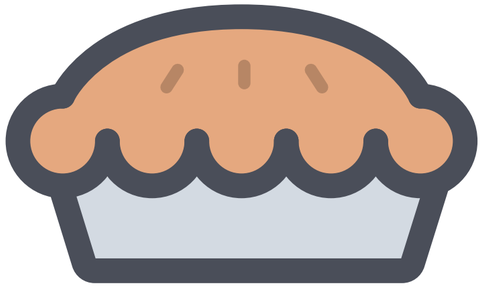


# BIG_PY
# LEVEL.2
# SESSION

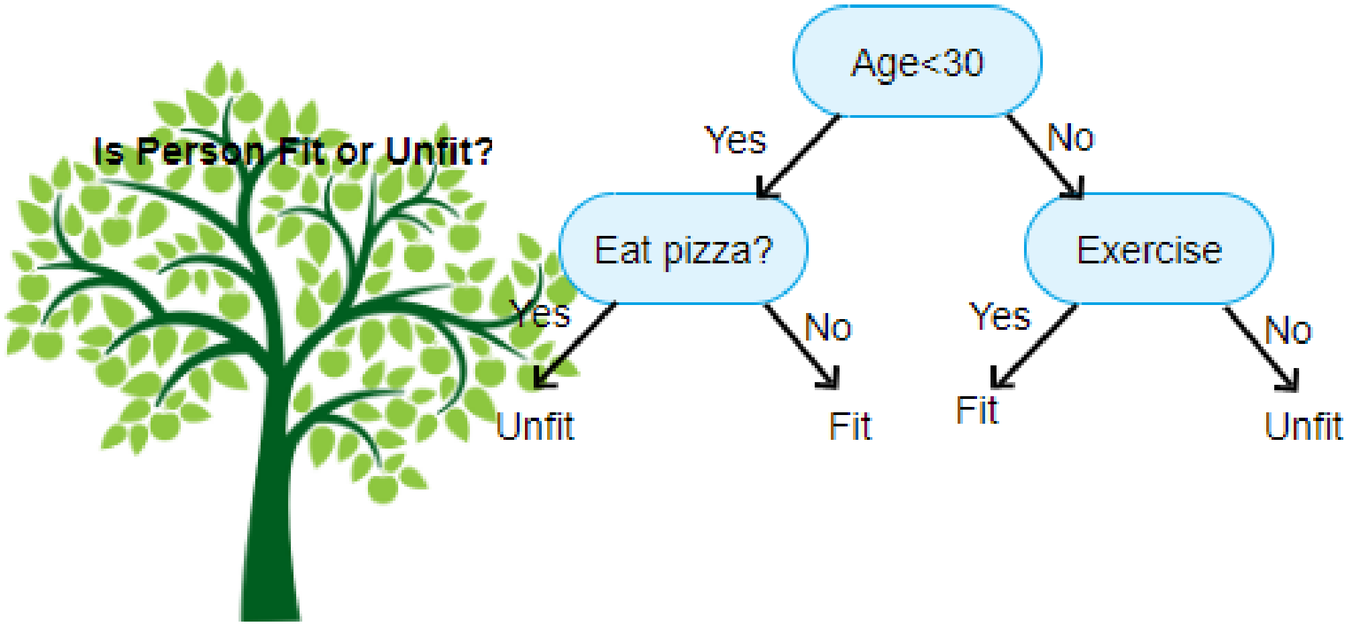
# Chapter.06
# 결정 트리

세션장
나마로

# 결정 트리(decision tree)



- 분류와 회귀 작업, 그리고 다중출력 작업도 가능한 머신러닝 알고리즘
- 최근 자주 사용되는 가장 강력한 머신러닝 알고리즘 중 하나인 랜덤 포레스트(random forest)의 기본 구성 요소임




https://m.blog.naver.com/pmw9440/222077642636

# 결정 트리 학습과 시각화



```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)
```

# 결정 트리 학습과 시각화



- export_graphviz() 함수를 사용해 그래프를 .dot 파일로 출력하여 시각화 가능

```
from graphviz import Source
from sklearn.tree import export_graphviz

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"))
```

