**Chapter 6. Decision Trees**

1. **Training and Visualizing a Decision Tree**
2. **Making Predictions**
3. **Estimating Class Probabilities**
4. **The CART Training Algorithm**
5. **Computational Complexity**
6. **Gini Impurity or Entropy?**
7. **Regularization Hyperparameters**
8. **Regression**
9. **Instability**

SVM과 마찬가지로 의사결정 트리(Decision Tree)는 분류 작업과 회귀 작업, 심지어는 다중 출력 작업까지 수행 할 수 있는 다용도의 기계 학습 알고리즘입니다. 또한 복잡한 데이터 세트를 피팅 할 수 있는 매우 강력한 알고리즘

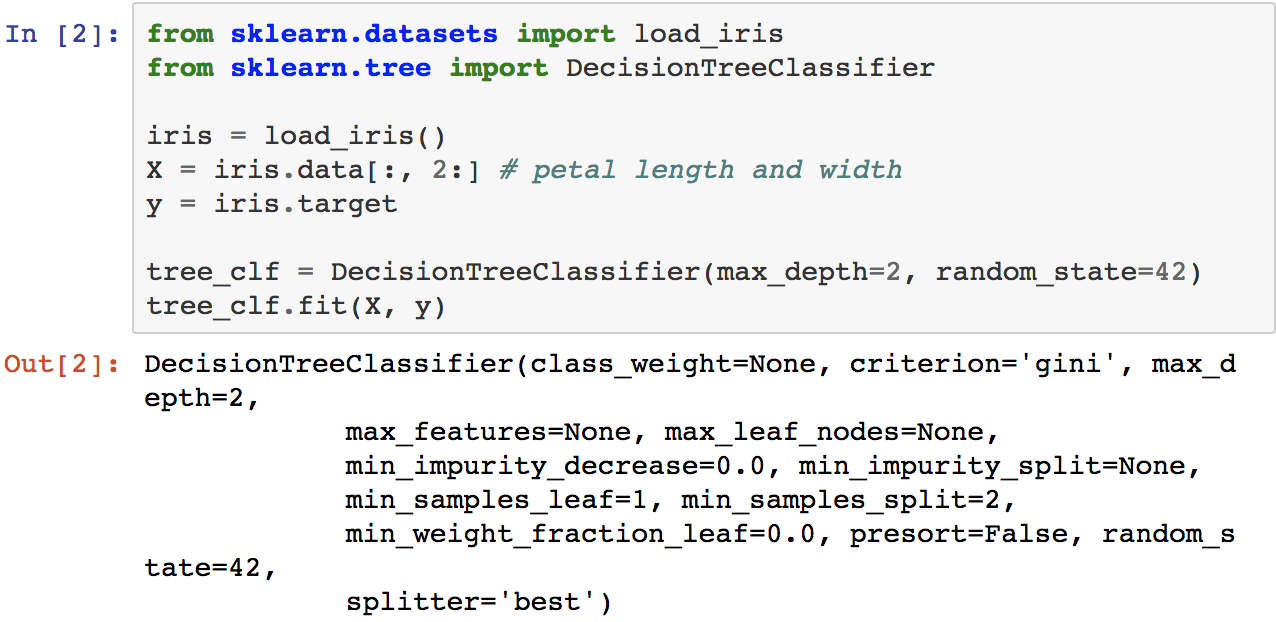
의사결정 트리는 오늘날 사용할 수 있는 가장 강력한 기계 학습 알고리즘 중 하나 인 Random Forest의 기본 구성 요소이기도 합니다.

* Decision Tree를 훈련, 시각화 및 예측하는 방법을 논의
* Scikit-Learn에서 사용한 CART 훈련 알고리즘을 살펴보고 트리를 정규화 하고 회귀 작업에 사용하는 방법에 대해 설명
* Decision Tree의 몇 가지 제한 사항에 대해 설명

1. **Training and Visualizing a Decision Tree**

의사 결정 트리(Decision tree)를 이해하기 위해 예측 트리를 작성하고 예측을 하는 방법을 살펴 보겠습니다.

다음 코드는 붓꽃(Iris) 데이터 셋을 DecisionTreeClassifier로 트레이닝합니다.



