Chap6

Decision Trees

Written by Jin Kim

177 Decision Trees 란 무엇인가

- SVM처럼 Classifier, Regressor로 모두 쓰일 수 있는 Machine Learning algorithms 임.
- Decision Trees는 Random Forest의 필수요소임.
  - Random Forest는 현존하는 가장 강력한 Machine Learning algorithms들중 하나임.
- 현 챕터에서 나올 내용들
- 1. Decision Tree에 대해
  - Train
  - Visualization
  - Prediction
- 2. the CART training algorithm
- 3. Tree의 정규화, Regression Task에 이용
- 4. Decision Trees의 한계점

177~179 Training and Visualizing a Decision Tree

- Iris dataset을 이용하여 해볼것임.
- Colab – 트레인, 시각화 하는 명령어가 소개 돼 있음

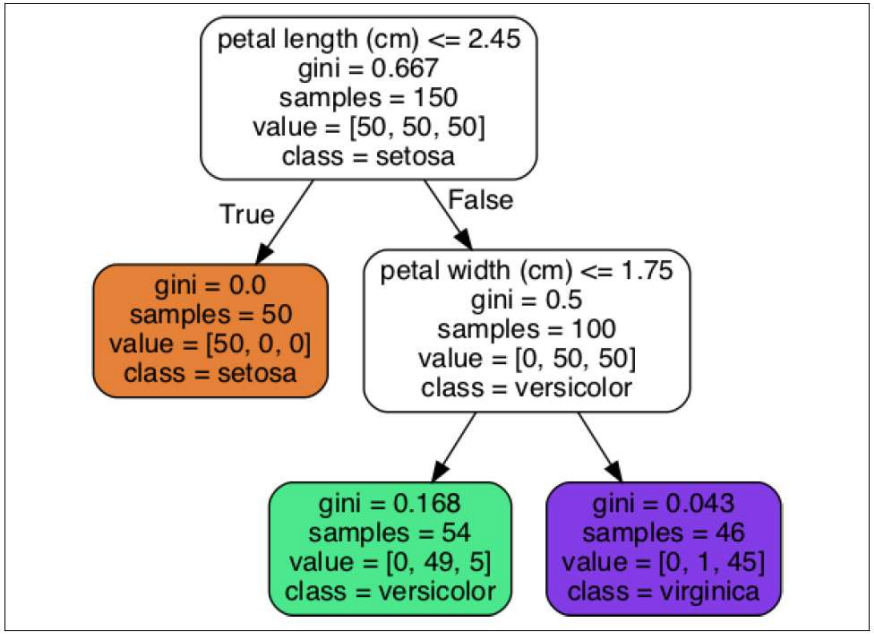


Figure 6-1. Iris Decision Tree

첫번째 Decision Tree를 시각화하면 좌측과 같이 될 것임.
Decision Tree를 분류기로 사용하기 위해 *root node* 에서 시작함.(꼭대기)
이 노드(depth=0)에서는 petal length가 2.45cm보다 작냐 크냐를 판별
만약 작으면 좌측 노드(depth=1,left)로 감. 현재 상황에서는 이를 *leaf node* 라고 부름. Children nodes 가 없음. 즉 아무런 질문도 하지 않음. 여기 오면 setosa로 분류됨
2.45cm보다 크면 우측(depth=1,right)으로 분류됨. Width가 1.75cm보다 작냐 크냐에서 다시 갈림. 작다면(depth=2,left) versicolor로, 아니면(depth=2,right) virginica로 분류됨.
또다른 특징 : Decision Tree는 Data preparatio이 작은것들중 하나임 : feature scaling이 필요없음.

180 node의 parameter들

- Node의 samples은 몇개의 instance들이 분류됐는지 갯수를 셈.
  - 예)100개의 training instance들의 petal length가 2.45cm를 넘고, 그중 54개의 petal width가 1.75cm보다 작음.
- Node의 value는 해당 노드에서 몇개의 instance들이 어떤 클래스로 분류됐는지 갯수를 셈
  - 우측 하단의 노드는 0개의 Iris-Setosa, 1 개의 Iris-Versicolor, 45 개의 Iris-Virginica만큼 분류함.
- Node의 gini는 impurity(불순도)를 측정함
  - Gini값이 0이면 pure한 상태
  - Detph가 1인 노드에서 Iris-Setosa로만 분류한다면 gini score은 0이 됨

180 node의 parameter들

Equation 6-1. Gini impurity

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

Gini score계산방법은 위와 같음.
만약 depth-2 left 노드에서 gini score이 $1 - (0/54)^2 - (49/54)^2 - (5/54)^2 \approx 0.168$ 와 같이 나올 수 있음.