# 결정 트리 학습과 시각화



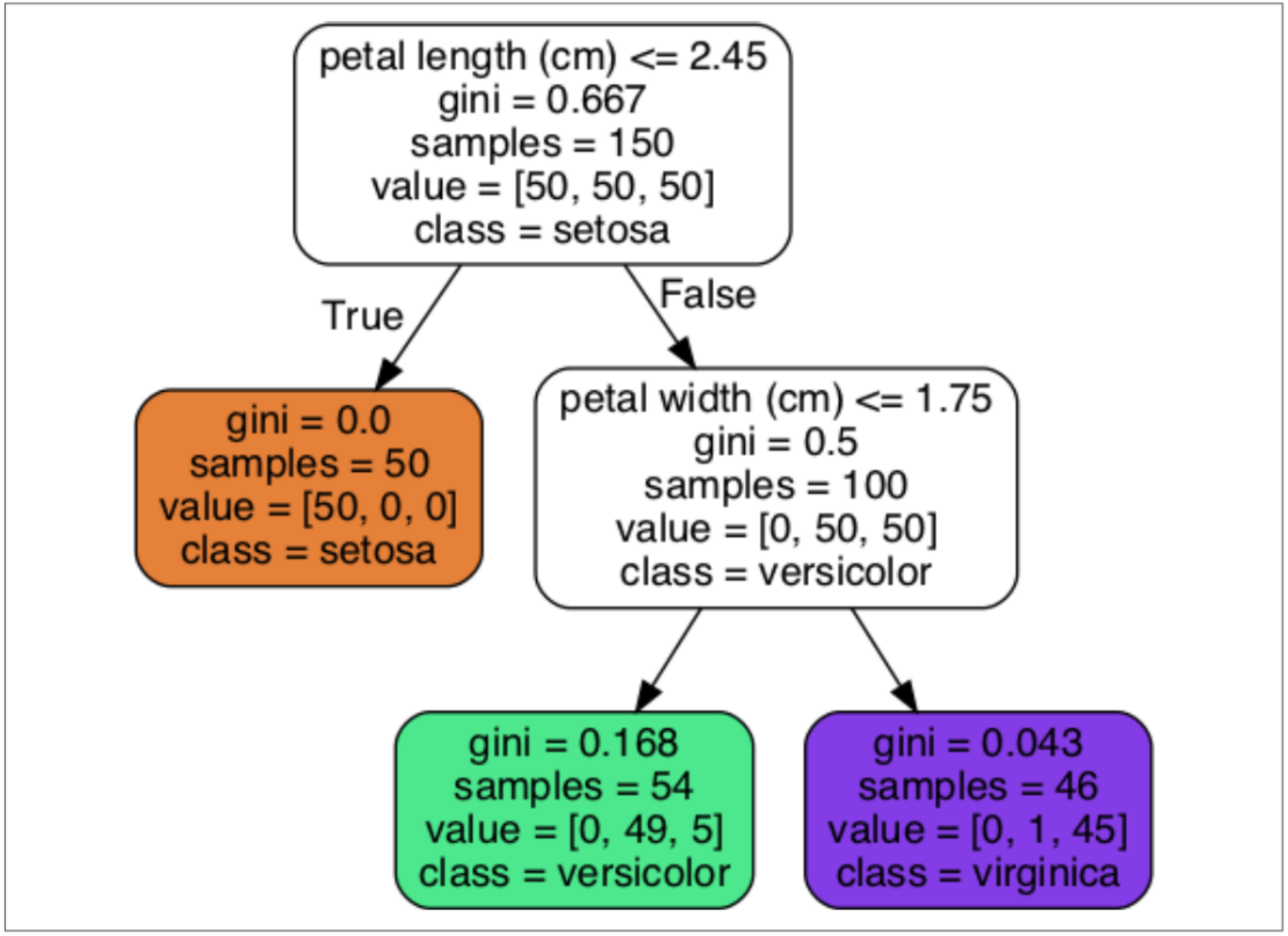



*Figure 6-1. Iris Decision Tree*

# 예측하기



- 루트 노트(root node)부터 리프 노드(leaf node)까지 도달하면서 최종적으로 클래스를 예측
- 대소 비교를 통해 구분하기 때문에 스케일을 맞추는 등 전처리가 필요하지 않음
- 그림의 각 항목은 다음 의미를 가짐

