머신러닝 이론 및 실습

5주차. 결정 트리

유선호

0 CONTEXT

1. 의사결정트리
2. 결정트리 훈련과 활용
3. CART 알고리즘
4. 지니 불순도 vs. 엔트로피
5. 결정트리 규제
6. 회귀 결정트리
7. 결정트리 단점

1 의사결정트리

의사결정트리(Decision Tree)

각 데이터들이 가진 속성들로부터 패턴을 예측 가능한 규칙들의 조합으로 나타내며, 이를 바탕으로 분류를 수행할 수 있도록 하는 지도학습 모델

의사결정나무는 분류와 회귀 모두 가능한 모델로, 범주형과 연속형 수치 모두 예측 가능

1 의사결정트리

의사결정트리의 구조

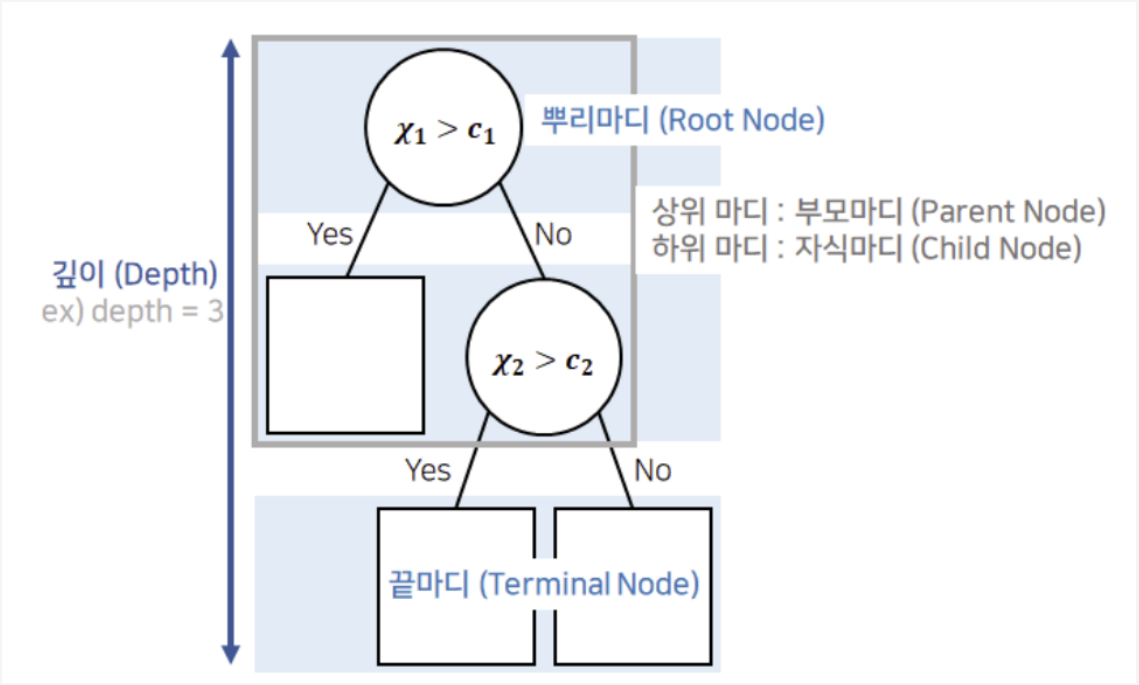
뿌리마디 (Root Node) : 나무가 시작되는 마디, 전체 데이터에 해당함

부모마디 (Parent Node) : 상위 마디

자식마디 (Child Node) : 하위 마디, 한 마디로부터 분리되어 나간 2개 이상의 마디들

끝마디 (Terminal Node / Leaf) : 최종 마디, 나무의 끝에 위치하는 마디

깊이 (Depth) : 가지를 이루고 있는 마디의 개수



WeTIE

1 의사결정트리

의사결정트리의 분류와 회귀 결과

의사결정트리의 분류와 회귀는 끝마디의 어떤 값을 반환하느냐의 차이점을 가진다.

분류의 경우, 새로운 데이터가 속하는 해당 끝마디에서 가장 빈도가 높은 범주로 새로운 데이터를 분류한다.

회귀의 경우, 해당 끝마디의 종속변수의 평균을 예측값을 반환한다. (예측값의 종류는 끝마디의 개수와 일치함)

1 의사결정트리

의사결정트리 수행 과정

1. 의사결정나무 형성

2. 가지치기 (Pruning)

3. 타당성 평가 & 해석 및 예측

1 의사결정트리

의사결정트리 수행 과정

1. 의사결정나무 형성

분석의 목적과 데이터 구조에 따라 적절한 분리기준과 정지규칙을 지정하여 의사결정나무 모형을 생성

분리기준 : 하나의 부모마디로부터 자식마디가 형성될 때, 어떤 입력변수로 어떻게 분리하는 것이 목표변수를 가장 잘 분류하는지를 파악하는 기준

ex. 순수도(purity), 불순도(impurity)

부모마디의 순수도에 비해, 자식마디들의 순수도가 증가하도록 (불순도가 감소하도록) 자식마디를 형성

정지규칙 : 더 이상 분리가 일어나지 않고 현재의 마디가 끝마디가 되도록 하는 규칙

깊이(Depth)나 끝마디의 개수가 몇 개가 될 때까지 나눌 것인지 정하여 규칙 설정

1 의사결정트리

의사결정트리 수행 과정

2. 가지치기 (Pruning)

분류오류(Classification Error)를 크게 할 위험이 높거나 부적절한 추론 규칙을 가지고 있는 가지를 제거하는 단계

의사결정나무의 분기 수가 증가하면, 새로운 데이터에 대한 오분류율이 감소하지만, 일정 수준 이상이 되면 오분류율이 증가하는 현상 발생

= 과적합 발생

적절한 가지치기를 수행하면 과적합을 막을 수 있음

3. 타당성 평가 & 해석 및 예측

1 의사결정트리

의사결정트리 장단점

장점.

(해석의 용이성) 사용자가 모형을 쉽게 이해 가능

(교호작용의 해석) 두 개 이상의 변수가 결합하여 목표변수에 어떻게 영향을 미치는지 알 수 있음

(비모수적 모형) 선형성, 정규성 혹은 등분산성 등의 가정이 필요하지 않음

단점.

