6장 의사결정나무

- 의사결정나무 학습과 시각화
- (의사결정나무) 예측하기
- (의사결정나무) 클래스 확률 추정
- (의사결정나무) CART 훈련 알고리즘
- (의사결정나무) 계산 복잡도
- 지니 불순도 대 엔트로피
- (의사결정나무) 규제 매개변수
- (의사결정나무) 회귀
- (의사결정나무) 불안정성

6.1 의사결정나무 학습과 시각화

- 붓꽃 데이터를 이용하여 사이킷런의 의사결정나무 모델을 학습시키고 학습결과 시각화 하기
- 붗꼿을 꽃잎의 길이와 너비 기준으로 분류하는 학습
- 사이킷런의 `DecisionTreeClassifier` 모델 활용
- 의사결정나무 방식의 최대 장점: 데이터 전처리 불필요

사이킷런의 의사결정나무 학습

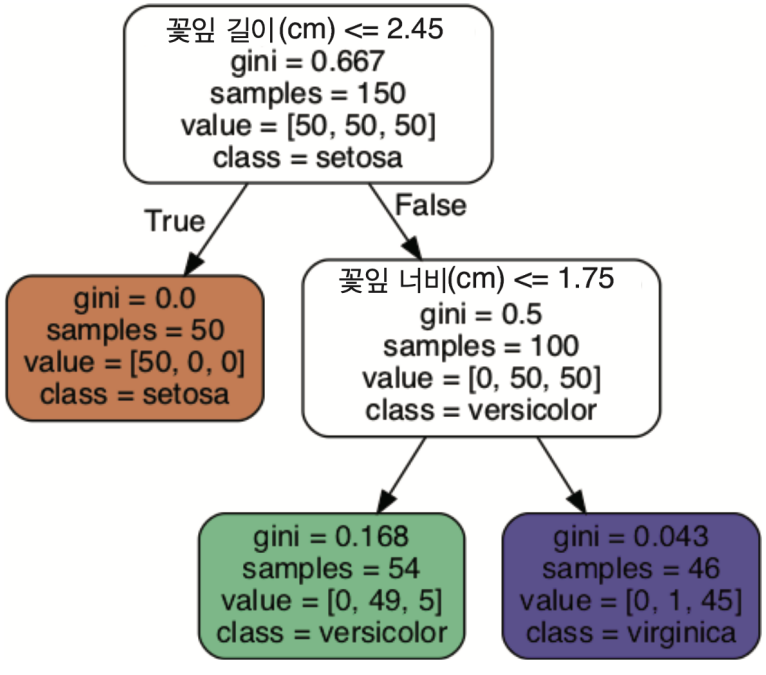
```python
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier

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)
```

사용된 옵션

- `max_depth`: 의사결정나무의 최대 깊이 지정.
    - 여기서는 2 사용. 즉, 연속된 가지치기가 최대 2번까기 가능.

의사결정나무 학습결과 시각화

- 사이킷런의 `export_graphviz()` 함수 활용
- 훈련 결과를 그래프 정보로 변환한 후 `iris_tree.dot` 파일에 저장
- pdf 또는 png 파일로 변환 가능


꽃잎 길이(cm) <= 2.45
gini = 0.667
samples = 150
value = [50, 50, 50]
class = setosa
True
False
gini = 0.0
samples = 50
value = [50, 0, 0]
class = setosa
꽃잎 너비(cm) <= 1.75
gini = 0.5
samples = 100
value = [0, 50, 50]
class = versicolor
gini = 0.168
samples = 54
value = [0, 49, 5]
class = versicolor
gini = 0.043
samples = 46
value = [0, 1, 45]
class = virginica

나무 구성 요소

- 마디(node): 가지치기가 시작되는 지점
- 나무뿌리(root node): 맨 상단에 위치한 마디
- 나뭇잎(leaf node): 더 이상의 가기치기가 발생하지 않는 마디