  ✔ samples: 얼마나 많은 훈련 샘플이 적용되었는지 측정

  ✔ value: 노드에서 각 클래스에 얼마나 많은 훈련 샘플이 있는지 측정

  ✔ gini: 지니 불순도(gini inpurity)값 측정

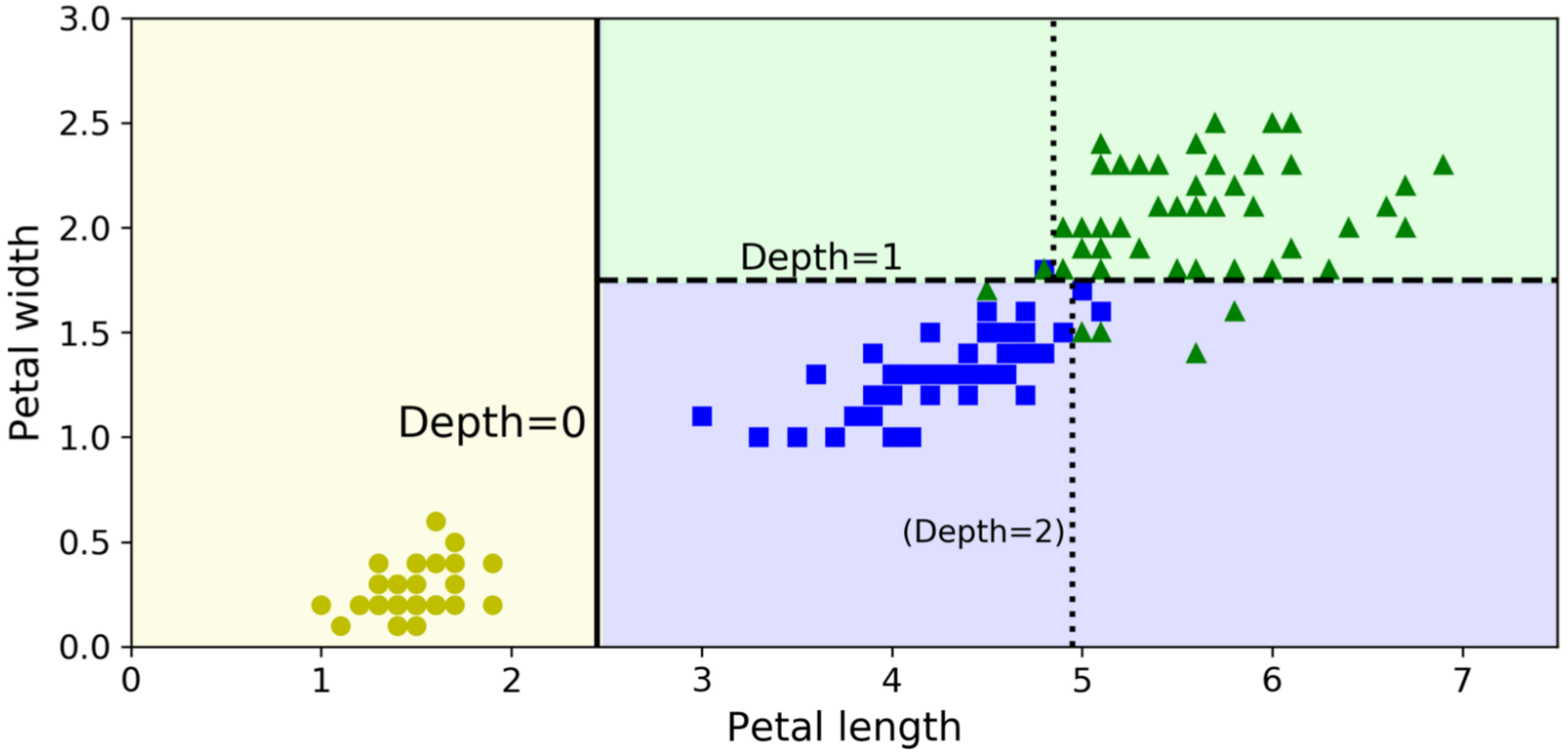
*Equation 6-1. Gini impurity*

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$

# 예측하기



- 왼쪽 영역은 순수 노드(Iris-Setosa만 있음)이기 때문에 더는 나눌 수 없음
- 오른쪽 영역은 순수 노드가 아니므로 깊이 1의 오른쪽 노드는 너비 1.75cm에서 나누어짐
- max_depth를 3으로 설정한다면 점선처럼 경계를 추가로 만들 것임