(비연속성) 연속형 변수를 비연속적 값으로 취급하기 때문에 분리의 경계 부근에서 예측 오류가 클 가능성이 있음

(선형성/주효과의 결여) 각 변수의 목표변수에 대한 영향력을 알 수 없음

(비안정성) 훈련용 데이터에 의존하므로 새로운 데이터의 예측에서 불안정할 수 있음

2 결정트리 훈련과 활용

1. 결정트리 훈련

결정트리 방식은 일반적으로 데이터 전처리가 불필요

사이킷런의 DecisionTreeClassifier 모델 활용

붓꽃 데이터 활용, 꽃잎의 길이와 너비 기준으로 분류

max_depth = 2 : 결정트리의 최대 깊이 지정, 최대 2번의 데이터셋 분할 허용

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

iris = load_iris(as_frame=True)
X_iris = iris.data[["petal length (cm)", "petal width (cm)"]].values
y_iris = iris.target

tree_clf = DecisionTreeClassifier(max_depth=2, random_state=42)
tree_clf.fit(X_iris, y_iris)
```

WeTIE

2. 결정트리 시각화

사이킷런의 export_graphviz() 함수 활용

```
from sklearn.tree import export_graphviz

export_graphviz(
        tree_clf,
        out_file=str(IMAGES_PATH / "iris_tree.dot"),
        feature_names=["petal length (cm)", "petal width (cm)"],
        class_names=iris.target_names,
        rounded=True,
        filled=True
    )
```

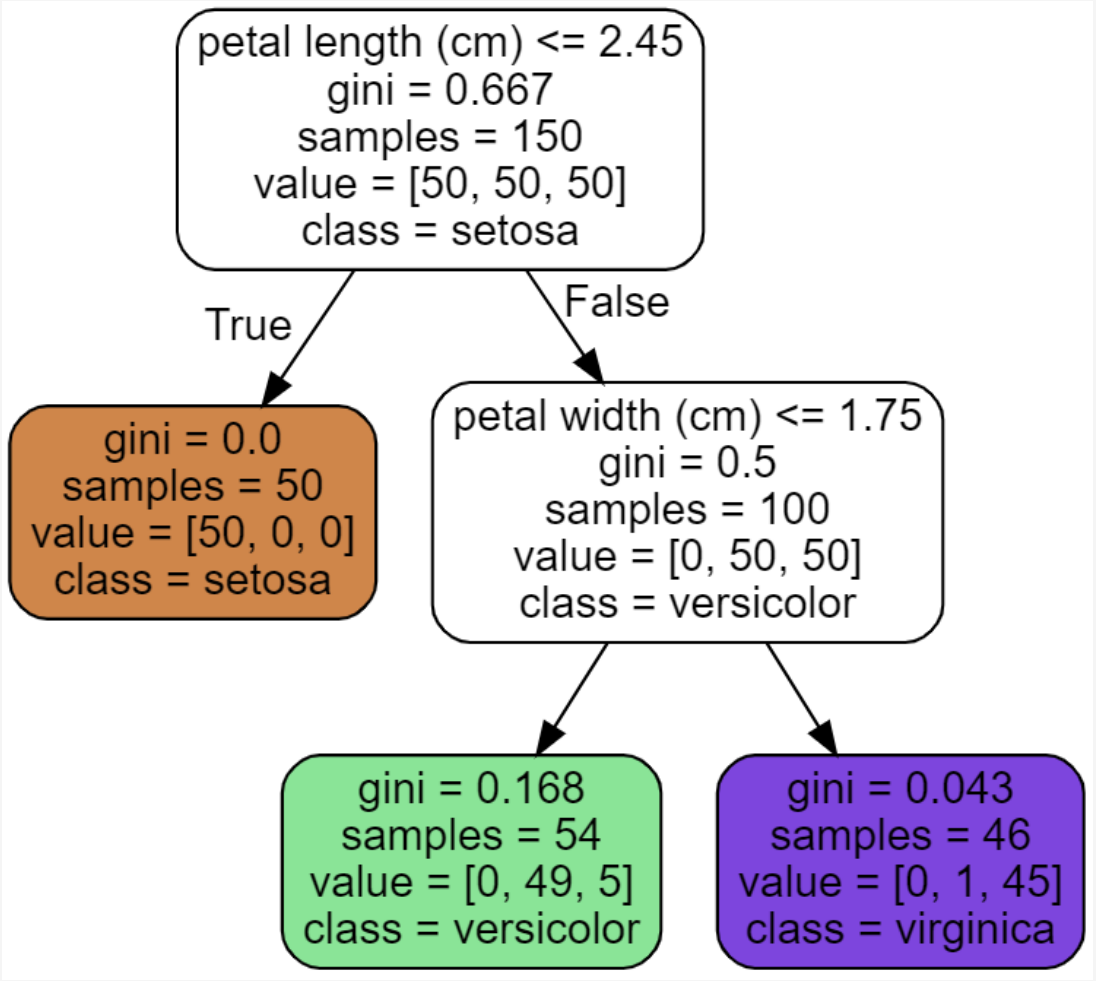
```
from graphviz import Source

