel(iris.feature_names[pair[1]])

  for i, color in zip(range(n_classes), plot_colors):
    idx = np.where(y == i)
    plt.scatter(X[idx, 0], X[idx, 1], c=color, label=iris.target_names[i],
```

```
                cmap=plt.cm.RdYlBu, edgecolor='b', s=15)

plt.suptitle("Decision surface")
plt.legend(loc='lower right', borderpad=0, handletextpad=0)
plt.axis('tight')
```

```
(0.0, 7.88, -0.9, 3.4800000000000044)
```



와인 데이터 학습

교차 검증

전처리 없이 학습

```
model = DecisionTreeClassifier()
```

```
cross_val_score(
    estimator=model,
    X=wine.data, y=wine.target,
    cv=5,
    n_jobs=multiprocessing.cpu_count()
)
```

```
array([0.94444444, 0.86111111, 0.86111111, 0.91428571, 0.85714286])
```

전처리 후 학습

```
model = make_pipeline(
    StandardScaler(),
    DecisionTreeClassifier()
)
```

```
cross_val_score(
    estimator=model,
    X=wine.data, y=wine.target,
    cv=5,
    n_jobs=multiprocessing.cpu_count()
)
```

```
array([0.94444444, 0.83333333, 0.91666667, 0.91428571, 0.85714286])
```

학습된 결정 트리 시각화

```
model = DecisionTreeClassifier()
model.fit(wine.data, wine.target)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

텍스트를 통한 시각화

[ ] ↳ 숨겨진 셀 1개

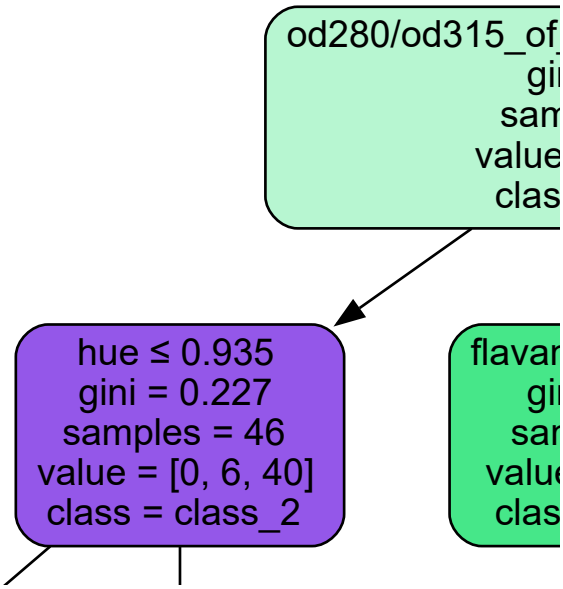
plot_tree를 사용한 시각화

```
tree.plot_tree(model)
```

```
[Text(189.42631578947368, 199.32, 'X[12] <= 755.0\ngini = 0.658\nsamples = 178\nvalu
 Text(114.53684210526316, 163.07999999999998, 'X[11] <= 2.115\ngini = 0.492\nsamples
 Text(70.48421052631579, 126.83999999999999, 'X[10] <= 0.935\ngini = 0.227\nsamples
 Text(35.242105263157896, 90.6, 'X[6] <= 1.58\ngini = 0.049\nsamples = 40\nvalue = [
 Text(17.621052631578948, 54.359999999999985, 'gini = 0.0\nsamples = 39\nvalue = [0,
 Text(52.863157894736844, 54.359999999999985, 'gini = 0.0\nsamples = 1\nvalue = [0,
 Text(105.72631578947369, 90.6, 'X[2] <= 2.45\ngini = 0.278\nsamples = 6\nvalue = [0
 Text(88.10526315789474, 54.359999999999985, 'gini = 0.0\nsamples = 5\nvalue = [0, 5
 Text(123.34736842105264, 54.359999999999985, 'gini = 0.0\nsamples = 1\nvalue = [0,
 Text(158.58947368421053, 126.83999999999999, 'X[6] <= 0.795\ngini = 0.117\nsamples
 Text(140.96842105263158, 90.6, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(176.21052631578948, 90.6, 'X[0] <= 13.175\ngini = 0.061\nsamples = 63\nvalue =
 Text(158.58947368421053, 54.359999999999985, 'gini = 0.0\nsamples = 58\nvalue = [0,
 Text(193.83157894736843, 54.359999999999985, 'X[12] <= 655.0\ngini = 0.48\nsamples
 Text(176.21052631578948, 18.119999999999976, 'gini = 0.0\nsamples = 3\nvalue = [0,
 Text(211.45263157894738, 18.119999999999976, 'gini = 0.0\nsamples = 2\nvalue = [2,
 Text(264.3157894736842, 163.07999999999998, 'X[6] <= 2.165\ngini = 0.265\nsamples =
 Text(229.07368421052632, 126.83999999999999, 'X[9] <= 3.605\ngini = 0.375\nsamples
 Text(211.45263157894738, 90.6, 'gini = 0.0\nsamples = 2\nvalue = [0, 2, 0]'),
 Text(246.69473684210527, 90.6, 'gini = 0.0\nsamples = 6\nvalue = [0, 0, 6]'),
```

▼ graphviz를 사용한 시각화

```
dot_data = tree.export_graphviz(decision_tree=model,
                                feature_names=wine.feature_names,
                                class_names=wine.target_names,
                                filled=True, rounded=True,
                                special_characters=True)
graph = graphviz.Source(dot_data)
graph
```