# 클래스 확률 추정



- 결정 트리는 한 샘플이 특성 클래스 k에 속할 확률을 추정할 수도 있음
  - ✔ 먼저 한 샘플에 대해 리프 노드를 찾기 위해 트리를 탐색
  - ✔ 탐색 후 그 노드에 있는 클래스 k의 훈련 샘플의 비율을 반환

```
tree_clf.predict_proba([[5, 1.5]])
```

```
array([[0.        , 0.90740741, 0.09259259]])
```

```
tree_clf.predict([[5, 1.5]])
```

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

# CART 훈련 알고리즘



- 사이킷런은 결정 트리를 훈련시키기 위해(즉, 트리를 성장시키기 위해) CART(Classification And Regression Tree) 알고리즘을 사용

  ✔ 먼저 훈련 세트를 하나의 특성 k의 임계값 $t_k$를 사용해 두 개의 서브셋으로 나눔

  ✔ 크기에 따른 가중치가 적용된 가장 순수한 서브셋으로 나눌 수 있는 $(k, t_k)$ 짝을 찾음

  ✔ 아래 비용함수를 최소화하는 방향으로 진행

  ✔ 최대 깊이가 되면 중지하거나 불순도를 줄이는 분할을 찾을 수 없을 때 멈추게 됨

*Equation 6-2. CART cost function for classification*

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

$$\text{where} \begin{cases} G_{\text{left/right}} \text{ measures the impurity of the left/right subset,} \\ m_{\text{left/right}} \text{ is the number of instances in the left/right subset.} \end{cases}$$

# 계산 복잡도



- 예측을 하려면 결정 트리를 루트 노드에서부터 리프 노드까지 탐색해야 함
- 결정 트리를 탐색하기 위해서는 약 *O(log2(m))*개의 노드를 거쳐야 함
- 훈련 세트가 작을 경우(수천개 이하) 사이킷런은 미리 데이터를 정렬하여 훈련 속도를 높일 수 있음.

  (presort=True)

# 불순도(Impurity)



- 기본적으로 결정 트리는 gini 불순도를 사용하지만, criterion="entropy"로 지정하면 엔트로피 불순도를 사용할 수 있음
  - ✔ 여기서 엔트로피란, 메세지의 평균 정보 양을 측정하는 섀넌의 정보이론에서 가져옴
  - ✔ 엔트로피가 0이라면, 모든 메세지가 동일하다는 의미
- 지니 불순도가 조금 더 계산이 빠르기 때문에 기본값으로 사용하기 좋음
- 다른 트리가 만들어지는 경우 gini 불순도가 가장 빈도 높은 클래스를 한쪽 가지로 고립시키는 경향이 있음
- 엔트로피 불순도는 gini보다 조금 더 균형 잡힌 트리를 만듬