Source.from_file(IMAGES_PATH / "iris_tree.dot")
```

결정트리 노드의 속성

gini : 노드의 지니 불순도, sample : 노드에 속하는 샘플 수

value : 각각의 범주에 속하면서 노드에 포함된 샘플 수

class : 가장 높은 비율을 차지하는 범주



2 결정트리 훈련과 활용

지니 불순도

G_i: i번째 노드의 지니 불순도

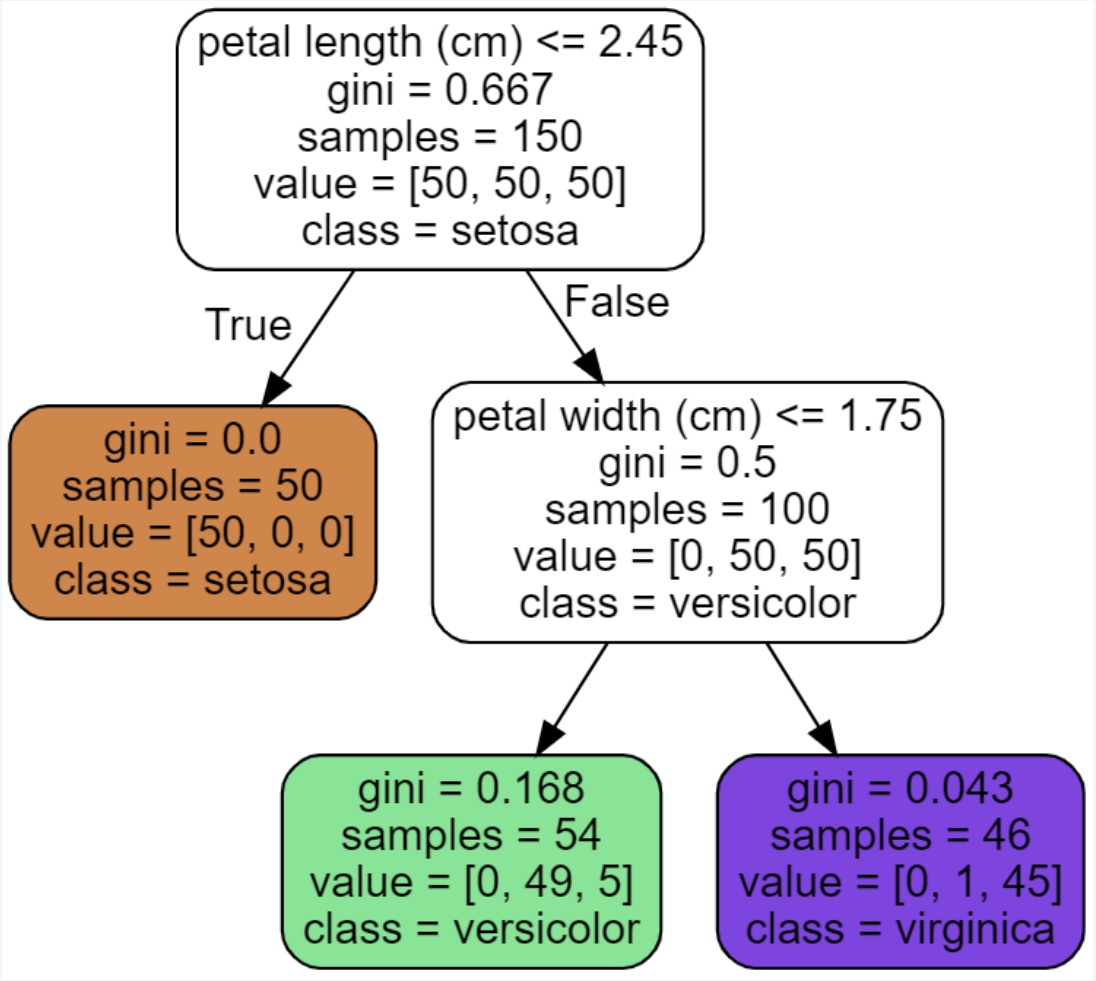
$$G_i = 1 - \sum_{k=0}^{K-1} (p_{i,k})^2$$

$p_{i,k}$는 i번째 노드에 있는 훈련 샘플 중 범주 k에 속한 샘플의 비율

K는 범주의 개수

깊이 2의 왼편 노드의 지니 불순도

$$G_4 = 1 - (0/54)^2 - (49/54)^2 - (5/54)^2 = 0.168$$



WeTIE

3. 범주 예측

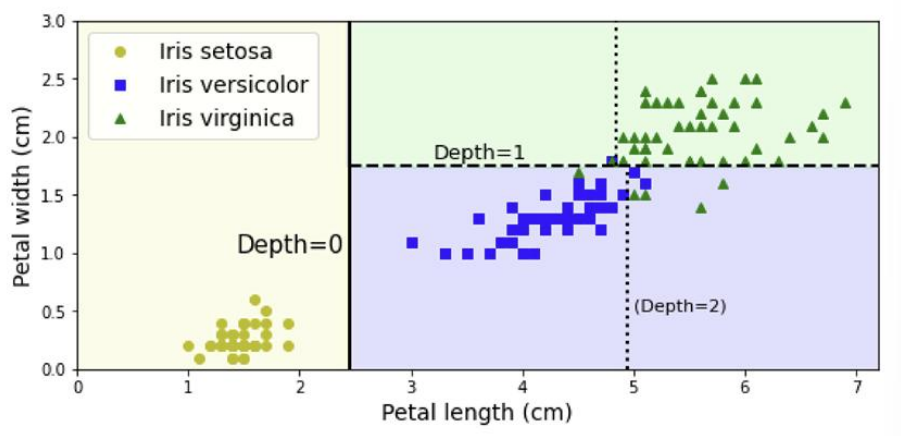
max_depth = 3 으로 지정해서 학습한 결정트리의 결정경계

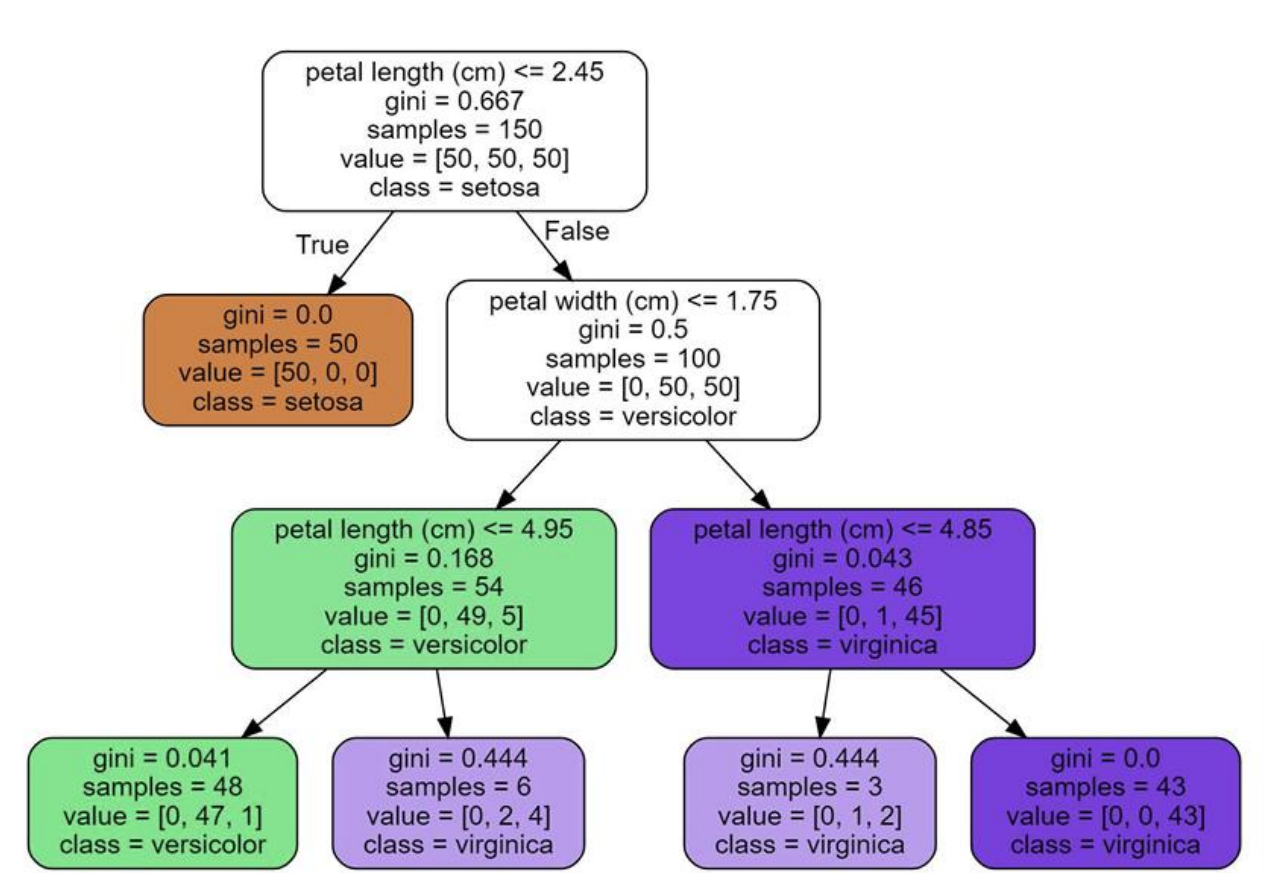
1차 분할 기준 : 꽃잎 길이 2.45cm. 트리 깊이 0에서 진행됨.

2차 분할 기준 : 꽃잎 너비 1.75cm. 트리 깊이 1에서 진행됨.

