

결정 트리

Decision Tree

- 결정 트리
 - 개념
 - 동작 원리
 - 결정 트리 시각화
 - 결정 트리 과대적합 및 해결방안
 - 결정 트리 하이퍼 파라미터 튜닝
 - 결정 트리 평가
- 결정 트리 실습

분류(Classification)의 개요

지도학습 -> 분류

학습 데이터로 주어진 피처와 레이블(결정값, 클래스값)을 머신러닝 알고리즘으로 학습해 모델을 생성하고, 이렇게 생성된 모델에 새로운 데이터값이 주어졌을 때 미지의 레이블을 예측하는 것.

기존 데이터가 어떤 레이블에 속하는지 패턴을 알고리즘으로 학습 -> 새롭게 관측된 데이터의 레이블을 판별

분류 알고리즘

1. 나이브베이즈 - 베이즈 통계와 생성 모델
2. 독립 변수와 종속 변수와 선형 관계성에 기반에 로지스틱 회귀
3. 데이터 균일도에 따른 규칙 기반 결정 트리
4. 개별 클래스 간의 최대 분류 마진을 효과적으로 찾아주는 SVM
5. 근접 거리를 기준으로 하는 최소 근접 알고리즘(KNN nearest Neighbor)
6. 심층 연결 기반의 신경망
7. 서로 다른(같은) 머신 러닝 알고리즘을 결합한 앙상블

분류(Classification)의 개요

앙상블 - 서로 다른 / 또는 같은 알고리즘 결합

1. 배깅 - 랜덤 포레스트: 뛰어난 예측 성능, 빠른 시간, 유연성

2. 부스팅 -

1) 그래디언트 부스팅 - 뛰어난 예측성능, 단, 수행시간 길다. 최적화 모델 튜닝 어렵다.

2) XGBoost, LightGBM - 예측 성능은 높이고 수행시간은 단축. 정형 데이터 분류 영역에서 가장 활용도가 높다.

3. 스태킹 - 앙상블의 앙상블

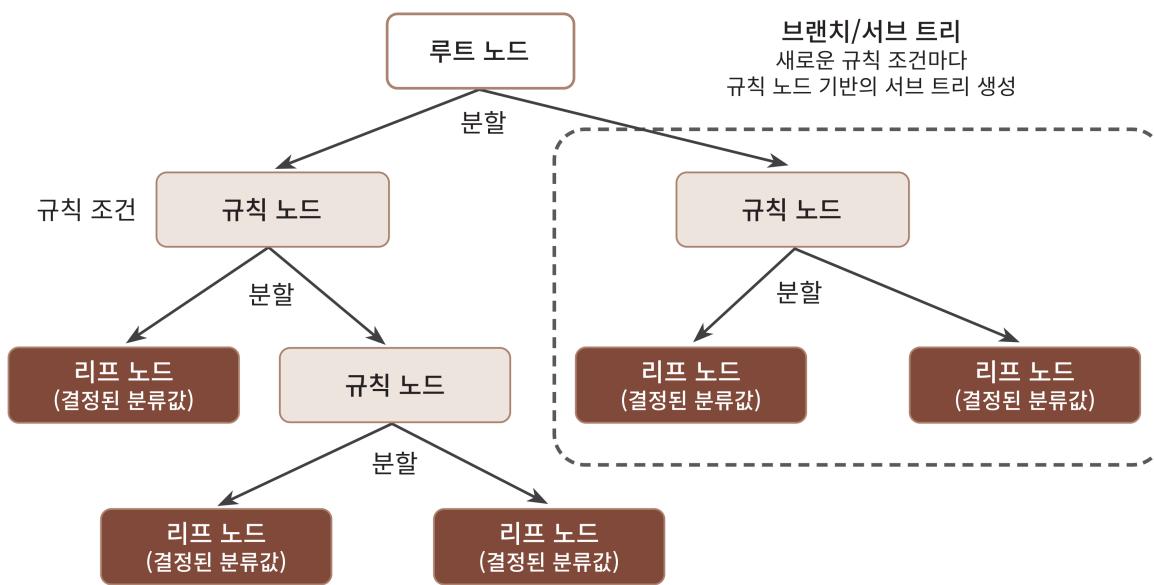
앙상블 - 분류에서 가장 각광을 받는 방법 중 하나, 정형 데이터의 예측 분석 영역(cf. 딥러닝 - 이미지, 영상, 음성, NLP 영역)

앙상블의 기본 알고리즘- 결정 트리

결정 트리

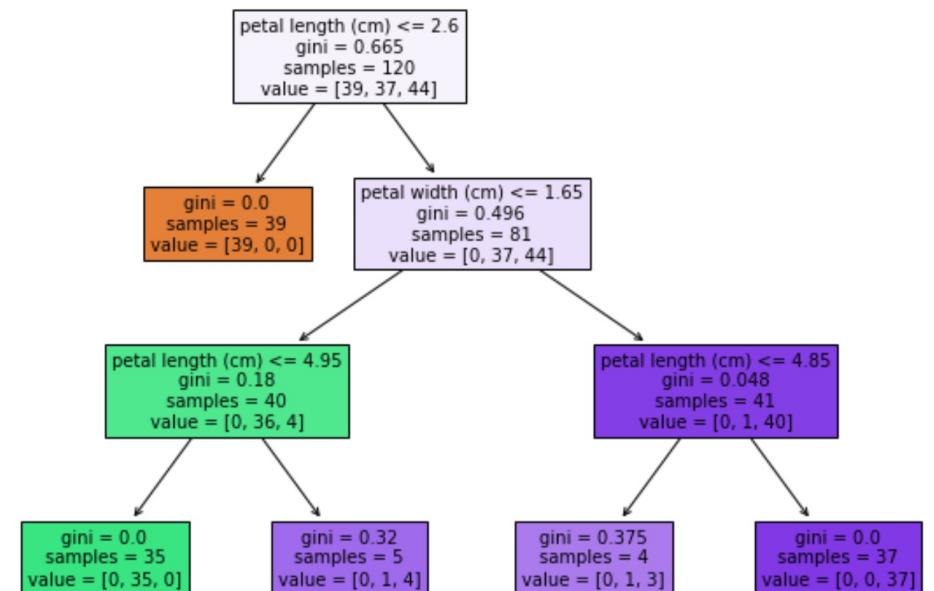
- 직관적으로 이해하기 쉬운 머신러닝 알고리즘
- 종속 변수가 범주형일 경우, 수치형일 경우 모두 사용할 수 있는 지도학습 방법론이다
- 결과에 대한 해석이 용이하여 정책 의사결정에 폭넓게 사용되는 방법론이다.
- 주어진 설명 변수(연속형, 범주형)를 활용해 의사결정 규칙(rule)을 자동으로 찾아내 트리 기반의 분류 규칙을 생성한다.
- 결정트리 알고리즘은 데이터나 오차 등에 대한 어떠한 가정도 필요 없는 비모수 방법(non-parametric method)으로 유연하게 사용할 수 있다.

<트리 구성 요소>



[참고]

