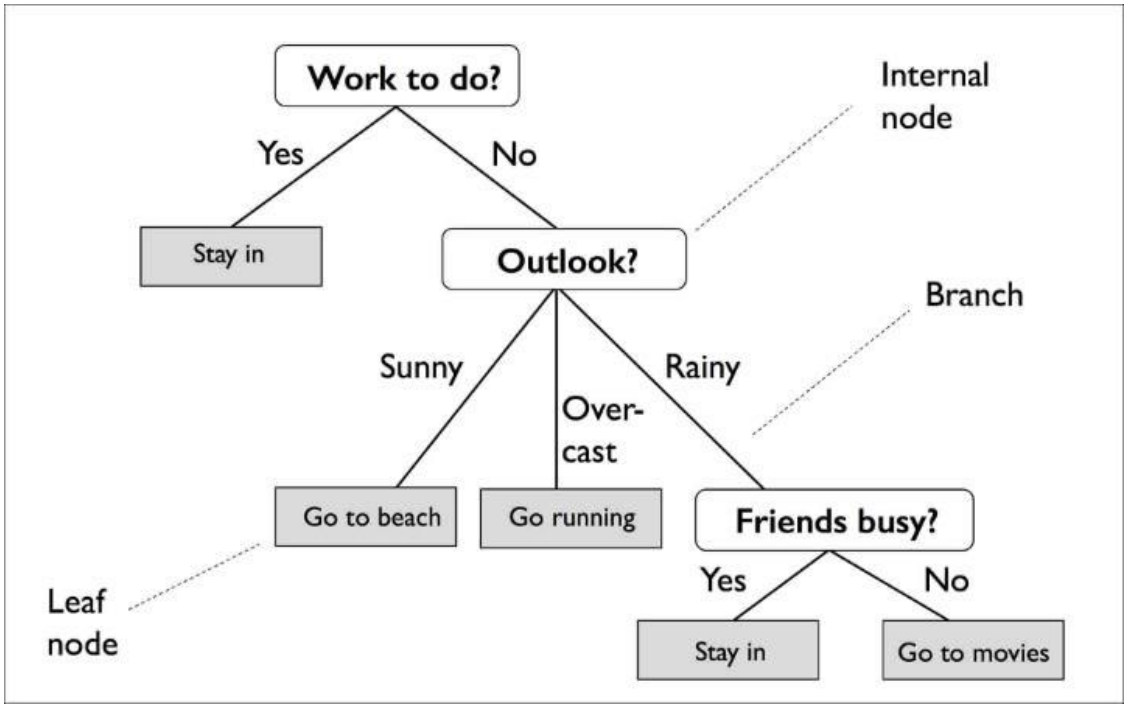
Machine Learning

Decision Tree

김선녕(sykim.lecture@gmail.com)

개요

- 트리의 루트$(root)$에서 시작해서 **정보이득**$(\boldsymbol{Information\ Gain, IG})$이 최대가 되는 특성으로 데이터를 분할
- 트리의 리프노드$(leaf\ node)$가 순수해질 때까지 모든 자식 노드에서 분할 작업을 반복
  - 해당 노드의 모든 샘플은 동일한 클래스에 속한다
  - 노드가 많은 깊은 트리는 과대적합 가능성이 높다
- 가지치기$(pruning)$: 트리의 최대 깊이를 제한



결정트리의 학습과 시각화(1)

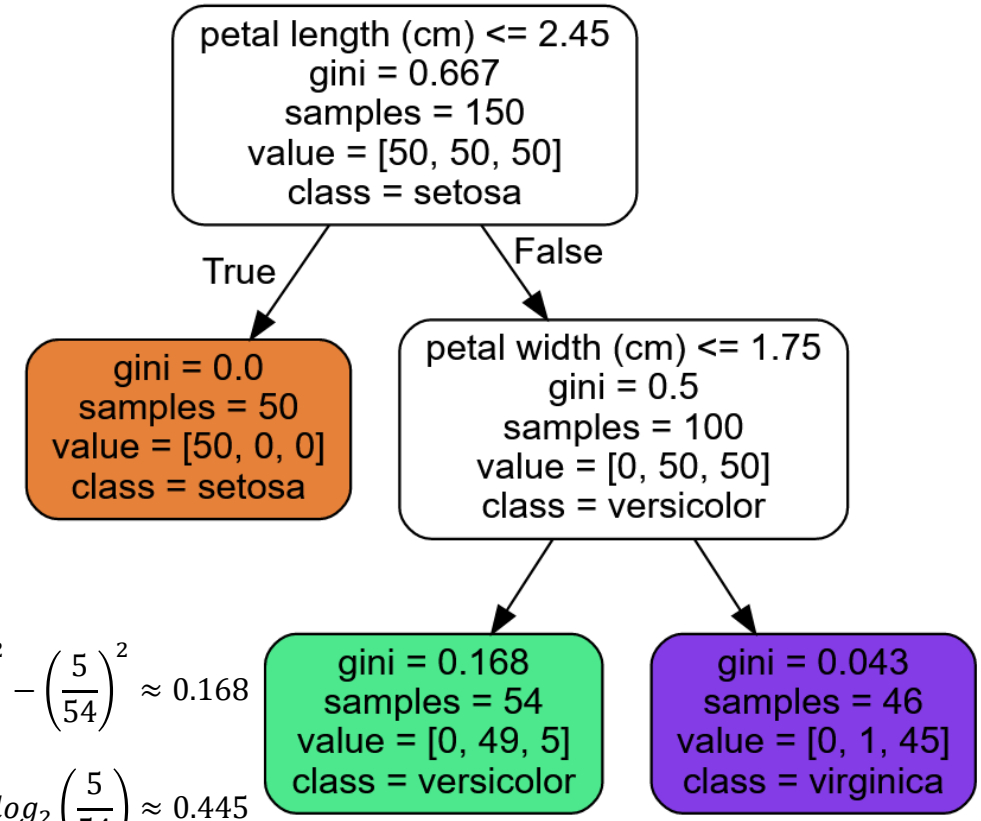
- 붓꽃 데이터셋
  - setosa, versicolor, virginica

```
import numpy as np
import os
from graphviz import Source
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_graphviz

# 그림을 저장할 위치
PROJECT_ROOT_DIR = "."
CHAPTER_ID = "decision_trees"
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, "images", CHAPTER_ID)
os.makedirs(IMAGES_PATH, exist_ok=True)

# 붓꽃 데이터셋에 DecisionTreeClassifier를 훈련
iris = load_iris()
X = iris.data[:, 2:] # 꽃잎 길이와 너비
y = iris.target

tree_clf = DecisionTreeClassifier(max_depth=2, random_state=42)
tree_clf.fit(X, y)
```

결정트리의 학습과 시각화(2)


```
# iris_tree.dot 파일로 시각화
# Graphviz(http://www.graphviz.org) : 오픈소스 그래프 시각화 패키지
dot_data = export_graphviz(
        tree_clf,
        out_file=os.path.join(IMAGES_PATH, "iris_tree.dot"),
        feature_names=iris.feature_names[2:],
        class_names=iris.target_names,
        rounded=True,
        filled=True
    )
Source.from_file(os.path.join(IMAGES_PATH, "iris_tree.dot"))
```

결정트리의 학습과 시각화(3)



$$1 - \left(\frac{0}{54}\right)^2 - \left(\frac{49}{54}\right)^2 - \left(\frac{5}{54}\right)^2 \approx 0.168$$

$$-\frac{49}{54} log_2\left(\frac{49}{54}\right) - \frac{5}{54} log_2\left(\frac{5}{54}\right) \approx 0.445$$

ref : Hands-On machine learning-O'REILLY

sample : 해당 노드에 적용된 훈련샘플 갯수

value :해당 노드에 적용된 각 클래스의 훈련샘플 갯수

gini : 불순도(impurity)

한 노드의 모든 샘플이 같은 클래스에 속해 있다면 gini = 0

정보 이득($Information\ Gain, IG$) 최대화

- 정보 이득은 부모 노드의 불순도와 자식 노드의 불순도 합의 차이
  - 자식 노드의 불순도가 낮을수록 정보 이득이 커진다
- 가장 정보가 풍부한 특성으로 노드를 나누기 위해 최적화할 목적 함수 정의
  - 목적함수 : 각 분할에서 정보 이득을 최대화
- 정보이득(IG)의 정의