*Equation 6-3. Entropy*

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

# 규제 매개변수



- 결정 트리는 훈련 데이터에 대한 제약 사항이 거의 없음

  ✔ 제한을 두지 않기에 트리가 훈련 데이터에 아주 가깝게 맞추려고 해서 대부분 과대적합되기 쉬움

- 결정트리는 훈련되기 전에 파라미터 수가 결정되지 않음

  ✔ 이러한 모델을 비파라미터 모델(nonparametric model)이라고 부름

  ✔ 반대로 선형회귀와 같은 선형 모델은 파라미터 모델(parametric model)이라 부름

  ✔ 파라미터 모델은 미리 정의된 모델 파라미터 수를 가지므로 자유도가 제한되고 과대적합될 위험이 줄어드나, 과소적합될 위험은 커진다

- 결정 트리에서는 최대 깊이를 제어할 수 있음

  ✔ max_depth로 조절하며, 이를 줄이면 모델을 규제하게 되고, 과대적합의 위험이 감소함

# 규제 매개변수



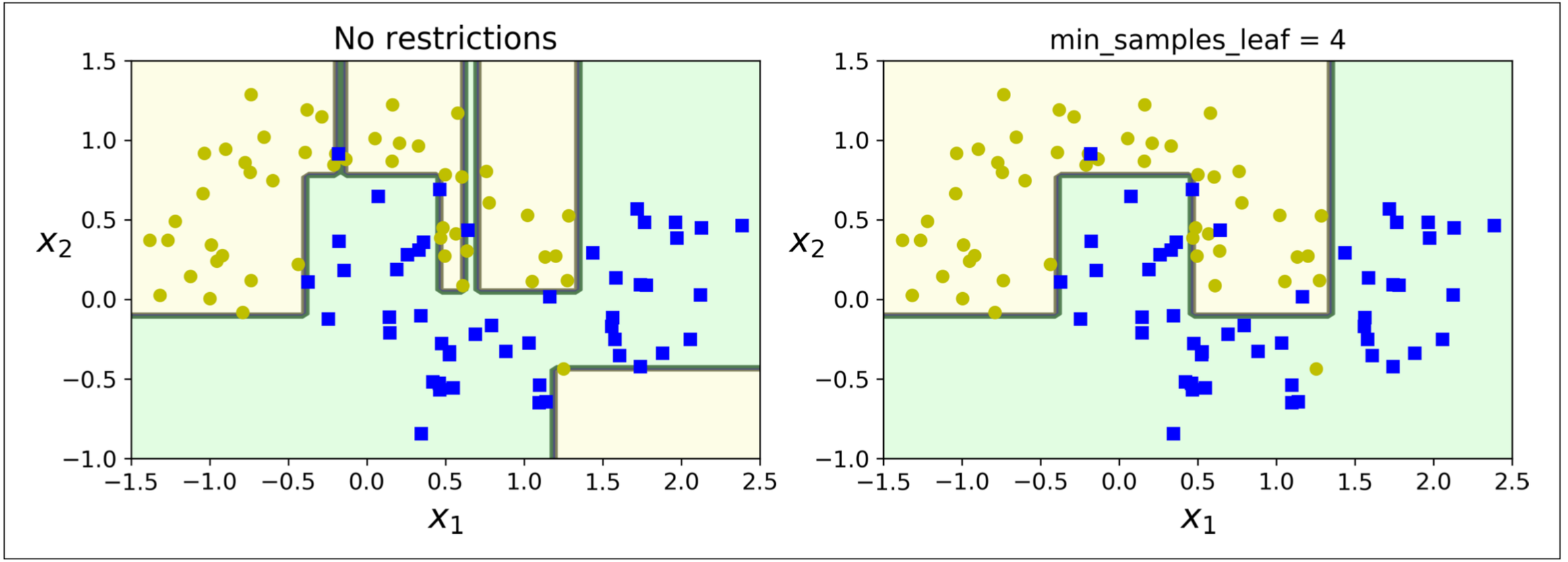



*Figure 6-3. Regularization using min_samples_leaf*

# 회귀(Regression)



- 결정 트리는 회귀 문제에도 사용할 수 있음