<붓꽃 분류>



※ 출처: 파이썬 머신러닝 완벽가이드, p185, 위키북스

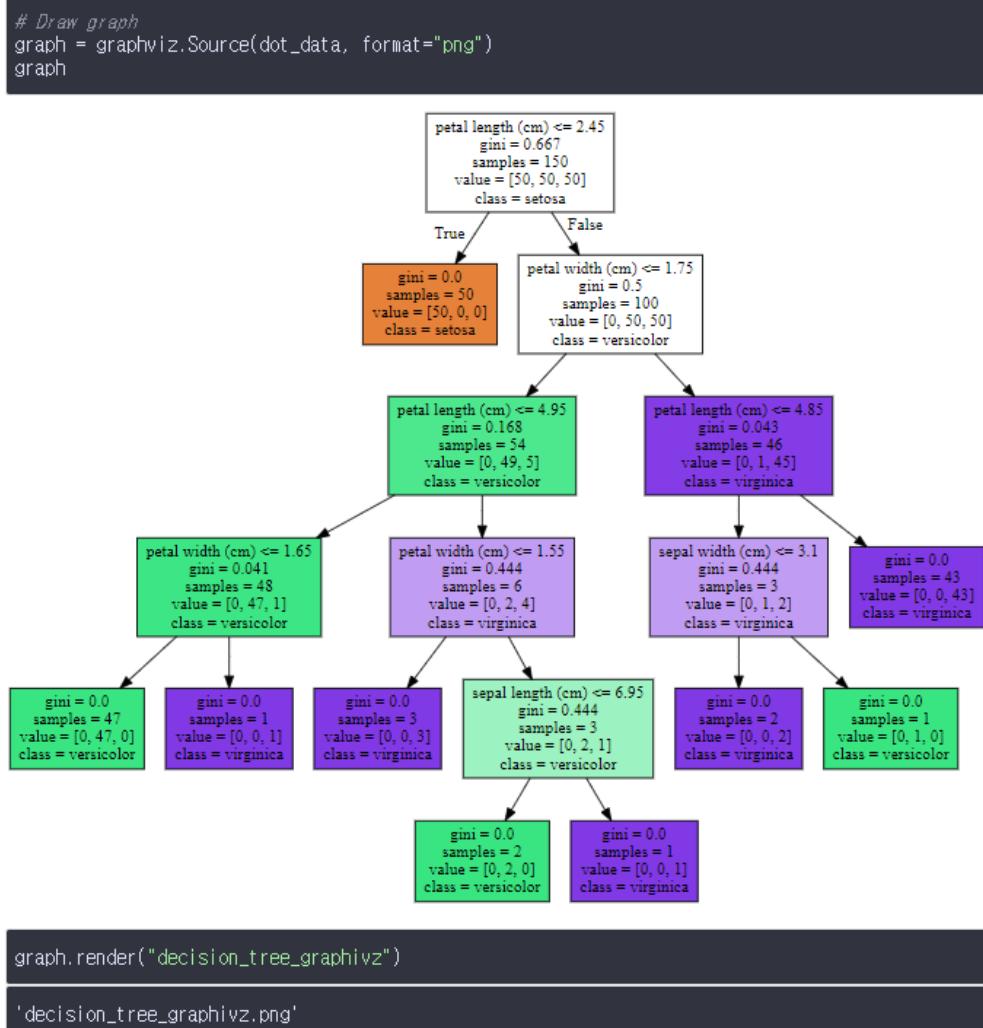
- 부모 노드, 자식 노드

결정 트리 시각화 - 4가지

- Python을 활용한 결정 트리 모델 시각화 방법

1. sklearn.tree.export_text method
2. skleran.tree.plot_tree method (matplotlib needed)
3. sklearn.tree.export_graphviz (graphviz needed)
4. dtreeviz (dtreeviz, graphviz needed)

※ 코드 예제: <https://mljar.com/blog/visualize-decision-tree/> 참고



[Plot Decision Tree with dtreeviz Package](#)

[실습] 붓꽃 데이터

- sklearn의 붓꽃 데이터를 결정 트리 알고리즘을 통해 분류

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

iris = load_iris()

X_train, X_test, y_train, y_test = train_test_split(iris.data,
                                                    iris.target,
                                                    test_size=0.2,
                                                    random_state=1)

from sklearn.tree import DecisionTreeClassifier

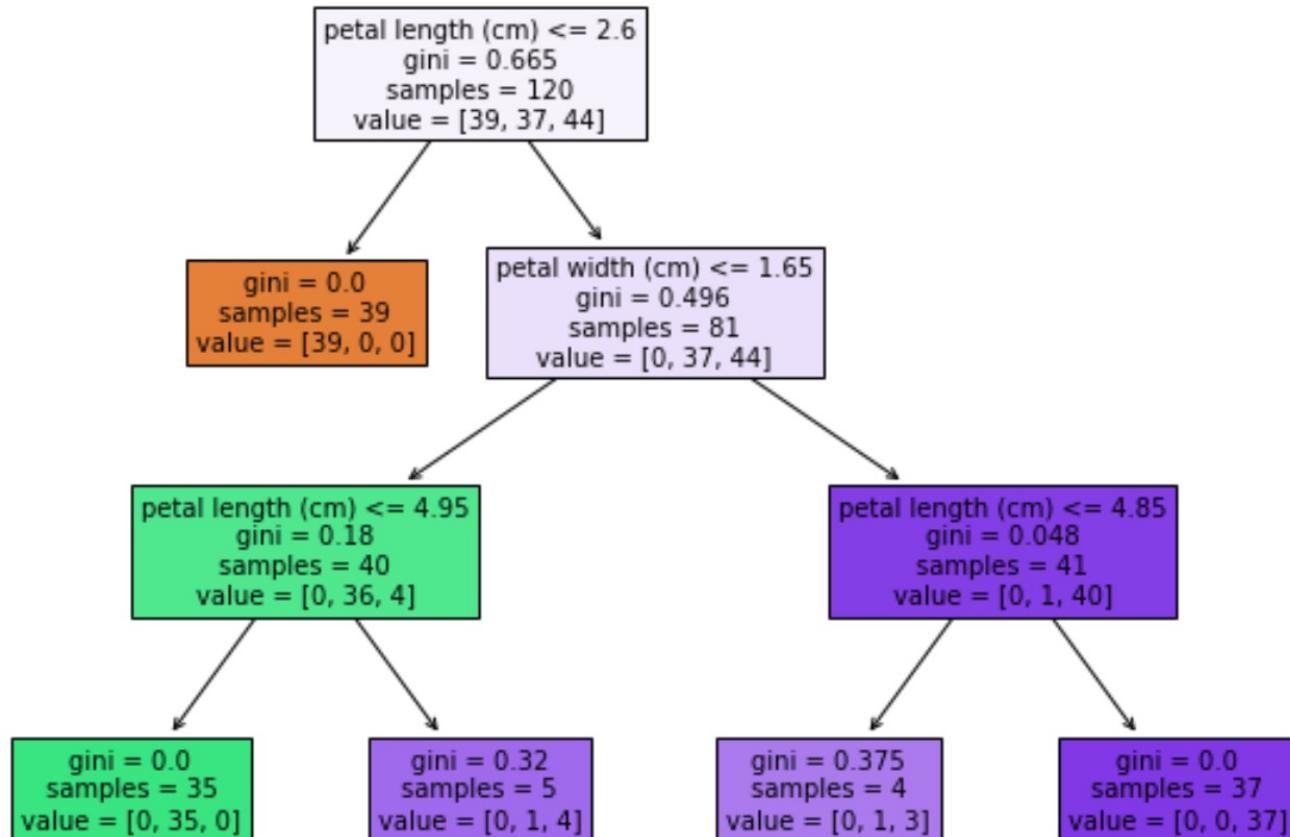
dt = DecisionTreeClassifier(max_depth=3)
dt.fit(X_train, y_train)
print(dt.score(X_train, y_train))
print(dt.score(X_test, y_test))
```

0.983333333333333
0.9666666666666667

[실습] 붓꽃 데이터

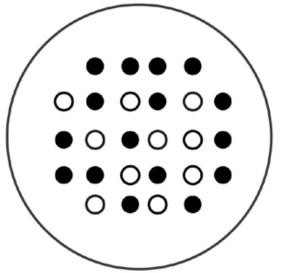
시각화