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$

f : 분할에 사용할 특성
D_p : 부모노드의 데이터 셋
D_j : j번째 자식 노드의 데이터셋
I : 불순도 지표
N_p : 부모노드에 있는 샘플 개수
N_j : j번째 자식 노드에 있는 샘플 개수

- 대부분의 라이브러리(사이킷런등..)는 **이진 결정 트리** 사용
  - 구현을 간단하게 하고 탐색 공간을 줄이기 위해
  - 부모노드는 두 개의 자식 노드 D_{left}, D_{right}로 나뉜다

$$ID(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

불순도(impurity) 지표

- 지니 불순도(gini impurity)

$$I_G(t) = 1 - \sum_{i=1}^{c} p(i|t)^2$$

$$1 - \left(\left(\frac{0}{54}\right)^2 + \left(\frac{49}{54}\right)^2 + \left(\frac{5}{54}\right)^2\right) \approx 0.168$$

- 엔트로피 불순도(entropy impurity)

$$I_H(t) = -\sum_{\substack{i=1 \\ p(i|t)\neq 0}}^{c} p(i|t) log_2 p(i|t)$$

$$-\left(\frac{49}{54} log_2\left(\frac{49}{54}\right) + \frac{5}{54} log_2\left(\frac{5}{54}\right)\right) \approx 0.445$$

- 분류오차 불순도(classification error impurity)

$$I_E = 1 - max\{p(i|t)\}$$

$$1 - \left(\frac{49}{54}\right) \approx 0.093$$

$\therefore$ $p(i|t)$: 특정 노드 t에서 클래스 i에 속한 샘플 비율

2 개의 분할 시나리오 - 지니 불순도

$$I_G(D_p) = 1 - (0.5^2 + 0.5^2) = 0.5$$

$$A: I_G(D_{left}) = 1 - \left(\left(\frac{3}{4}\right)^2 + \left(\frac{1}{4}\right)^2\right) = \frac{3}{8} = 0.375$$

$$A: I_G(D_{right}) = 1 - \left(\left(\frac{1}{4}\right)^2 + \left(\frac{3}{4}\right)^2\right) = \frac{3}{8} = 0.375$$

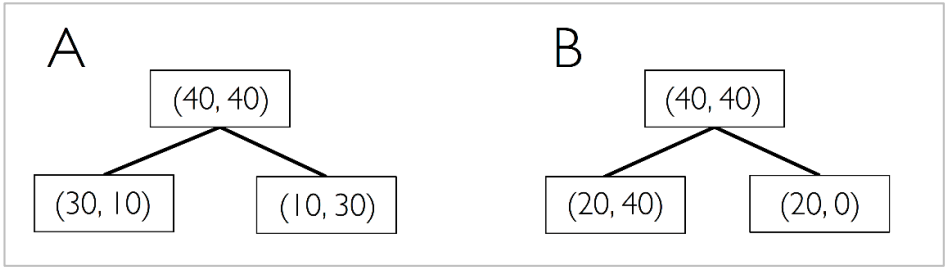
$$A: IG_G = 0.5 - \frac{4}{8}0.375 - \frac{4}{8}0.375 = \mathbf{0.125}$$

$$B: I_G(D_{left}) = 1 - \left(\left(\frac{2}{6}\right)^2 + \left(\frac{4}{6}\right)^2\right) = \frac{4}{9} = 0.\overline{4}$$

$$B: I_G(D_{right}) = 1 - (1^2 + 0^2) = 0$$

$$B: IG_G = 0.5 - \frac{6}{8}0.\overline{4} - 0 = \mathbf{0.1\overline{6}}$$

$$I_G(t) = 1 - \sum_{i=1}^{c} p(i|t)^2$$



ref : python machine learning 2nd

2 개의 분할 시나리오 - 엔트로피 불순도

$$I_H(D_P) = -\left(0.5log_2(0.5) + 0.5log_2(0.5)\right) = 1$$

$$A: I_H(D_{left}) = -\left(\frac{3}{4}log_2\left(\frac{3}{4}\right) + \frac{1}{4}log_2\left(\frac{1}{4}\right)\right) = 0.81$$

$$A: I_H(D_{right}) = -\left(\frac{1}{4}log_2\left(\frac{1}{4}\right) + \frac{3}{4}log_2\left(\frac{3}{4}\right)\right) = 0.81$$

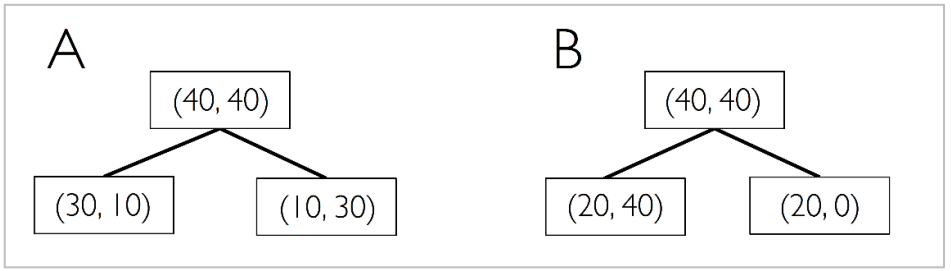
$$A: IG_H = 1 - \frac{4}{8}0.81 - \frac{4}{8}0.81 = \mathbf{0.19}$$

$$B: I_H(D_{left}) = -\left(\frac{2}{6}log_2\left(\frac{2}{6}\right) + \frac{4}{6}log_2\left(\frac{4}{6}\right)\right) = 0.92$$

$$B: I_H(D_{right}) = 0$$

$$B: IG_H = 1 - \frac{6}{8}0.92 - 0 = \mathbf{0.31}$$

$$I_H(t) = -\sum_{\substack{i=1 \\ p(i|t)\neq 0}}^{c} p(i|t)log_2 p(i|t)$$

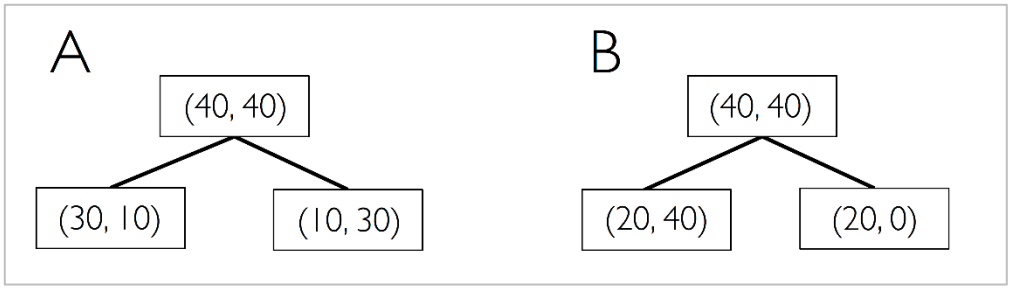


ref : python machine learning 2nd

2 개의 분할 시나리오 - 분류 오차

$$I_E(D_p) = 1 - 0.5 = 0.5$$

$$I_E = 1 - max\{p(i|t)\}$$



$$A: I_E(D_{left}) = 1 - \frac{3}{4} = 0.25$$

$$A: I_E(D_{right}) = 1 - \frac{3}{4} = 0.25$$

$$A: IG_E = 0.5 - \frac{4}{8}0.25 - \frac{4}{8}0.25 = \mathbf{0.25}$$

$$B: I_E(D_{left}) = 1 - \frac{4}{6} = \frac{1}{3}$$

$$B: I_E(D_{right}) = 1 - 1 = 0$$

$$B: IG_E = 0.5 - \frac{6}{8} \times \frac{1}{3} - 0 = \mathbf{0.25}$$

ref : python machine learning 2nd

3개의 불순도 비교(1)