3차 분할 기준 : 꽃잎 길이 4.95cm. 트리 깊이 2에서 진행됨.

4차 분할 기준 : 꽃잎 길이 4.85cm. 트리 깊이 2에서 진행됨.





WeTIE

3. 범주 예측

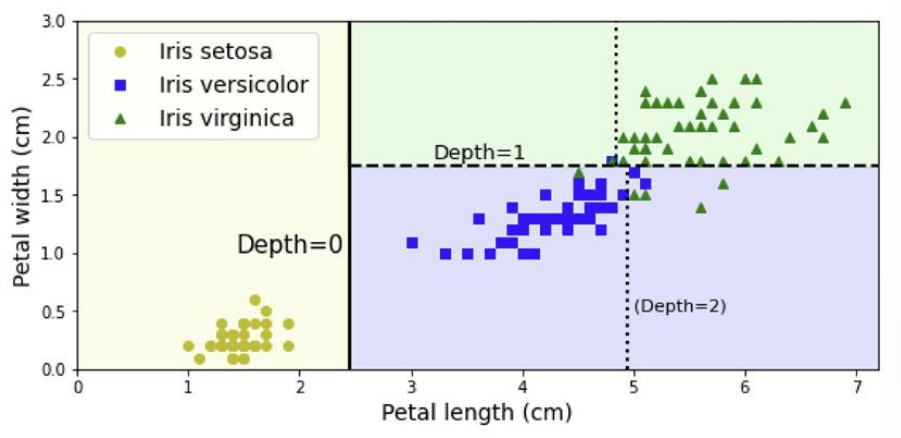
꽃잎 길이와 너비 : 각각 5cm, 1.5cm

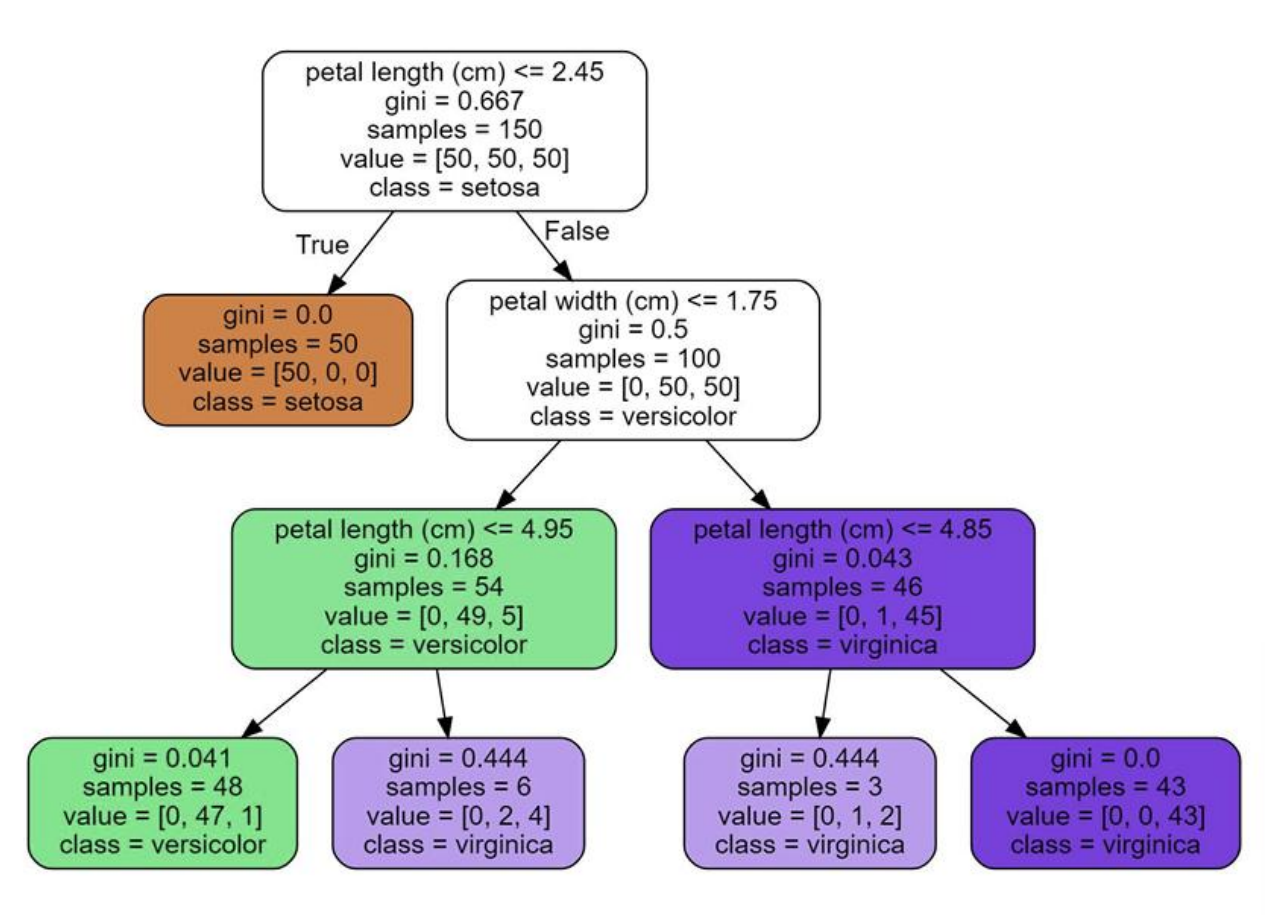
데이터가 주어지면 루트에서 시작

분할 1단계 : 꽃잎 길이가 2.45cm 이하가 아니기에 오른편으로 이동

분할 2단계 : 꽃잎 너비가 1.75cm 이하이기에 왼편으로 이동.

버시컬러로 판정





4. 범주에 속할 확률

주어진 샘플에 대해 예측된 노드에 속한 샘플들의 범주별 비율

max_depth = 2로 훈련된 tree_clf 모델은 예를 들어 꽃잎 길이와 너비가 각각 5cm, 1.5cm인 붓꽃을 깊이 2의 왼편 잎 노드인 G3에 포함

G3 노드에 포함된 샘플들의 범주별 비율 : (세토사, 버시컬러, 버지니카)

(0/54, 49/54, 5/54) = (0, 0.907, 0.093)

버시컬러에 속할 확률이 90.7%로 가장 높기에 해당 샘플은 버시컬러로 예측

Predict_proba() vs. Predict()

`predict_proba()` 메서드: 지정된 샘플의 범주별 추정 확률을 계산