$p_{i,k}$ 는 i번째 노드에서 k클래스가 있게 되는 비율임.

CART algorithm은 *binary trees* 만을 만들어냄. 무슨말이냐면 nonleaf node들은 항상 2개의 children 을 만들어냄.
다른 algorithm (ID3 등) 들은 2개보다 많은 children을 만들어 낼 수 있음.

180~181 Decision Tree's decision boundaries

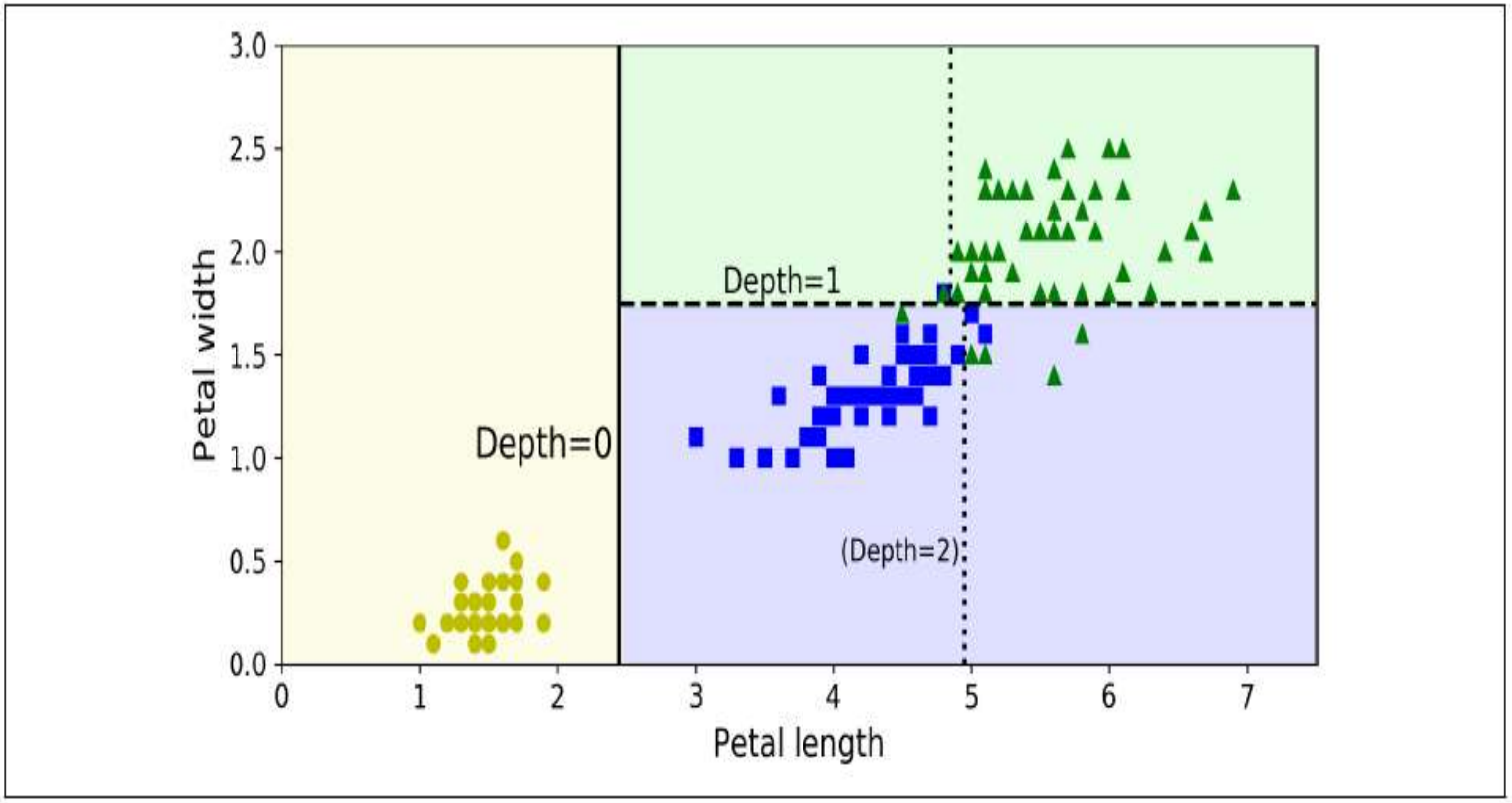


Figure 6-2. Decision Tree decision boundaries

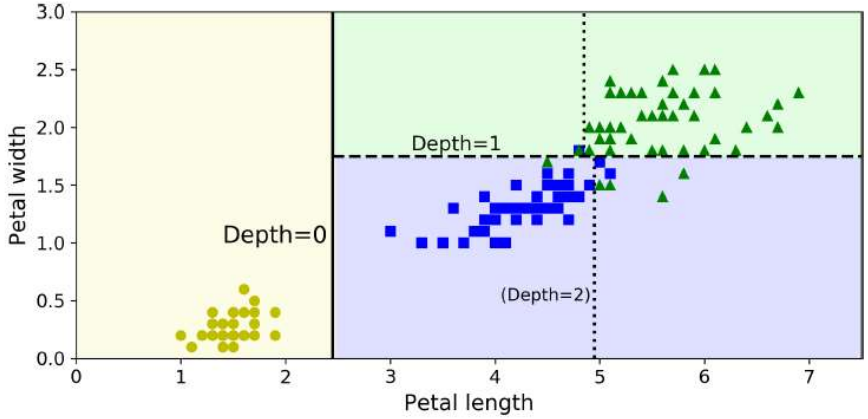
굵은 수직선은 root node의 decision boundary를 나타냄.(Depth = 0):petal length = 2.5cm
왼쪽면은 pure한 상태(Iris-Setosa만 있음)기 때문에, 더이상 쪼개질 수 없음.
Depth=1 노드는 Petal width로 한번 더 쪼개짐.(Dashed line)
max_depth값이 2기 때문에, 여기서 멈췄음
만약 max_depth값이 3이었으면 점선이 하나 더 생길수도 있음.

181 Model Interpretation: White Box Versus Black Box

- Decision Trees 들은 직관적임.
- 이런 직관적인 형태를 *white box models* 라 부름.
- 반면 Random Forests 나 neural networks 는 보통 *black box models*라 불림.
  - 예측을 잘 하며 어떤 계산을 통해 예측을 한지 알 수 있음.
  - 그러나 왜 그런 예측이 됐는지는 설명하기 어려움.(예. 사진속 사람찾기 neural network를 통해 했을경우 어떤것이 그 결과에 기여했는지 알기 어려움)

181 Estimating Class Probabilities

- Decision Tree는 instance 가 특정 클래스 k에 속할 가능성도 측정 가능함.
  - Ex. Petal 길이 5cm, 너비 1.5cm 일때 depth-2 left node 가 가장 일치하는 노드임.
  - 결과 : 0% for Iris-Setosa (0/54), 90.7% for Iris-Versicolor (49/54), and 9.3% for Iris-Virginica (5/54)
  - 클래스를 물으면 가장 높은 가능성인 Iris-Versicolor로 예측될것임.