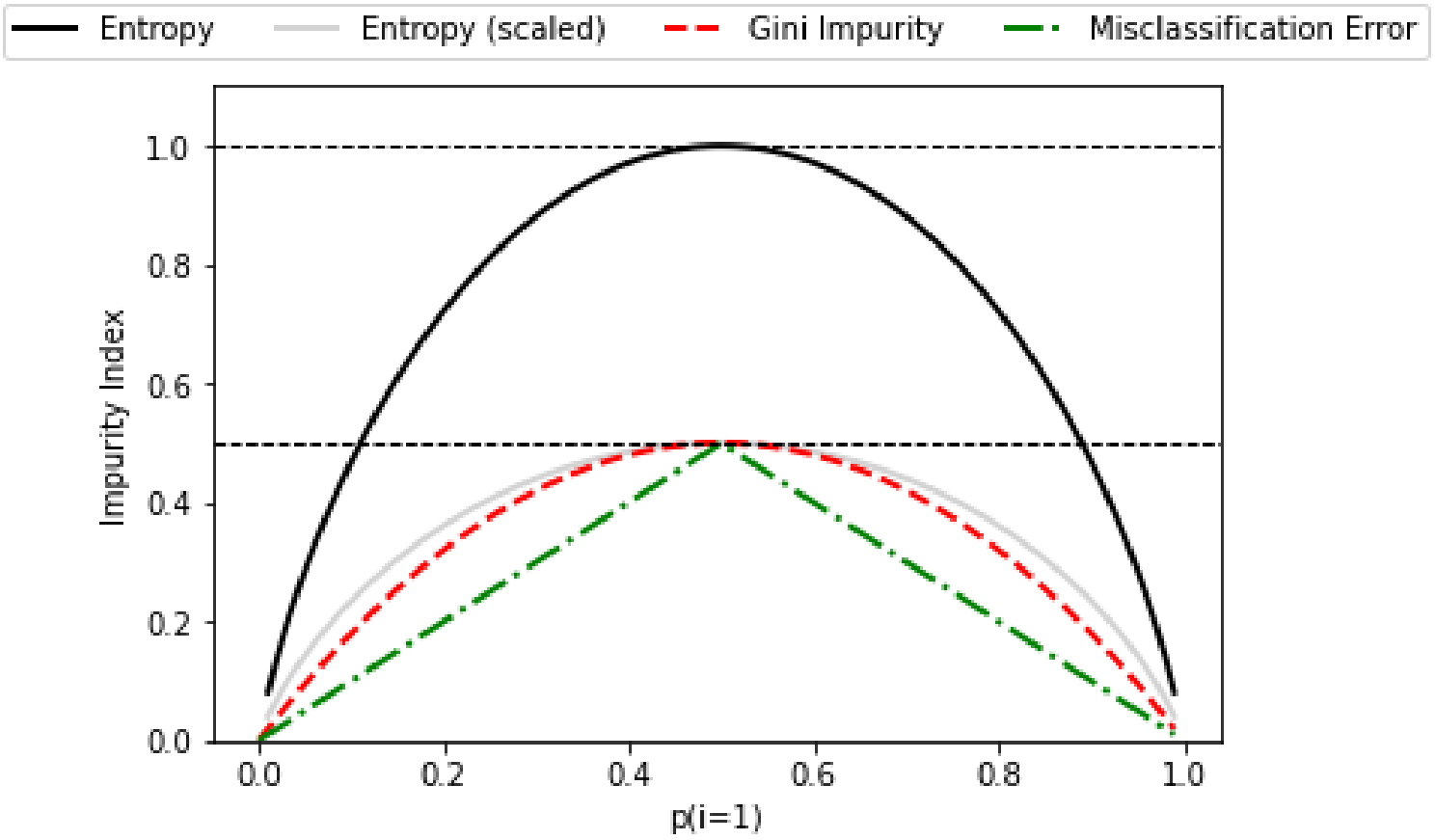

```
import matplotlib.pyplot as plt
import numpy as np

def gini(p):
    return p * (1 - p) + (1 - p) * (1 - (1 - p))

def entropy(p):
    return - p * np.log2(p) - (1 - p) * np.log2((1 - p))

def error(p):
    return 1 - np.max([p, 1 - p])

x = np.arange(0.0, 1.0, 0.01)

ent = [entropy(p) if p != 0 else None for p in x]

# 스케일 조정된 엔트로피(entropy/2) 추가
sc_ent = [e * 0.5 if e else None for e in ent]
err = [error(i) for i in x]
```

3개의 불순도 비교(2)


```
fig = plt.figure()
ax = plt.subplot(111)
for i, lab, ls, c, in zip([ent, sc_ent, gini(x), err],
                          ['Entropy', 'Entropy (scaled)',
                           'Gini Impurity', 'Misclassification Error'],
                          ['-', '-', '--', '-.'],
                          ['black', 'darkgray', 'red', 'green', 'cyan']):
    line = ax.plot(x, i, label=lab, linestyle=ls, lw=2, color=c)

ax.legend(loc='upper center', bbox_to_anchor=(0.5, 1.15),
          ncol=5, fancybox=True, shadow=False)

ax.axhline(y=0.5, linewidth=1, color='k', linestyle='--')
ax.axhline(y=1.0, linewidth=1, color='k', linestyle='--')
plt.ylim([0, 1.1])
plt.xlabel('p(i=1)')
plt.ylabel('Impurity Index')
plt.show()
```

3개의 불순도 비교(3)

- 클래스 1의 확률 범위 $[0, 1]$ 에 대한 불순도
- 스케일 조정된 엔트로피$(entropy/2)$ 추가
  - 지니 불순도가 엔트로피와 분류 오차의 중간임을 관찰



ref : python machine learning 2nd

3개의 불순도 비교(4)

- 지니 불순도와 엔트로피 사용에서 큰 차이가 없다.
  - 둘 다 비슷한 트리를 만든다
  - 지니 불순도가 조금 더 빠르기 때문에 기본값으로 좋다
- 트리가 만들어지는 경우 지니 불순도가 가장 빈도 높은 클래스를 한쪽 가지 $(branch)$로 고립시키는 경향이 있는 반면 엔트로피는 조금 더 균형 잡힌 트리를 만든다
- 분류오차 : 가지치기에는 좋은 기준이지만 결정 트리를 구성하는데 권장되지 않는다.
  - 노드의 클래스 확률 변화에 덜 민감하기 때문
- 보통 불순도 조건을 바꾸어 트리를 평가하는 것보다 가지치기 수준을 바꾸면서 튜닝하는 것이 훨씬 낫다

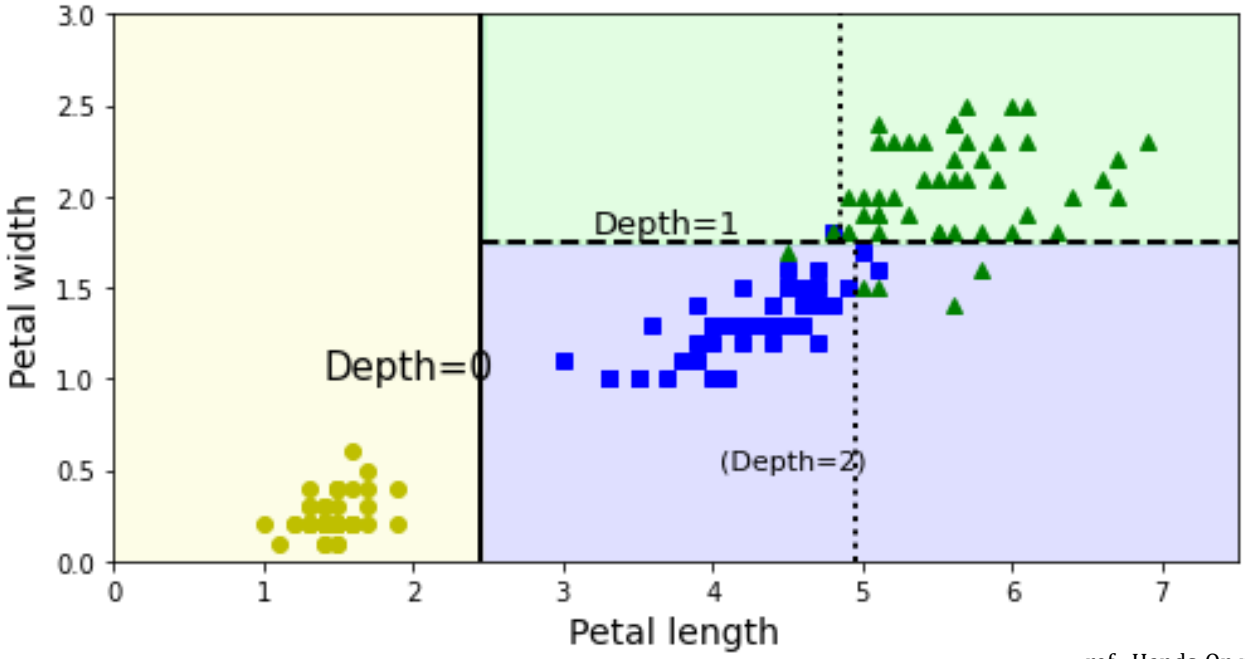
결정트리의 경계(1)


```
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt

def plot_decision_boundary(clf, X, y, axes=[0, 7.5, 0, 3], iris=True, legend=False
, plot_training=True):
    x1s = np.linspace(axes[0], axes[1], 100)
    x2s = np.linspace(axes[2], axes[3], 100)
    x1, x2 = np.meshgrid(x1s, x2s)
    X_new = np.c_[x1.ravel(), x2.ravel()]
    y_pred = clf.predict(X_new).reshape(x1.shape)
    custom_cmap = ListedColormap(['#fafab0','#9898ff','#a0faa0'])
    plt.contourf(x1, x2, y_pred, alpha=0.3, cmap=custom_cmap)
    if not iris:
        custom_cmap2 = ListedColormap(['#7d7d58','#4c4c7f','#507d50'])
        plt.contour(x1, x2, y_pred, cmap=custom_cmap2, alpha=0.8)
    if plot_training:
        plt.plot(X[:, 0][y==0], X[:, 1][y==0], "yo", label="Iris setosa")
        plt.plot(X[:, 0][y==1], X[:, 1][y==1], "bs", label="Iris versicolor")
        plt.plot(X[:, 0][y==2], X[:, 1][y==2], "g^", label="Iris virginica")
        plt.axis(axes)
    if iris:
        plt.xlabel("Petal length", fontsize=14)
        plt.ylabel("Petal width", fontsize=14)
    else:
        plt.xlabel(r"$x_1$", fontsize=18)
        plt.ylabel(r"$x_2$", fontsize=18, rotation=0)
    if legend:
        plt.legend(loc="lower right", fontsize=14)
```