```
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
plt.figure(figsize=(10, 7))
plot_tree(dt, max_depth=3, filled=True, feature_names=iris.feature_names)
plt.show()
```

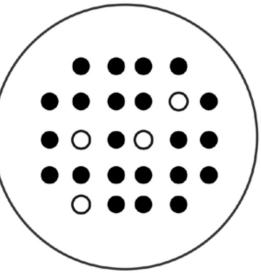


트리의 분할 원리

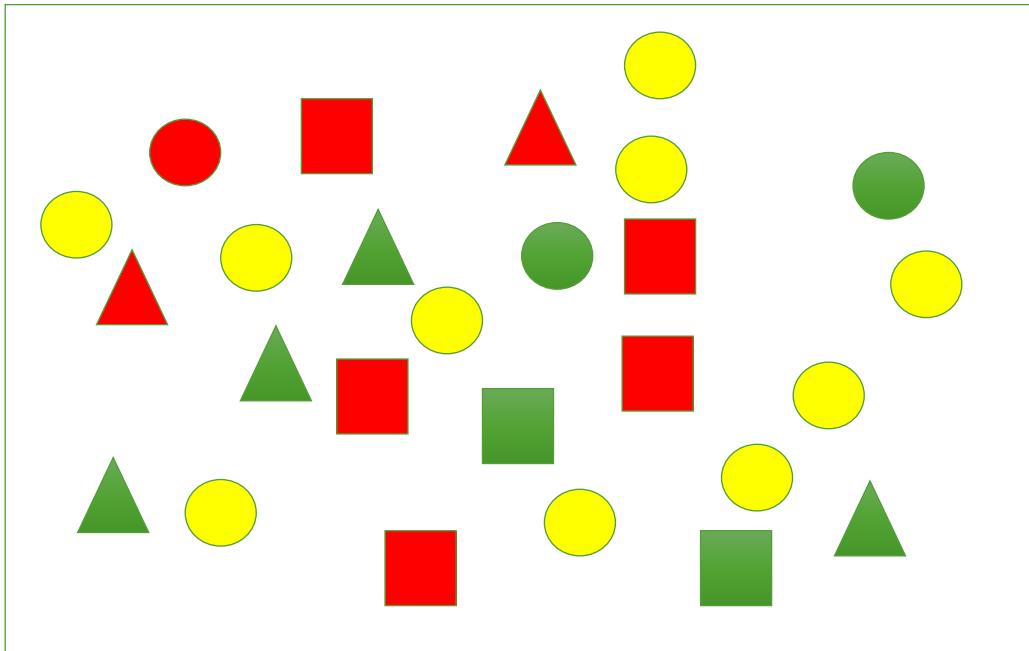
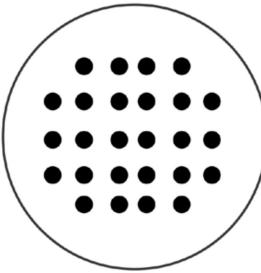
데이터 세트 A



데이터 세트 B



데이터 세트 C



트리의 분할 원리

- 결정트리의 알고리즘에는 CHAID(Kass, 1980), CART(Breiman et al., 1984), C4.5(Quinlan, 1993) 등과 이들의 장점을 결합한 다양한 알고리즘이 있음.
- 가지 분할(tree split, branch split) 단계는 결정트리 모형을 만들기 위한 초기 단계이다.
- 주어진 데이터의 설명 변수를 이용해 순수도(purity)가 최대가 되도록(불순도가 감소하도록) 규칙을 만들어 나간다. (불순도 기준에 의해 정보이득^{information Gain}이 최대가 되도록 나눈다)
- 가지 분할 기준은 분류 문제와 회귀 문제 있어서 상이한 기준이 존재하며, 대표적인 예는 다음과 같다.

알고리즘	특징	분류(Classification)	회귀(Regression)
CHAID(Chi-Squared Automatic Iteration Detection)	다지 분리(multiway split)	카이제곱 통계량	F-통계량
CART(Classification And Regression Tree)	이진 분리(binary split)	지니 지수	분산 감소량
C4.5, C5.0	다진 분리, 이진 분리	엔트로피 지수	n/a

$$\phi(g) = \chi^2 = \sum_i \frac{(O_i - E_i)^2}{E_i}$$

$$\phi(g) = 1 - \sum_{i=1}^l \hat{P}_i(g)^2$$

$$\phi(g) = -\sum_i \hat{p}_i(g) \log \hat{p}_i(g)$$

트리의 분할 원리

순수도 Purity 또는 불순도 Impurity에 의해 측정

- 순수도 Purity : 특정 범주의 개체들이 포함된 정도
- 불순도 Impurity : 얼마나 다양한 범주의 개체들이 포함되어 있는가를 계산

분할 속성의 선택

- 부모 마디의 순수도에 비해서 자식 마디의 순수도가 증가하도록 자식 마디를 형성
- 예) 그룹0과 그룹1의 비율이 45%와 55%인 마디는 각 그룹의 비율이 90%와 10%인 마디에 비해 순수도가 낮음(불순도가 높음)
- 마지막에 도달한 노드의 클래스 비율을 보고 예측을 만듦.
- 단, 결정트리는 끝까지(greedy) 분할을 하기 때문에 과적합이 생기기 쉽다. (가지치기 또는 과적합 개선 처리가 필요)

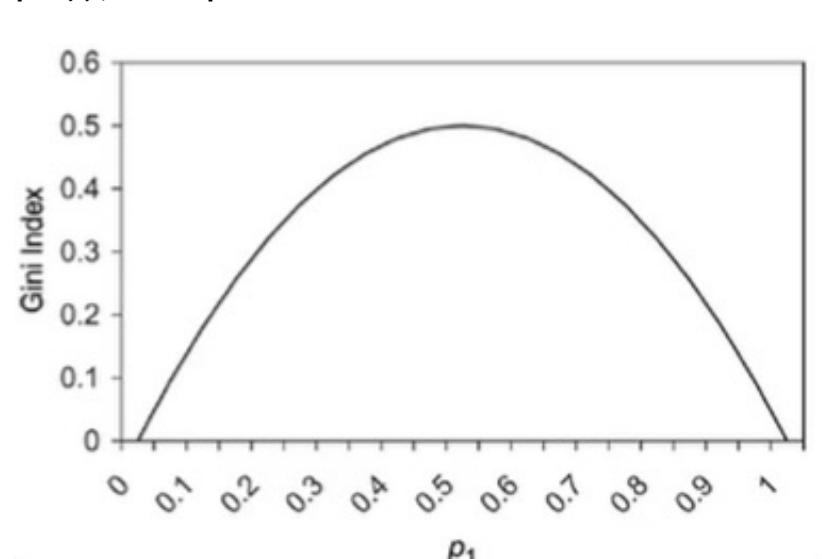
가지치기

- 트리의 최대 깊이 지정(max_depth)

결정트리 분류 기준- CART

- CART(Classification And Regression Tree)
 - Breiman 등이 개발
 - 분류 및 회귀 양쪽 모두 사용 가능
 - 불순도 알고리즘: Gini 지수(Gini Index) - 불확실성의 증감으로 판정
 - 분리: 2지 분리(Binary Split)
 - 가지치기(교차 타당도) : 학습 데이터를 이용하여 나무를 성장시키고 검증용 데이터를 이용하여 가지치기
 - 2번 복원추출을 했을 때 나올 수 있는 확률

$$G(A) = 1 - \sum_{K=1}^n p_k^2$$



결정트리 분류 기준- CART

지니 계수로 분할 하기

- 특징에 의한 분리가 이진 분류로 나타날 경우 지니 계수(Gini coefficient)를 사용할 수 있음.
- 지니 계수의 특성

특징이 항상 이진 분류로 나뉠 때 사용됨

두 개의 범주 개체가 50:50일 때 최대 불순도값 0.5

지니 계수가 높을수록 순도가 높음

- 지니 계수를 통해 의사결정 트리의 노드를 결정하는 순서.

지니 특징으로 분리된 두 노드의 지니 계수를 구함($P^2 + Q^2$)

특징 계수를 구함

- 지니 계수가 더 높은 특성을 상위 노드로 선택함.

결정트리 분류 기준- CART

[예제]

A 홈쇼핑에서는 충성고객(LC: Loyal Customer)과 탈퇴고객(CC: Churn Customer)을 구분하는 규칙을 생성하고자 한다.

총 10명의 고객을 대상으로 성별, 결혼여부 중 어느 변수가 더 분류를 잘 하는 변수인지 찾고, 분류규칙을 찾으시오

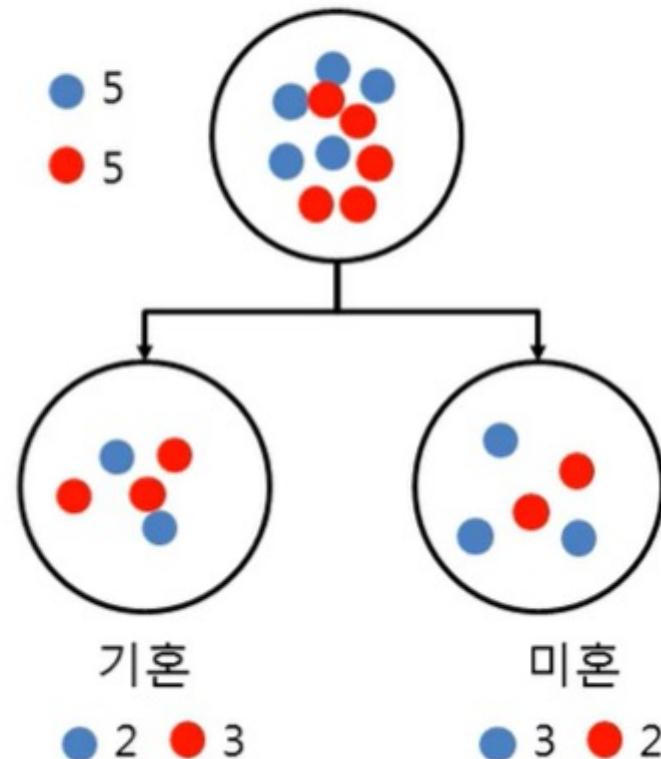
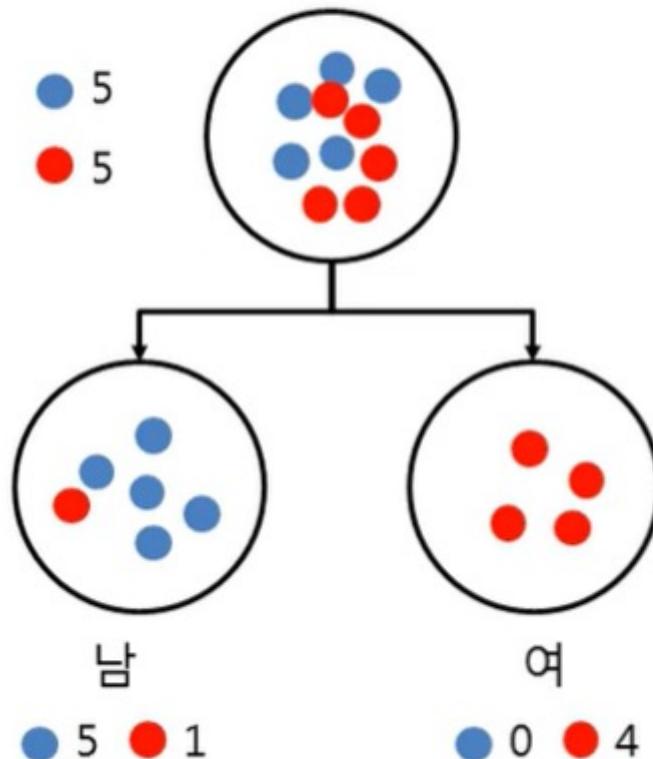


어느 속성이 더 중요한 속성인가?

출처: https://www.youtube.com/watch?v=A2h_PqWQQM4&list=PLEUKy_nwlzwEVNZEjg0zv9oiR8TXRU2PI&index=44

결정트리 분류 기준- CART

- : 충성고객(LC)