시각화

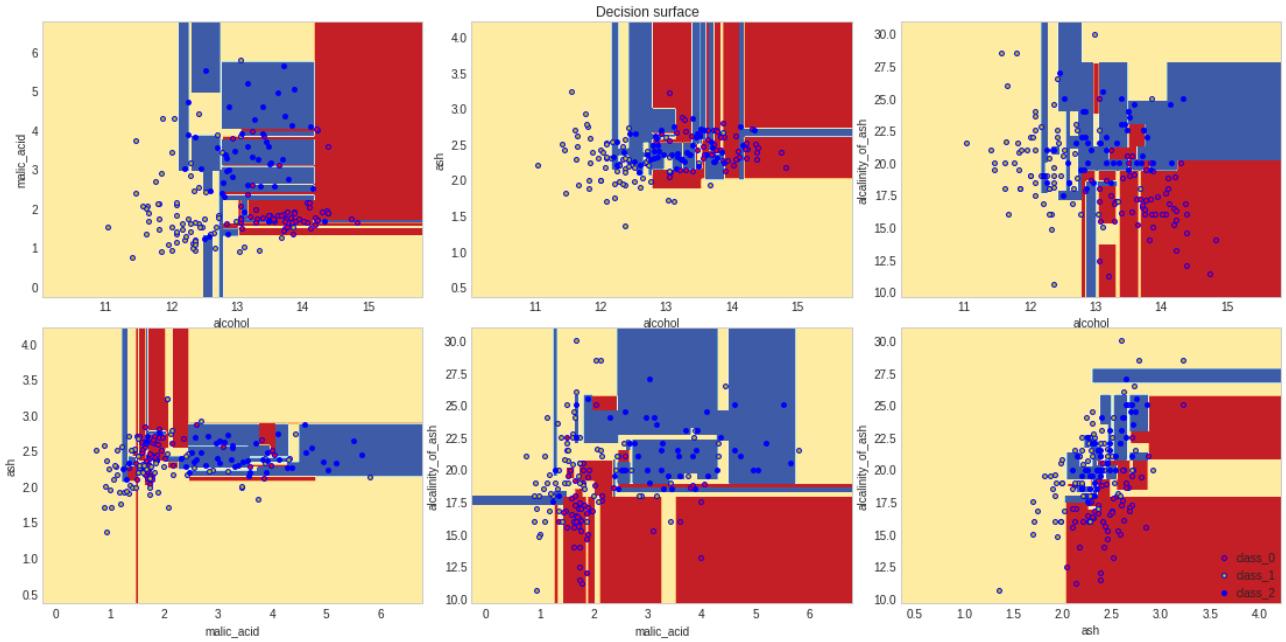
```
n_classes = 3
plot_colors = 'ryb'
plot_step = 0.02
```

결정 경계 시각화

```
plt.figure(figsize=(16, 8))

for pairidx, pair in enumerate([[0, 1], [0, 2], [0, 3],
                                [1, 2], [1, 3], [2, 3]]):
  X = wine.data[:, pair]
  y = wine.target

  model = DecisionTreeClassifier()
  model = model.fit(X, y)

  plt.subplot(2, 3, pairidx + 1)

  x_min, x_max = X[:, 0].min()-1, X[:, 0].max()+1
  y_min, y_max = X[:, 1].min()-1, X[:, 1].max()+1
  xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step),
                       np.arange(y_min, y_max, plot_step))
  plt.tight_layout(h_pad=-0.5, w_pad=0.5, pad=2.5)

  Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
```

```
  Z = Z.reshape(xx.shape)
  cs = plt.contourf(xx, yy, Z, cmap=plt.cm.RdYlBu)

  plt.xlabel(wine.feature_names[pair[0]])
  plt.ylabel(wine.feature_names[pair[1]])

  for i, color in zip(range(n_classes), plot_colors):
    idx = np.where(y == i)
    plt.scatter(X[idx, 0], X[idx, 1], c=color, label=wine.target_names[i],
                cmap=plt.cm.RdYlBu, edgecolor='b', s=15)

plt.suptitle("Decision surface")
plt.legend(loc='lower right', borderpad=0, handletextpad=0)
plt.axis('tight')
```

```
(0.3600000000000001, 4.220000000000003, 9.6, 30.979999999999542)
```