```
>>> tree_clf.predict_proba([[5, 1.5]]).round(3)
array([[0.   , 0.907, 0.093]])
```

`predict()` 메서드: 품종 범주를 예측하며, 가장 높은 추정 확률을 갖는 품종으로 지정.

```
>>> tree_clf.predict_proba([[5, 1.5]]).round(3)
array([1])
```

3 CART 알고리즘

CART 알고리즘

Classification and Regression Tree

각 노드에서 아래 비용함수를 최소화하는 특성 k와 특성의 임계값 t_k를 결정해서 사용함

- m, m_{left}, m_{right}: 각각 부모와 양쪽 자식 노드에 속한 샘플 수
- G_{left}, G_{right}: 두 자식 노드의 지니 불순도

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

비용함수가 작을수록 불순도가 낮은 두 개의 부분집합으로 분할됨

분할 과정 반복 : max_depth 등 규제의 한계에 다다르거나 더 이상 불순도를 줄이는 분할이 불가능할 때까지 진행

4 지니 불순도 vs. 엔트로피

지니 불순도 vs. 엔트로피

DecisionTreeClassifier의 criterion = “entropy” 옵션 설정:

지니 불순도 대신에 샘플들의 무질서 정도를 측정하는 엔트로피 사용

지니 불순도를 사용할 때와 비교해서 큰 차이가 나지 않음

엔트로피 방식이 노드를 보다 균형 잡힌 두 개의 자식 노드로 분할함

i 번 인덱스 노드의 엔트로피

$$H_i = -\sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^{K} p_{i,k} \log_2(p_{i,k})$$

엔트로피 방식이 보다 균형 잡힌 이진탐색트리를 만드는 이유

특정 k 에 대해 $p_{i,k}$ 가 0 에 매우 가까우면 $-\log(p_{i,k})$ 가 매우 커진다. 이는 엔트로피 증가로 이어지기 때문에 결국 비용함수 $J(k, t_k)$ 가 증가한다. 따라서 $p_{i,k}$ 가 매우 작게되는 경우는 피하도록 학습하게 되고 결국 보다 균형 잡힌 두 개의 부분집합으로 분할하는 방향으로 유도된다.

두 방식의 큰 차이가 없고 지니 불순도 방식보다 빠르게 훈련되어 기본값으로 사용함

5 결정트리 규제

비파라미터 모델 vs. 파라미터 모델

결정트리 모델은 데이터에 대한 어떤 가정도 하지 않음

노드르 분할할 때 어떤 제한도 가해지지 않으며,

노드를 분할할 때마다 새로운 파라미터가 학습되기 때문에 학습되어야 하는 파라미터의 개수를 미리 알 수 없다.

이러한 모델을 비파라미터 모델이라 부른다.

비파라미터 모델의 자유도는 제한되지 않기에 과대적합될 가능성이 높다.

선형 모델과 같은 모델은 파라미터 수가 훈련 시작 전에 규정되기에 과대적합 가능성이 상대적으로 적어진다.

이러한 모델을 파라미터 모델이라 한다.

5 결정트리 규제

사이킷런 DecisionTreeClassifier 규제 하이퍼파라미터

규제 강화 방법

min_ 접두사 사용 규제 : 값을 키울 것

max_ 접두사 사용 규제 : 값을 감소시킬 것

하이퍼파라미터	기능
max_depth	결정트리의 높이 제한
min_samples_split	노드 분할해 필요한 최소 샘플 개수
min_samples_leaf	잎 노드에 포함된 최소 샘플 개수
max_leaf_nodes	최대 잎 노드 개수
max_features	분할에 사용되는 특성 개수

5 결정트리 규제

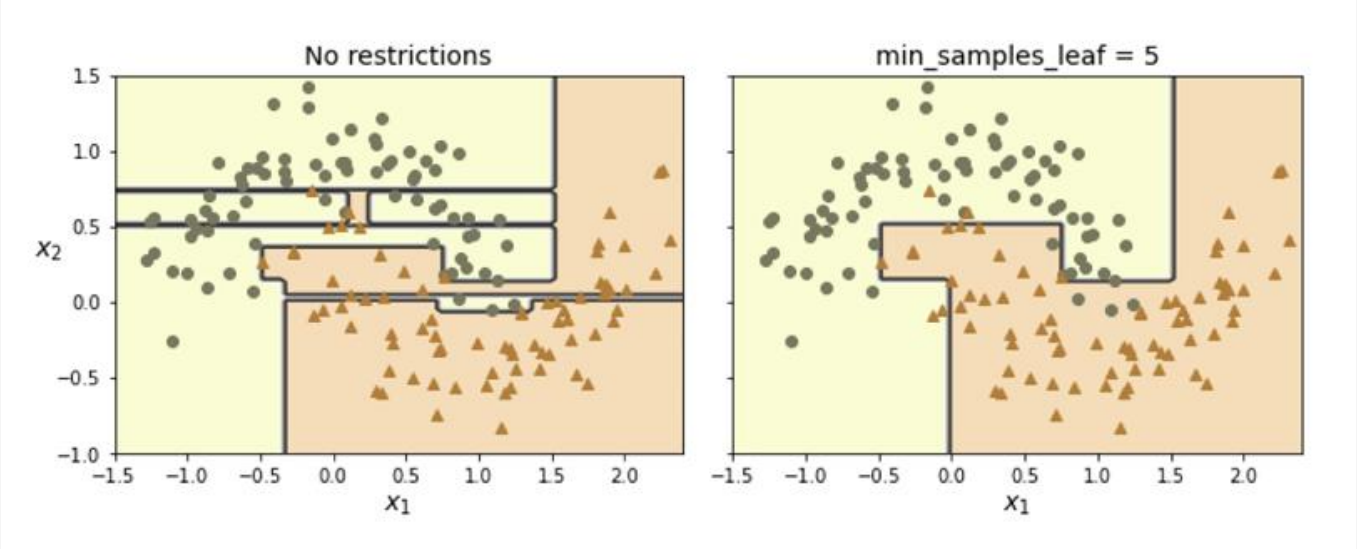
규제 적용 예제

예제 : 초승달 데이터셋에 대한 결정트리 모델 학습

왼쪽 : 규제 전형 없음, 보다 정교하며 과대적합됨

오른쪽 : min_samples_leaf = 5, 일반화 성능이 보다 좋음

