

1. 트리 (tree):

1) node와 edge로 구성. 계층적 구조를 표현하기 위한 자료 구조.

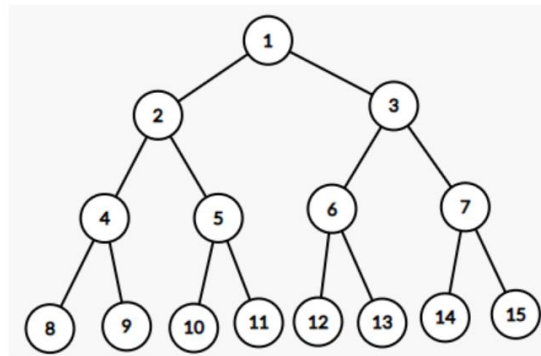
2) Edge는 node와 node를 연결한 선.

3) 맨 꼭대기 node는 root 노드. 자식이 없는 node를 leaf node.

4) 최대 두개의 자식을 갖는 트리를 이진 트리 (binary tree)라고 한다.

5) 트리의 depth: root노드로부터 특정 node까지의 경로 상의 edge 수

예: root node는 node 1, node 1의 자식 node는 node 2와 node 3, node 2의 부모는 node 1



<이진 트리의 예>

6) 실생활에서 tree 구조의 대표적 예

윈도우 파일 탐색기.

디렉토리 (node), 파일 (leaf node)

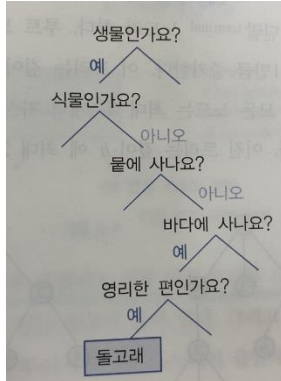
상위 디렉토리와 하위 디렉토리는: 부모와 자식 관계

```
C:\
├─ Program Files
│   ├── Application1
│   └─ Application2
├─ Users
│   ├── User1
│   └─ User2
└─ Windows
```

2. 결정 트리 (decision tree):

1) 자료구조에서 나오는 트리 (tree) 구조를 사용한 머신 러닝 (ML) 모델.

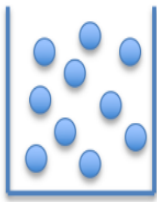
- 2) 훈련 데이터에 대하여 자동으로 질문을 만들어 새로운 데이터를 분류 또는 값을 예측.
- 3) 쉽게 생각하면 아래 그림과 같은 "스무 고개 놀이". 이것도 트리구조 이다. 훈련 (학습) 데이터를 통해 만들고 새로운 데이터를 예측하는 모델이다.



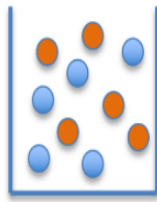
- 4) 결정 트리의 경우 대부분 이진 트리 형태
- 5) 결정 트리는 머신 러닝 방법 중에서 분류기에서 가장 많이 쓰이는 방법 중의 하나인 **random forest** 기본이 되는 방법으로 반드시 알아야 하는 중요한 방법.
- 6) 결정 트리는 **non-parametric model (비모수적 모델)**에 속함. 학습하기 전에 미리 정해진 파라미터의 수를 가정하지 않음.
- 7) 대표적인 parametric model은 linear regression (선형 회귀). 입력 데이터 (X)와 출력 데이터 (Y)간의 특정 형태를 가정. 예측해야 하는 파라미터가 고정되어 있음. $y=aX+b$
- 9) Scikit-learn의 decision tree 설명: [1.10. Decision Trees — scikit-learn 1.3.0 documentation](https://scikit-learn.org/stable/modules/tree.html)

3. 지니 불순도 (Gini impurity): decision tree에는 지니 불순도라는 개념이 등장한다.

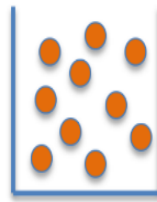
- 1) 지니 불순도란 어떤 집합의 "순도 (purity)" 혹은 동질성을 측정
- 2) 아래 예를 보면 항아리에 10개의 구슬이 있다. 항아리 1은 파란색 구슬만 있음 (불순도=0). 항아리 3은 빨간색 구슬만 있음 (불순도=0).
- 3) 항아리 2는 빨간색 반, 파란색 반. 빨간 구슬의 비율이 1/2. 불순도가 가장 높음



항아리 1.