- : 이탈고객(CC)



출처: https://www.youtube.com/watch?v=A2h_PqWQQM4&list=PLEUKy_nwlzwEVNZEjg0zv9oiR8TXRU2PI&index=44

결정트리 분류 기준- CART

- 성별에 따른 Gini Index

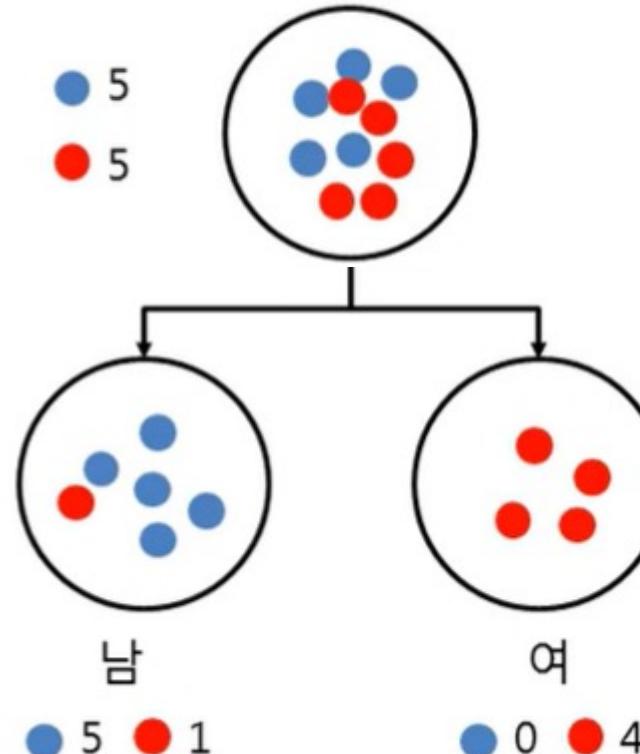
$$G(\text{상위}) = 1 - \left(\frac{5}{10} \right)^2 - \left(\frac{5}{10} \right)^2 = 0.5$$

$$G(\text{남}) = 1 - \left(\frac{5}{6} \right)^2 - \left(\frac{1}{6} \right)^2 = 0.278$$

$$G(\text{여}) = 1 - \left(\frac{0}{4} \right)^2 - \left(\frac{4}{4} \right)^2 = 0$$

$$G(\text{성별}) = \left(\frac{6}{10} \right)(0.278) + \left(\frac{4}{10} \right)(0) = 0.167$$

지니 불순도 = $1 - (\text{음성클래스비율})^2 + (\text{양성클래스비율})^2$



결정트리 분류 기준- CART

- 결혼에 따른 Gini Index

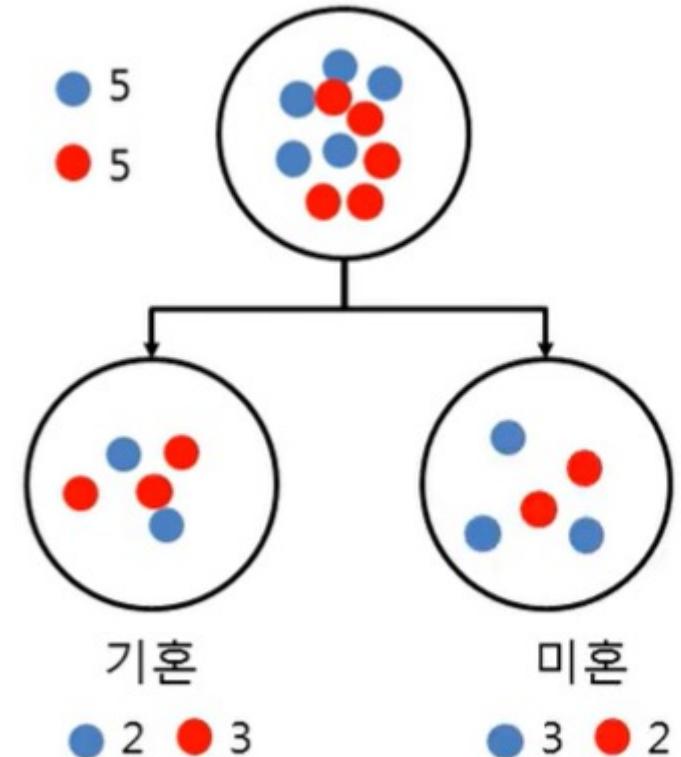
$$G(\text{상위}) = 1 - \left(\frac{5}{10}\right)^2 - \left(\frac{5}{10}\right)^2 = 0.5$$

$$G(\text{기혼}) = 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = 0.48$$

$$G(\text{미혼}) = 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2 = 0.48$$

$$G(\text{결혼}) = \left(\frac{5}{10}\right)(0.48) + \left(\frac{5}{10}\right)(0.48) = 0.48$$

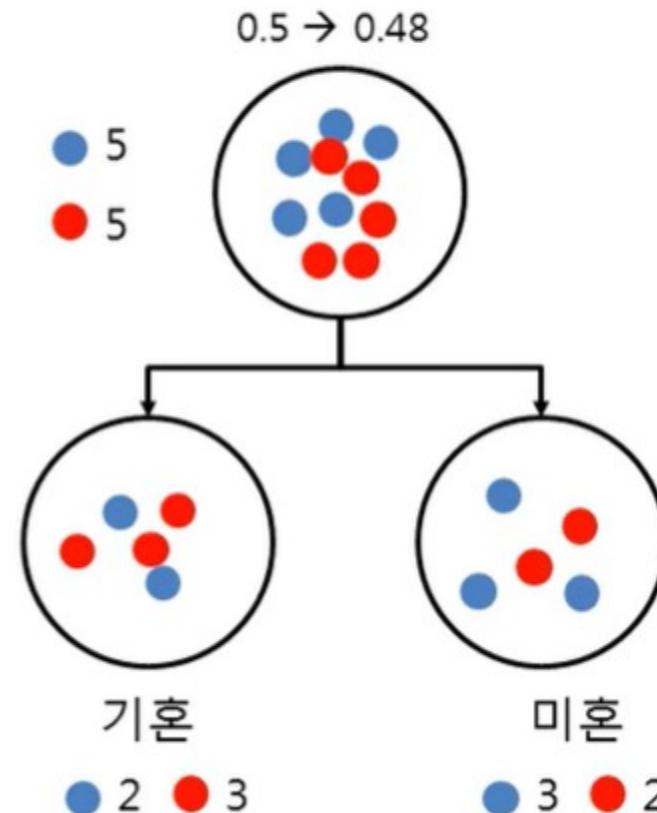
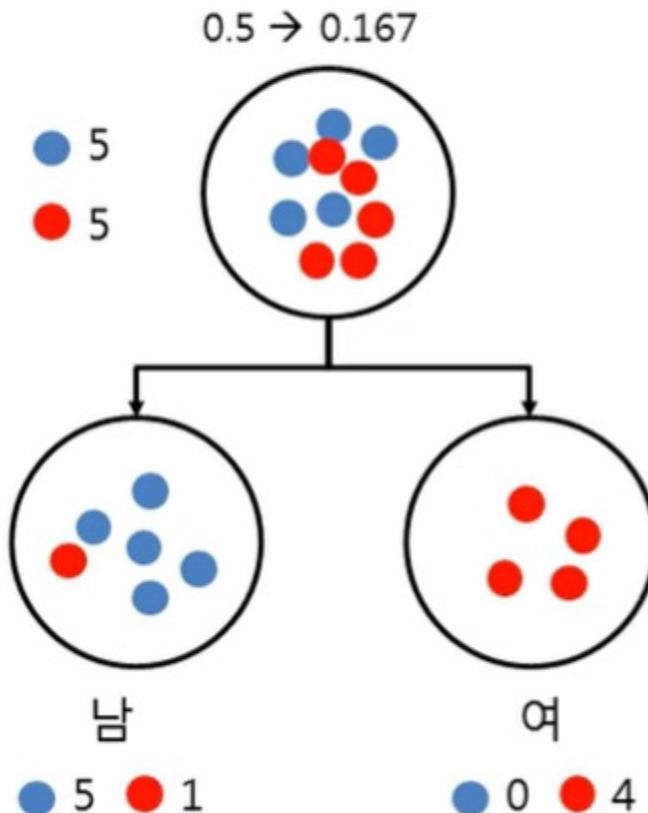
지니 불순도 = $1 - (\text{음성클래스비율})^2 + (\text{양성클래스비율})^2$



결정트리 분류 기준- CART

$$G(\text{성별}) = \left(\frac{6}{10}\right)(0.278) + \left(\frac{4}{10}\right)(0) = 0.167$$

$$G(\text{결 혼}) = \left(\frac{5}{10}\right)(0.48) + \left(\frac{5}{10}\right)(0.48) = 0.48$$



※ 위 경우에서 성별과 혼인 여부 중 어떤 속성을 기준으로 가지를 분할해야 하는가?

결정트리 분류 기준- 정보엔트로피

- 엔트로피 - 정보량을 측정하는 기준으로 엔트로피가 높으면 정보가 많이 담겨있음을 의미함.
- 정보 이득(information gain)은 질문 이전의 엔트로피에서 질문후의 엔트로피를 뺀 값
- 즉, 리프 노드가 낮은 정보량을 가지도록 분할을 함.
- 낮은 정보량을 가진다는 것
 - 새로운 결과를 기대할 수 없다는 것.(모두가 동일하므로)
 - 엔트로피가 낮음.

질문 후의 정보 이득 = 질문 전의 엔트로피 - 질문 후의 엔트로피

$$Gain(T, X) = Entropy(T) - Entropy(T, x)$$

확률을 바탕으로 정보 엔트로피를 구하는 공식

$$E(S) = \sum_{i=1}^c (-)p_i \log_2 p_i$$

결정트리 분류 기준- 정보 엔트로피

속성 선택 기준

- 어떻게 분류 기준(속성)을 선택할까
- 모든 속성에 대해 정보이득을 계산하여 결정
- 정보 이득 계산 방법: 엔트로피(entropy) 방식, 지니(gini) 방식
- 각 노드별로 위와 같이 정보 이득 계산값으로 선택된 속성 선택

정보 이득 계산(Entropy)

- 분류하기 전과 분류 후의 변화량
- 계산값이 가장 높은 값(변화량이 큰 것)을 선택

결정트리 분류 기준- 지니 불순도 또는 엔트로피?

기본적으로 지니 불순도가 사용되지만 criterion 매개변수를 "entropy"로 지정하여 엔트로피 불순도를 사용할 수 있음

- 엔트로피는 문자의 무질서함을 측정하는 열역학의 개념

문자가 안정되고 질서 정연하면 엔트로피가 0에 가까움

메시지의 평균 정보 양을 측정하는 샐런의 정보 이론: 모든 메시지가 동일할 때 엔트로피가 0

머신러닝에서는 불순도의 측정 방법: 어떤 세트가 한 클래스의 샘플만 담고 있다면 엔트로피가 0

- 지니 불순도와 엔트로피 중 어떤 것을 사용해야 할까?

실제로는 큰 차이가 없이 둘 다 비슷한 트리를 만듦

지니 불순도가 조금 더 계산이 빠르기 때문에 기본값으로 좋음

그러나 다른 트리가 만들어지는 경우 지니 불순도가 가장 빈도 높은 클래스를 한쪽 가지(branch)로 고립시키는 경향이 있는 반면 엔트로피는 조금 더 균형 잡힌 트리를 만듦

[실습] 와인 분류 - Kaggle 와인 데이터

Kaggle wine dataset을 활용하여 화이트와인을 분류하기