- 각 노드에서 클래스를 예측하는 대신 어떤 값을 예측함

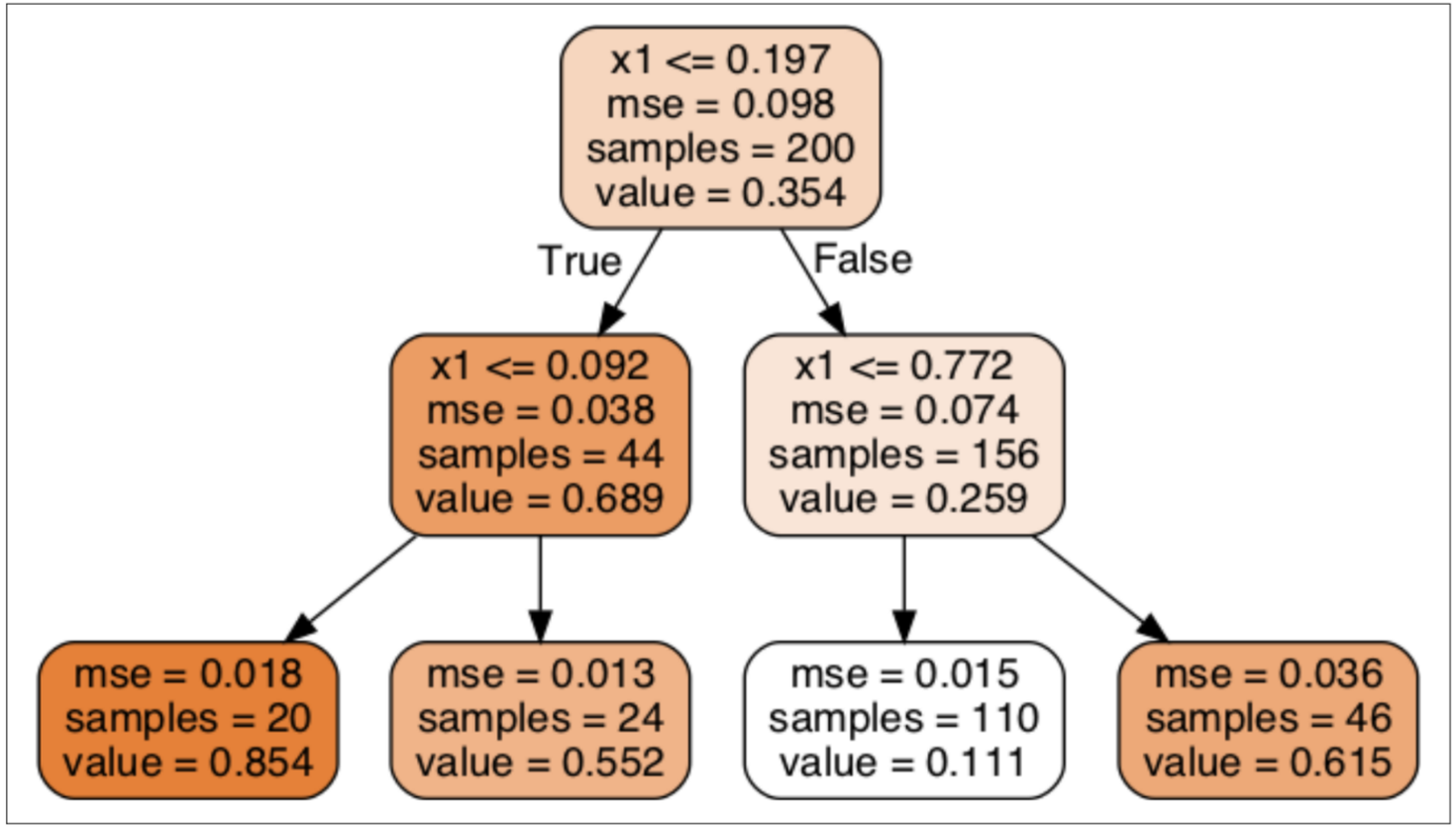



*Figure 6-4. A Decision Tree for regression*

# 회귀(Regression)



- CART 알고리즘은 훈련 세트를 평균제곱오차(MSE)를 최소화하도록 분할하는 것을 제외하고는 분류 작업과 거의 비슷하게 작동함

*Equation 6-4. CART cost function for regression*

$$J(k, t_k) = \frac{m_{\text{left}}}{m}\text{MSE}_{\text{left}} + \frac{m_{\text{right}}}{m}\text{MSE}_{\text{right}} \quad \text{where} \begin{cases} \text{MSE}_{\text{node}} = \sum_{i \in \text{node}} \left(\hat{y}_{\text{node}} - y^{(i)}\right)^2 \\ \hat{y}_{\text{node}} = \frac{1}{m_{\text{node}}} \sum_{i \in \text{node}} y^{(i)} \end{cases}$$