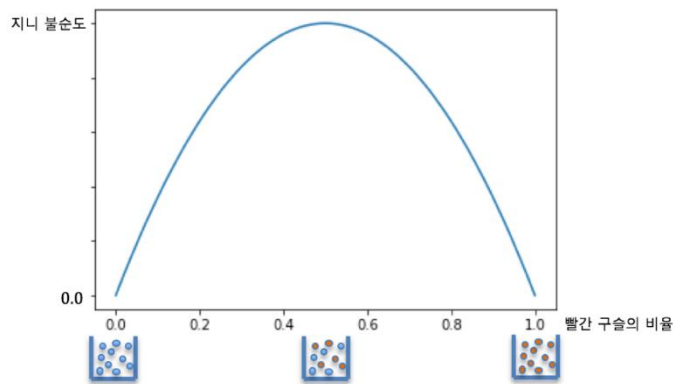


항아리 2.



항아리 3.

- 4) 아래 그림을 보면 빨간 구슬의 비율이 0이거나 1일 경우 불순도=0이다. 빨간 구슬과 파란 구슬이 반반씩 섞인 경우 불순도가 0.5로 가장 큼. 즉, 순도가 가장 낮음



- 4. 지니 불순도 측정 수식:** 결정 트리에서 i번째 node의 Gini 불순도 측정

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

$p_{i,k}$: i번째 node에 있는 훈련 sample중 class k에 속한 sample의 비율

- 쉽게 생각하면 해당 node에서 class들이 혼합되어진 정도

- 5. 간단한 지니 불순도 측정 예:** 어떤 노드 T에 9개의 sample이 있고 총 3개의 class가 있다. 9개의 sample 중에 class 1인 sample이 3개, class 2인 sample이 4개, class 3인 sample이 2개. 이 노드의 Gini 불순도 계산

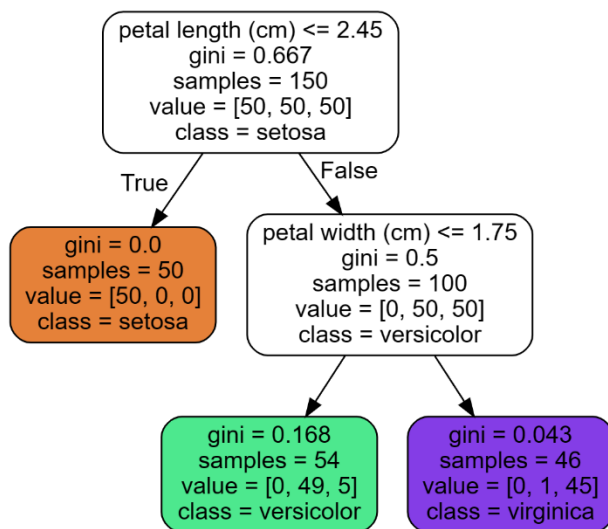
$$1 - (3/9)^2 - (4/9)^2 - (2/9)^2 \approx 0.642$$

- Q: 만일 노드 T에 9개의 sample이 있고 총 3개의 class가 있는데 9개의 sample이 모두 class 1이면. 이때의 지니 불순도는? 0

6. (실습) Iris dataset: iris dataset에 대한 설명.

1) 150개의 sample로 구성, 3개의 class (setosa, versicolor, virginica), 4개의 특징: sepal length (꽃받침 길이), sepal width (꽃받침 너비), petal length (꽃잎 길이), petal width (꽃잎 너비).

8. (실습) Iris dataset에 대한 결정 트리 분류기 학습 및 적용 예 (max_depth=2)



- Tree 구조에서 depth란: node가 root node로부터 얼마나 떨어져 있다?