```
import pandas as pd  
wine_kaggle = pd.read_csv('../data/wine_dataset.csv')
```

```
wine_kaggle.iloc[0]
```

```
fixed_acidity      7.4  
volatile_acidity   0.7  
citric_acid        0.0  
residual_sugar     1.9  
chlorides          0.076  
free_sulfur_dioxide 11.0  
total_sulfur_dioxide 34.0  
density            0.9978  
pH                 3.51  
sulphates          0.56  
alcohol             9.4  
quality             5  
style               red  
Name: 0, dtype: object
```

[실습] 와인 분류 - Kaggle 와인 데이터

```
# 과적합 모델
```

```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train_scaled, y_train)
print(dt.score(X_train_scaled, y_train))
print(dt.score(X_test_scaled, y_test))
```

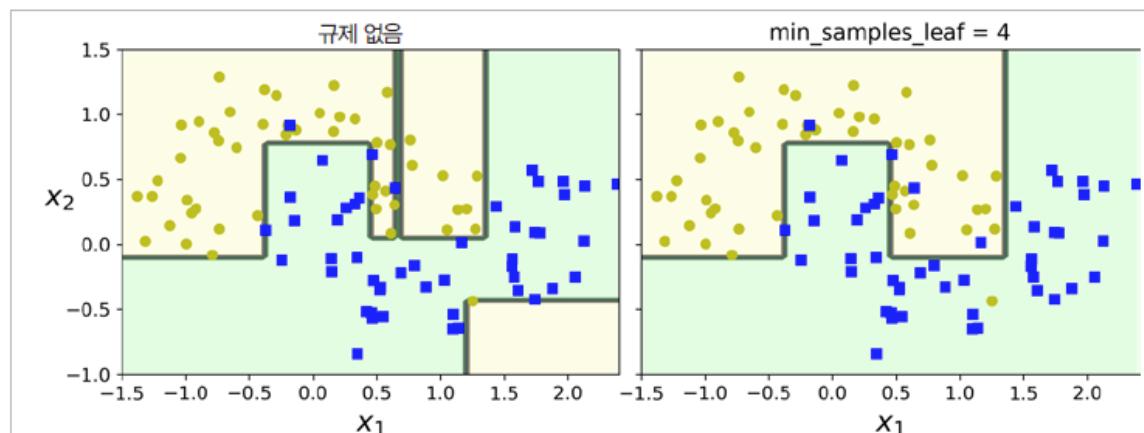
✓ 0.0s