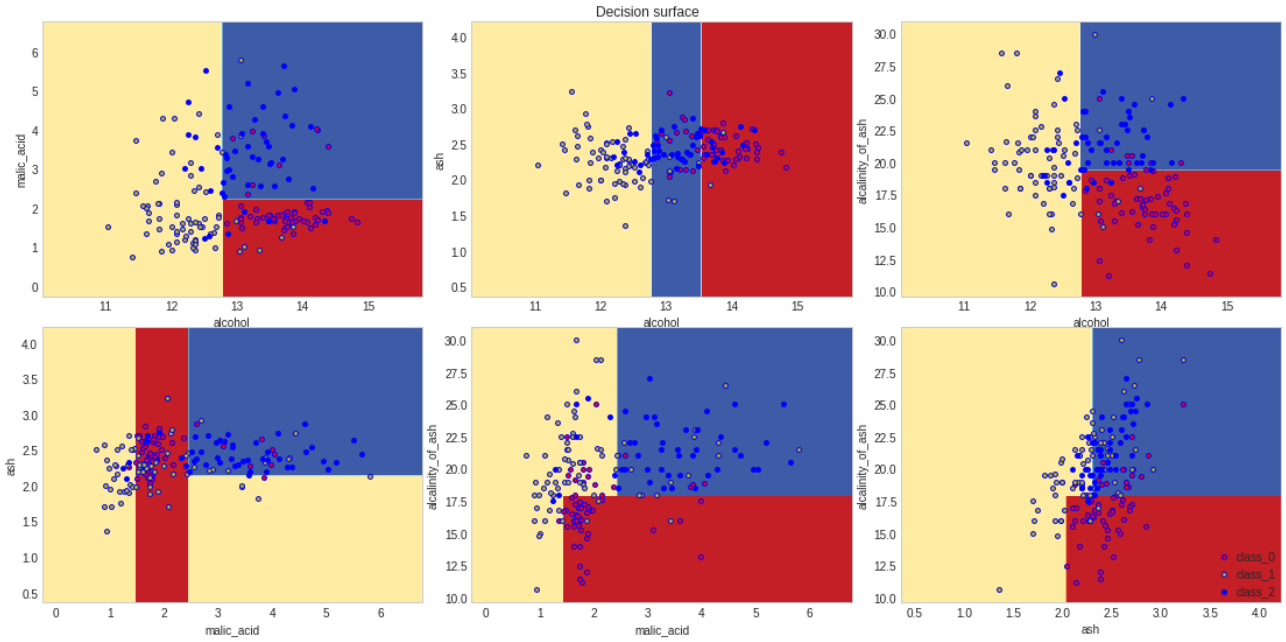


하이퍼파리미터를 변경해 보면서 결정 경계의 변화 확인

```
plt.figure(figsize=(16, 8))

for pairidx, pair in enumerate([[0, 1], [0, 2], [0, 3],
                                [1, 2], [1, 3], [2, 3]]):
  X = wine.data[:, pair]
  y = wine.target

  model = DecisionTreeClassifier(max_depth=2)
  model = model.fit(X, y)

  plt.subplot(2, 3, pairidx + 1)

  x_min, x_max = X[:, 0].min()-1, X[:, 0].max()+1
```

```
  y_min, y_max = X[:, 1].min()-1, X[:, 1].max()+1
  xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step),
                       np.arange(y_min, y_max, plot_step))
  plt.tight_layout(h_pad=-0.5, w_pad=0.5, pad=2.5)

  Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
  Z = Z.reshape(xx.shape)
  cs = plt.contourf(xx, yy, Z, cmap=plt.cm.RdYlBu)

  plt.xlabel(wine.feature_names[pair[0]])
  plt.ylabel(wine.feature_names[pair[1]])

  for i, color in zip(range(n_classes), plot_colors):
    idx = np.where(y == i)
    plt.scatter(X[idx, 0], X[idx, 1], c=color, label=wine.target_names[i],
                cmap=plt.cm.RdYlBu, edgecolor='b', s=15)

plt.suptitle("Decision surface")
plt.legend(loc='lower right', borderpad=0, handletextpad=0)
plt.axis('tight')
```

```
(0.3600000000000001, 4.220000000000003, 9.6, 30.979999999999542)
```



유방암 데이터 학습

교차 검증

전처리 없이 학습

```
model = DecisionTreeClassifier()
```

```
cross_val_score(
    estimator=model,
    X=cancer.data, y=cancer.target,
    cv=5,
    n_jobs=multiprocessing.cpu_count()
)
```

```
array([0.90350877, 0.90350877, 0.9122807 , 0.93859649, 0.90265487])
```

전처리 후 학습

```
model = make_pipeline(
    StandardScaler(),
    DecisionTreeClassifier()
)
```

```
cross_val_score(
    estimator=model,
    X=cancer.data, y=cancer.target,
    cv=5,
    n_jobs=multiprocessing.cpu_count()
)
```

```
array([0.9122807 , 0.90350877, 0.92105263, 0.96491228, 0.90265487])
```

학습된 결정 트리 시각화

```
model = DecisionTreeClassifier()
model.fit(cancer.data, cancer.target)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

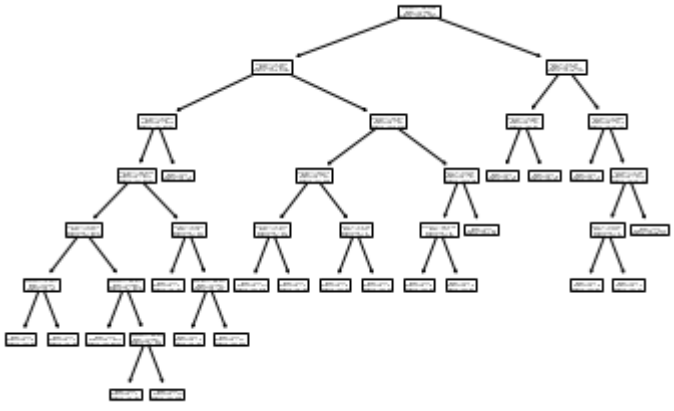
텍스트를 통한 시각화

[ ] ↳숨겨진 셀 *1*개

plot_tree를 사용한 시각화