의사결정나무 마디 속성

- `gini`: 해당 마디의 불순도 측정값
    - 모든 샘플이 동일 클래스에 속하면 불순도가 0이 됨. 즉, `gini=0`.
    - 의사결정나무 학습 과정에 사용되는 알고리즘의 비용함수에 사용됨. (아래에서 자세히 설명)

- `samples`: 해당 마디 결정에 사용된 샘플 수

- `value`: 해당 마디 결정에 사용된 샘플을 클래스 별로 구분한 결과
    - 훈련 샘플의 레이블 정보를 이용하여 분류

- `class` : 각 클래스별 비율을 계산하여 가장 높은 비율에 해당하는 클래스 선정
    - 동일한 비율이면 낮은 인덱스 선정
    - 예를 들어, 깊이 2의 왼편 마디의 클래스별 비율은 아래와 같음

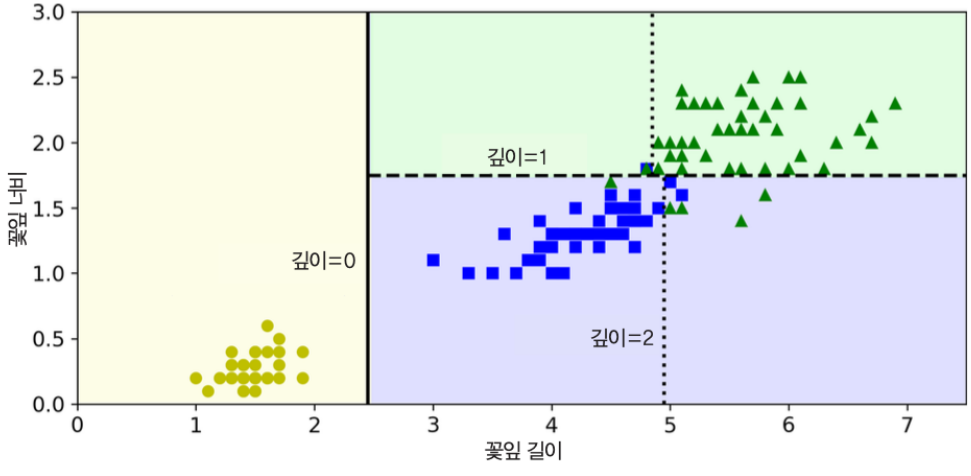
$$p_0 = 0/54, \quad p_1 = 49/54, \quad p_2 = 5/54$$

6.2 (의사결정나무) 예측하기

- 데이터가 주어지면 나무뿌리에서 시작

- 꽃잎 길이: 2.45cm 이하
    - 왼편으로 이동. setosa로 판정

- 꽃잎 길이: 2.45cm 초과
    - 오른편으로 이동
    - 꽃잎 너비: 1.75cm 이하
        - 왼편으로 이동. Iris-Versicolor로 판정
    - 꽃잎 너비: 1.75cm 초과
        - 오른편으로 이동. Iris-Virginica로 판정



- 점선: `max_depth=3`으로 지정할 경우를 보여줌.
- `max_depth` 값을 크게 잡으면 과대적합 위험도 커짐.

6.3 (의사결정나무) 클래스 확률 추정

- 계산된 클래스별 비율을 이용하여 새로운 샘플에 대한 예측 실행

- 예제: 꽃잎 길이와 너비가 각각 5cm, 1.5cm인 붓꽃에 대한 클래스 별 예측확률:

$$[0/54,\ 49/54,\ 5/54]$$

- 판정: 가장 높은 확률을 가진 Iris-Versicolor!