182 The CART Training Algorithm =*Classification And Regression Tree* (CART)

- 알고리즘 : 트레이닝셋을 a single feature *k* 과 a threshold *tk* 를 이용해 나눔.
  - e.g., “petal length ≤ 2.45 cm”
  - k와 tk 쌍으로 the purest한 값들을 찾음.
  - Cost function은 아래와 같음

Equation 6-2. CART cost function for classification

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

where $\begin{cases} G_{\text{left/right}} \text{ 는 좌/우측 부분집합의 impurity(불순도)를 측정함} \\ m_{\text{left/right}} \text{ 는 좌/우측 부분집합의 instance의 수임} \end{cases}$

182~183 The CART Training Algorithm =*Classification And Regression Tree* (CART)

- 위의 방법으로 트레이닝 세트를 2개로 나누는데 성공했다면, 같은 방법으로 부분집합을 계속해서 2개로 나눈다. max_depth hyperparameter 에 의해 정해진 최대 depth에 도달하거나, impurity를 줄일 수 있는 방법이 없으면 멈춤.
- min_samples_split, min_samples_leaf, min_weight_fraction_leaf, and max_leaf_nodes 등의 hyperparameter들도 멈추는 상태에 관여하는것들임.
- CART는 Greedy algorithm이기도 함. 이것은 괜찮은(reasonably good)솔루션을 제공하기도 하지만, 최적의 솔루션을 확신하지는 못함.

183 Computational Complexity

- Decision Tree의 Computation al Complexity는 $O(log_2(m))$ 를 따름.
- -> Prediction은 큰 데이터셋에서도 매우 빠르다.

- 그러나, 매 노드에서 전체 featur를 비교한다. $O(n \times m\ log(m))$
- 적은 training set(만개 미만의 instance)에서는 scikit-learn기능 중 presorting을 사용하여 속도를 올릴 수 있음. 그러나 많은 데이터셋에서는 굉장히 느려 질 수 있다.