# 회귀(Regression)



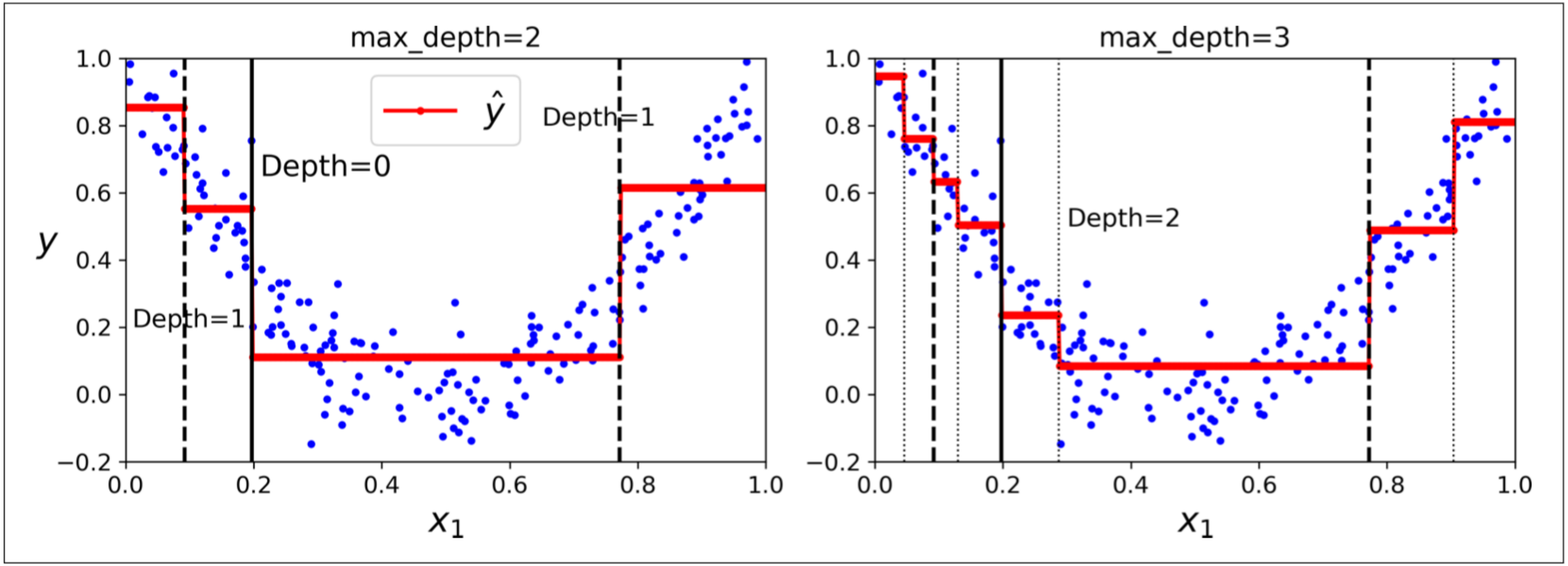



*Figure 6-5. Predictions of two Decision Tree regression models*

# 회귀(Regression)



- min_samples_leaf=10 으로 지정하면 과대적합을 어느정도 해소할 수 있음

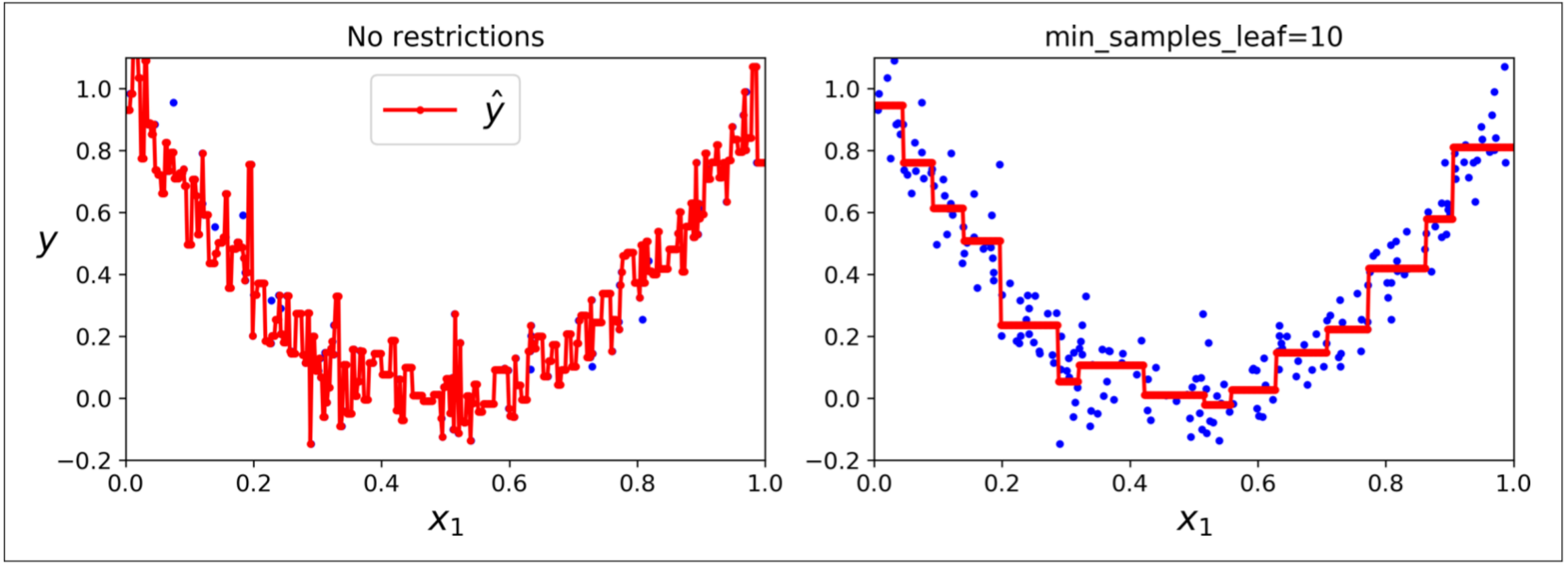



*Figure 6-6. Regularizing a Decision Tree regressor*