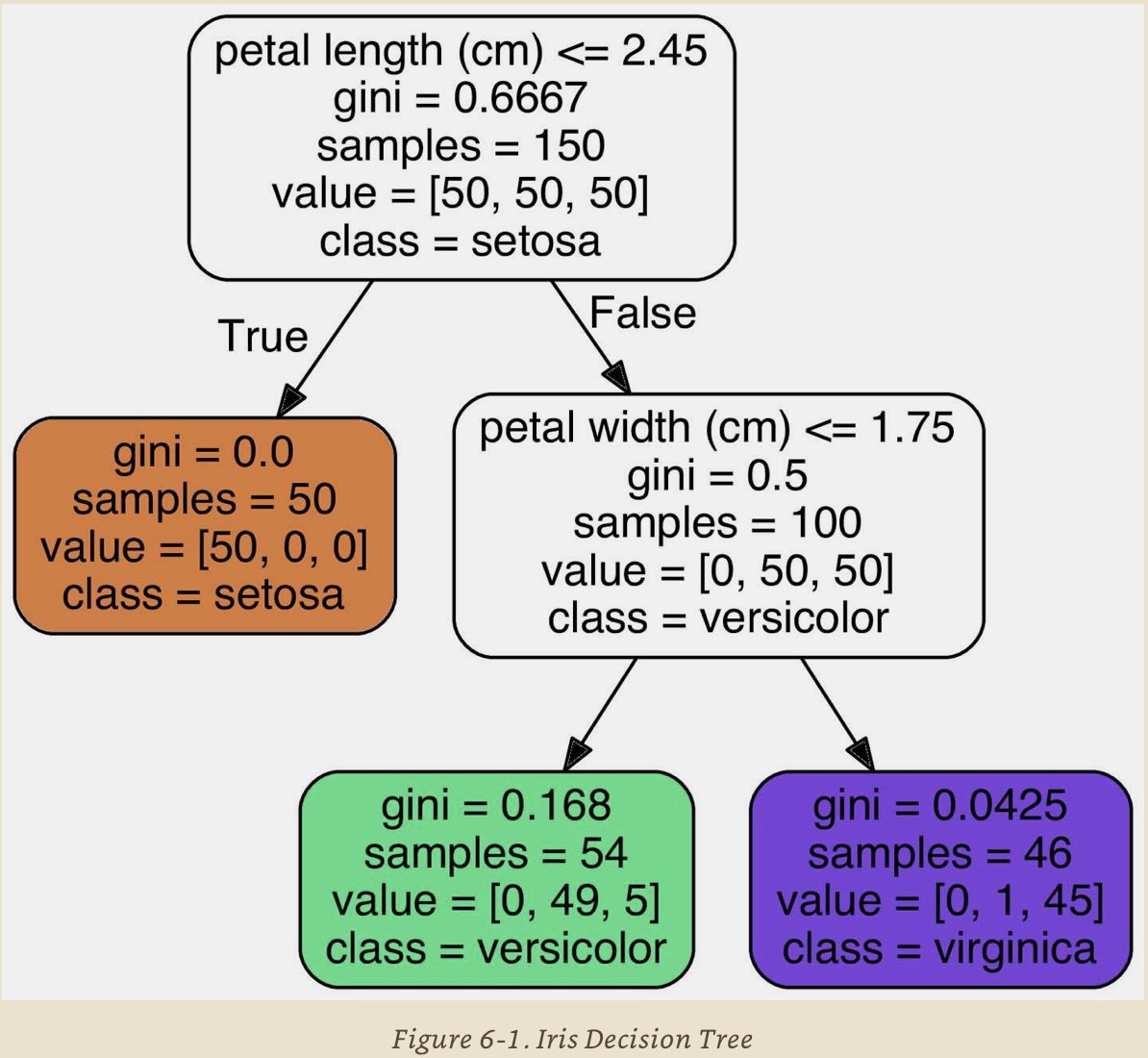
먼저 export\_graphviz () 메서드를 사용하여 iris\_tree.dot라는 그래프 정의 파일을 출력하여 훈련 된 Decision Tree를 **시각화** 할 수 있습니다.



그런 다음이 .dot 파일을 graphviz 패키지의 dot command-line tool을 사용하여 PDF 또는 PNG와 같은 다양한 형식으로 변환 할 수 있습니다.



이 명령 줄은 .dot 파일을 .png 이미지 파일로 변환합니다. 첫 번째 Decision Tree는 그림 6-1과 같습니다.



1. **Making Predictions**

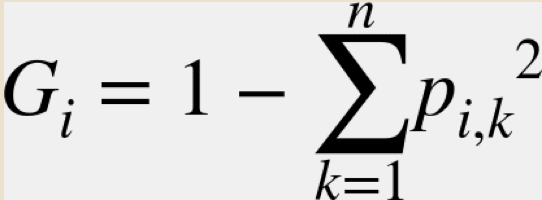
그림 6-1에 표시된 트리가 어떻게 예측되는지 봅니다.

붓꽃을 찾아 분류하고 싶다고 합시다.

* 루트 노드(깊이 0, 상단)에서 시작합니다. 이 노드는 꽃잎의 길이가 2.45cm보다 작은 지를 묻습니다. 그럴 경우 루트의 왼쪽 자식 노드(child)(깊이 1, 왼쪽)로 이동합니다. 이 경우를 잎 노드(leaf node)라고 합니다. 자식 노드가 없어서 더이상 질문을 하지 않습니다. 노드의 클래스를 보면 꽃이 아이리스 - 세토사(Iris setosa)라고 예측하였다.
* 이번에 꽃잎 길이는 2.45cm보다 큽니다. 루트의 오른쪽 자식 노드 (깊이 1, 오른쪽)로 내려와야 하므로 다른 질문을 합니다. 꽃잎 너비가 1.75cm보다 작습니까? 그렇다면 꽃은 아이리스 - 베르시컬러(Iris versicolor)(깊이 2, 왼쪽)일 가능성이 큽니다. 그렇지 않다면 아이리스 - 버지니카(Iris virginica)(깊이 2, 오른쪽) 일 가능성이 큽니다.