- 동일한 마디에 속한 샘플에 대한 예측값은 언제나 동일

6.4 (의사결정나무) CART 훈련 알고리즘

지니 불순도 계산

- 불순도: 마디에 포함된 `gini` 속성
    - K는 클래스 수이고, p_k는 클래스 k에 속한 샘플의 비율

$$G = 1 - \sum_{k=0}^{K-1} p_k^2$$

- 예를 들어, 깊이 2의 왼편 마디의 지니 불순도는 0.168

$$G = 1 - (0/54)^2 - (49/54)^2 - (5/54)^2 = 0.168$$

분류와 회귀 나무(CART, classification and regression tree) 알고리즘

- 아래 비용함수를 최소화 하는 특성 k와 해당 특성의 임곗값 t_k을 결정
    - 탐욕적 알고리즘(greedy algorithm) 활용

$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

- G_{left}(G_{right}): 지정된 특성 k와 특성 임곗값 t_k로 구분된 왼편(오른편) 부분집합의 지니 불순도
    - 즉, 각 마디의 지니 불순도를 낮추는 방향으로 학습

- m: 해당 마디의 전체 샘플 수

- m_{left}(m_{right}): 지정된 특성 k와 특성 임곗값 t_k로 구분된 왼편(오른편) 부분집합의 크기

- $J(k, t_k)$가 작을수록 불순도가 낮은 두 개의 부분집합으로 분할됨

- 탐욕적 알고리즘은 해당 마디에 포함된 샘플을 지니 불순도가 가장 낮은, 즉, 가장 순수한(pure) 두 개의 부분집합으로 분할

- 이렇게 나누는 과정은 `max_depth` 깊이에 다다르거나 불순도를 줄이는 분할을 더 이상 찾을 수 없을 때, 또는 다른 규제의 한계에 다다를 때까지 반복

6.5 (의사결정나무) 계산 복잡도

최적 의사결정나무 찾기

- 최적의 의사결정나무를 찾는 문제는 NP-완전(NP-complete)임.
- 이런 문제의 시간 복잡도는 $O(\exp(m))$
- 매우 작은 훈련 세트에 대해서도 제대로 적용하기 어려움

예측 시간 복잡도

- 학습된 의사결정나무가 예측에 필요한 시간: $O(\log m)$
    - 훈련 샘플 수 m에만 의존하며 매우 빠름
    - 특성 수와 무관: 각 마디에서 하나의 특성만 분류기준으로 사용되기 때문

학습 시간 복잡도

훈련 샘플이 크기순으로 정렬된 경우

- 각 마디에서 분류하는 데 걸리는 시간: $O(n \cdot m \cdot \log(m))$
- 의사결정나무를 완성하는 데 걸리는 시간: $O(n \cdot m^2 \cdot \log(m))$
- 규제가 있는 경우 좀 더 빨라짐.

훈련 샘플을 정렬하는 데 걸리는 시간

- `DecisionTreeClassifier`의 `presort=True` 옵션 설정
    - 훈련 세트를 미리 퀵정렬 시킨 후 학습 시작
- 훈련 세트가 크면 이 방식은 속도가 늦어짐
    - 퀵정렬 자체의 복잡도: $O(m \log m)$

6.6 지니 불순도 대 엔트로피

- `DecisionTreeClassifier`의 `criterion="entropy"` 옵션 설정:
    - gini 불순도 대신에 엔트로피 불순도 사용

- 특정 마디의 엔트로피(H) 계산

$$H = -\sum_{\substack{k=0 \\ p_k \neq 0}}^{K-1} p_k \log(p_k)$$

- 두 불순도의 차이는 크지 않으며, 비슷한 의사결정나무를 생성

- 엔트로피 불순도 특징:
    - 특정 k에 대해 만약 p_k 0에 가까운 경우
    - $\log(p_k)$: 음의 무한대로 수렴
    - 엔트로피 증가
    - 비용함수 $J(k, t_k)$ 증가
    - 그런 조합은 피하게 됨
    - 따라서 마디를 보다 균형 잡힘 두 개의 부분집합으로 분할하는 방향으로 유도