```
0.9978833942659227
0.8692307692307693
```

결정트리의 과적합 및 규제

- 결정 트리는 훈련 데이터에 대한 제약 사항이 거의 없음(vs. 선형 모델은 데이터가 선형일 거라 가정)
- 미리 생성 로직을 제어하지 않으면 완벽하게 클래스 값을 구별해내기 위해 트리노드를 계속 만들어감. 이로 인해 결국 매우 복잡한 규칙 트리가 만들어져 모델이 쉽게 과적합되는 문제점을 가짐.
- 제한을 두지 않으면 트리가 훈련 데이터에 아주 가깝게 맞추려고 해서 대부분 과대적합되기 쉬움
- 결정 트리는 훈련되기 전에 파라미터 수가 결정되지 않기 때문에, 이런 모델을 ‘비파라미터 모델’이라고 부름(vs. 선형 모델은 미리 파라미터 수가 결정되어 있음)
- 모델 구조가 데이터에 맞춰져서 고정되지 않고 자유로움
- 훈련 데이터 과적합을 피하기 위해 학습할 때 결정트리의 자유도를 제한(규제)
- sklearn의 경우 min_, max_로 시작하는 파라미터를 통해 규제 설정

결정 트리 분류 모델의 규제



사이킷런 DecisionTreeClassifier

Parameters: **criterion : string, optional (default="gini")**

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

트리를 구성할 때 사용하는 불순도
[gini, entropy]

splitter : string, optional (default="best")

The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

트리를 split할 때의 전략[best, random]

max_depth : int or None, optional (default=None)

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

최종 생성되는 트리의 Depth의 최대값

min_samples_split : int, float, optional (default=2)

The minimum number of samples required to split an internal node:

- If int, then consider min_samples_split as the minimum number.
- If float, then min_samples_split is a fraction and ceil(min_samples_split * n_samples) are the minimum number of samples for each split.

노드를 split할 때 필요한 최소 샘플 수

Changed in version 0.18: Added float values for fractions.

min_samples_leaf : int, float, optional (default=1)

Leaf node에 있는 최소 샘플 수

The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

- If int, then consider min_samples_leaf as the minimum number.
- If float, then min_samples_leaf is a fraction and ceil(min_samples_leaf * n_samples) are the minimum number of samples for each node.

Changed in version 0.18: Added float values for fractions.

사이킷런 DecisionTreeClassifier

Parameters: `max_features : int, float, string or None, optional (default=None)` split할 때 고려하는 feature의 개수

The number of features to consider when looking for the best split:

- If int, then consider `max_features` features at each split.
- If float, then `max_features` is a fraction and `int(max_features * n_features)` features are considered at each split.
- If "auto", then `max_features=sqrt(n_features)`.
- If "sqrt", then `max_features=sqrt(n_features)`.
- If "log2", then `max_features=log2(n_features)`.
- If None, then `max_features=n_features`.

Note: the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than `max_features` features.

`class_weight : dict, list of dicts, "balanced" or None, default=None`

Weights associated with classes in the form `{class_label: weight}`. If not given, all classes are supposed to have weight one. For multi-output problems, a list of dicts can be provided in the same order as the columns of `y`.

Note that for multioutput (including multilabel) weights should be defined for each class of every column in its own dict. For example, for four-class multilabel classification weights should be `[{0: 1, 1: 1}, {0: 1, 1: 5}, {0: 1, 1: 1}, {0: 1, 1: 1}]` instead of `[{1:1}, {2:5}, {3:1}, {4:1}]`.

The "balanced" mode uses the values of `y` to automatically adjust weights inversely proportional to class frequencies in the input data as `n_samples / (n_classes * np.bincount(y))`

For multi-output, the weights of each column of `y` will be multiplied.

Note that these weights will be multiplied with `sample_weight` (passed through the `fit` method) if `sample_weight` is specified.

<http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier>

Target label의 가중치 부여

- **default = None:** 모두 같은 가중치 (1)

- **balanced:** target label 분포에 따라 weight 설정

[실습] 와인 데이터와 결정 트리 규제

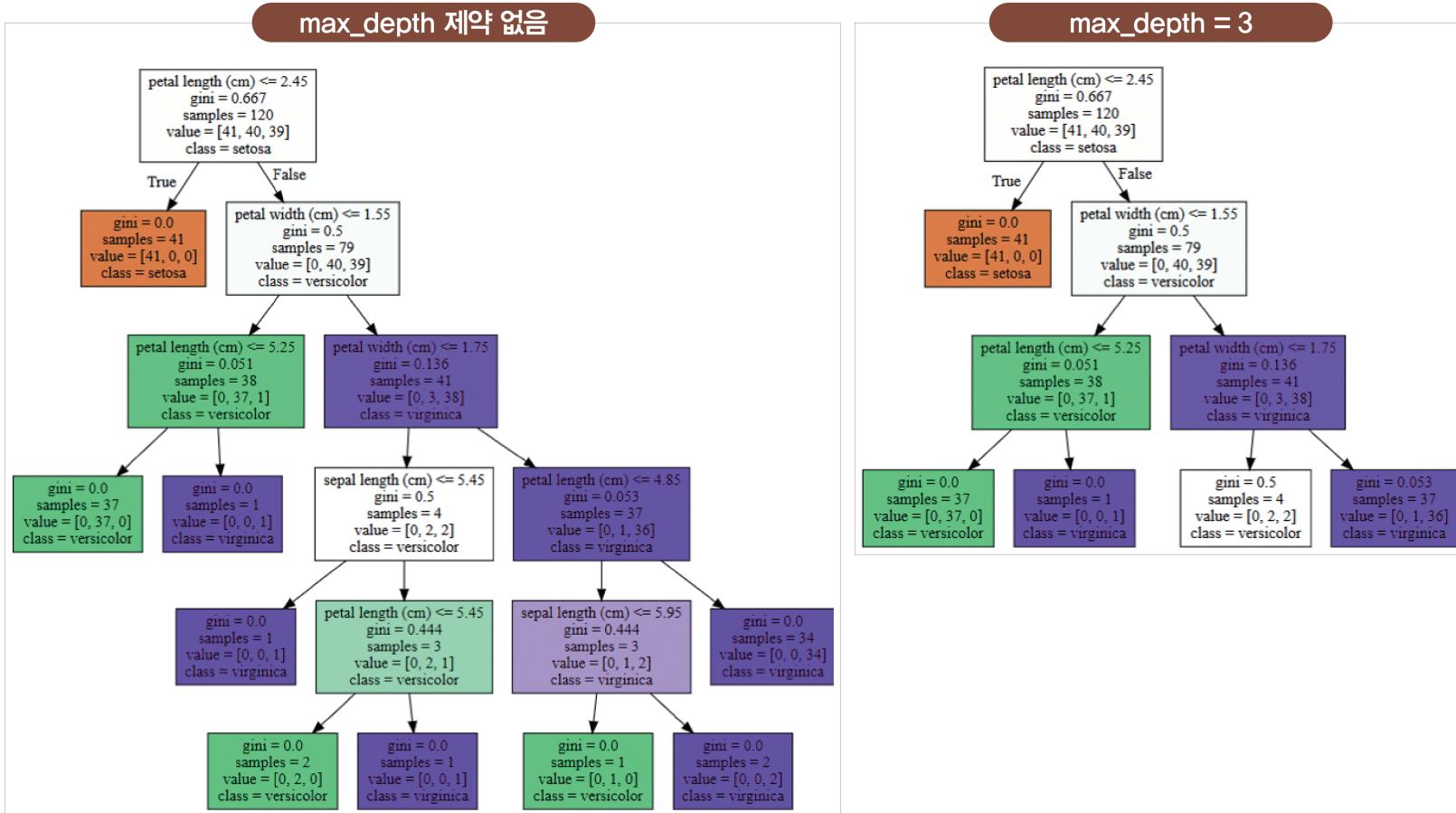
규제를 적용한 결정 트리