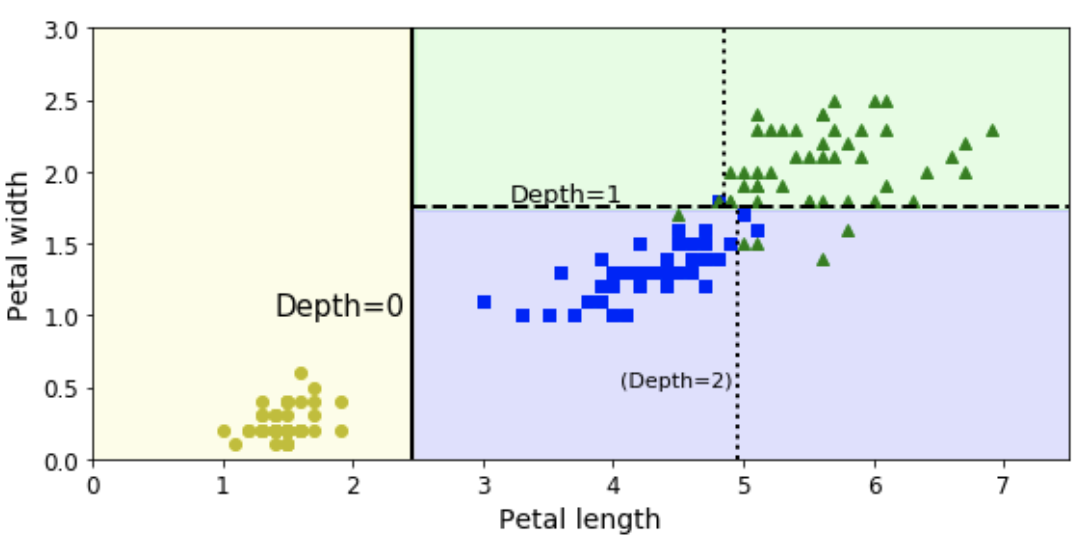
Decision tree의 많은 특성 중 하나는 데이터 준비가 거의 필요 없다는 것입니다. 특히, feature scaling이나 centering을 전혀 필요로 하지 않습니다.

* 노드의 샘플 속성은 적용되는 트레이닝 인스턴스의 수를 계산합니다.
* 노드의 value 속성은 이 노드가 적용되는 각 클래스의 트레이닝 인스턴스 수를 나타냅니다.
* 마지막으로 노드의 gini 속성은 해당 불순물을 측정합니다.



pi, k는 i 번째 노드의 트레이닝 인스턴스 중 클래스 k 인스턴스의 비율입니다.

Scikit-Learn은 이진 트리 만 생성하는 CART 알고리즘을 사용합니다. nonleaf nodes는 항상 두 개의 자식을 가집니다 (예 : 질문에 예 / 아니오 응답 만 있음). 그러나 ID3과 같은 다른 알고리즘은 세 개 이상의 자식이 있는 노드로 Decision Tree를 생성 할 수 있습니다.

 이 그림 Decision Tree의 의사 결정 경계를 보여줍니다.

* 두꺼운 수직선은 루트 노드 (깊이 0)의 결정 경계를 나타냅니다. 꽃잎 길이=2.45cm로 분할
* 오른쪽 영역은 불순물이므로 오른쪽 노드(깊이 1)은 꽃잎 너비=1.75cm 로 분할(가로 점선)
* max\_depth가 2로 설정 되었으므로 Decision Tree가 바로 중지됩니다.
* max\_depth를 3으로 설정하면 깊이 2 노드는 각각 다른 결정 경계를 추가합니다(세로 점선)

알 수 있듯이 Decision Tree는 매우 직관적이며 의사 결정은 쉽게 해석 할 수 있습니다. 이러한 모델을 종종 화이트 박스 모델이라고 합니다. Decision Tree는 필요에 따라 수동으로 적용 할 수도 있는 훌륭하고 간단한 분류 규칙을 제공합니다.

반대로, Random Forest 또는 Neural Networks는 일반적으로 블랙 박스 모델로 간주됩니다.

이것들은 훌륭한 예측을 하고 예측을 하기 위해 수행한 계산을 쉽게 확인할 수 있습니다. 그럼에도 불구하고 예측이 왜 이루어졌는지를 간단히 설명하는 것은 일반적으로 어렵습니다.

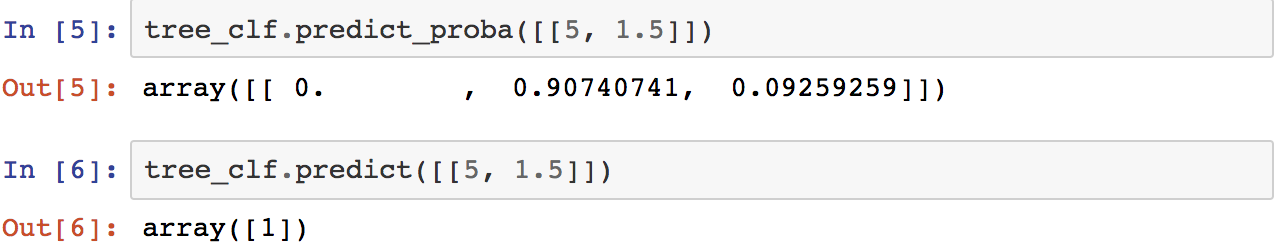
1. **Estimating Class Probabilities**

Decision Tree는 인스턴스가 특정 클래스 k에 속할 확률을 평가할 수도 있습니다. 먼저 이 인스턴스에 대한 리프 노드를 찾기 위해 트리를 탐색 한 다음이 노드에서 클래스 k의 트레이닝 인스턴스 비율을 반환합니다.

꽃잎이 길이 5cm, 너비 1.5cm 인 꽃을 발견했다고 가정합니다.

Decision Tree는 Iris-Setosa 0 %, Iris-Versicolor 90.7 %, Iris-Virginica 9.3%로 나타난다.

물론 클래스를 예측하면 Iris-Versicolor (클래스 1)가 가장 높은 확률을 가지므로 출력된다.