결정트리의 경계(2)


```
plt.figure(figsize=(8, 4))
plot_decision_boundary(tree_clf, X, y)
plt.plot([2.45, 2.45], [0, 3], "k-", linewidth=2)
plt.plot([2.45, 7.5], [1.75, 1.75], "k--", linewidth=2)
plt.plot([4.95, 4.95], [0, 1.75], "k:", linewidth=2)
plt.plot([4.85, 4.85], [1.75, 3], "k:", linewidth=2)
plt.text(1.40, 1.0, "Depth=0", fontsize=15)
plt.text(3.2, 1.80, "Depth=1", fontsize=13)
plt.text(4.05, 0.5, "(Depth=2)", fontsize=11)

plt.show()
```

결정트리의 경계(3)

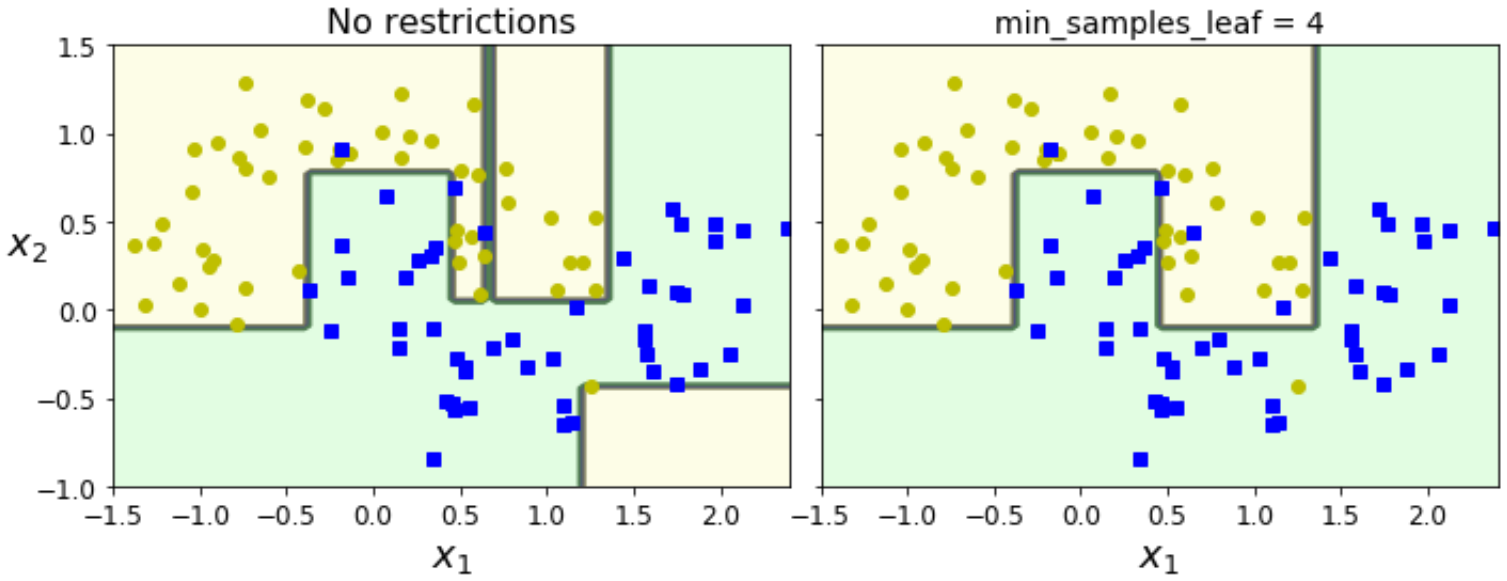


ref : Hands-On machine learning-O'REILLY

- Depth 0 : 결정경계($Petal\ length\ =\ 2.45$)
- 왼쪽 영역 : 순수 노드(Setosa만 있음)이므로 나눌 수 없다
- 오른쪽 영역 : 순수 노드가 아니므로 나눌 수 있다
- Depth 1 : 결정경계($Petal\ width\ =\ 1.75$)
- max_depth를 3으로 하면 Depth 2의 두 노드가 결정경계로 추가(작은 점선)

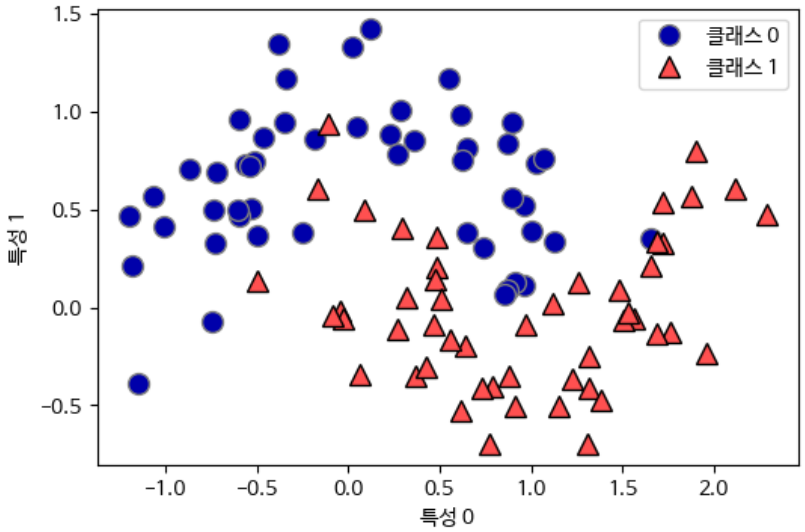
규제 매개변수

- DecisionTreeClassifier 의 매개변수
  - min_samples_split : 분할되기 위해 노드가 가져야 하는 최소 샘플 수
  - min_samples_leaf : 리프노드가 가지고 있어야 할 최소 샘플 수
  - min_weight_fraction_leaf : min_samples_leaf 와 같지만 가중치가 부여된 전체 샘플 수에서의 비율
  - max_leaf_nodes : 리프노드의 최대 수
  - max_features : 각 노드에서 분할에 사용할 특성의 최대 수
- min_ / max_ 로 시작하는 매개변수를 증가/감소시키면 모델에 규제가 커진다