```
from sklearn.datasets import make_moons

X_moons, y_moons = make_moons(n_samples=150, noise=0.2, random_state=42)

tree_clf1 = DecisionTreeClassifier(random_state=42)
tree_clf2 = DecisionTreeClassifier(min_samples_leaf=5, random_state=42)
tree_clf1.fit(X_moons, y_moons)
tree_clf2.fit(X_moons, y_moons)
```



WeTIE

6 회귀 결정트리

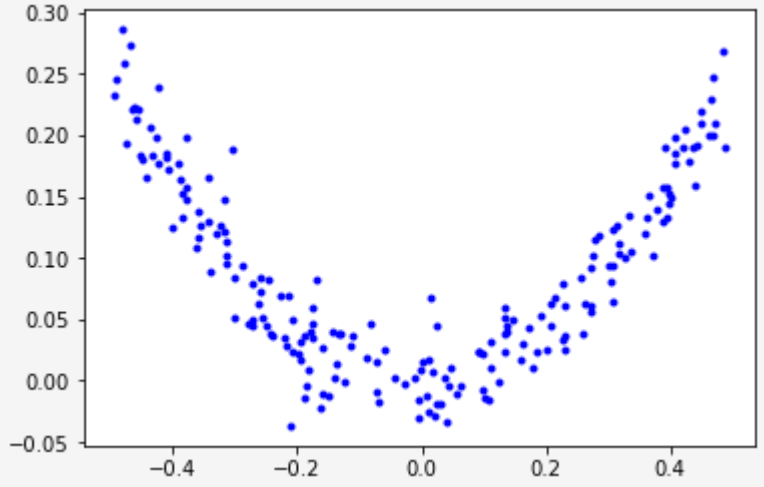
사이킷런의 DecisionTreeRegressor 예측기 활용

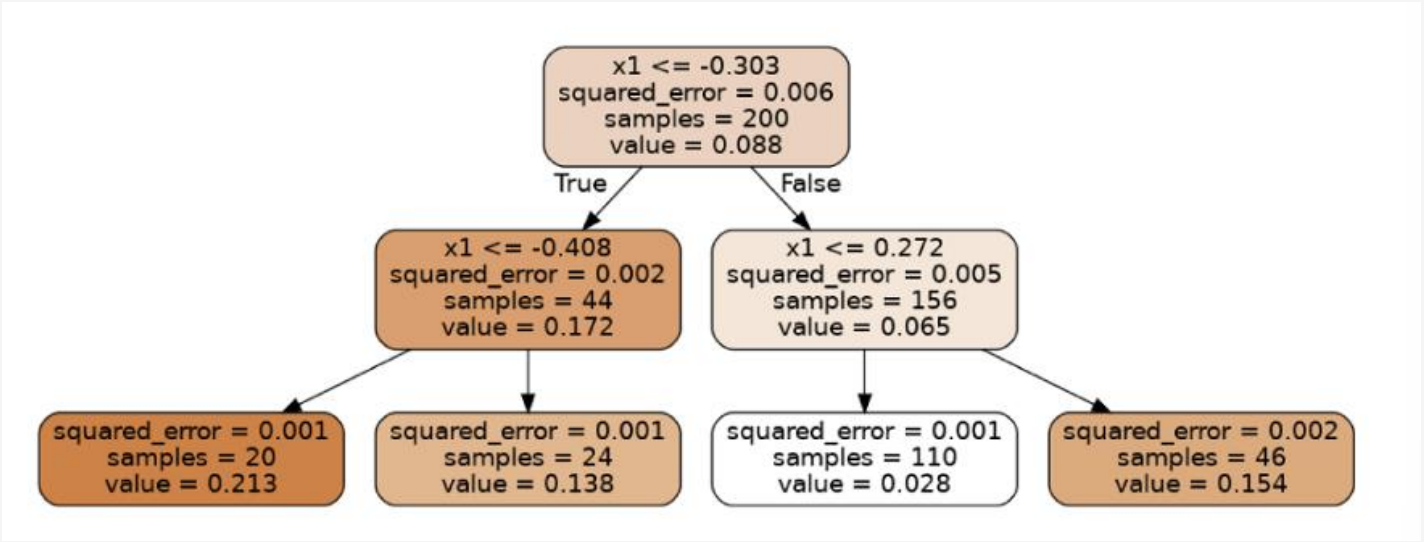
결정트리 알고리즘 아이디어를 그대로 이용하여 회귀 문제에 적용 가능

잡음이 포함된 2차 함수 형태의 데이터셋을 이용하여 결정트리 회귀 모델을 훈련시켜보자