꽃잎의 길이가 6cm, 너비가 1.5cm 인 경우, 예상 확률은 그림 6-2의 하단 오른쪽 직사각형의 다른 곳과 동일합니다

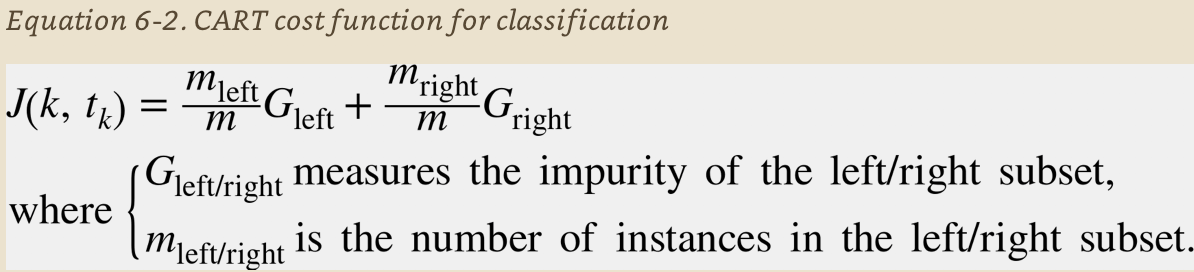
1. **The CART Training Algorithm**

Scikit-Learn은 분류 및 회귀 트리(CART) 알고리즘을 사용하여 Decision tree를 트레이닝 시킵니다.

알고리즘은 먼저 단일 feature k와 임계 값 tk (예 : "꽃잎 길이 ≤2.45cm")를 사용하여 트레이닝 세트를 두 개의 하위 세트로 분할합니다.

크기에 따라 가중치를 생성하는 쌍 (k, tk)을 찾습니다.

알고리즘이 최소화 하려고 하는 비용 함수는 식 6-2에 의해 주어진다.



훈련 세트를 두 개로 성공적으로 분할하면 동일한 논리를 사용하여 하위 집합을 계속해서 재귀적으로 분할합니다. 최대 깊이(max\_depth 하이퍼 매개 변수로 정의 됨)에 도달하거나 불순물을 줄이는 분할을 찾을 수 없는 경우 재귀를 중지합니다.

하이퍼 매개 변수는 다음과 같은 추가 중지 조건 을 제어합니다.

min\_samples\_split, min\_samples\_leaf, min\_weight\_fraction\_leaf, max\_leaf\_nodes

[CART 알고리즘은 최상위 레벨에서 최적의 분할을 탐독한 다음 각 레벨에서 프로세스를 반복합니다. 분할로 인해 가능한 한 가장 낮은 수준의 불순물이 생성되는지 여부는 확인하지 않습니다. 합리적으로 좋은 솔루션을 생성하지만 최적의 솔루션이라고 보장 할 수는 없습니다.]

최적의 트리를 찾는 것은 NP-Complete 문제로 알려져 있습니다 : O (exp (m))의 시간이 필요하므로 상당히 작은 트레이닝 세트에서도 문제를 다루기가 어렵습니다. 이것이 우리가 "합리적으로 좋은"해결책을 찾기 위해 정착해야하는 이유입니다.

1. **Computational Complexity**

예측을 하려면 Decision tree를 루트에서 리프로 이동해야 합니다. Decision tree는 탐색하려면 대략 O (log2 (m))개의 노드를 통과해야 합니다. 각 노드는 하나의 지형지물의 값을 확인하기만 하면 되므로 지형지물의 수에 관계없이 전반적인 예측 복잡성은 O (log2 (m))에 불과합니다. 따라서 대규모 트레이닝 세트를 다룰 때에도 예측은 매우 빠릅니다.

그러나, 트레이닝 알고리즘은 각 노드의 모든 샘플에서 모든 기능을 비교합니다.

이것은 O (n × m log (m))의 훈련 복잡성을 초래한다. 작은 훈련 세트 (수천 개 미만의 인스턴스)의 경우, Scikit-Learn은 데이터를 사전 정렬(presort = True로 설정)하여 트레이닝 속도를 높일 수 있지만 대규모 트레이닝 세트의 경우 트레이닝 속도가 상당히 느려집니다.

1. **Gini Impurity or Entropy?**

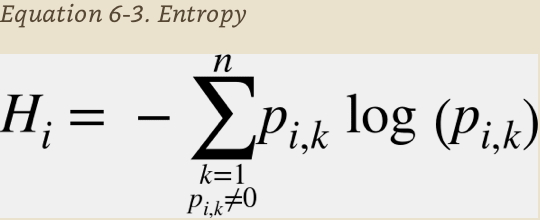
기본적으로 Gini 불순물 측정 값이 사용되지만 조건 하이퍼 매개 변수를 "엔트로피"로 설정하여 엔트로피 불순물 측정 값을 선택할 수 있습니다.

엔트로피의 개념은 분자 역학의 척도로서 열역학에서 유래했습니다(분자가 잘 정렬 되어있을 때 엔트로피는 0에 접근합니다.)

모든 메시지가 동일 할 때 엔트로피는 0입니다.(세트에 한 클래스만 있으면 0이다)

기계 학습에서는 불순물 측정으로 자주 사용됩니다.

식 6-3은 i 번째 노드의 엔트로피의 정의를 보여줍니다.



언제Gini 불순물이나 엔트로피를 사용해야하나?

대부분의 경우 큰 차이를 만들지 않는다는 것입니다. 비슷한 Tree로 이어집니다. Gini 불순물은 계산 속도가 약간 빠르기 때문에 기본값으로 되어있다.

그러나 서로 다른 점은 Gini불순물은 트리의 자체 분기에서 가장 빈번한 클래스를 고립시키는 경향이 있고 엔트로피는 좀 더 균형 잡힌 Tree를 생성하는 경향이 있습니다.