- Tree 구조의 max depth: 최대 깊이

- Iris dataset에 대한 ML 결정 트리 적용 예'

- Petal length와 Petal width 2개의 특징만 분류에 사용

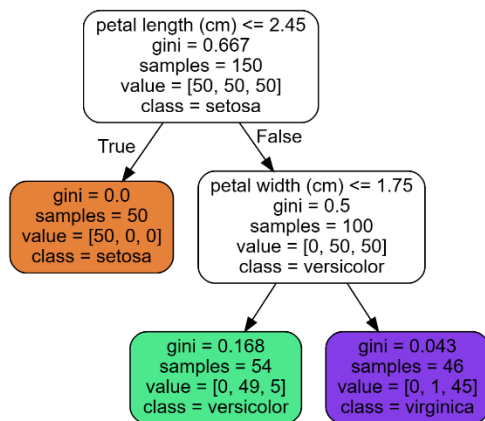
특징 추출

```
X = iris.data[:, 2:] # petal length and width
```

1) 먼저 root node에서 시작

- 2) root node에서 petal length가 2.45cm보다 짧은지 질문 (지니 불순도 0.67)
- 3) petal length가 2.45cm보다 짧으면 왼쪽 자식 node로 이동
- 4) 이 경우 node가 leaf node이므로 추가적인 질문을 하지 않음. 지니 불순도가 0임 예측 시 Node에 있는 예측 class를 보고 결정 트리가 새로 발견한 꽃의 품종을 Setosa라고 판단함
 - 첫번째 질문으로 Setosa class를 완벽하게 분리할 수 있다는 것을 의미함
- 5) root node에서 petal length가 2.45cm보다 크면 오른쪽 자식 node로 이동. 이 경우에는 leaf node가 아니기 때문에 추가로 질문. 지니 불순도=0.5. petal width가 1.75cm보다 작은가?.
- 6) 참이라면 꽃의 품종을 versicolor로 판단. 거짓이라면 virginica로 판단.

9. (실습) 위의 예에서 결정 트리의 각 node의 속성들



1) node의 sample 속성: 얼마나 많은 훈련 sample이 적용되었는지 따짐.

- 주황색 node: 전체 150개의 sample 중 50개의 훈련 sample이 petal length가 2.45cm보다 짧거나 같다.
- 100개의 훈련 sample이 petal length가 2.45cm보다 크다. 그중 54개의 sample이 petal width가 1.75cm보다 짧거나 같고 46개의 sample이 petal width가 1.75cm보다 길다.

2) node의 value 속성: 각 node에서 각각의 class에 얼마나 많은 sample이 있는지

- Value=[setosa, versicolor, virginica]

- root node: value=[50, 50, 50]. setosa=50개, versicolor=50개, virginica=50개
- 보라색 node: value=[0, 1, 45]. setosa=0개, vericolor=1개, virginica=45개

3) node의 Gini 속성: 각 node의 Gini 불순도 측정

- 한 노드의 모든 sample이 같은 class (주황색 node): Gini 불순도=0
- Q: root node의 Gini 불순도: $1-(1/3)^2-(1/3)^2-(1/3)^2=2/3=0.667$

10. 결정 트리 훈련 알고리즘: CART 알고리즘

1) 기본 idea: 최대한 각 자식 node (subset)가 불순도가 낮도록 (즉, 동일한 class를 갖도록) 분류해야 한다. 즉, 그런 특징 (feature)과 그 경계값을 선택해야 한다. 불순도가 낮다는 것은 한 node에 동일한 class들끼리 있다는 것이고 그래야 분류가 잘된 것이다.

3) 어떻게? 각 특징에 대해서 분류 후에 Gini 불순도 계산. 그 중에 어떤 특징으로 분류했을 때 Gini 불순도가 가장 낮아지는지 따진다. 그 특징으로 가장 먼저 나눈다.

- 구체적으로 아래 J를 최소화 하는 특징 선택

$$J = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$$

G_{left} : 왼쪽 subset의 불순도, G_{right} : 오른쪽 subset의 불순도

m_{left} : 왼쪽 subset의 sample수, m_{right} : 오른쪽 subset의 sample 수

4) 최초에 Gini 불순도가 가장 낮아지는 feature를 결정 트리의 root node로 둔다.

5) 이를 각각의 자식 node에 대해서도 recursive하게 반복

6) 언제 멈춤: 최초에 정해진 max_depth (tree의 깊이) 까지

11. CART 알고리즘 예제

CART 알고리즘은 각 Feature에 대해 가능한 모든 분할을 테스트하고, **지니 불순도(Gini Impurity)**를 최소화하는 분할을 선택합니다.

분할을 반복하여 재귀적으로 트리를 확장합니다.