```
tree_reg = DecisionTreeRegressor(max_depth=2, random_state=42)
tree_reg.fit(X, y)
```

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

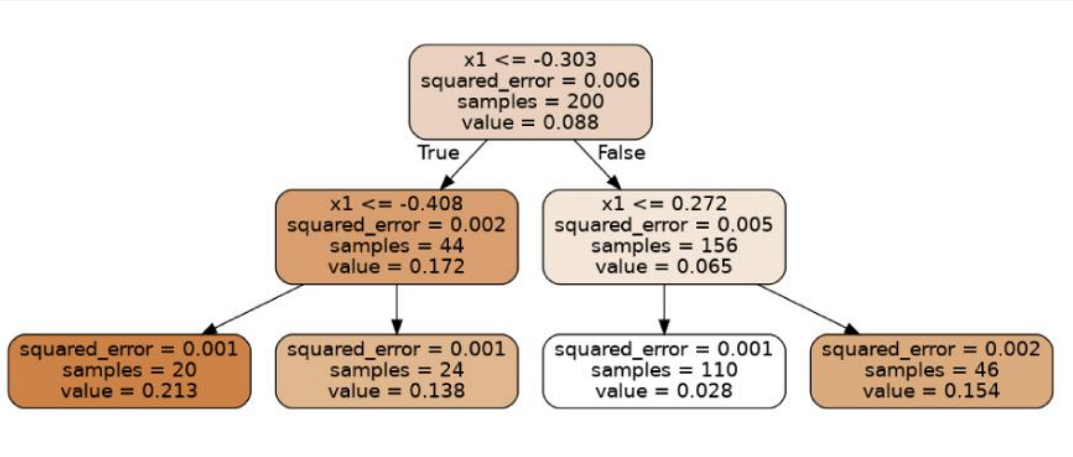


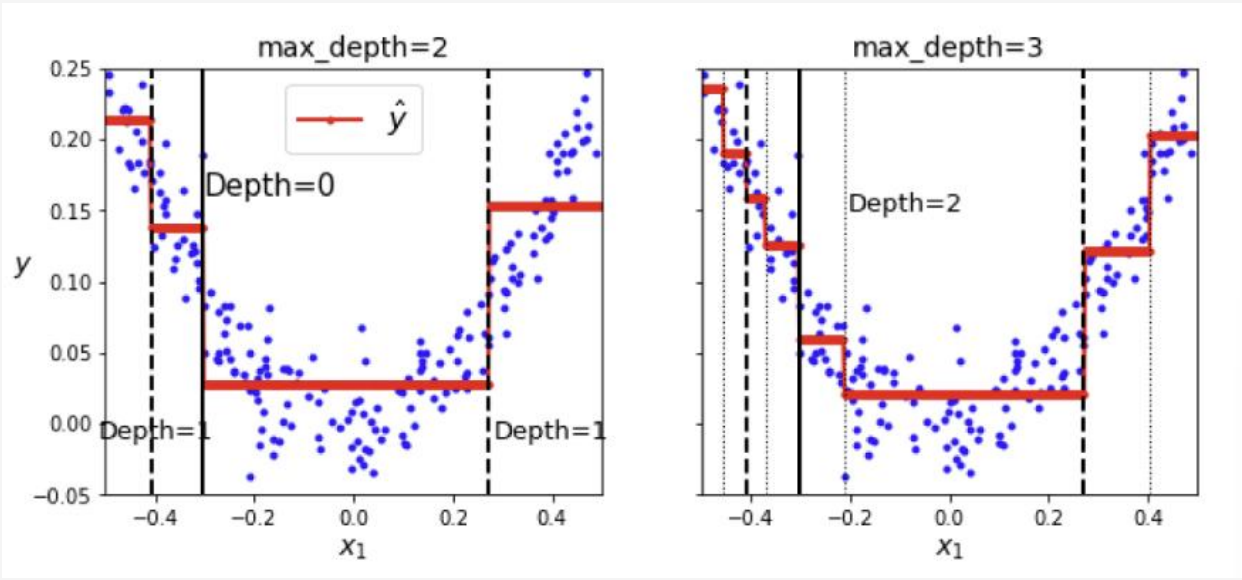


WeTIE

6 회귀 결정트리

사이킷런의 DecisionTreeRegressor 예측기 활용





WeTIE

6 회귀 결정트리

회귀용 CART 알고리즘과 비용함수

분류의 경우처럼 아래 비용함수를 최소화하는 특성 k와 해당 특성의 임계값 t_k를 결정해서 사용함

$\mathrm{MSE}_{\mathrm{node}}$: 해당 노드의 평균제곱오차(MSE).
m_{node}: 해당 노드에 속하는 샘플 수
$y^{(i)}$: 샘플 i에 대한 실제 타깃값

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

$$\mathrm{MSE}_{\mathrm{node}} = \frac{1}{m_{\mathrm{node}}} \sum_{i \in \mathrm{node}} (\hat{y}_{node} - y^{(i)})^2$$

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

WeTIE