확률이 적은 일이 얼마나 일어나는지

1. **Regularization Hyperparameters**

Decision tree는 트레이닝 데이터에 대해 거의 가정하지 않습니다

제한없이 두면 트리 구조가 트레이닝 데이터에 적응하여 매우 밀접하게 적용되며 over-fitting가능성이 높습니다. 그러한 모델은 nonparametric model이라 하는데 매개 변수가 없는 것이 아니라 매개 변수의 수가 트레이닝 이전에 결정되지 않았기 때문이다.

대조적으로 선형 모델과 같은 매개 변수 모델은 미리 결정된 수의 매개 변수를 가지므로 자유도가 제한되어 over-fitting의 위험을 줄입니다 (그러나 under-fitting의 위험이 증가합니다).

트레이닝 데이터의 over-fitting을 피하려면 Decision tree의 트레이닝 중 정규화를 해야합니다.

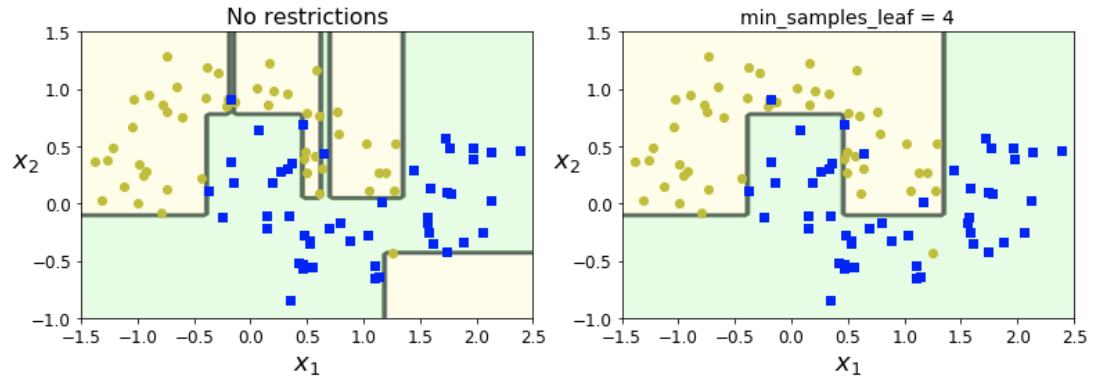
정규화 하이퍼 매개 변수는 사용 된 알고리즘에 따라 다르지만 일반적으로 Decision tree의 최대 깊이를 제한 할 수 있습니다. Scikit-Learn에서 max\_depth를 줄이면 모델을 정규화 하여 over-fitting의 위험을 줄일 수 있습니다.

DecisionTreeClassifier 클래스에는 Decision Tree의 모양을 비슷하게 제한하는 몇 가지 매개 변수가 있습니다.

* min\_samples\_split (노드가 분할되기 전에 가지고 있어야하는 샘플의 최소 수)
* min\_samples\_leaf (리프 노드에 있어야하는 샘플의 최소 수)
* min\_weight\_fraction\_leaf (min\_samples\_leaf와 같지만 가중치가 부여된 인스턴스의 총 개수의 부분을 표현)
* max\_leaf\_nodes (최대 리프 노드 수)
* max\_features (각 노드에서 분할을 위해 평가되는 최대 피처 수)

min\_ \* hyperparameters를 늘리거나 max\_ \* hyperparameters를 줄이면 모델이 정규화 됩니다.

다른 알고리즘은 먼저 Decision tree를 제한없이 트레이닝 한 다음 불필요한 노드를 제거 (삭제)함으로써 작동합니다.



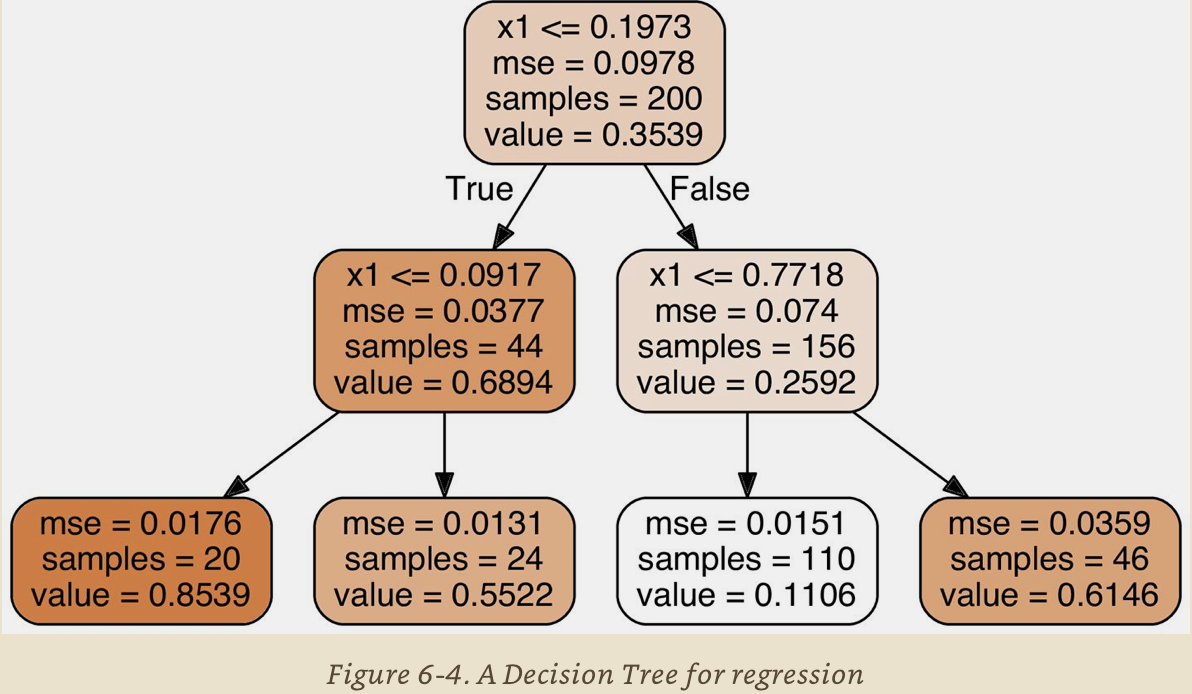
위 그림은 달 데이터 세트(5 장에서 소개)에서 훈련 된 2 개의 Decision tree를 보여줍니다. 왼쪽에서 Decision tree는 기본 하이퍼 매개 변수로 트레이닝 되며,