CART 알고리즘 예제 (3개의 Feature)

문제

학생들이 시험에 합격(Pass)할지 실패(Fail)할지를 예측하는 문제입니다. 학생 데이터는 다음과 같고, Feature는 세 개입니다:

공부 시간(Hours)	과목 선호도(Preference)	집중도(Attention)	시험 결과(Pass/Fail)
2	5	4	Fail
3	7	3	Fail
6	6	7	Pass
8	9	8	Pass
4	4	5	Fail

Feature 1: 공부 시간(Hours)

분할 예시: $\text{Hours} \leq 4$ 와 $\text{Hours} > 4$

1. 왼쪽 그룹 ($\text{Hours} \leq 4$): [2, 3, 4] → 결과: [Fail, Fail, Fail]

- 지니 불순도:

$$G_{\text{left}} = 1 - (1^2 + 0^2) = 0$$

2. 오른쪽 그룹 ($\text{Hours} > 4$): [6, 8] → 결과: [Pass, Pass]

- 지니 불순도:

$$G_{\text{right}} = 1 - (0^2 + 1^2) = 0$$

3. 전체 지니 불순도(가중 평균):

$$G_{\text{total}} = \frac{3}{5} \cdot 0 + \frac{2}{5} \cdot 0 = 0$$

분석: 이 분할에서 지니 불순도는 완벽히 0입니다.

Feature 2: 과목 선호도(Preference)

분할 예시: $\text{Preference} \leq 6$ 와 $\text{Preference} > 6$

1. 왼쪽 그룹 ($\text{Preference} \leq 6$): [5, 7, 6, 4] → 결과: [Fail, Fail, Pass, Fail]

- 클래스 비율: $\text{Fail} = \frac{3}{4}$, $\text{Pass} = \frac{1}{4}$
- 지니 불순도:

$$G_{\text{left}} = 1 - \left(\left(\frac{3}{4} \right)^2 + \left(\frac{1}{4} \right)^2 \right) = 1 - \left(\frac{9}{16} + \frac{1}{16} \right) = \frac{6}{16} = 0.375$$

2. 오른쪽 그룹 ($\text{Preference} > 6$): [9] → 결과: [Pass]

- 지니 불순도:

$$G_{\text{right}} = 1 - (1^2 + 0^2) = 0$$

3. 전체 지니 불순도(가중 평균):

$$G_{\text{total}} = \frac{4}{5} \cdot 0.375 + \frac{1}{5} \cdot 0 = 0.3$$

분석: 이 분할에서 지니 불순도는 0.3으로 비교적 낮지만, 공부 시간보다 덜 효과적입니다.

Feature 3: 집중도(Attention)

분할 예시: $\text{Attention} \leq 5$ 와 $\text{Attention} > 5$

1. 왼쪽 그룹 ($\text{Attention} \leq 5$): [4, 3, 5] → 결과: [Fail, Fail, Fail]

- 지니 불순도:

$$G_{\text{left}} = 1 - (1^2 + 0^2) = 0$$

2. 오른쪽 그룹 ($\text{Attention} > 5$): [7, 8] → 결과: [Pass, Pass]

- 지니 불순도:

$$G_{\text{right}} = 1 - (0^2 + 1^2) = 0$$

3. 전체 지니 불순도(가중 평균):

$$G_{\text{total}} = \frac{3}{5} \cdot 0 + \frac{2}{5} \cdot 0 = 0$$

분석: 이 분할에서도 지니 불순도는 0입니다.

3단계: 최적 분할 선택

모든 Feature에 대해 지니 불순도를 계산한 결과:

- 공부 시간(Hours): $G_{\text{total}} = 0$
- 과목 선호도(Preference): $G_{\text{total}} = 0.3$
- 집중도(Attention): $G_{\text{total}} = 0$