```
tree.plot_tree(model)
```

```
[Text(209.25, 203.85, 'X[20] <= 16.795\ngini = 0.468\nsamples = 569\nvalue = [212, 3
 Text(136.01250000000002, 176.67000000000002, 'X[27] <= 0.136\ngini = 0.159\nsamples
 Text(78.46875, 149.49, 'X[12] <= 6.597\ngini = 0.03\nsamples = 333\nvalue = [5, 328
 Text(68.00625000000001, 122.31, 'X[13] <= 38.605\ngini = 0.024\nsamples = 332\nvalu
 Text(41.85, 95.13, 'X[14] <= 0.003\ngini = 0.012\nsamples = 319\nvalue = [2, 317]')
 Text(20.925, 67.94999999999999, 'X[27] <= 0.101\ngini = 0.245\nsamples = 7\nvalue =
 Text(10.4625, 40.77000000000001, 'gini = 0.0\nsamples = 6\nvalue = [0, 6]'),
 Text(31.387500000000003, 40.77000000000001, 'gini = 0.0\nsamples = 1\nvalue = [1, 0
 Text(62.775000000000006, 67.94999999999999, 'X[21] <= 33.27\ngini = 0.006\nsamples
 Text(52.3125, 40.77000000000001, 'gini = 0.0\nsamples = 292\nvalue = [0, 292]'),
 Text(73.2375, 40.77000000000001, 'X[21] <= 33.56\ngini = 0.095\nsamples = 20\nvalue
 Text(62.775000000000006, 13.590000000000003, 'gini = 0.0\nsamples = 1\nvalue = [1,
 Text(83.7, 13.590000000000003, 'gini = 0.0\nsamples = 19\nvalue = [0, 19]'),
 Text(94.16250000000001, 95.13, 'X[18] <= 0.016\ngini = 0.26\nsamples = 13\nvalue =
 Text(83.7, 67.94999999999999, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(104.625, 67.94999999999999, 'X[13] <= 39.15\ngini = 0.153\nsamples = 12\nvalue
 Text(94.16250000000001, 40.77000000000001, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]
 Text(115.0875, 40.77000000000001, 'gini = 0.0\nsamples = 11\nvalue = [0, 11]'),
 Text(88.93125, 122.31, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(193.55625, 149.49, 'X[21] <= 25.67\ngini = 0.476\nsamples = 46\nvalue = [28, 1
 Text(156.9375, 122.31, 'X[23] <= 810.3\ngini = 0.332\nsamples = 19\nvalue = [4, 15]
 Text(136.01250000000002, 95.13, 'X[24] <= 0.179\ngini = 0.124\nsamples = 15\nvalue
 Text(125.55000000000001, 67.94999999999999, 'gini = 0.0\nsamples = 14\nvalue = [0,
 Text(146.475, 67.94999999999999, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(177.8625, 95.13, 'X[0] <= 14.19\ngini = 0.375\nsamples = 4\nvalue = [3, 1]'),
 Text(167.4, 67.94999999999999, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(188.32500000000002, 67.94999999999999, 'gini = 0.0\nsamples = 1\nvalue = [0, 1
 Text(230.175, 122.31, 'X[7] <= 0.054\ngini = 0.198\nsamples = 27\nvalue = [24, 3]')
 Text(219.7125, 95.13, 'X[21] <= 28.545\ngini = 0.5\nsamples = 6\nvalue = [3, 3]'),
 Text(209.25, 67.94999999999999, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(230.175, 67.94999999999999, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(240.63750000000002, 95.13, 'gini = 0.0\nsamples = 21\nvalue = [21, 0]'),
 Text(282.4875, 176.67000000000002, 'X[1] <= 16.11\ngini = 0.109\nsamples = 190\nval
 Text(261.5625, 149.49, 'X[15] <= 0.021\ngini = 0.498\nsamples = 17\nvalue = [8, 9]'
 Text(251.10000000000002, 122.31, 'gini = 0.0\nsamples = 9\nvalue = [0, 9]'),
 Text(272.02500000000003, 122.31, 'gini = 0.0\nsamples = 8\nvalue = [8, 0]'),
 Text(303.4125, 149.49, 'X[24] <= 0.088\ngini = 0.023\nsamples = 173\nvalue = [171,
 Text(292.95, 122.31, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(313.875, 122.31, 'X[26] <= 0.18\ngini = 0.012\nsamples = 172\nvalue = [171, 1]
 Text(303.4125, 95.13, 'X[6] <= 0.053\ngini = 0.375\nsamples = 4\nvalue = [3, 1]'),
 Text(292.95, 67.94999999999999, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(313.875, 67.94999999999999, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(324.33750000000003, 95.13, 'gini = 0.0\nsamples = 168\nvalue = [168, 0]')]
```



▼ graphviz를 사용한 시각화

```
dot_data = tree.export_graphviz(decision_tree=model,
                                feature_names=cancer.feature_names,
                                class_names=cancer.target_names,
                                filled=True, rounded=True,
                                special_characters=True)
graph = graphviz.Source(dot_data)
graph
```

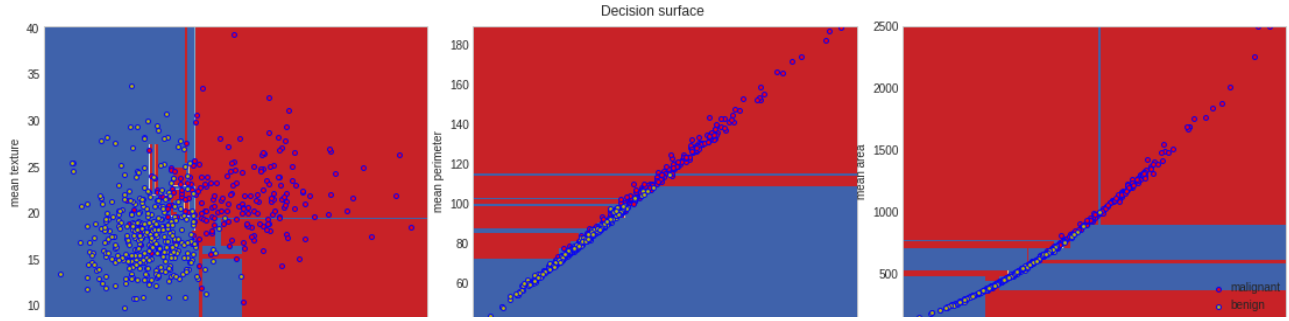
시각화

```
n_classes = 2
plot_colors = 'ryb'
plot_step = 0.02
```

결정 경계 시각화

```
plt.figure(figsize=(16, 8))

for pairidx, pair in enumerate([[0, 1], [0, 2], [0, 3]]):
  X = cancer.data[:, pair]
  y = cancer.target

  model = DecisionTreeClassifier()
  model = model.fit(X, y)

  plt.subplot(2, 3, pairidx + 1)

  x_min, x_max = X[:, 0].min()-1, X[:, 0].max()+1
  y_min, y_max = X[:, 1].min()-1, X[:, 1].max()+1
  xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step),
                       np.arange(y_min, y_max, plot_step))
  plt.tight_layout(h_pad=-0.5, w_pad=0.5, pad=2.5)

  Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
  Z = Z.reshape(xx.shape)
  cs = plt.contourf(xx, yy, Z, cmap=plt.cm.RdYlBu)

  plt.xlabel(cancer.feature_names[pair[0]])
  plt.ylabel(cancer.feature_names[pair[1]])

  for i, color in zip(range(n_classes), plot_colors):
    idx = np.where(y == i)
    plt.scatter(X[idx, 0], X[idx, 1], c=color, label=cancer.target_names[i],
                cmap=plt.cm.RdYlBu, edgecolor='b', s=15)

plt.suptitle("Decision surface")
plt.legend(loc='lower right', borderpad=0, handletextpad=0)
plt.axis('tight')
```

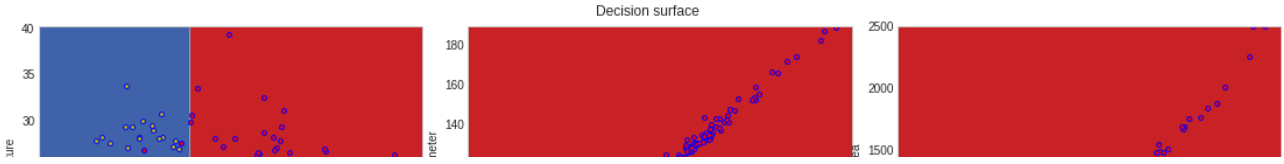
```
(5.981, 29.10099999999951, 142.5, 2501.980000001207)
```



▼ 하이퍼파리미터를 변경해 보면서 결정 경계의 변화 확인

```
plt.figure(figsize=(16, 8))

for pairidx, pair in enumerate([[0, 1], [0, 2], [0, 3]]):
  X = cancer.data[:, pair]
  y = cancer.target

  model = DecisionTreeClassifier(max_depth=2)
  model = model.fit(X, y)

  plt.subplot(2, 3, pairidx + 1)

  x_min, x_max = X[:, 0].min()-1, X[:, 0].max()+1
  y_min, y_max = X[:, 1].min()-1, X[:, 1].max()+1
  xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step),
                       np.arange(y_min, y_max, plot_step))
  plt.tight_layout(h_pad=-0.5, w_pad=0.5, pad=2.5)

  Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
  Z = Z.reshape(xx.shape)
  cs = plt.contourf(xx, yy, Z, cmap=plt.cm.RdYlBu)

  plt.xlabel(cancer.feature_names[pair[0]])
  plt.ylabel(cancer.feature_names[pair[1]])

  for i, color in zip(range(n_classes), plot_colors):
    idx = np.where(y == i)
    plt.scatter(X[idx, 0], X[idx, 1], c=color, label=cancer.target_names[i],
                cmap=plt.cm.RdYlBu, edgecolor='b', s=15)

plt.suptitle("Decision surface")
plt.legend(loc='lower right', borderpad=0, handletextpad=0)
plt.axis('tight')
```

```
(5.981, 29.10099999999951, 142.5, 2501.980000001207)
```



회귀 - DecisionTreeRegressor()

보스턴 주택 가격 데이터 학습

교차 검증

전처리 없이 학습

```
model = DecisionTreeRegressor()
```

```
cross_val_score(
    estimator=model,
    X=boston.data, y=boston.target,
    cv=5,
    n_jobs=multiprocessing.cpu_count()
)
```

```
array([ 0.58639239,  0.61588248,  0.6300357 ,  0.38807055, -1.44578502])
```

전처리 후 학습

```
model = make_pipeline(
    StandardScaler(),
    DecisionTreeRegressor()
)
```

```
cross_val_score(
    estimator=model,
    X=boston.data, y=boston.target,
    cv=5,
    n_jobs=multiprocessing.cpu_count()
)
```

```
array([ 0.60166808,  0.59866933,  0.60669357,  0.39741031, -1.84407281])
```

학습된 결정 트리 시각화

```
model = DecisionTreeRegressor()
model.fit(boston.data, boston.target)
```

```
model.fit(boston.data, boston.target)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

▶ 텍스트를 통한 시각화

[ ] ↳ 숨겨진 셀 1개

▼ plot_tree를 사용한 시각화

```
tree.plot_tree(model)
```

```
[Text(227.6398862631712, 212.004, 'X[5] <= 6.941\nmse = 84.42\nsamples = 506\nvalue
 Text(143.39095612911376, 201.132, 'X[12] <= 14.4\nmse = 40.273\nsamples = 430\nvalu
 Text(63.55613858617206, 190.26, 'X[7] <= 1.385\nmse = 26.009\nsamples = 255\nvalue
 Text(35.82953197170901, 179.388, 'X[12] <= 10.83\nmse = 78.146\nsamples = 5\nvalue
 Text(35.21096384237875, 168.516, 'mse = 0.0\nsamples = 4\nvalue = 50.0'),
 Text(36.448100101039266, 168.516, 'mse = -0.0\nsamples = 1\nvalue = 27.9'),
 Text(91.2827452006351, 179.388, 'X[5] <= 6.543\nmse = 14.885\nsamples = 250\nvalue
 Text(37.68523635969977, 168.516, 'X[12] <= 7.57\nmse = 8.39\nsamples = 195\nvalue =
 Text(13.985438799076213, 157.644, 'X[9] <= 222.5\nmse = 3.015\nsamples = 43\nvalue
 Text(13.366870669745959, 146.772, 'mse = 0.0\nsamples = 1\nvalue = 28.7'),
 Text(14.604006928406466, 146.772, 'X[2] <= 5.48\nmse = 2.541\nsamples = 42\nvalue =
 Text(10.438337182448038, 135.9, 'X[10] <= 19.95\nmse = 1.328\nsamples = 19\nvalue =
 Text(6.649607390300231, 125.02799999999999, 'X[11] <= 393.33\nmse = 0.72\nsamples =
 Text(3.09284064665127, 114.156, 'X[2] <= 3.895\nmse = 0.337\nsamples = 8\nvalue = 2
 Text(1.8557043879907622, 103.28399999999999, 'X[6] <= 26.05\nmse = 0.096\nsamples =
 Text(1.2371362586605081, 92.412, 'X[0] <= 0.066\nmse = 0.002\nsamples = 2\nvalue =
 Text(0.6185681293302541, 81.53999999999999, 'mse = 0.0\nsamples = 1\nvalue = 22.5')
 Text(1.8557043879907622, 81.53999999999999, 'mse = -0.0\nsamples = 1\nvalue = 22.6'
```