오른쪽에서 Decision tree는 min\_samples\_leaf = 4로 트레이닝 됩니다.

왼쪽 모델이 over-fitting이고 오른쪽 모델이 아마 일반화가 더 잘 될 것입니다

1. **Regression**

Decision tree는 회귀 작업을 수행 할 수도 있습니다. Scikit-Learn의 DecisionTreeRegressor 클래스를 사용하여 회귀 트리를 작성하고 max\_depth = 2 인 노이즈가 있는 quadratic 데이터 세트를 트레이닝시켜본다.

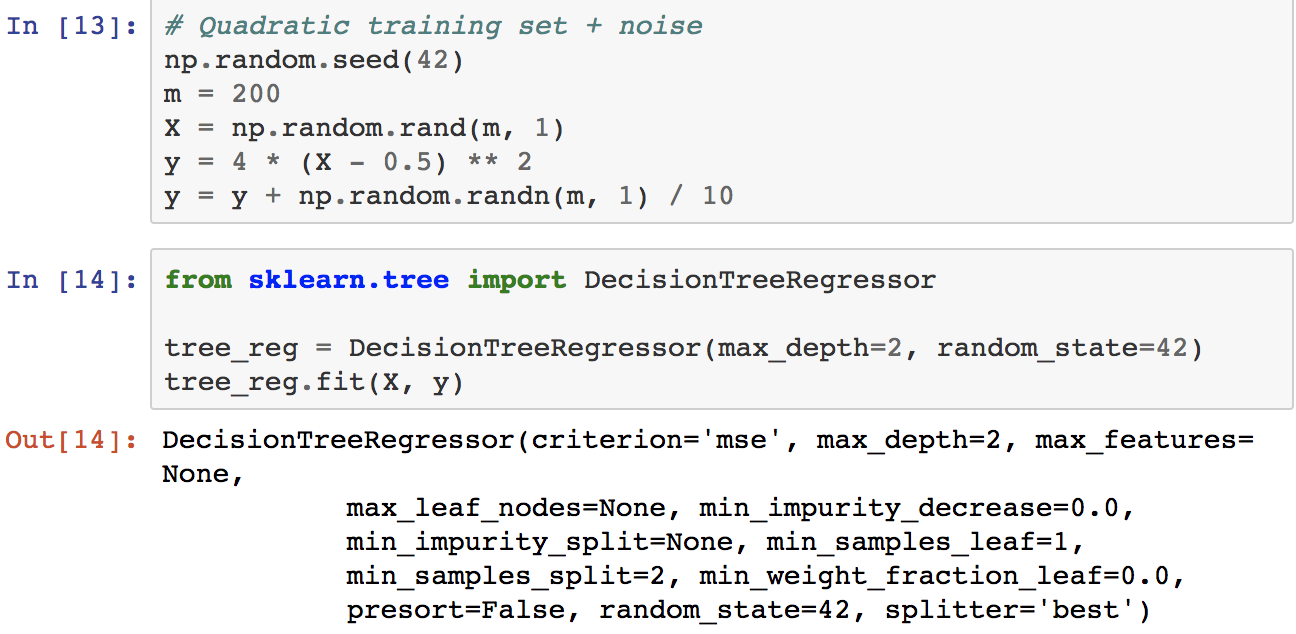


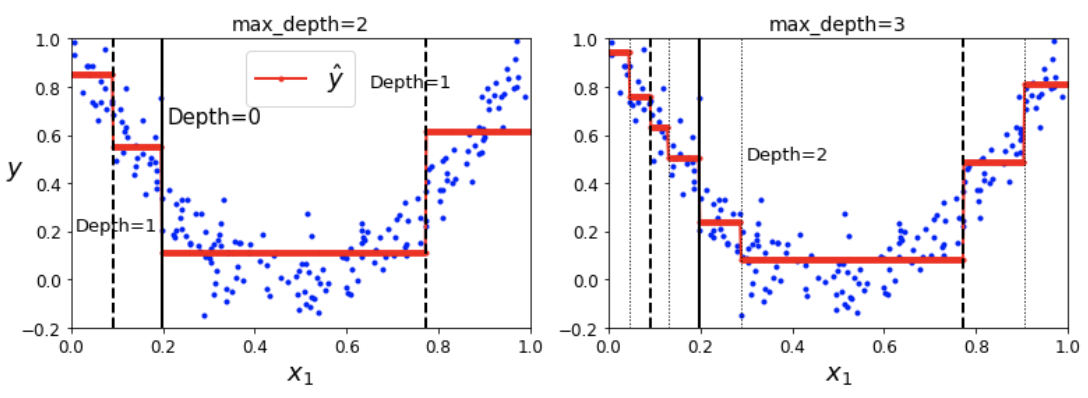
이전에 작성한 분류 트리와 매우 유사합니다. 주요 차이점은 각 노드의 클래스를 예측하는 대신 값을 예측한다는 것입니다.