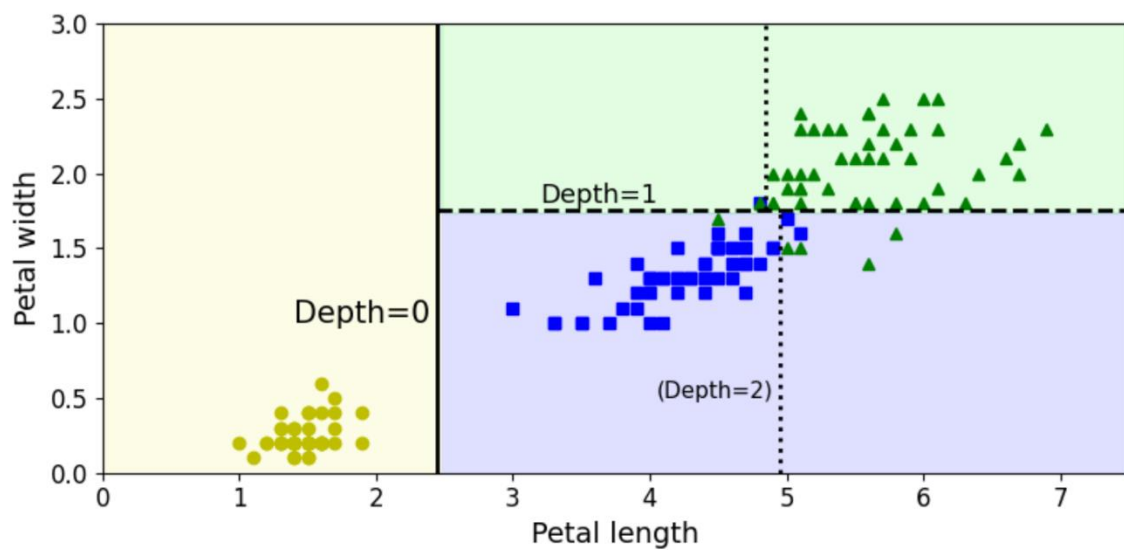
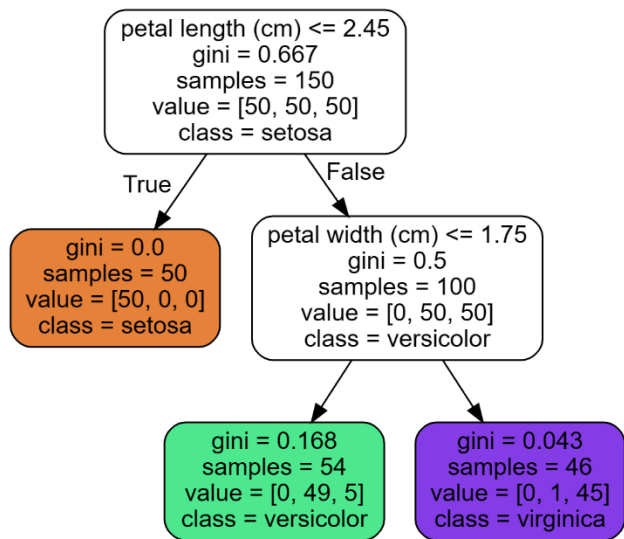


→ 공부 시간(Hours) 또는 집중도(Attention)가 최적의 분할 기준입니다. 여기서는 공부 시간(Hours)을 선택합니다.

4단계: 트리 확장

공부 시간(Hours)을 기준으로 트리를 나누고, 각 그룹에서 다시 CART 알고리즘을 반복합니다.

12. (실습) 결정 트리를 사용한 결정경계 시각화



- 1) Depth 0에서의 결정 경계: Petal length = 2.45cm
- 2) 왼쪽은 Iris-setosa만 있어서 더 나눌게 없음
- 3) 오른쪽은 Petal width 1.75cm에서 나뉨. 1.76cm보다 작으면 versicolor, 크면 virginica
- 4) Max_depth를 2로 했기 때문에 결정트리는 더 이상 분할되지 않음
- 5) 만일, max_depth가 3이면 깊이 2의 두 node가 각각 결정 경계를 추가로 만듭니다 (점선 부분)

13. Decision tree의 동작

- 1) 데이터 준비
- 2) 트리 구조 준비: root node (트리 시작점)
- 3) 특징 선택 및 분할: CART 알고리즘으로 node의 지니 불순도를 최소화하도록 분할
- 4) 트리를 반복적으로 분할. 데이터가 더 이상 분할할 수 없을 때 leaf 노드 생성
- 5) 특정 종료 조건에 다다르면 트리 생성 멈춤

예: 1. 노드에 있는 데이터들이 모두 같은 class

2. 사전 설정된 "최대 깊이" 도달
3. 사전 설정된 "최소 샘플 수 조건" (노드에 있는 데이터가 너무 적음) 만족
4. 분할로 인한 지니 불순도의 변화가 미미함등