```
dt = DecisionTreeClassifier(max_depth=3, random_state=42)
dt.fit(X_train_scaled, y_train)
print(dt.score(X_train_scaled, y_train))
print(dt.score(X_test_scaled, y_test))
```

0.8454877814123533 훈련 세트의 성능은 낮아졌지만, 테스트 세트의 성능은 그대로임
0.8415384615384616

결정트리 하이퍼파라미터 효과

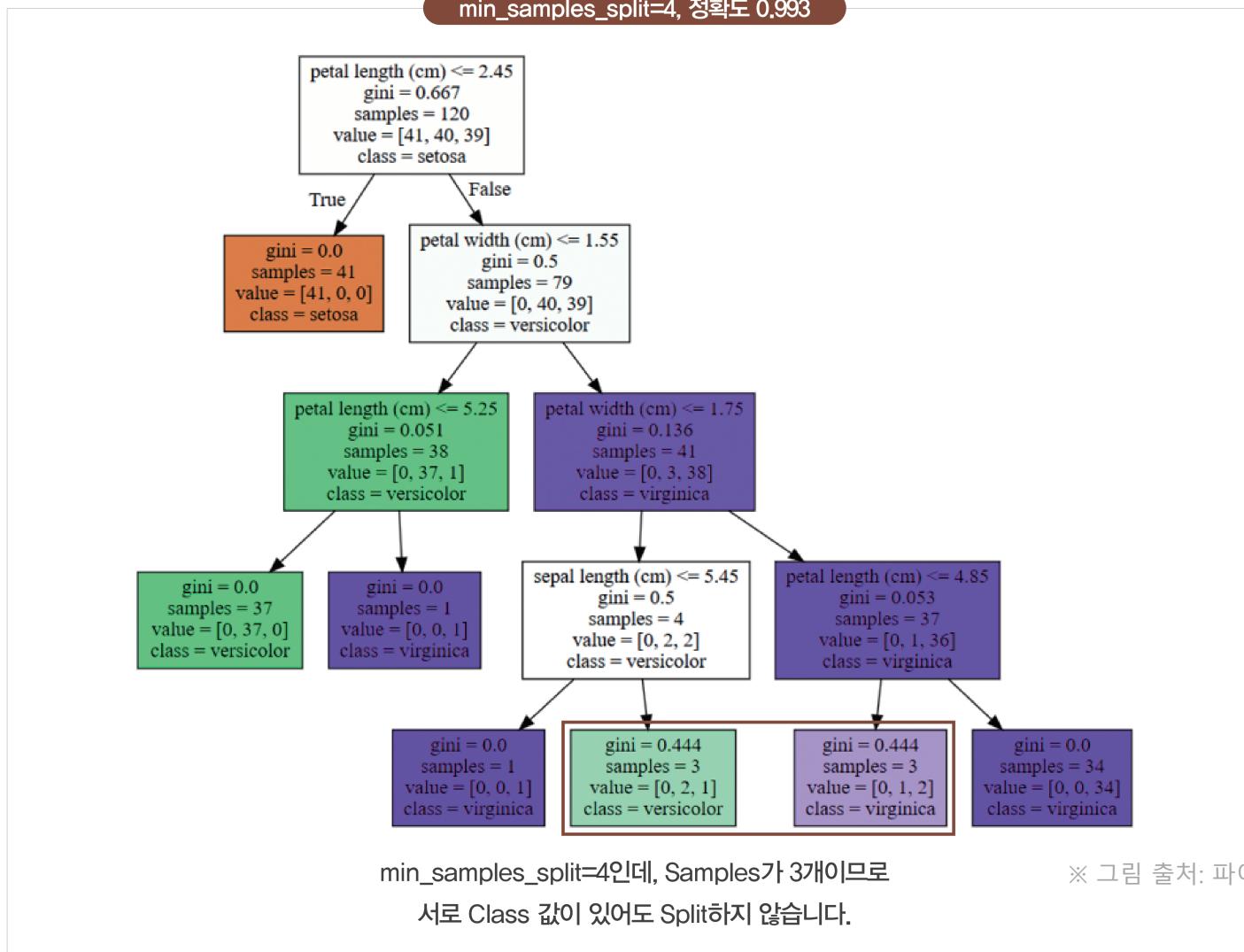
max_depth : 결정트리의 최대 깊이 제한



※ 그림 출처: 파이썬 머신러닝 완벽가이드, p195, 위키북스

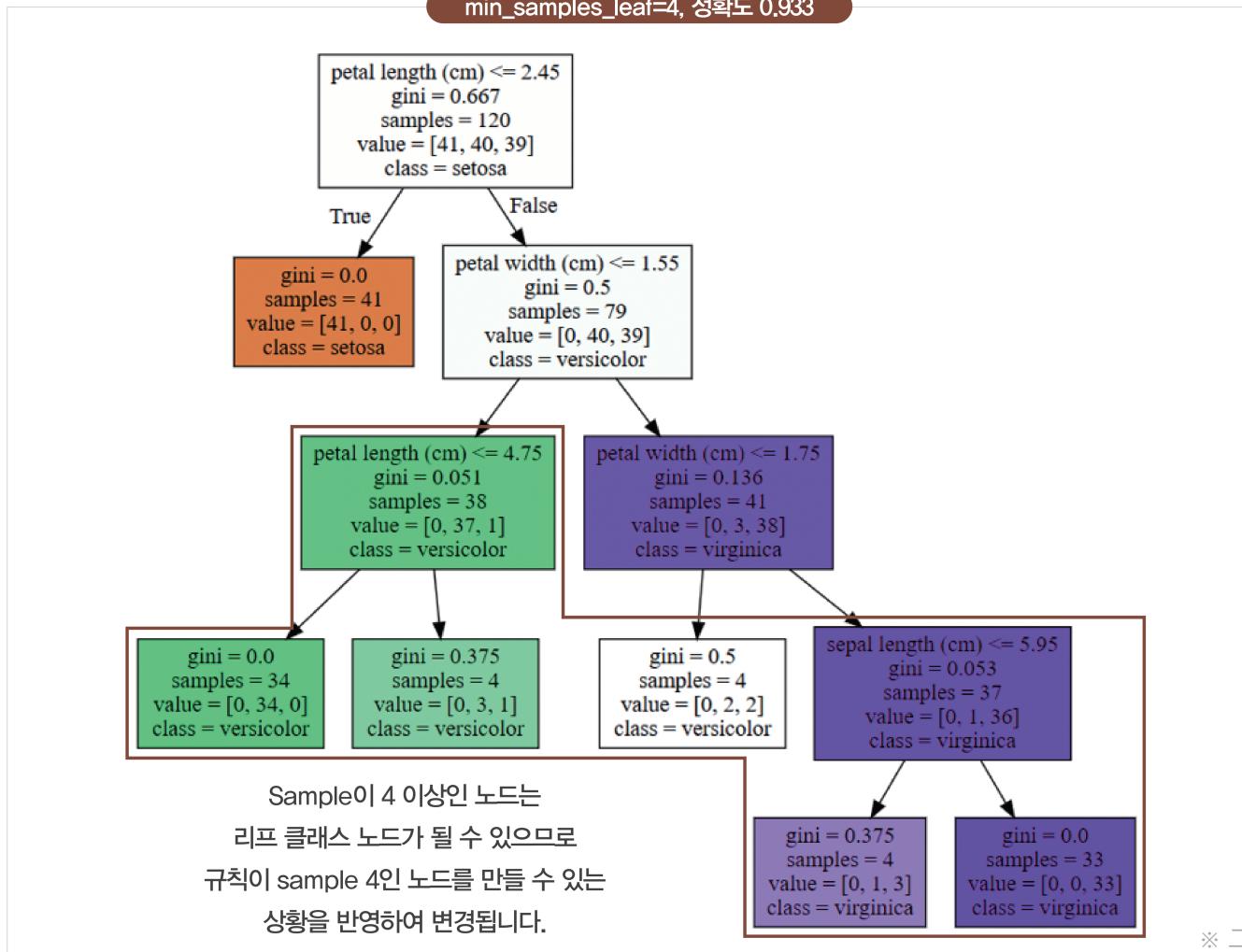
결정트리 하이퍼파라미터 효과

max_samples_split: 자식 노드를 만들기 위한 최소한의 샘플 데이터 개수



결정트리 하이퍼파라미터 효과

min_samples_leaf: 리프 노드에서 필요한 최소 샘플 데이터 개수



※ 그림 출처: 파이썬 머신러닝 완벽가이드, p195, 위키북스

결정트리 과적합과 파라미터 제어

가지 분할을 멈추는 여러 가지 방법

1. 분할을 통해 얻어지는 잎의 크기가 너무 작다면 분할을 멈춘다

min_samples_split

min_samples_leaf

2. 새로운 분할 영역이 “유의미” 한 정도로 불순도를 줄이지 않으면 분할을 멈춘다

min_impurity_decrease

과적합과 최고의 점수 간의 균형을 맞추기 위해 파라미터 제어

탐색할 매개변수 지정

훈련 세트에서 그리드 서치를 수행하여 최상의 검증 점수가 나오는 매개변수 선정

최상의 매개 변수로 지정된 모델에서 훈련

별도로 준비해둔 테스트 데이터셋으로 과적합 예상 여부 판단

※ RandomizedSearchCV - 매개 변수의 범위는 지정하되, 간격을 랜덤하게 지정

결정트리와 특성 중요도(**feature_importances_** 속성)

결정 트리의 **feature_importances_** 속성

- 어떤 특성이 가장 유용한 특성인지 나타냄
- **feature_importances_** 속성에 저장되어 있음
- 특성 중요도 계산 원리: 각 노드의 정보 이득 * 전체 샘플에 대한 비율을 곱한 후 특성별로 더하여 계산함
- 특성 중요도의 활용: 특성 선택에 활용(즉, 결정 트리 모델을 특성 선택에 활용)

property feature_importances_

Return the feature importances.

The importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as the Gini importance.

Warning: impurity-based feature importances can be misleading for high cardinality features (many unique values). See [`sklearn.inspection.permutation_importance`](#) as an alternative.

Returns: **feature_importances_** : *ndarray of shape (n_features,*

Normalized total reduction of criteria by feature (Gini importance).

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.feature_importances_

결정트리와 특성 중요도(feature_importances_ 속성)

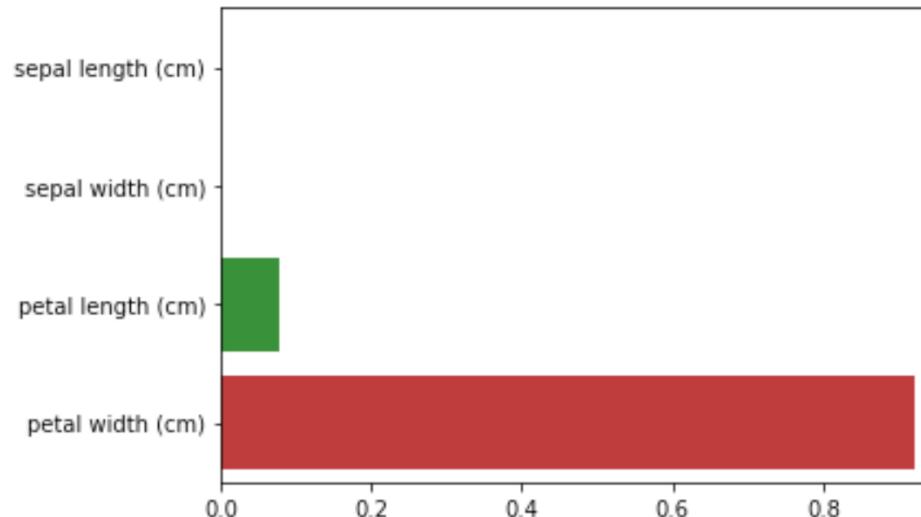
```
import seaborn as sns
import numpy as np
%matplotlib inline