예를 들어, x1 = 0.6 인 새 인스턴스에 대한 예측을 하고 싶다고 가정 해보십시오.

루트에서 시작하여 트리를 탐색하면 value = 0.1106을 예측하는 리프 노드에 도달합니다. 이 예측은 리프 노드와 연관된 110 개의 트레이닝 인스턴스의 평균 목표 값입니다.

이 모델의 예측은 아래 그림의 왼쪽에 표시된다.





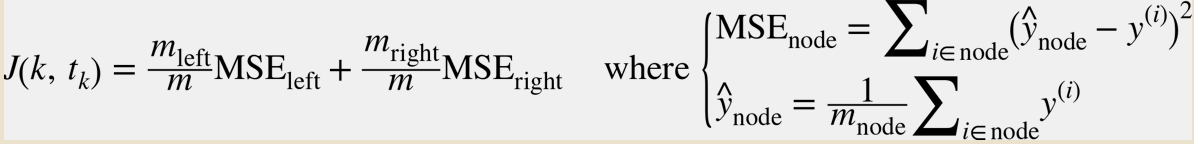
max\_depth = 3으로 설정하면 오른쪽에 표시된 예측 값을 얻을 수 있습니다.

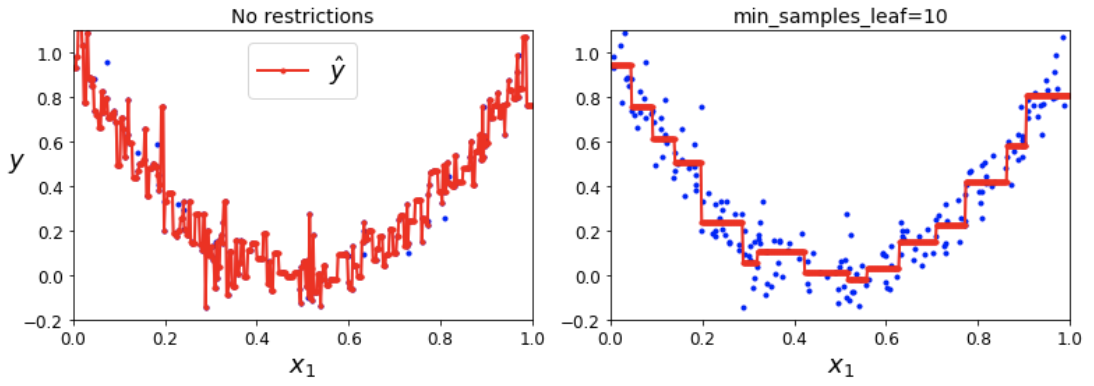
각 영역의 예상 값이 해당 영역의 인스턴스의 평균 목표 값인지 주의

알고리즘은 각 영역을 대부분의 트레이닝 인스턴스를 예측된 값에 최대한 가깝게 만드는 방식으로 분할합니다.

CART 알고리즘은 불순물을 최소화하는 방식으로 트레이닝 세트를 분할하는 대신, MSE를 최소화하는 방식으로 트레이닝 세트를 분리하려고 한다는 점을 제외하고는 이전과 거의 동일하게 작동합니다.

식 6-4는 알고리즘이 최소화하려고 하는 비용 함수를 보여줍니다.





분류 작업과 마찬가지로 Decision tree는 회귀 작업을 처리할 때 over-fitting되기 쉽다.

정규화(기본 하이퍼 매개 변수 사용)가 없으면 위 그림의 왼쪽과 같이 예측값이 표시됩니다.

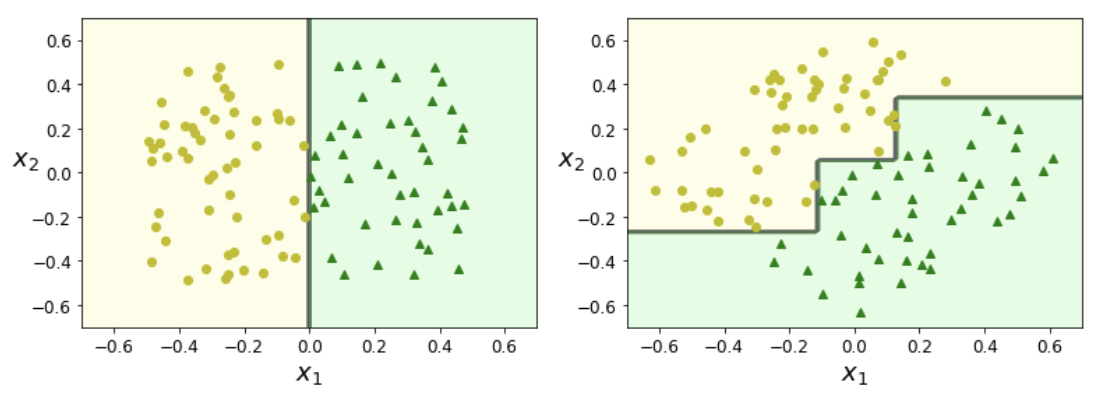
분명히 트레이닝 세트에 너무 심하게 over-fitting합니다.

min\_samples\_leaf = 10으로 설정하면 그림의 오른쪽처럼 훨씬 합리적인 모델이 됩니다.