14. Overfitting 문제 and regularization (규제):

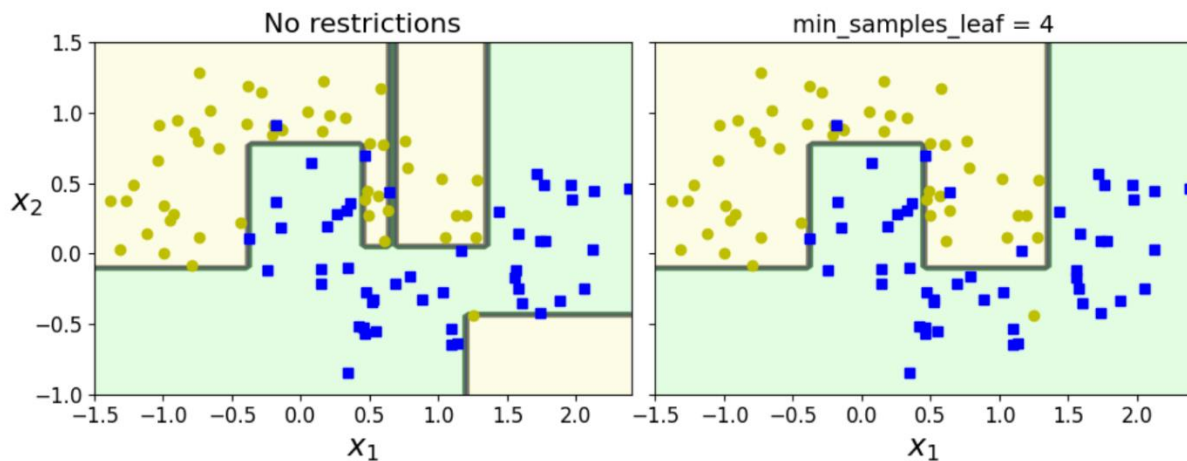
- 1) overfitting 문제: 머신 러닝 모델이 훈련 데이터에 과하게 너무 맞춰져서, 새로운 데이터에 대한 일반화 (generalization) 능력이 떨어지는 것. 훈련 데이터에는 높은 성능을 보이지만 새로운 데이터에 대해서는 성능이 낮은 문제
- 2) 결정 트리는 overfitting 문제에 취약할 수 있음. 훈련 데이터에 맞춰서 너무 세분화된 결정경계가 만들어 질 수 있음. 이 경우에 일반화 능력이 떨어질 수 있음.
- 3) 간단한 규제 방법은 결정 트리의 최대 깊이 (max_depth)를 두어 제어 가능. 너무 많은 질문을 하

지 못하게 함.

15. (실습) Overfitting

왼쪽 그림: 규제가 없음. 훈련 데이터에 지나치게 overfitting 됨.

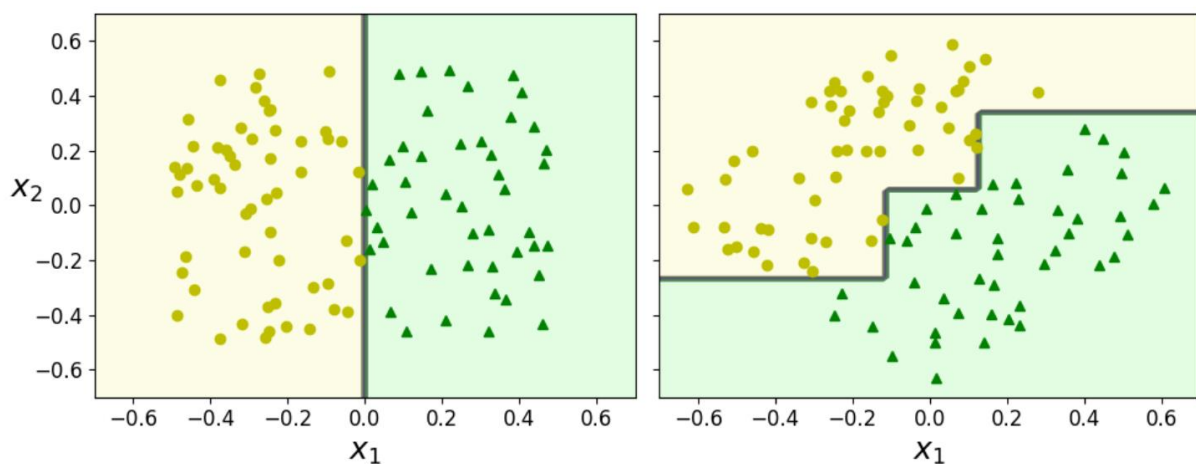
오른쪽 그림: 규제가 있음. leaf node가 가져야할 최소 sample 수=4 지정. Overfitting 문제없음