- 하지만 지니 불순도가 좀 더 계산이 빠르기에, 기본값으로 사용

6.7 (의사결정나무) 규제 매개변수

비매개변수 모델 대 매개변수 모델

비매개변수 모델(nonparametric model)

- 훈련 시작 전에 파라미터 수가 결정되지 않는 모델
- 예제: 의사결정나무. 어떤 모델일지 미리 지정하지 않음.
    - 마디를 분할할 수 있는 자유도(degree of freedom) 제함 없음
- 과대적합 위험 높음

매개변수 모델(parametric model)

- 미리 정의된 모델 파라미터 사용
- 예제: 선형 모델
- 과대적합 위험도 줄어듦.
- 과소적합 위험도 커딤.

사이킷런 `DecisionTreeClassifier` 규제하기

- `max_depth`: 의사결정나무의 최대 높이 제한
- `min_samples_split`: 마디를 분할하기 위해 필요한 최소 샘플 수
- `min_samples_leaf`:나뭇잎에 포함되어야 하는 최소 샘플 수
- `min_weight_fraction_leaf`:
    - 샘플 별로 가중치가 설정된 경우: 가중치의 전체 합에서 해당 나뭇잎에 포함된 샘플의 가중치의 합이 차지하는 비율
    - 샘플 별로 가중치가 없는 경우: `min_samples_leaf`와 동일한 역할 수행

- `max_leaf_nodes`: 허용된 나뭇잎의 최대 개수

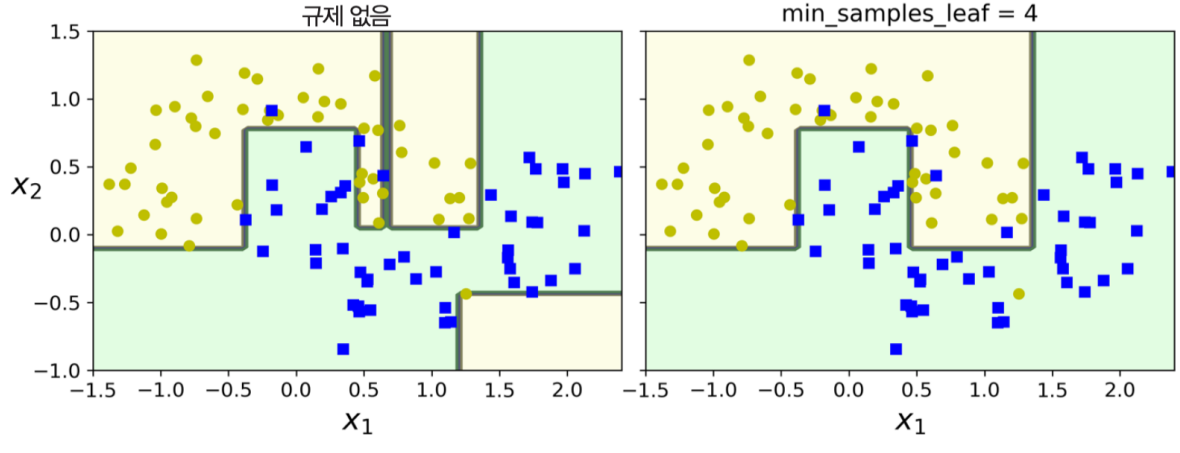
- `max_features`: 각 마디에서 분할 평가에 사용될 수 있는 최대 특성 수

- 규제를 높이는 방법
    - `min_` 접두사 사용 규제: 값을 키울 것
    - `max_` 접두사 사용 규제: 값을 감소시킬 것

예제: 사이킷런 `DecisionTreeClassifier` 규제 사용

moons 데이터셋에 대한 의사결정나무 모델 학습

- 왼편: 규제 전혀 없음
    - 보다 정교함
    - 과대적합됨
- 오른편: `min_samples_leaf=4`
    - 일반화 성능 보다 좋음



사전 가지치기 대 사후 가지치기

- 사전 가지치기: 사이킷런의 `DecisionTreeClassifier` 처럼 학습 과정에 사용되는 규제에 따라 분할을 제한하는 것

- 사후 가지치기: 우선 제한 없이 의사결정나무를 훈련 시킨 뒤에 통계적 가설검정을 이용하여 별로 의미 없는 마디를 잘라내는 기법
    - 사이킷런은 사후 가지치기를 지원하지 않음.