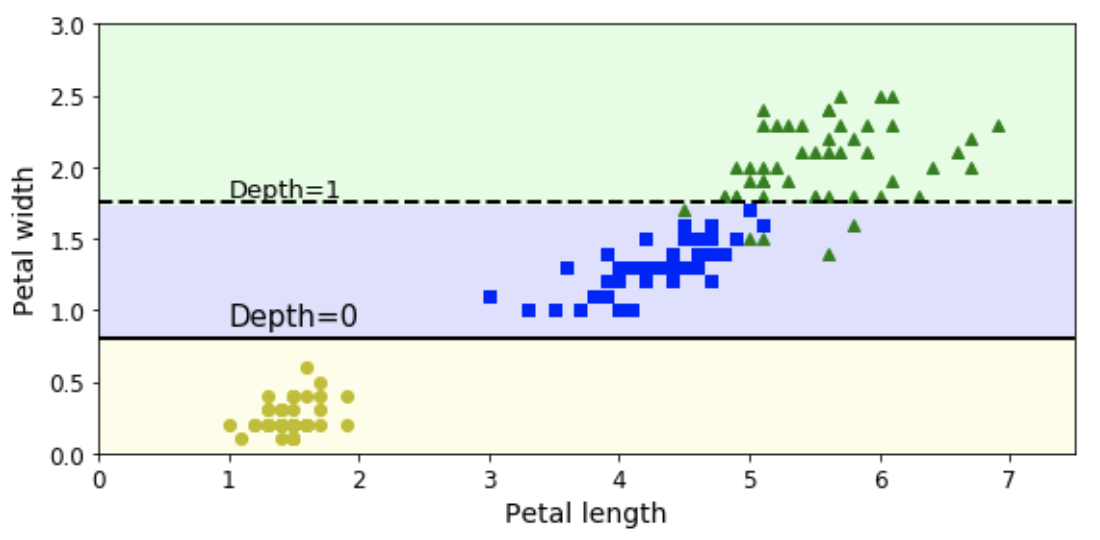
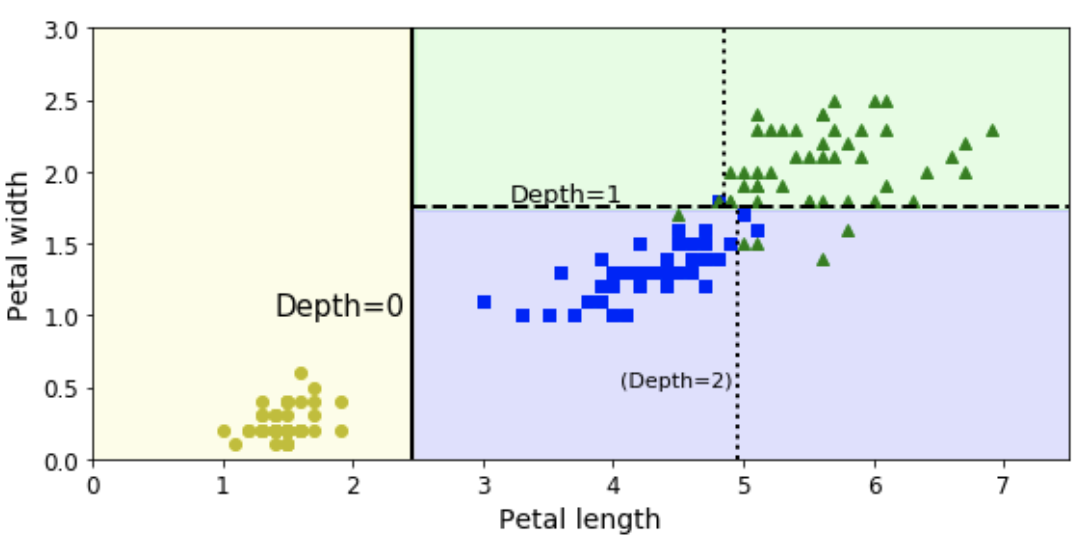
1. **Instability**

다행히도 의사 결정 나무는 이해하기 쉽고 해석하기 쉽고 사용하기 쉽고 다양하며 강력합니다. 그러나 몇 가지 제한 사항이 있습니다.(불안정)

* **모든 분할은 축에 수직적으로 나타내는 직교 결정 경계를 추구한다**.  
  이는 의사 결정 집합 순환에 민감합니다.  
  예를 들어, 아래 그림은 단순한 선형 분리형 데이터 세트를 보여줍니다. 왼쪽에서 Decision Tree는 쉽게 분리 할 수 ​​있지만 오른쪽에서는 데이터 세트가 45 ° 회전 된 후 결정 경계가 불필요하게 복잡하게 보입니다.  
  Decision tree가 모두 훈련 세트에 완벽하게 들어 맞았지만 오른쪽 모델이 잘 일반화되지는 않을 것입니다. 이 문제를 제한하는 한 가지 방법은 PCA를 사용하는 것입니다



* **훈련 데이터의 작은 변화에 매우 민감하다는 것입니다.**예를 들어 Iris 꽃 트레이닝 세트 (꽃잎이 길이 4.8cm, 너비가 1.8cm 인 트레이닝 세트)에서 가장 넓은 iris-versicolor를 제거하고 새로운 Decision tree를 트레이닝하면 두 번째 그림과 같은 모델을 얻을 수 있습니다. 보시다시피, 첫 번째 그림과 매우 다릅니다.   
  실제로, Scikit-Learn에 의해 사용 된 훈련 알고리즘이 확률 적이므로 (random\_state hyperparameter를 설정하지 않는 한) 동일한 훈련 데이터에서도 매우 다른 모델을 얻을 수 있습니다.



Random Forests는 다음 장에서 볼 수 있듯이 많은 트리에 대한 예측을 평균하여 이러한 불안정성을 제한 할 수 있습니다.