183~184 Gini Impurity or Entropy

- 기본적으로 Gini Impurity measure가 사용된다.
  - criterion hyperparameter을 'entropy'로 변경하여 *entropy* impurity measure 도 사용 가능.
- Entropy의 개념은 thermodynamics(열역학)에서 왔음(molecular disorder = 분자의 장애(?))
  - 분자가 still and well ordered(정지되고 안정된 상태)에서는 Entropy는 거의 0임.
  - 그 후에 다른 도메인으로 퍼지게 됐음.
  - Machine learning에서는 impurity measure에 사용됨. 하나의 class인 instance만을 가지고 있을 경우 Entropy가 0이 됨.
  - the depth-2 left node의 Entropy :

$$-\frac{49}{54}\log_2\left(\frac{49}{54}\right) - \frac{5}{54}\log_2\left(\frac{5}{54}\right) \approx 0.445$$

Equation 6-3. Entropy

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

184 Gini impurity vs entropy

- 대부분의 경우 두 measure간에는 큰 차이가 없음.
- Gini impurity가 살짝 빠르긴 함.
- Gini impurity는 가장 많이 나오는 class를 isolate시키려는 경향이 있음.
- Entropy는 balanced tree를 만들려는 경향이 있음.

184 Regularization Hyperparameters

- Decision Tree들은 Regularized 되지 않을 경우 overfit되는경우가 많음.
- 이런 모델들은 *Nonparametric model* 이라 불림.
  - 파라미터가 없어서가 아니라, 파라미터들이 train 한 후에 결정되어서임.
  - 이것은 Model strcture가 data에 근접할 수 있게 만듦
- 반대로, *parametric model* 이라 불리는 모델들도 있음.
  - DOF가 제한적이게 되나 overfit의 가능성이 낮음.
  - Underfit의 가능성은 상대적으로 높음.
- Overfit을 예방하려면, Decision Tree를 제한시켜야함.(=regularization해야함)

184~185 Regularization Hyperparameters

- The regularization hyperparameters들은 algorithm에 따라 다르게 사용되긴 하지만, maximum depth는 보통은 제한 할 수 있음.
  - Scikit-Learn에서는 max_depth hyperparameter 에 의해 제어됨.
  - Default는 None (= unlimited)상태임.
  - 이것을 바꿔주면 overfit을 방지 할 수 있다.
- DecisionTreeClassifier class는 몇 가지의 다른 parameter도 있음.
  - min_samples_split : 나눠지기 전 노드가 반드시 가져야 하는 최소한의 샘플의 갯수(the minimum number of samples a node must have before it can be split)
  - Min_samples_leaf : leaf node가 가져야 하는 최소한의 샘플의 갯수
  - min_weight_fraction_leaf : min_samples_leaf와 같지만, fraction of weighted instance의 총 수로 나타내어짐.
  - max_leaf_nodes : leaf node의 최대 수
  - max_features : feature의 최대 수
  - Min~ 파라미터를 늘리거나, max~ 파라미터를 줄이면 모델을 regularize함

185 regularize 하지 않고 잘 작동하게 하는 다른 방법

- 어떤 노드의 자식노드가 모두 leaf node이면서 *statistically significant* 한 purity improvement 를 제공하지 않는 경우, 불필요한 노드로 고려됨. 그 불필요한 노드를 *pruning*(=delete) 하는것임.
- *χ2 test* 와 같은 standard statistical test에서는 improvement가 순수한 chance의 결과일 가능성을 예측하곤 했다. (*null hypothesis*라 불림) *p-value* 라 불리는 이 가능성이 threshold 보다 높을 경우(보통 5%, hyperparameter에 의해 제어됨), node는불필요한것으로 고려되고 그 자식노드들은 지워진다.
- Pruning 은 모든 불필요한 노드들이 지워질때까지 계속됨.