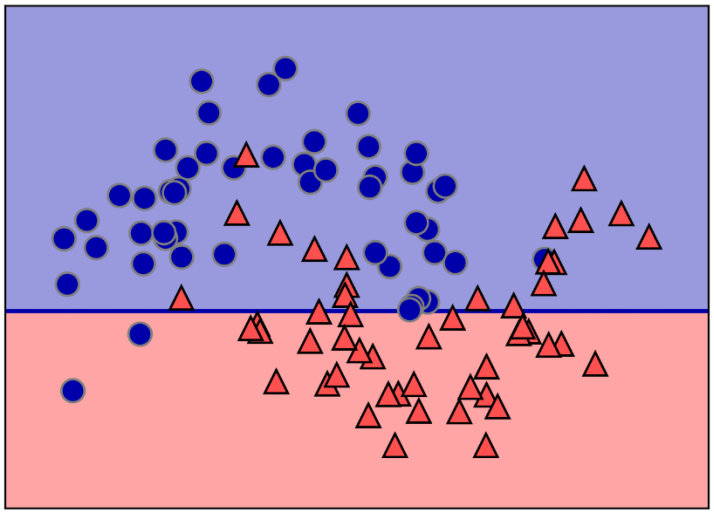


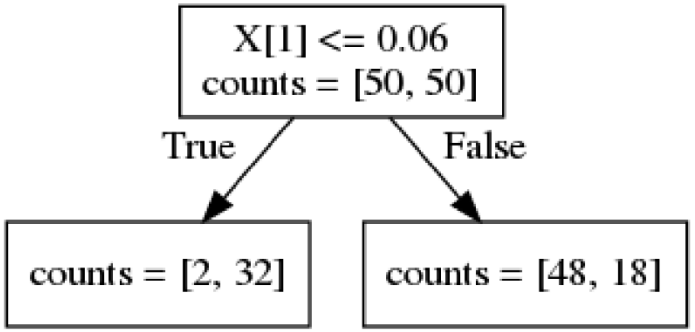
결정트리 만들기(1)


```
import mglearn as mglearn
mglearn.plots.plot_tree_progressive()
```



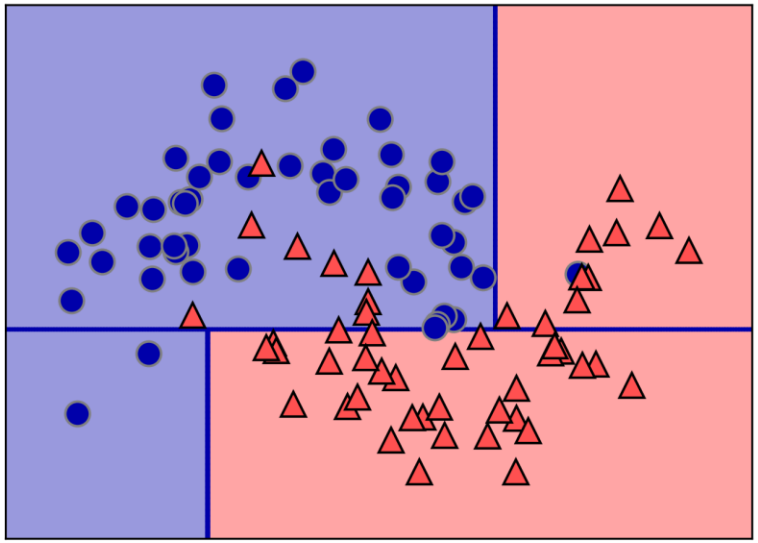
깊이 = 1

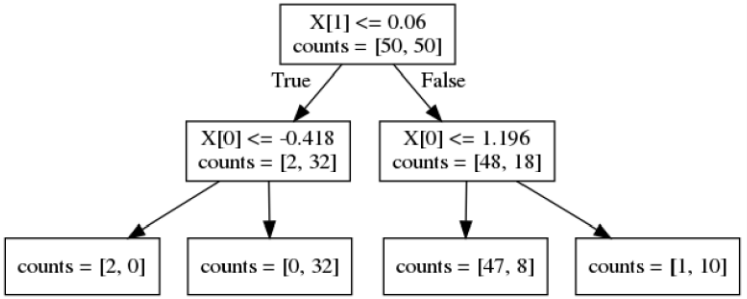




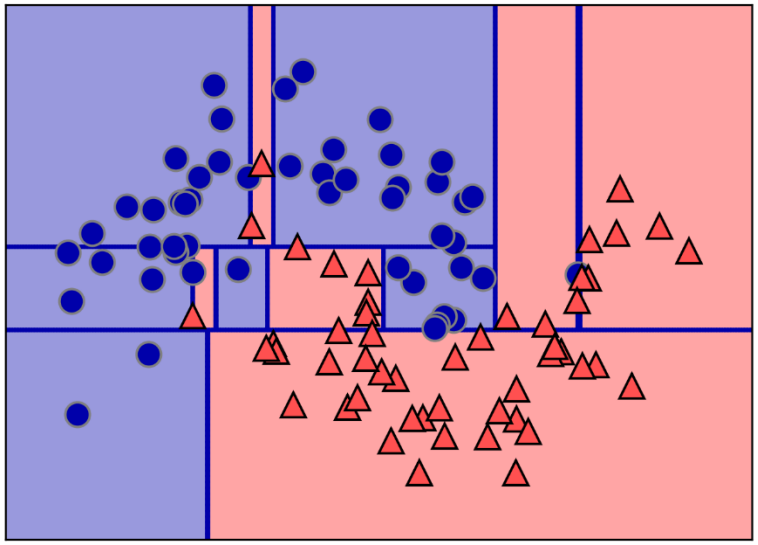
결정트리 만들기(2)

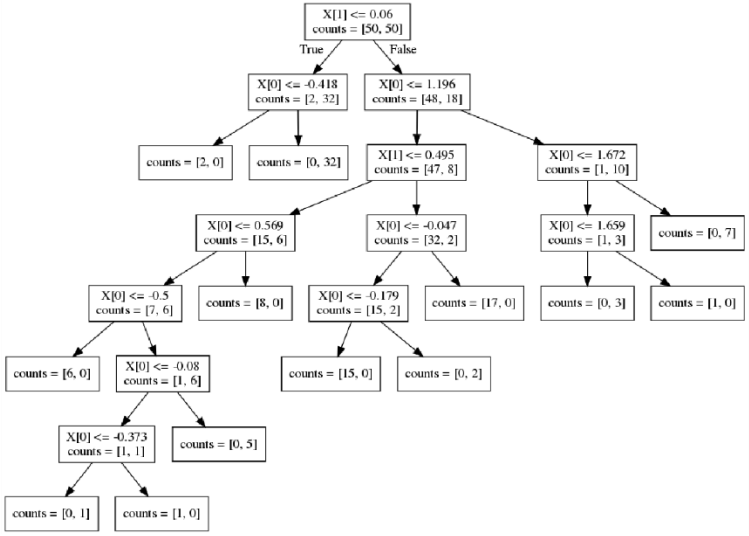
깊이 = 2





깊이 = 9





가지치기(pruning)