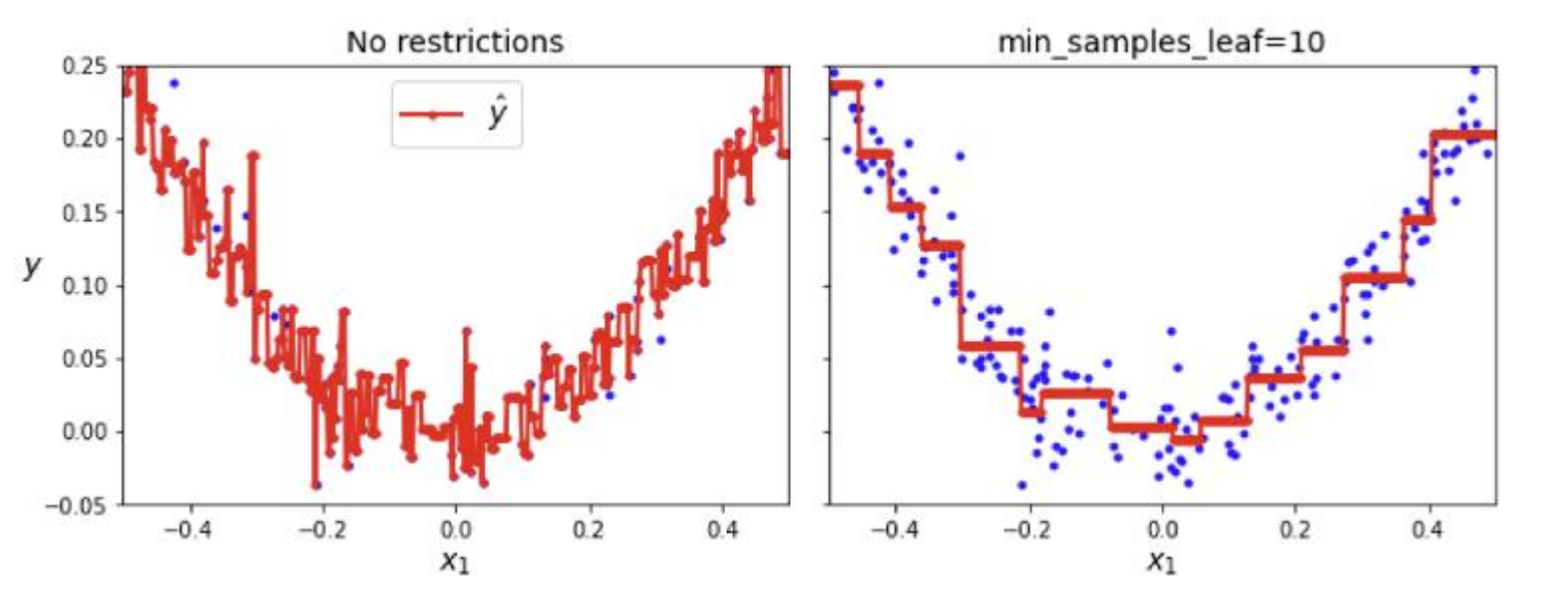
6 회귀 결정트리

규제

분류의 경우처럼 규제가 없으면 과대적합이 발생할 수 있음

왼쪽 : 규제가 없는 경우, 과대적합 발생

오른쪽 : min_sample_leaf = 10



WeTIE

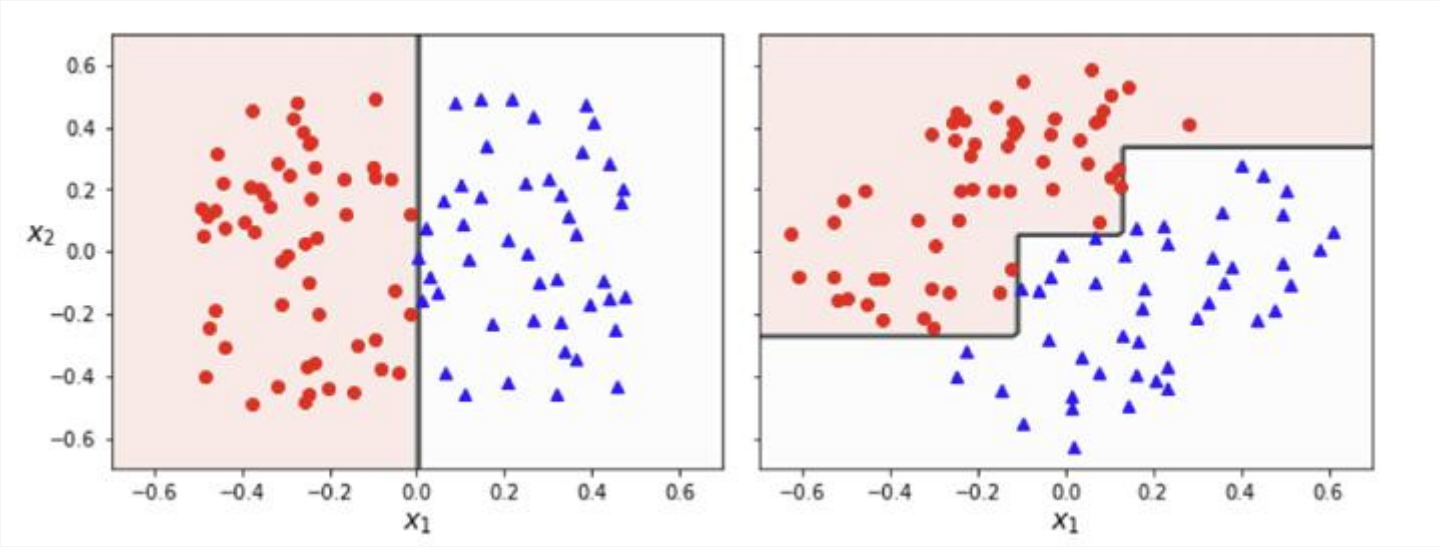
7 결정트리 단점

1. 훈련셋 회전 민감도

결정트리 알고리즘은 성능이 매우 우수하지만 기본적으로 주어진 훈련셋에 민감하게 반응함

결정트리는 항상 축에 수직인 분할을 사용, 따라서 조금만 회전을 가해도 결정 경계가 많이 달라짐

오른쪽 그래프는 왼쪽 그래프를 45도 회전시킨 훈련셋 학습



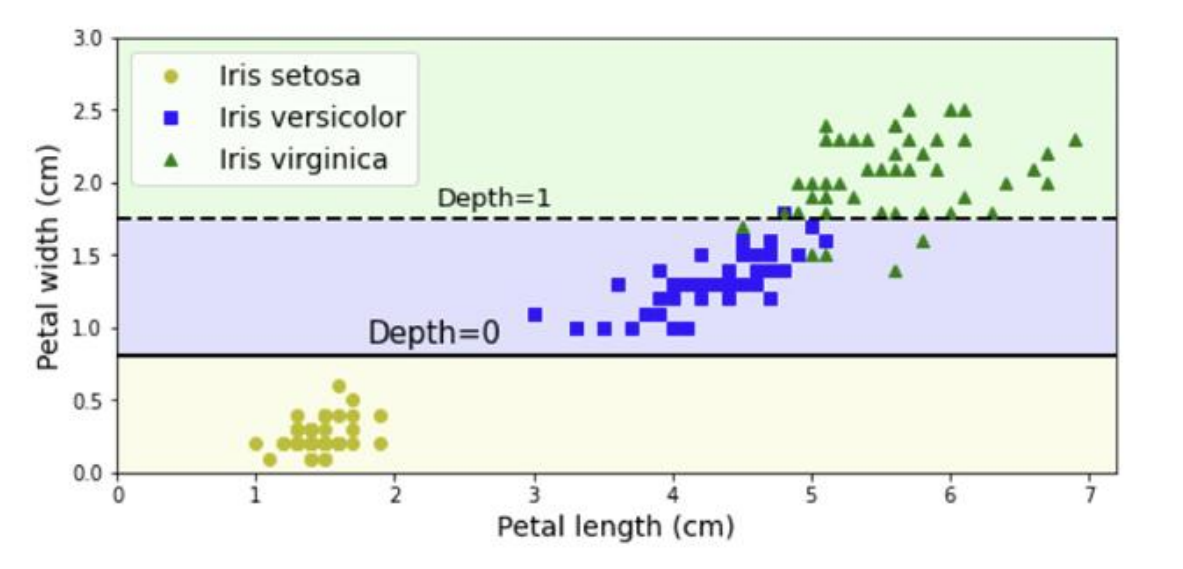
7 결정트리 단점

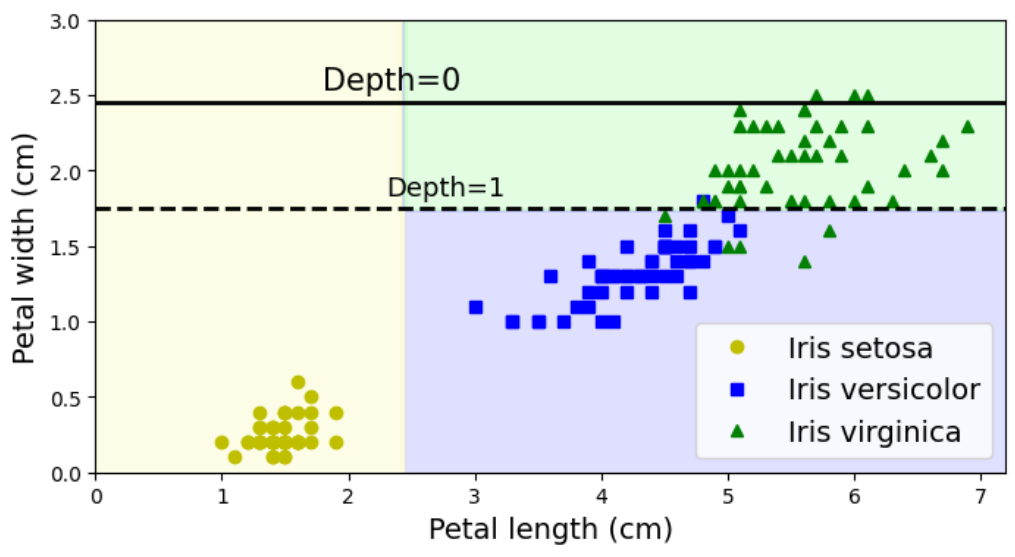
2. 높은 분산

훈련 데이터의 작은 변화에도 매우 민감함

```
tree_clf_tweaked = DecisionTreeClassifier(max_depth=2, random_state=40)
tree_clf_tweaked.fit(X_iris, y_iris)
```

random_state를 지정하지 않으면서 동일한 모델을 훈련시키면 다른 결과가 나온다.





높은 분산 문제는 여러 개의 결정트리를 동시에 훈련시킨 후 평균값을 활용하는 랜덤 포레스트 모델을 이용하면 해결할 수 있다.

WeTIE

E.O.D

6주차. 결정 트리