185 min_samples_leaf의 예시

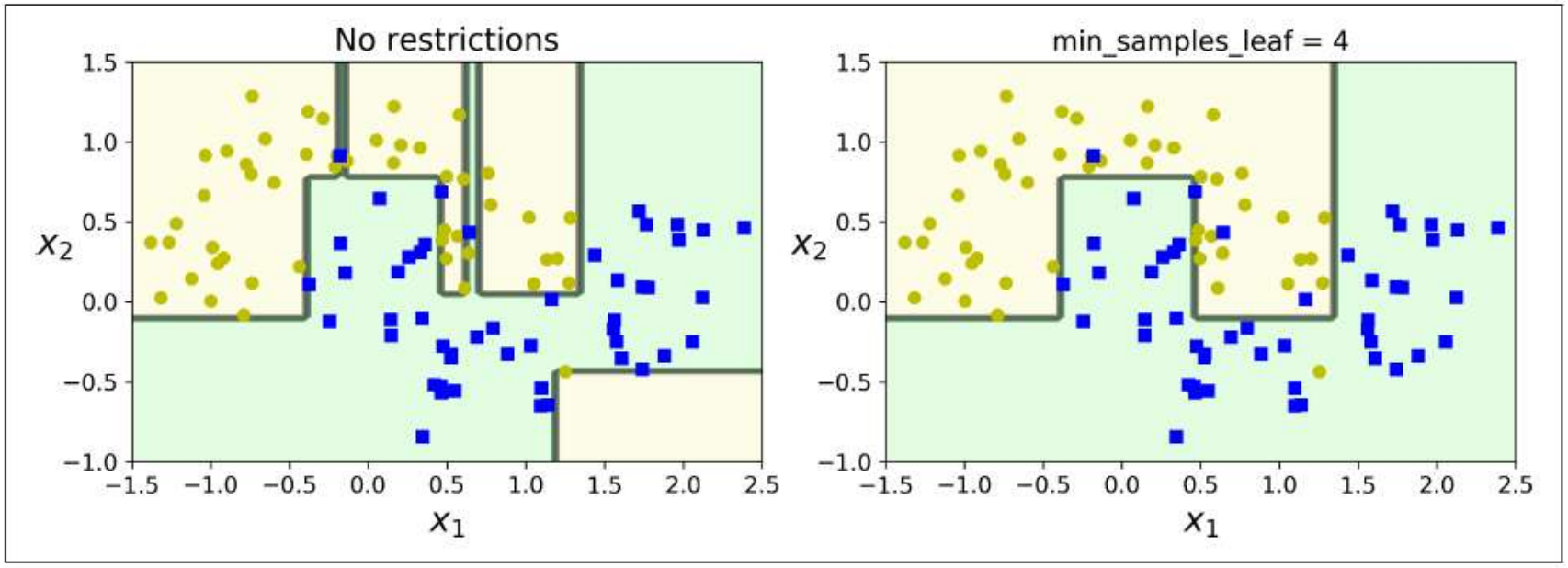


Figure 6-3. Regularization using min_samples_leaf

좌측의 Decision Tree는 min_samples_leaf=4 로 트레인됨. 또한 overfitting되어있다는것을 쉽게 알 수 있음.
우측의 모델이 일반적으로 더 잘 작동할것.

185~186 Regression

- Decision Tree는 Regression에도 사용 가능함.
- Scikit-learn의 DecisionTreeRegressor을 사용하면 됨.

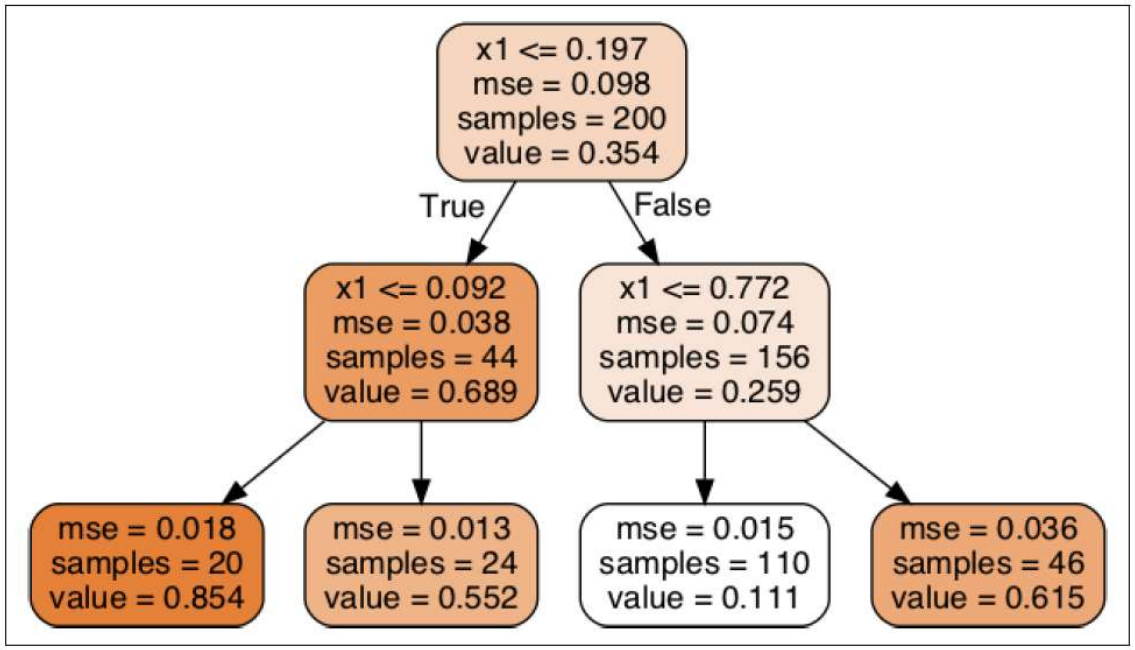


Figure 6-4. A Decision Tree for regression

max_depth=2 에서 트레인 했음.
데이터셋은 noisy quadratic dataset임.