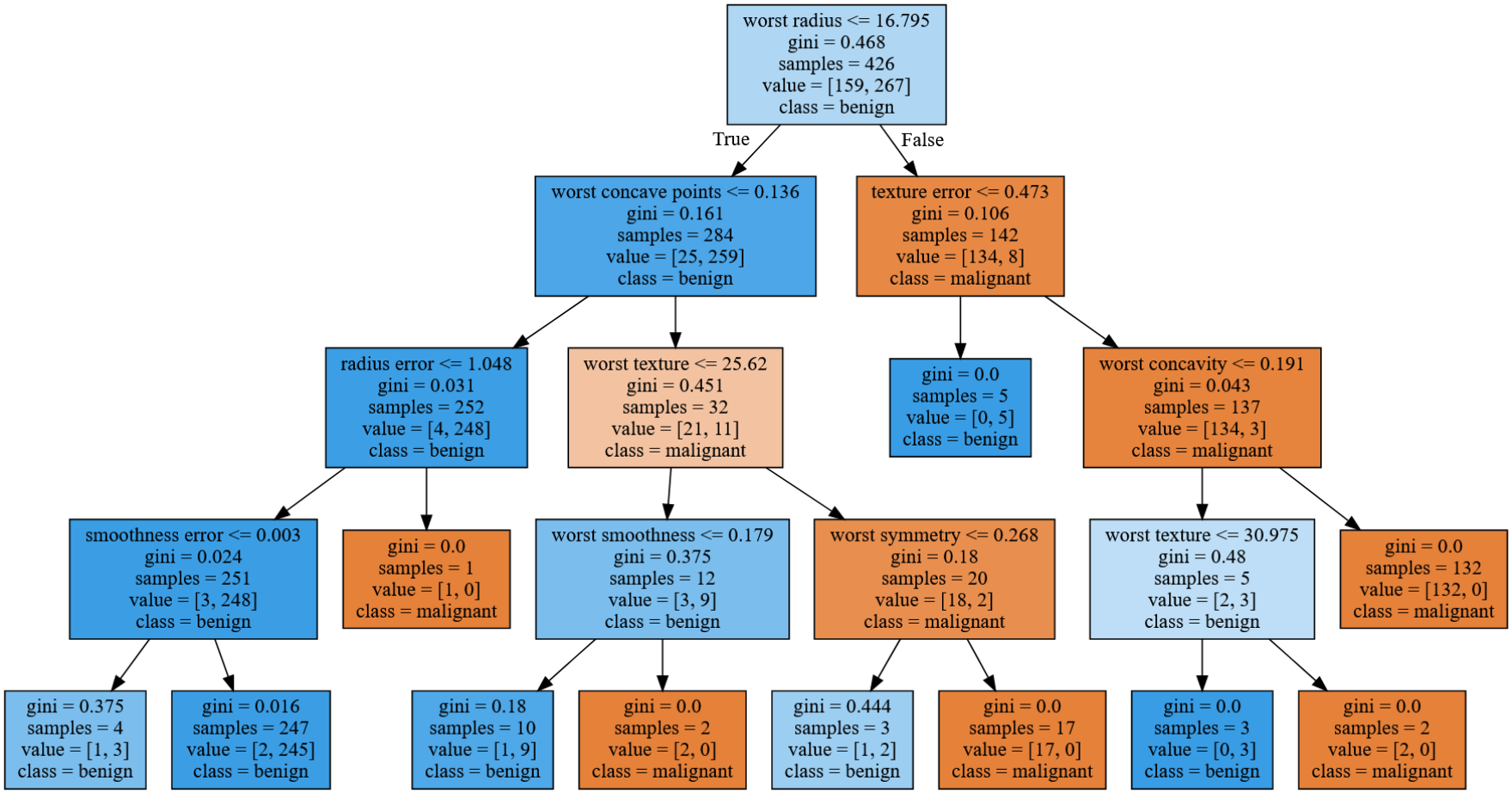

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

# 유방암 데이터셋
cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(
    cancer.data, cancer.target, stratify=cancer.target, random_state=42)
tree = DecisionTreeClassifier(random_state=0)
tree.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(tree.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(X_test, y_test)))
```

Accuracy on training set: 1.000
Accuracy on test set: 0.937

```
tree = DecisionTreeClassifier(max_depth=4, random_state=0)
tree.fit(X_train, y_train)

print("Accuracy on training set: {:.3f}".format(tree.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(X_test, y_test)))
```

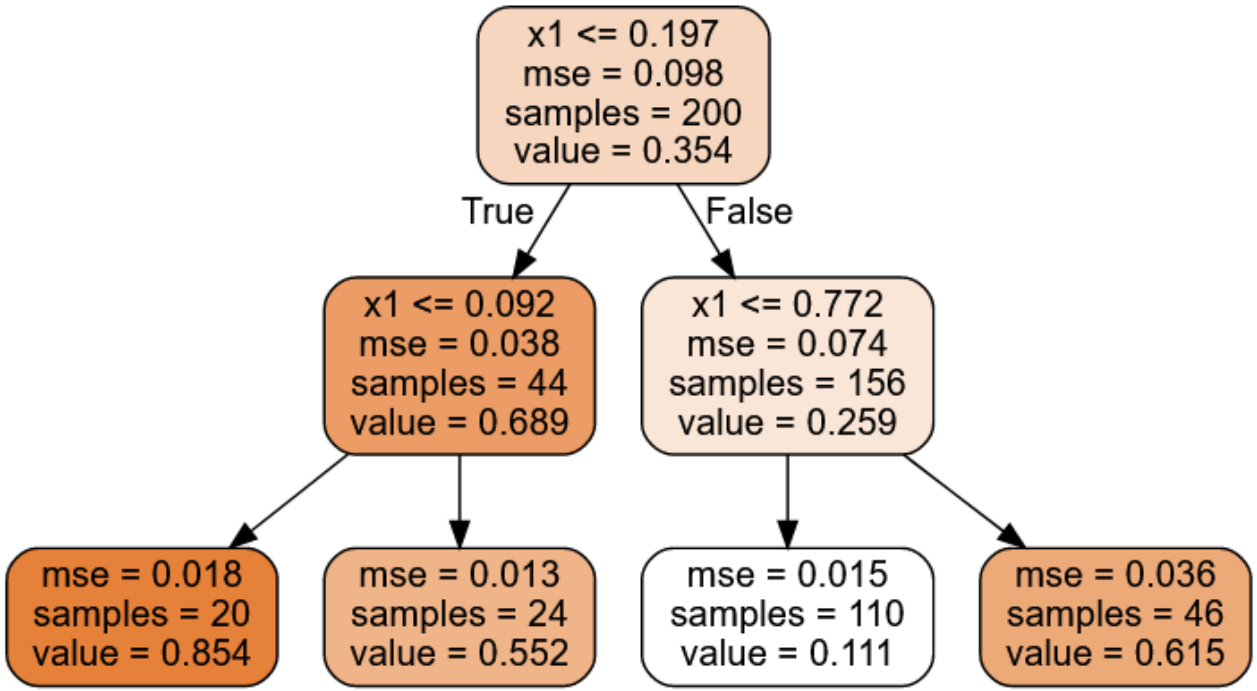
Accuracy on training set: 0.988
Accuracy on test set: 0.951

```
from sklearn.tree import export_graphviz
import graphviz

export_graphviz(tree, out_file="tree.dot", class_names=["malignant", "benign"],
                feature_names=cancer.feature_names, impurity="gini", filled=True)