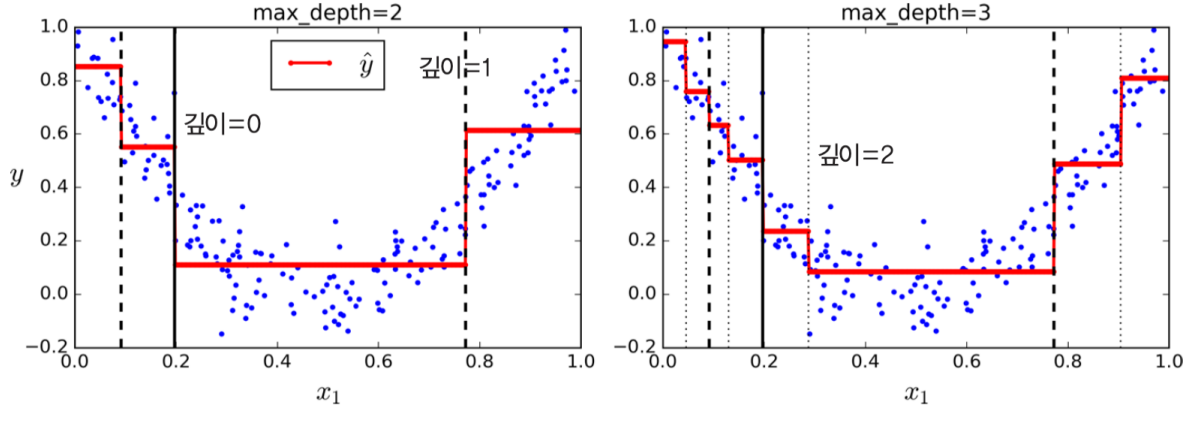
6.8 (의사결정나무) 회귀

- 의사결정나무 알고리즘 아이디어를 거의 그대로 이용하여 회귀 문제에 적용 가능

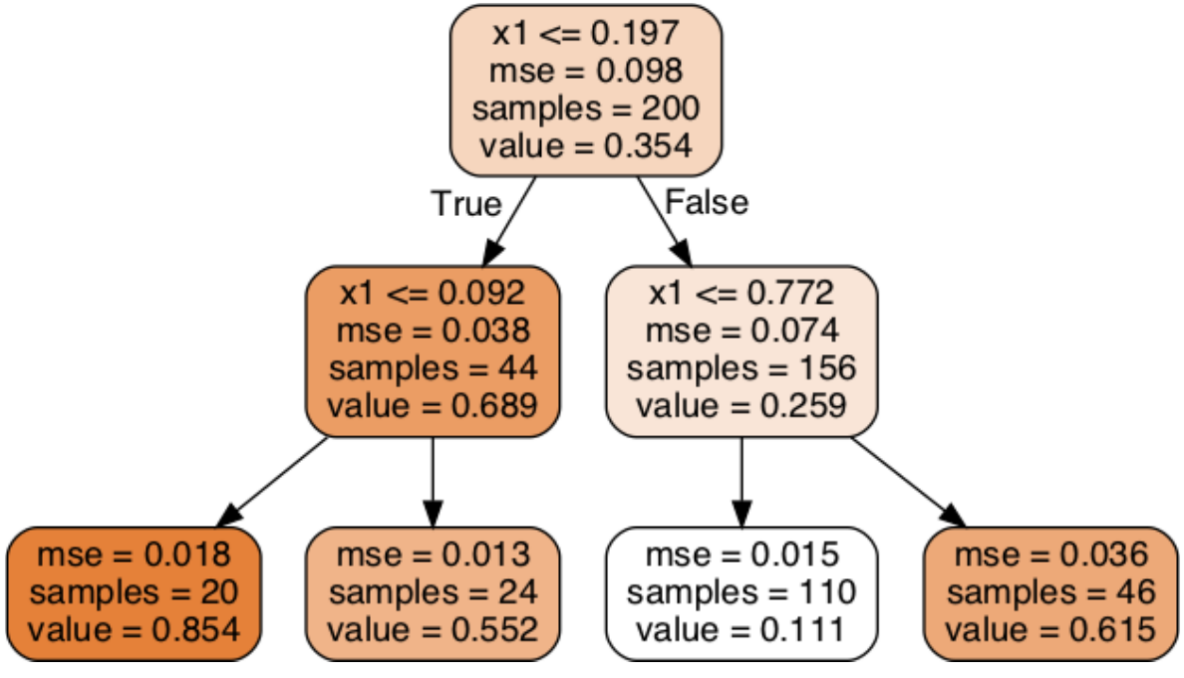
사이킷런의 DecisionTreeRegressor 예측기 활용

예제: 잡음이 포함된 2차 함수 형태의 데이터셋

- 왼편: `max_depth=2`
- 오른편: `max_depth=3`



- 왼편 그래프에 대한 회귀 그래프



- 각 마디에 포함된 속성
    - `samples`: 해당 마디에 속한 훈련 샘플 수
    - `value`: 해당 마디에 속한 훈련 샘플의 평균 타깃값
    - `mse`: 해당 마디에 속한 훈련 샘플의 평균제곱오차(mse)
        - 오차 기준은 `value` 사용.

회귀용 CART 알고리즘의 비용함수