16. (실습) Sensitivity to training set

1) 데이터를 45도만큼 회전한 경우 (오른쪽)

2) 불필요하게 구불구불해져서 두 결정 트리 모두 훈련 세트를 완벽하게 학습하지만 오른쪽 모델은 잘 일반화가 되지 않음.

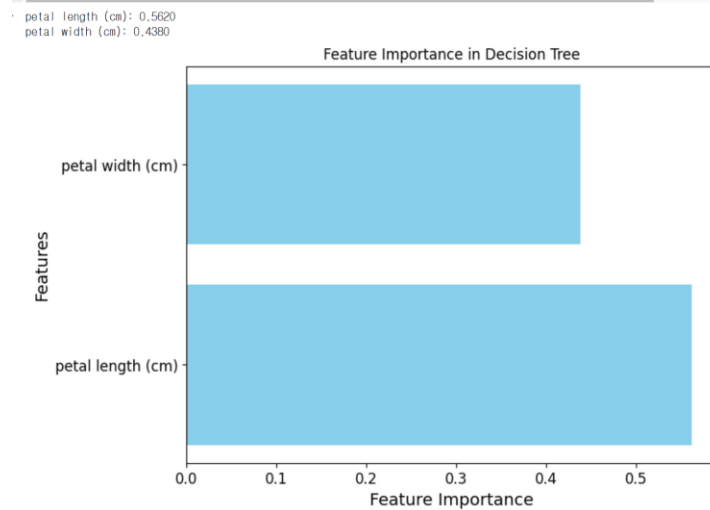


Decision tree에서 특징 중요도 확인 방법 (실습)

- 분류 시에 어떤 특징이 가장 큰 영향을 끼쳤는지는 매우 중요한 정보이다
- 트리에서 분할기준으로 선택된 횟수와 지니 불순도 감소와 같은 측정값 기반
- 특정 특징이 분할에 자주 사용되거나, 해당 특징으로 인하여 지니 불순도가 크게 줄어든다면, 그 특징은 중요도가 높게 평가됨
- 각 특징의 중요도 값은 0과 1 사이로 scaling되어 총합은 1
- 중요도가 높을수록 해당 특징이 모델의 예측 성능에 더 큰 영향을 끼침

코드:

```
# 특징 중요도 출력
importances = tree_clf.feature_importances_
```



Decision tree에서 overfitting 방지하는 법

[DecisionTreeClassifier — scikit-learn 1.5.2 documentation](#)

여러가지가 있지만 아래 3가지가 많이 쓰인다

1. 트리의 최대 깊이 제한.

- 너무 깊은 tree는 데이터를 과도하게 분할하여 학습 데이터에 over-fitting될 수 있음

```
DecisionTreeClassifier(max_depth=5)
```

2. 최소 샘플 수 설정

- min_samples_split: 노드를 분할하기 위해 필요한 최소 샘플 수를 설정합니다. 값이 클수록 트리가 덜 복잡해집니다.
- min_samples_leaf: 리프 노드(종료 노드)에 있어야 하는 최소 샘플 수를 설정합니다. 값이 클수록 트리가 간소화됩니다.

```
DecisionTreeClassifier(min_samples_split=10, min_samples_leaf=5)
```

3. 최대 leaf node 수 제한

트리의 리프 노드 개수를 제한하여 과도한 분기를 방지합니다.

```
DecisionTreeClassifier(max_leaf_nodes=10)
```

Decision tree의 장, 단점

- 1) 대표적인 decision tree의 장점: 이해가 쉽고 해석 가능, data 정규화와 같은 data 전처리가 필요 없음, white box model (설명가능한 모델)
- 2) 대표적인 decision tree의 단점: generalization 문제, **overfitting** 가능성