기존 트레인 했던것과 매우 유사하게 생겼지만, 가장 큰 차이점은 결과값이 value를 예측하는것. 좌측은 예를들어 $x1 = 0.6$ 인 인스턴스의 결과를 예측할때, value=0.1106 를 얻게 됨. 이값은 110개의 샘플의 평균을 구한것임. MSE도 해당 110개의 값에 대한것임.

186~187 max_depth

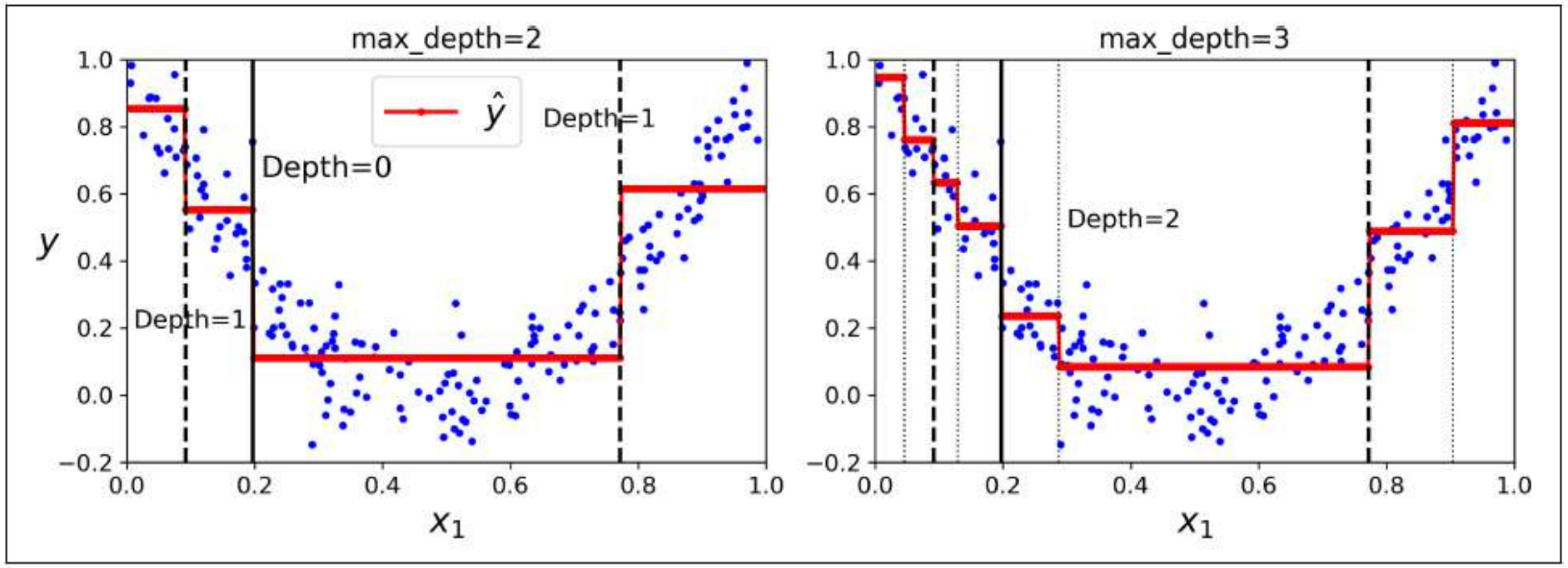


Figure 6-5. Predictions of two Decision Tree regression models

구역마다 평균값을 따로 가짐.
알고리즘이 나누는 구역은 대부분의 training instance들이 해당 구역의 평균값과 가장 가깝게 만드는 방향으로 함.

187 The CART algorithm

- 위(트레이닝셋을 최소의 impurity값을 가지게 나누는것)와는 다르게, CART algorithm은 트레이닝셋을 최소의 MSE값을 가지게 나눔.