- 아래 비용함수를 최소화 하는 특성 k와 해당 특성의 임곗값 t_k을 결정
    - 탐욕적 알고리즘(greedy algorithm) 활용
    - 각 마디의 평균제곱오차 MSE를 최소화하는 방향으로 학습

$$J(k, t_k) = \frac{m_{\text{left}}}{m} \text{MSE}_{\text{left}} + \frac{m_{\text{right}}}{m} \text{MSE}_{\text{right}}$$

- $\text{MSE}_{\text{left}}(\text{MSE}_{\text{right}})$: 지정된 특성 k와 특성 임곗값 t_k로 구분된 왼편(오른편) 부분집합의 평균 제곱오차(mse)

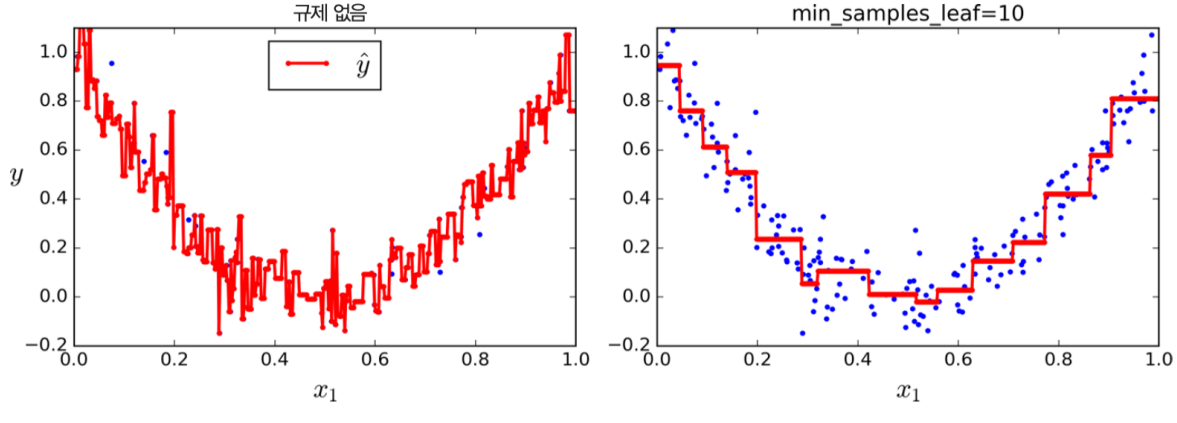
    - 해당 마디에 속한 샘플들의 평균 타깃값 기준
    - m_{node}: 해당 마디에 속하는 샘플 수
    - $y^{(i)}$: 샘플 i에 대한 레이블