with open("tree.dot") as f:
    dot_graph = f.read()
display(graphviz.Source(dot_graph))
```


worst radius <= 16.795
gini = 0.468
samples = 426
value = [159, 267]
class = benign
True
False
worst concave points <= 0.136
gini = 0.161
samples = 284
value = [25, 259]
class = benign
texture error <= 0.473
gini = 0.106
samples = 142
value = [134, 8]
class = malignant
radius error <= 1.048
gini = 0.031
samples = 252
value = [4, 248]
class = benign
worst texture <= 25.62
gini = 0.451
samples = 32
value = [21, 11]
class = malignant
gini = 0.0
samples = 5
value = [0, 5]
class = benign
worst concavity <= 0.191
gini = 0.043
samples = 137
value = [134, 3]
class = malignant
smoothness error <= 0.003
gini = 0.024
samples = 251
value = [3, 248]
class = benign
gini = 0.0
samples = 1
value = [1, 0]
class = malignant
worst smoothness <= 0.179
gini = 0.375
samples = 12
value = [3, 9]
class = benign
worst symmetry <= 0.268
gini = 0.18
samples = 20
value = [18, 2]
class = malignant
worst texture <= 30.975
gini = 0.48
samples = 5
value = [2, 3]
class = benign
gini = 0.0
samples = 132
value = [132, 0]
class = malignant
gini = 0.375
samples = 4
value = [1, 3]
class = benign
gini = 0.016
samples = 247
value = [2, 245]
class = benign
gini = 0.18
samples = 10
value = [1, 9]
class = benign
gini = 0.0
samples = 2
value = [2, 0]
class = malignant
gini = 0.444
samples = 3
value = [1, 2]
class = benign
gini = 0.0
samples = 17
value = [17, 0]
class = malignant
gini = 0.0
samples = 3
value = [0, 3]
class = benign
gini = 0.0
samples = 2
value = [2, 0]
class = malignant

회귀(1)

- 분류트리와 차이점 : 각 노드에서 클래스를 예측하는 대신 어떤 값을 예측한다는 점
- (예) $x_1 = 0.6$인 샘플의 타깃 값을 예측한다고 가정
  - 루트 노드부터 시작해서 트리를 순회하면 결국 value=0.111인 리프 노드에 도달
  - 이 리프 노드에 있는 110개 훈련 샘플의 평균 타깃 값이 예측 값이 된다.
  - 이 예측 값을 사용해 110개 샘플에 대한 평균제곱오차(MSE)를 계산하면 0.015가 된다



ref : Hands-On machine learning-O'REILLY

회귀(2)

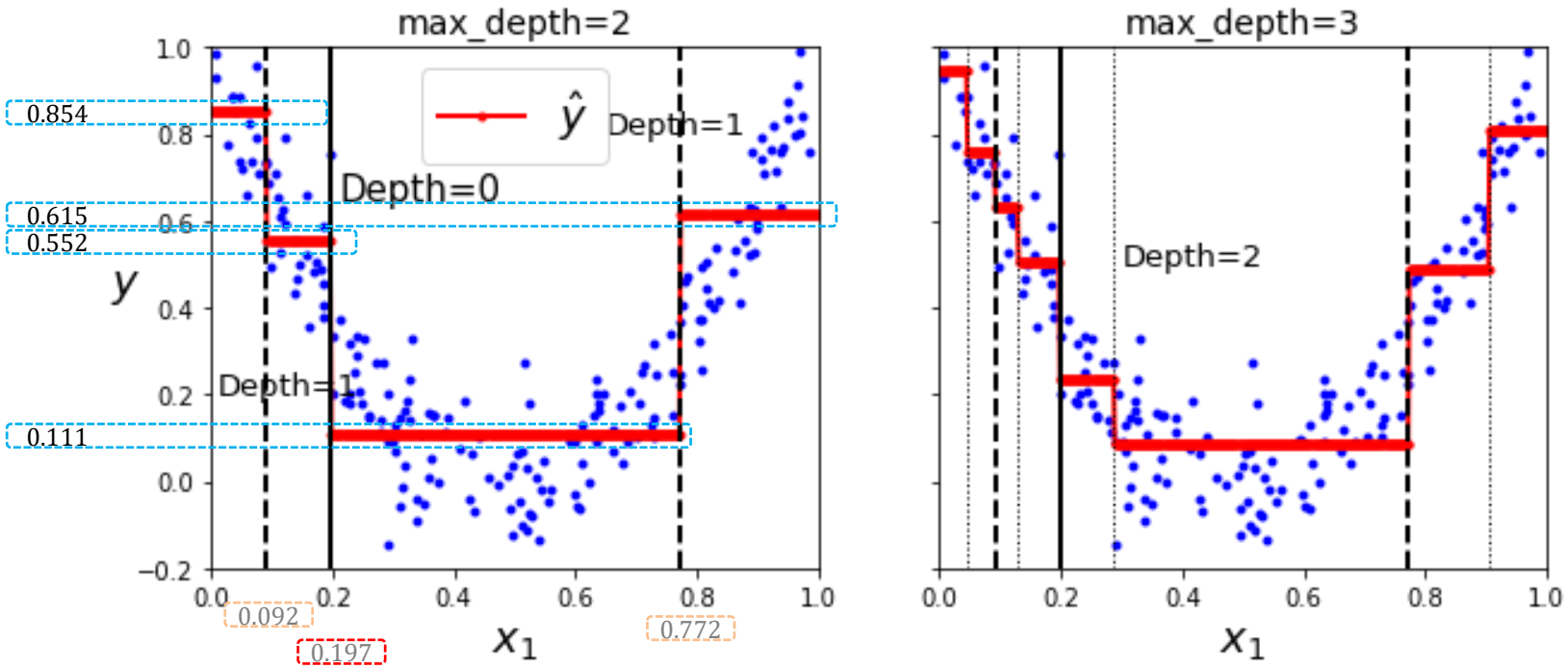
- 회귀 트리 생성 : 사이킷런의 DecisionTreeRegressor 이용

```
from sklearn.tree import DecisionTreeRegressor

# 2차식으로 만든 데이터셋 + 잡음
np.random.seed(42)
m = 200
X = np.random.rand(m, 1)
y = 4 * (X - 0.5) ** 2
y = y + np.random.randn(m, 1) / 10

tree_reg1 = DecisionTreeRegressor(random_state=42, max_depth=2)
tree_reg2 = DecisionTreeRegressor(random_state=42, max_depth=3)
tree_reg1.fit(X, y)
tree_reg2.fit(X, y)

def plot_regression_predictions(tree_reg, X, y, axes=[0, 1, -
0.2, 1], ylabel="$y$"):
    x1 = np.linspace(axes[0], axes[1], 500).reshape(-1, 1)
    y_pred = tree_reg.predict(x1)
    plt.axis(axes)
    plt.xlabel("$x_1$", fontsize=18)
    if ylabel:
        plt.ylabel(ylabel, fontsize=18, rotation=0)
    plt.plot(X, y, "b.")
    plt.plot(x1, y_pred, "r.-", linewidth=2, label=r"$\hat{y}$")
```

회귀(3)


```
fig, axes = plt.subplots(ncols=2, figsize=(10, 4), sharey=True)
plt.sca(axes[0])
plot_regression_predictions(tree_reg1, X, y)
for split, style in ((0.1973, "k-"), (0.0917, "k--"), (0.7718, "k--")):
    plt.plot([split, split], [-0.2, 1], style, linewidth=2)
plt.text(0.21, 0.65, "Depth=0", fontsize=15)
plt.text(0.01, 0.2, "Depth=1", fontsize=13)
plt.text(0.65, 0.8, "Depth=1", fontsize=13)
plt.legend(loc="upper center", fontsize=18)
plt.title("max_depth=2", fontsize=14)

plt.sca(axes[1])
plot_regression_predictions(tree_reg2, X, y, ylabel=None)
for split, style in ((0.1973, "k-"), (0.0917, "k--"), (0.7718, "k--")):
    plt.plot([split, split], [-0.2, 1], style, linewidth=2)
for split in (0.0458, 0.1298, 0.2873, 0.9040):
    plt.plot([split, split], [-0.2, 1], "k:", linewidth=1)
plt.text(0.3, 0.5, "Depth=2", fontsize=13)
plt.title("max_depth=3", fontsize=14)