Equation 6-4. CART cost function for regression

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

187 The CART algorithm

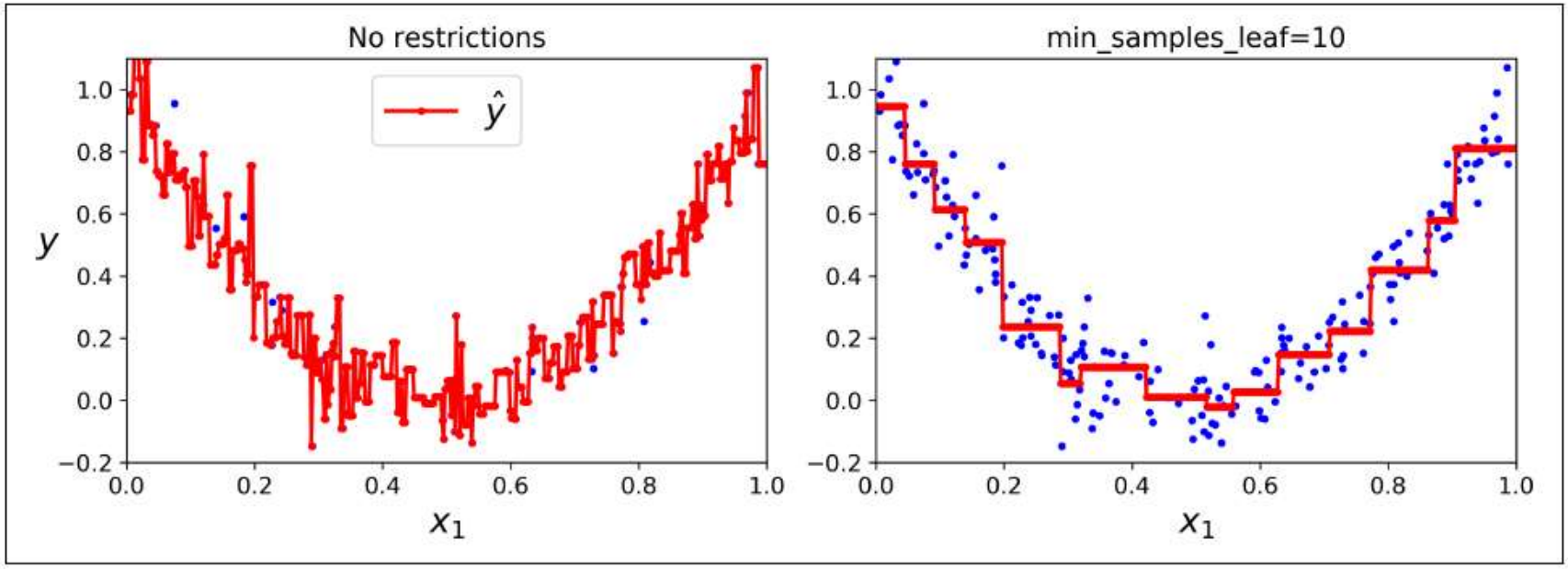


Figure 6-6. Regularizing a Decision Tree regressor

Decision Trees 알고리즘도 Overfit하는 경향이 있음.
Regularization을 하지 않을경우 좌측그림과 같은 prediction 을 얻게될것임. Overfit된것.
min_samples_leaf=10으로 세팅하게 되면 우측그림과 같은 prediction을 얻음.

188 Instability

- Decision Trees은 앞 내용대로 해석이 쉽고, 유연하고, 사용하기 쉽고, 강력하다.
- 그러나 몇가지 단점도 있음.
- 1. orthogonal decision boundaries 를 사용함. -> 모든 split은 축에 대해 수직(perpendicular)임 -> training set rotation에 대해 민감하게 반응함.

좌측 : 데이터를 쉽게 나눌 수 있음

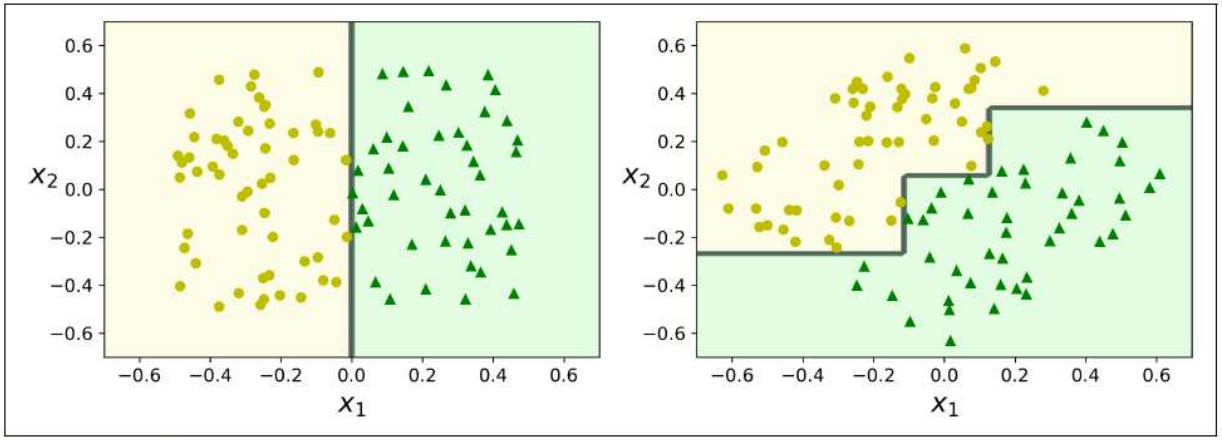


Figure 6-7. Sensitivity to training set rotation

우측 : 데이터를 45도 회전시켰는데, 불필요하게 많이 나누어짐.