$$\text{MSE}_{\text{node}} = \sum_{i \in \text{node}} (\hat{y}_{node} - y^{(i)})^2$$

$$\hat{y}_{\text{node}} = \frac{1}{m_{\text{node}}} \sum_{i \in \text{node}} y^{(i)}$$

예제

- 왼편: 규제가 없는 경우
    - 과대적합 발생
- 오른편: `min_samples_leaf=10`
    - 나름 괜찮음.



6.9 (의사결정나무) 불안정성

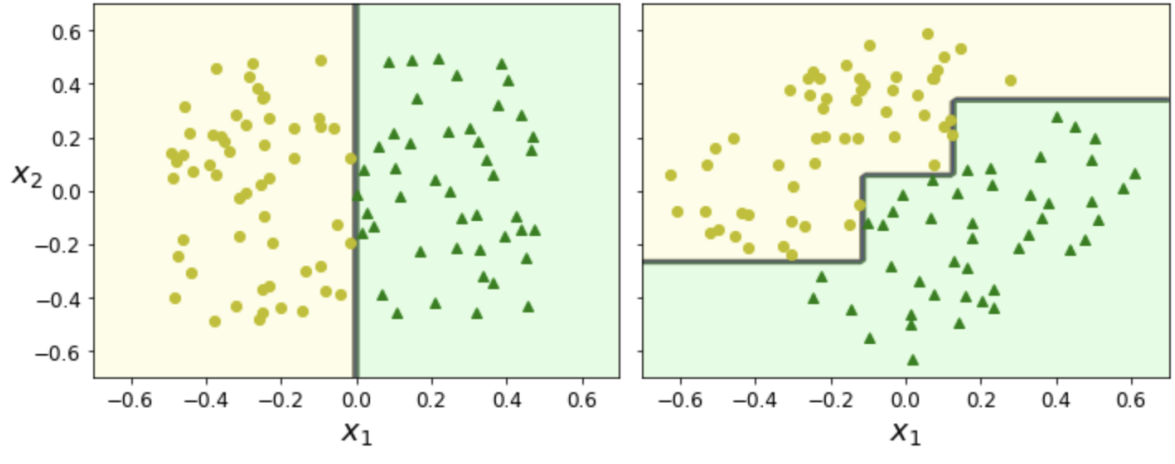
- 의사결정나무 알고리즘은 성능이 매우 우수하지만 기본적으로 주어진 훈련 세트에 민감하게 반응함.

단점 1: 훈련 세트의 회전에 민감

- 의사결정나무는 항상 축에 수직인 분할을 사용
- 따라서 조금만 회전을 가해도 결정 경계가 많이 달라짐

예제

- 오른편 그래프: 왼편 그래프를 45도 회전시킨 훈련 세트 학습
- PCA 기법 등을 사용하여 훈련 샘플 회전시킨 후 학습 가능. (8장 참조)



단점 2: 훈련 세트의 작은 변화에 민감

예제

- 붓꽃 데이터에서 하나의 샘플을 제거한 후 학습시킬 때 매우 다르게 학습할 수 있음.
- 왼편 그래프: 모든 샘플 대상 훈련
- 오른편 그래프: 가장 넓은 Iris-Versicolor 샘플 제거 후 훈련