▼ graphviz를 사용한 시각화

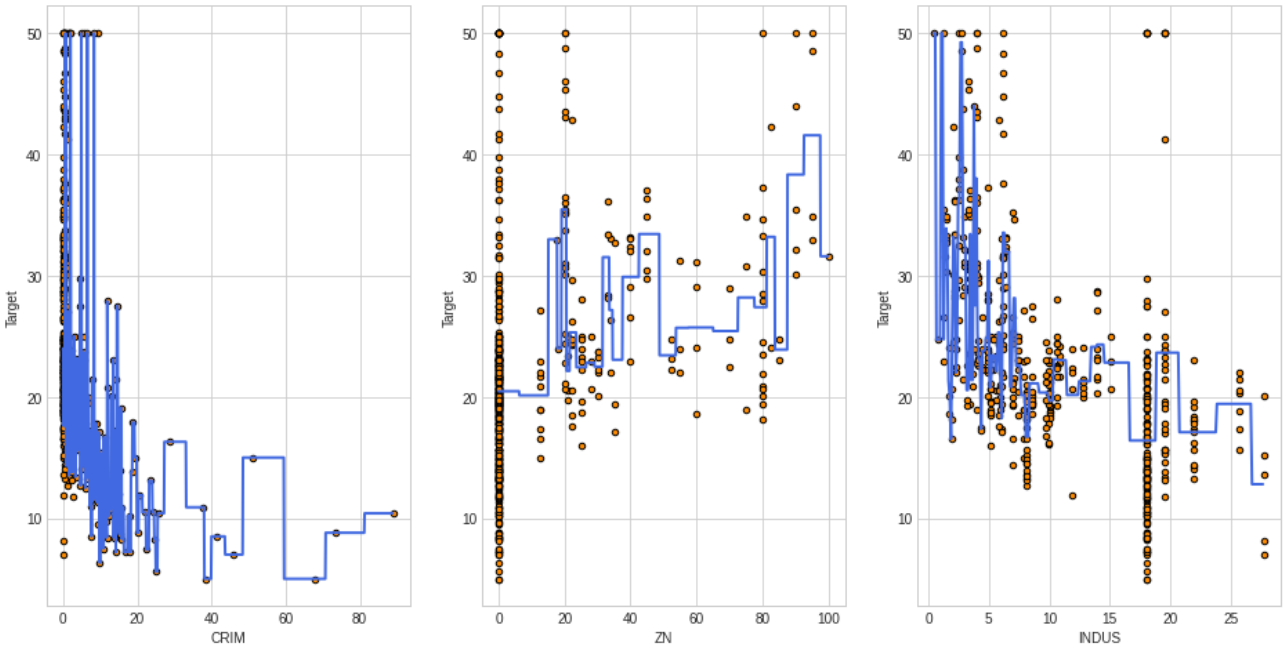
```
 Text(3.7114087759815244, 92.412, 'mse = 0.0\nsamples = 2\nvalue = 23.7'),
```

```
dot_data = tree.export_graphviz(decision_tree=model,
                                feature_names=boston.feature_names,
                                filled=True, rounded=True,
                                special_characters=True)
graph = graphviz.Source(dot_data)
graph
```

시각화

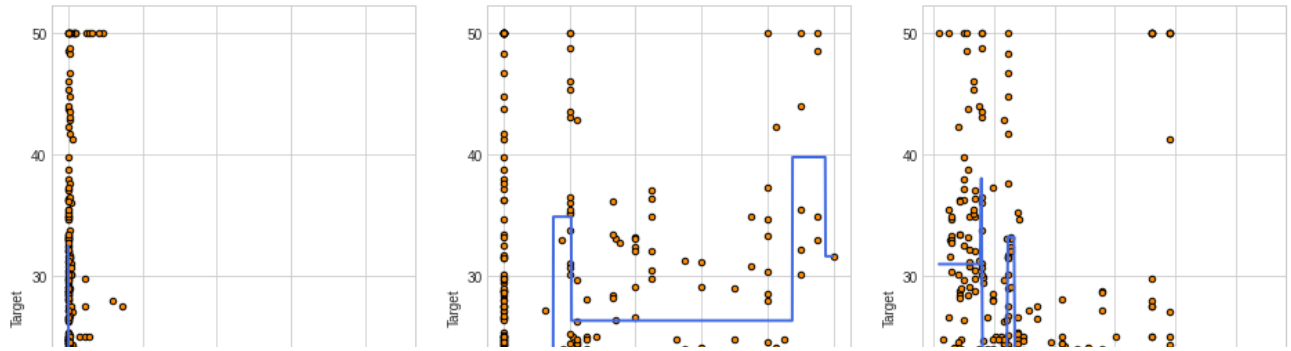
회귀식 시각화

```
plt.figure(figsize=(16, 8))

for pairidx, pair in enumerate([0, 1, 2]):
  X = boston.data[:, pair].reshape(-1, 1)
  y = boston.target

  model = DecisionTreeRegressor()
  model.fit(X, y)

  X_test = np.arange(min(X), max(X), 0.1)[:, np.newaxis]
  predict = model.predict(X_test)

  plt.subplot(1, 3, pairidx + 1)
  plt.scatter(X, y, s=20, edgecolors='k',
              c='darkorange', label='data')
  plt.plot(X_test, predict, color='royalblue', linewidth=2)
  plt.xlabel(boston.feature_names[pair])
  plt.ylabel('Target')
```



▼ 하이퍼파리미터를 변경해 보면서 회귀식 시각화

```
plt.figure(figsize=(16, 8))

for pairidx, pair in enumerate([0, 1, 2]):
  X = boston.data[:, pair].reshape(-1, 1)
  y = boston.target

  model = DecisionTreeRegressor(max_depth=3)
  model.fit(X, y)

  X_test = np.arange(min(X), max(X), 0.1)[:, np.newaxis]
  predict = model.predict(X_test)

  plt.subplot(1, 3, pairidx + 1)
  plt.scatter(X, y, s=20, edgecolors='k',
              c='darkorange', label='data')
  plt.plot(X_test, predict, color='royalblue', linewidth=2)
  plt.xlabel(boston.feature_names[pair])
  plt.ylabel('Target')
```