좌/우측 다 데이터를 완벽히 나누기는 했지만, 우측의 모델은 generalize하기에 부족해보임.
PCA를 사용하면 해결이 됨.(챕터8)

188~189 Instability

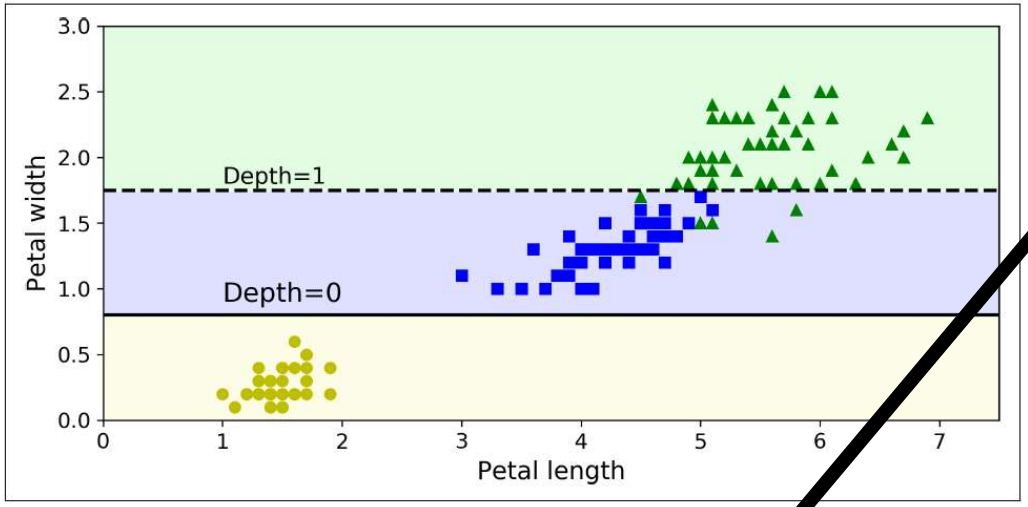


Figure 6-8. Sensitivity to training set details

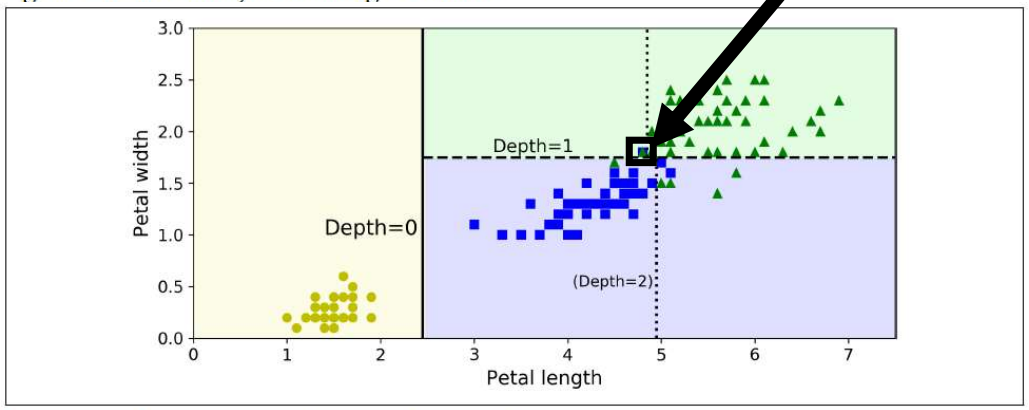


Figure 6-2. Decision Tree decision boundaries

Dicision Trees의 가장 큰 문제점은 작은 variation에 대해 매우 민감하게 반응하는것임.
예를들어서, Iris 트레이닝 셋에서 Iris-Versicolor를 제거한다고 해 보자.(length 4.8, width 1.8 인 데이터)
새 Decision Tree를 만들게 되면 좌측의 결과를 얻게 된다.

Scikit-Learn의 training algorithm 은 stochastic이기 때문에 같은 트레인 데이터라도 매우 다른 모델을 얻을 수 있다.(random_state hyperparameter을 사용할 경우는 같음)

Random Forests에서는 많은 tree들의 prediction값 평균화를 통해서 위와 같은 문제점들을 제한시킬 수 있다.
-> 다음챕터에 나옴