# 불안정성(Instability)



- 결정 트리는 계단 모양의 결정 경계를 만듦
  - ✔ 즉 모든 분할은 축에 수직임
  - ✔ 결정 트리는 훈련 세트의 회전에 민감함
- 훈련 데이터를 더 좋은 방향으로 회전시키는 PCA 기법을 사용하면 효과가 좋아질 수 있음

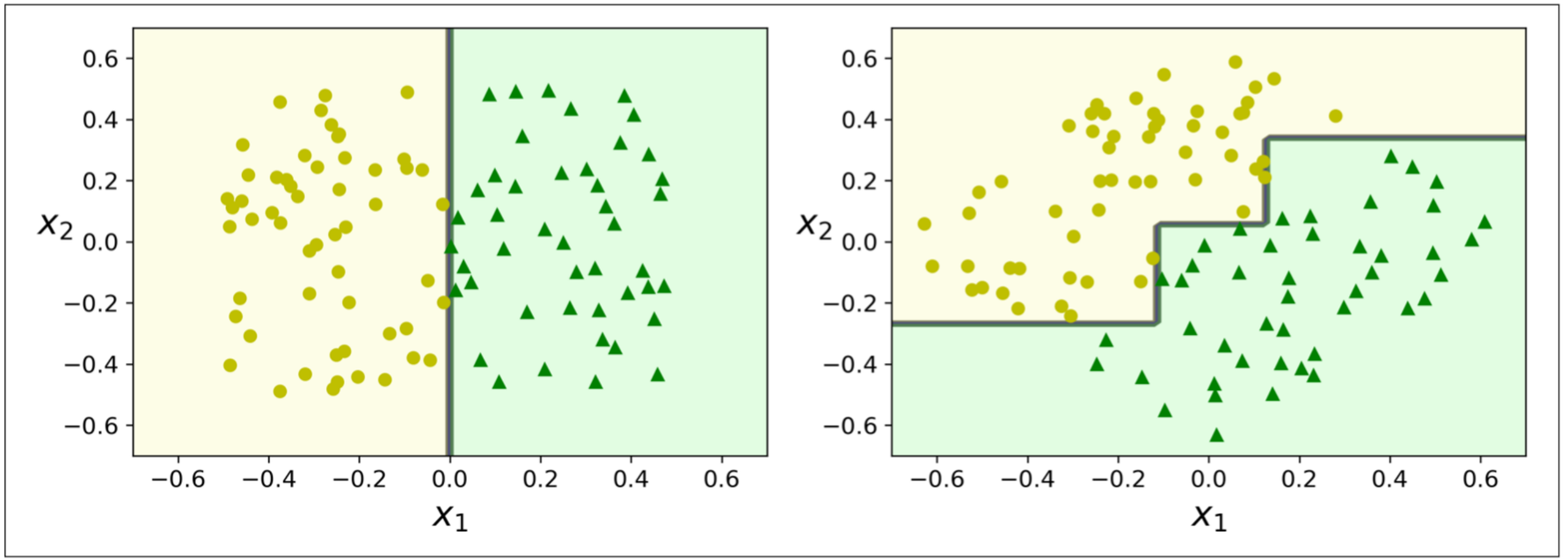



*Figure 6-7. Sensitivity to training set rotation*

# QnA

LEVEL.2
SESSION