당뇨병 데이터 학습

교차 검증

전처리 없이 학습

```
model = DecisionTreeRegressor()
```

```
cross_val_score(
    estimator=model,
    X=diabetes.data, y=diabetes.target,
    cv=5,
    n_jobs=multiprocessing.cpu_count()
)
```

```
array([-0.27174991, -0.03353101, -0.20303123,  0.14985239, -0.25872968])
```

전처리 후 학습

```
model = make_pipeline(
    StandardScaler(),
    DecisionTreeRegressor()
)
```

```
Text(77.78494226527943, 58.051999999999996, 'mse = 0.0\nsamples = 1\nvalue = 19.4 ');
```

```
cross_val_score(
    estimator=model,
    X=diabetes.data, y=diabetes.target,
    cv=5,
    n_jobs=multiprocessing.cpu_count()
)
```

```
array([-0.33304976,  0.00776503, -0.18311132, -0.05614469, -0.22256672])
```

```
Text(83.35205542725174, 70.668, 'X[7] <= 2.266\nmse = 0.85\nsamples = 5\nvalue = 21
```

학습된 결정 트리 시각화

```
Text(82.73348729792149, 38.05199999999999, 'mse = 0.0\nsamples = 2\nvalue = 20.8')
```

```
model = DecisionTreeRegressor()
model.fit(diabetes.data, diabetes.target)
```

```
model.fit(diabetes.data, diabetes.target)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

```
Text(92.12799076212471, 125.02799999999999, 'X[0] <= 0.046\nmse = 5.482\nsamples =
```

텍스트를 통한 시각화

[ ] ↳ 숨겨진 셀 1개

```
Text(86.44489607390301, 81.53999999999999, 'mse = 0.0\nsamples = 1\nvalue = 19.8'),
```

plot_tree를 사용한 시각화

```
tree.plot_tree(model)
```

```
[Text(179.37361789772729, 212.26285714285714, 'X[8] <= -0.004\nmse = 5929.885\nsampl
 Text(104.42594074675324, 201.90857142857143, 'X[2] <= 0.006\nmse = 3240.821\nsample
 Text(71.02950487012987, 191.5542857142857, 'X[6] <= 0.021\nmse = 2143.968\nsamples
 Text(40.46949350649351, 181.2, 'X[4] <= 0.063\nmse = 2856.847\nsamples = 87\nvalue
 Text(31.545116883116883, 170.84571428571428, 'X[1] <= 0.003\nmse = 2496.899\nsample
 Text(15.783428571428571, 160.49142857142857, 'X[3] <= -0.035\nmse = 2955.72\nsample
 Text(9.21787012987013, 150.13714285714286, 'X[0] <= -0.051\nmse = 2357.959\nsamples
 Text(4.17412987012987, 139.78285714285715, 'X[3] <= -0.092\nmse = 2049.0\nsamples =
 Text(3.4784415584415584, 129.42857142857144, 'mse = 0.0\nsamples = 1\nvalue = 55.0'
 Text(4.869818181818182, 129.42857142857144, 'X[3] <= -0.05\nmse = 1335.633\nsamples
 Text(3.4784415584415584, 119.07428571428571, 'X[9] <= -0.026\nmse = 401.6\nsamples
 Text(2.7827532467532468, 108.72, 'X[4] <= -0.039\nmse = 50.75\nsamples = 4\nvalue =
 Text(1.3913766233766234, 98.36571428571429, 'X[5] <= -0.067\nmse = 16.0\nsamples =
 Text(0.6956883116883117, 88.01142857142858, 'mse = 0.0\nsamples = 1\nvalue = 142.0'
 Text(2.087064935064935, 88.01142857142858, 'mse = 0.0\nsamples = 1\nvalue = 150.0')
 Text(4.17412987012987, 98.36571428571429, 'X[0] <= -0.075\nmse = 1.0\nsamples = 2\n
 Text(3.4784415584415584, 88.01142857142858, 'mse = 0.0\nsamples = 1\nvalue = 158.0'
 Text(4.869818181818182, 88.01142857142858, 'mse = 0.0\nsamples = 1\nvalue = 160.0')
```

▼ graphviz를 사용한 시각화

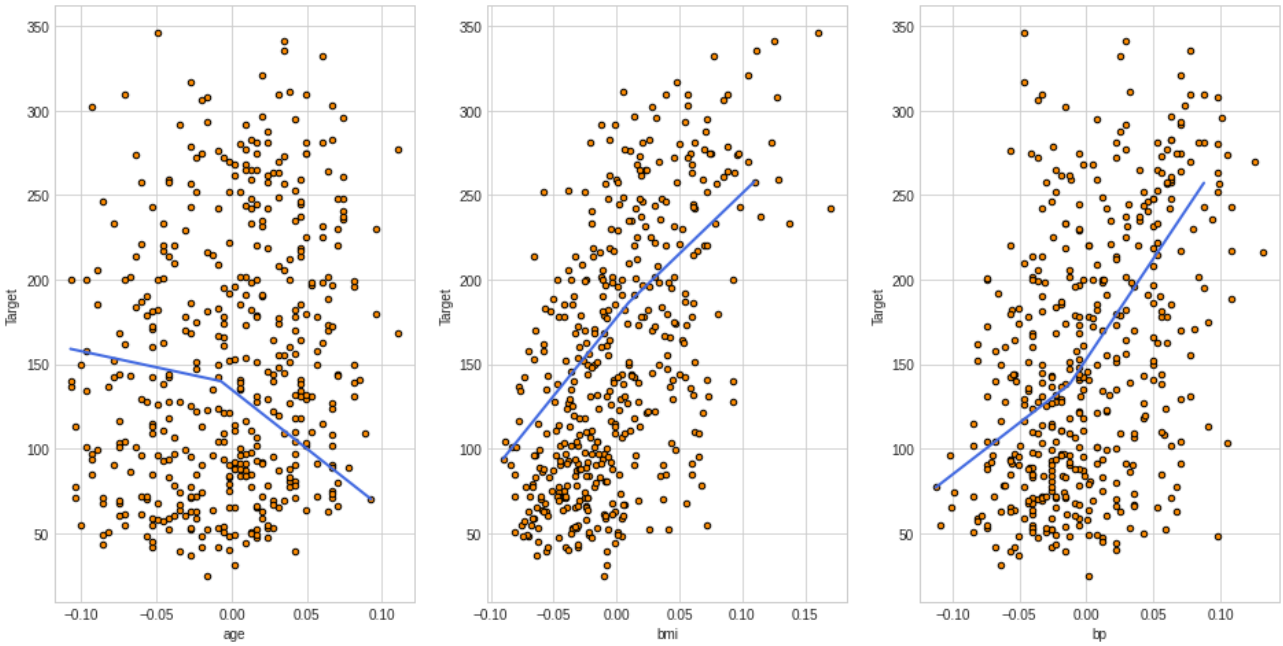
```
 Text(5.565506493506493, 108.72, 'mse = 0.0\nsamples = 1\nvalue = 134.0'),
```