plt.show()
```

회귀(4)

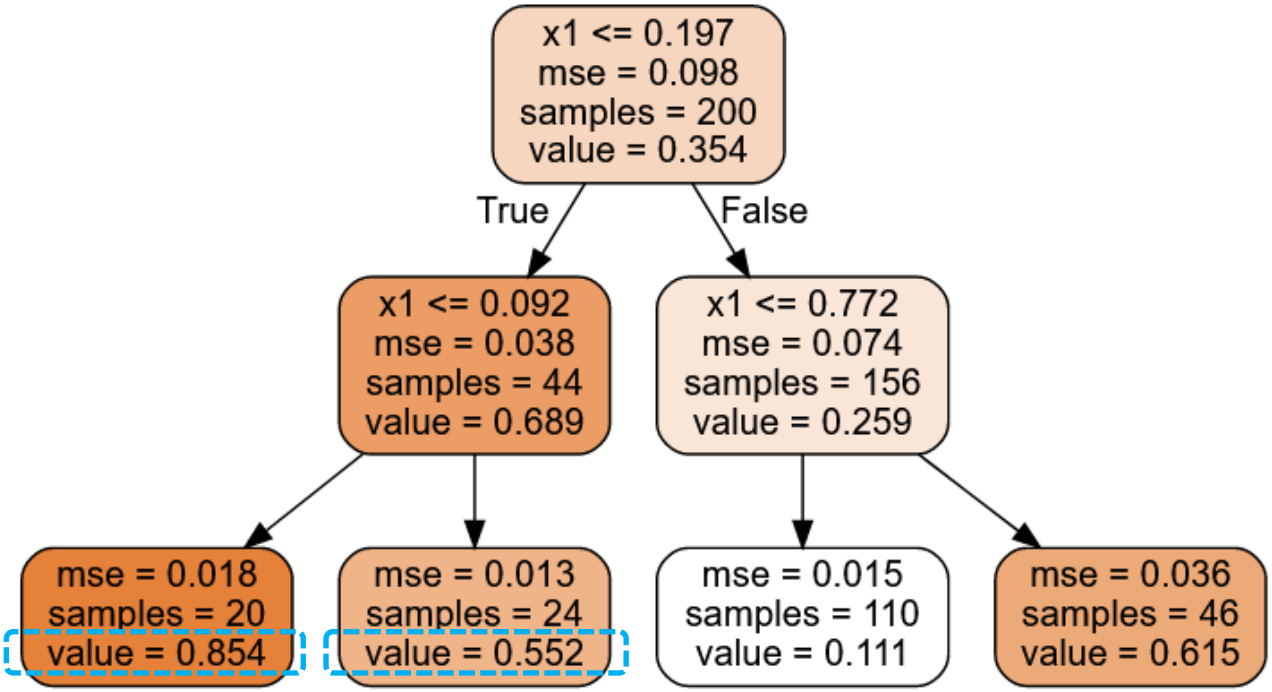


- 각 영역의 예측 값은 항상 그 영역에 있는 타깃 값의 평균
- 알고리즘은 예측 값과 가능한 한 많은 샘플이 가까이 있도록 영역을 분할한다

ref : Hands-On machine learning-O'REILLY

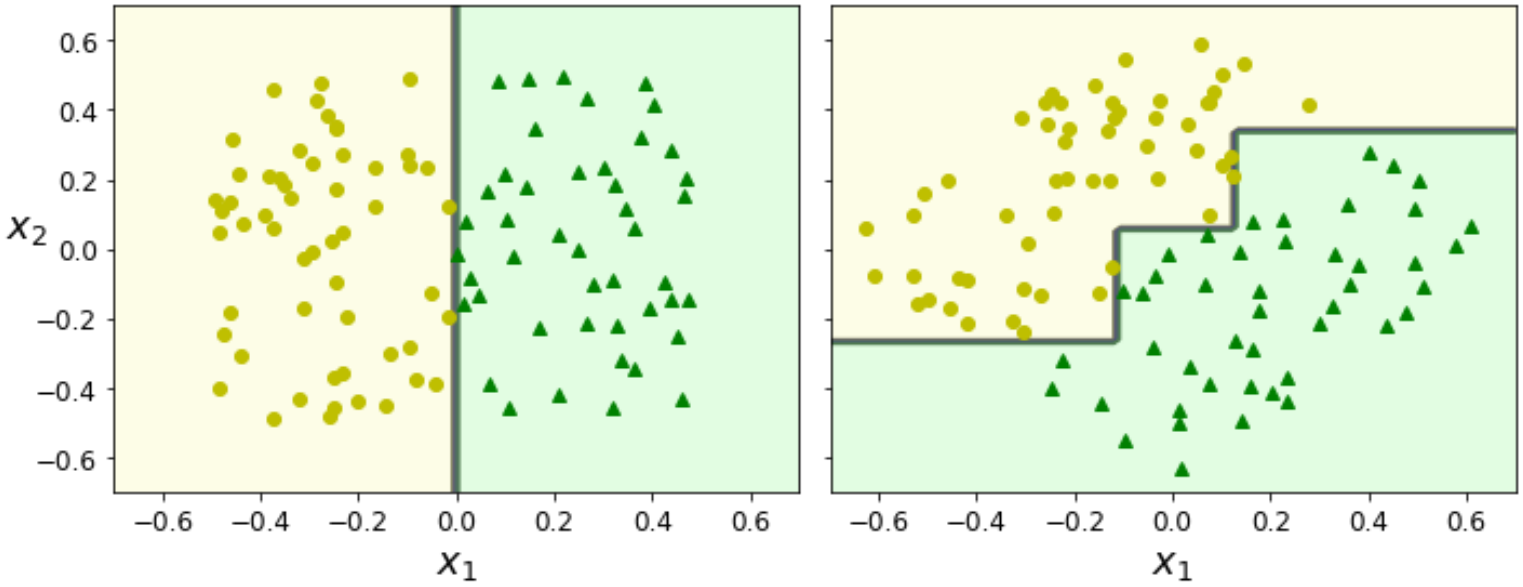
회귀(5)


```
export_graphviz(
        tree_reg1,
        out_file=os.path.join(IMAGES_PATH, "regression_tree.dot"),
        feature_names=["x1"],
        rounded=True,
        filled=True
    )
Source.from_file(os.path.join(IMAGES_PATH, "regression_tree.dot"))
```



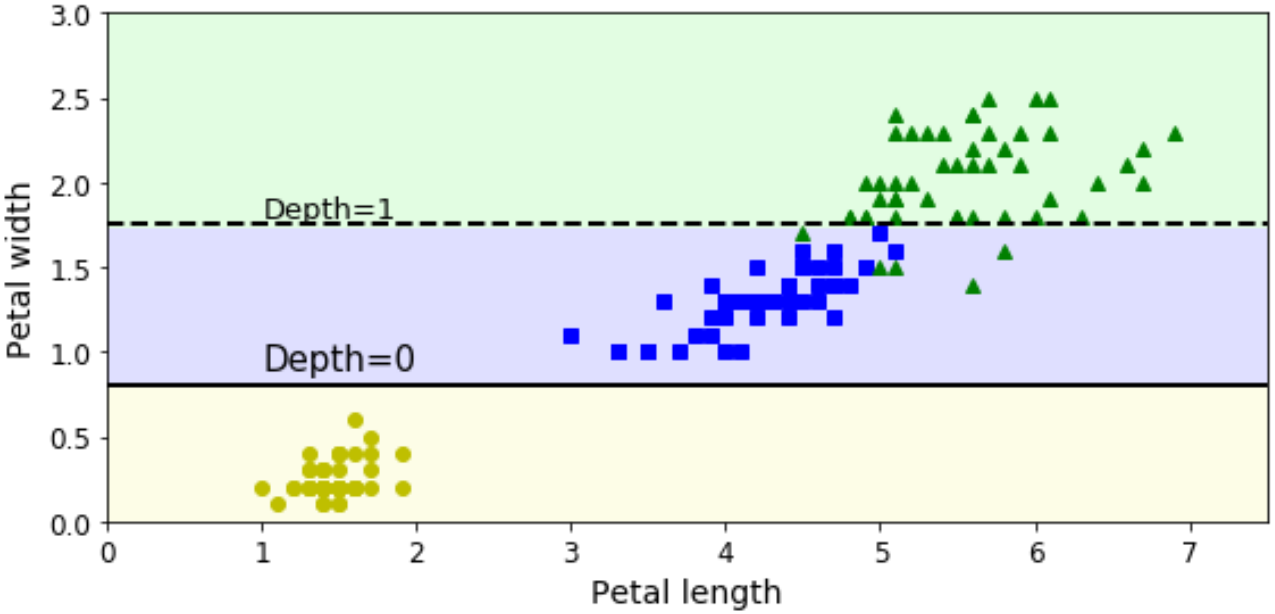
불안정성(1)

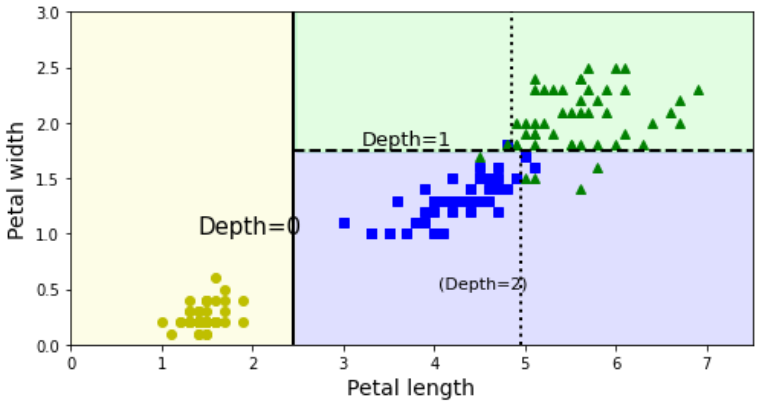
- 결정트리는 계단 모양의 결정 경계를 만든다.
  - 훈련세트의 회전에 민감
- 오른쪽 결정트리 : 데이터셋을 45도 회전한 결정트리
  - 구불구불한 불필요한 형태
  - 일반화되기 쉽지 않음 : 훈련 데이터를 더 좋은 방향으로 회전시키는 PCA기법 사용



불안정성(2)

- 훈련데이터에 있는 작은 변화에도 매우 민감
  - 훈련데이터세트에서 가장 넓은 versicolor(꽃잎길이4.9cm & 너비 1.8cm)를 제거하고 결정 트리를 훈련시의 모델





원래 훈련셋트 결과

결정트리의 불안정성은
랜덤 포레스트에서 극복할 수 있다