```
dot_data = tree.export_graphviz(decision_tree=model,
                                feature_names=diabetes.feature_names,
                                filled=True, rounded=True,
                                special_characters=True)
graph = graphviz.Source(dot_data)
graph
```

시각화

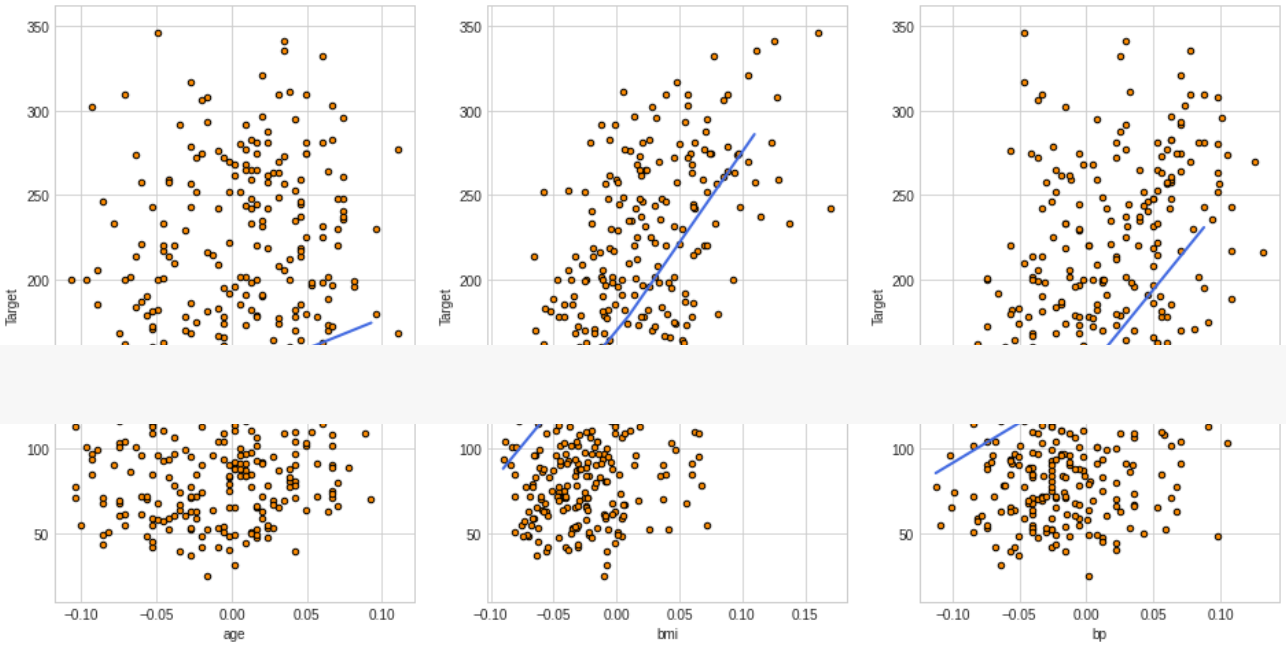
회귀식 시각화

```
plt.figure(figsize=(16, 8))

for pairidx, pair in enumerate([0, 2, 3]):
  X = diabetes.data[:, pair].reshape(-1, 1)
  y = diabetes.target

  model = DecisionTreeRegressor()
  model.fit(X, y)

  X_test = np.arange(min(X), max(X), 0.1)[:, np.newaxis]
  predict = model.predict(X_test)

  plt.subplot(1, 3, pairidx + 1)
  plt.scatter(X, y, s=20, edgecolors='k',
              c='darkorange', label='data')
  plt.plot(X_test, predict, color='royalblue', linewidth=2)
  plt.xlabel(diabetes.feature_names[pair])
  plt.ylabel('Target')
```



▼ 하이퍼파리미터를 변경해 보면서 회귀식 시각화

```
plt.figure(figsize=(16, 8))

for pairidx, pair in enumerate([0, 2, 3]):
  X = diabetes.data[:, pair].reshape(-1, 1)
  y = diabetes.target

  model = DecisionTreeRegressor(max_depth=3)
  model.fit(X, y)

  X_test = np.arange(min(X), max(X), 0.1)[:, np.newaxis]
  predict = model.predict(X_test)

  plt.subplot(1, 3, pairidx + 1)
  plt.scatter(X, y, s=20, edgecolors='k',
              c='darkorange', label='data')
  plt.plot(X_test, predict, color='royalblue', linewidth=2)
  plt.xlabel(diabetes.feature_names[pair])
  plt.ylabel('Target')
```


Target
age
bmi
bp