# feature importance 추출
print("Feature importances:\n{0}".format(np.round(dt.feature_importances_, 3)))

# feature별 importance 매핑
for name, value in zip(iris.feature_names , dt.feature_importances_):
    print('{0} : {1:.3f}'.format(name, value))

# feature importance를 column 별로 시각화 하기
sns.barplot(x=dt.feature_importances_ , y=iris.feature_names)
```

Feature importances:
[0. 0. 0.079 0.921]
sepal length (cm) : 0.000
sepal width (cm) : 0.000
petal length (cm) : 0.079
petal width (cm) : 0.921



DecisionTree GridSearchCV

▪ GridSearchCV 주요 파라미터

- estimator: classifier, regressor, pipeline 등
- param_grid : Key + 리스트 조합의 딕셔너리
- scoring: 예측 성능을 측정할 평가 방법
- refit : 가장 최적의 파라미터를 찾은 뒤 입력된 estimator 객체를 해당 파라미터로 재학습시킴.

▪ 주요 속성

cv_results_ : param_grid에 기술된 하이퍼 파라미터를 차례로 학습한 결과를 저장하고 있음.
데이터프레임으로 변환하면 더 쉽게 볼 수 있음.

best_params_ : 최고 성능을 나타낸 하이퍼파라미터의 값

best_score_ : 최고 점수

▪ 일반적 머신 러닝 모델 훈련 방법

학습데이터를 GridSearchCV를 통해 최적 하이퍼파라미터 튜닝을 수행한 뒤 별도의 테스트 데이터 셋에서 이를 평가함.

[실습] 하이퍼 파라미터 튜닝과 GridSearchCV

- 기본 매개 변수를 사용한 결정 트리 모델에서 min_impurity_decrease 매개변수의 최적값 찾기

```
from sklearn.model_selection import GridSearchCV
params={'min_impurity_decrease':[0.0001, 0.0002, 0.0003, 0.0004, 0.0005]}

gs = GridSearchCV(DecisionTreeClassifier(random_state=42), params, n_jobs=-1)
# 결정트리 클래스의 객체를 생성하자마자 바로 전달
# cv 기본값 5
# min_impurity_decrese값마다 5번의 교차검증 25개의 모델을 훈련
# n_jobs= 병렬 실행에 사용할 CPU 코어 수(기본값 1, 모든 코어 사용: -1)

gs.fit(X_train, y_train)
```

```
GridSearchCV(estimator=DecisionTreeClassifier(random_state=42), n_jobs=-1,
            param_grid={'min_impurity_decrease': [0.0001, 0.0002, 0.0003,
                                                0.0004, 0.0005]})
```

DecisionTreeClassifier 구현 tip

1. 차원(feature)이 너무 많거나, 샘플의 크기가 너무 작으면 overfitting되는 경향이 있으므로, 차원이 많은 경우, 차원 축소(PCA, ICA, Feature Selection)를 고려할 필요가 있다.
2. max_depth를 3으로 하고, 우선 트리를 생성해 보고, 점점 max_depth를 점점 늘려가면서 테스트하라.
3. 트리 모델을 도식화해서 보라.
4. 오버피팅 발생 시, 아래 파라미터들을 조절해 보라.
 1. max_depth 낮추기
 2. min_sample_split 높이기 - split 노드의 최소 샘플 개수. 이 개수에 도달하는 노드가 만들어지면 더 이상 트리를 확장하지 않는다.
 3. min_sample_leaf 높이기 - leaf 노드의 최소 샘플 개수. 이 개수에 도달하는 리프 노드가 만들어지면 더 이상 트리를 확장하지 않는다.

※ 트리가 bias되지 않도록 데이터 밸런스를 맞춰라. 예를 들어, 각 클래스별로 동일한 개수의 샘플 데이터 사용

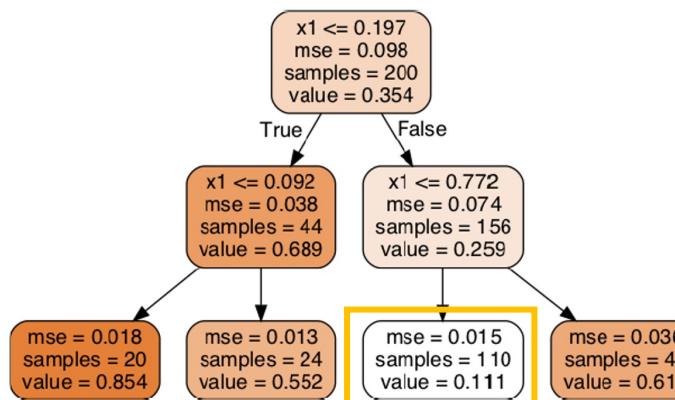
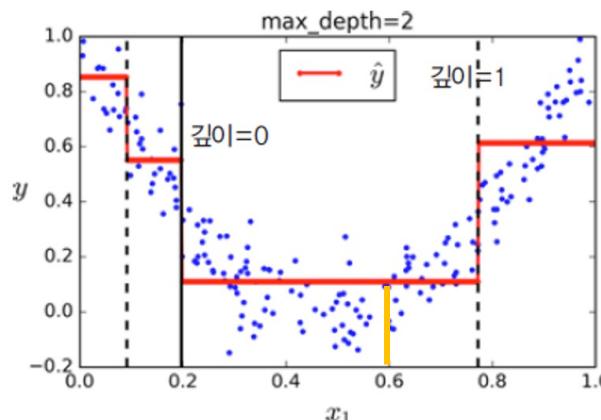
<http://scikit-learn.org/stable/modules/tree.html#tree>

결정트리 회귀 DecisionTreeRegressor

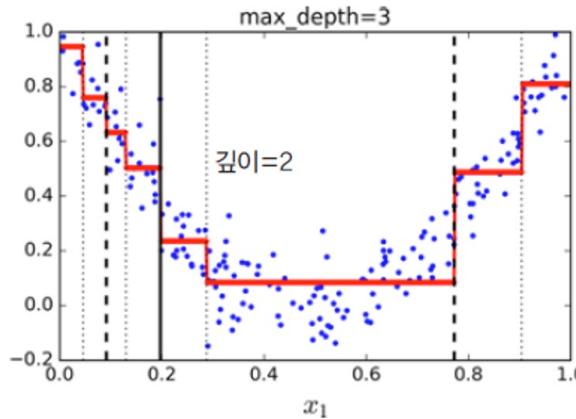
회귀 결정 트리

각 노드에서 클래스를 예측하는 것이 아닌 어떤 값을 예측함

```
from sklearn.tree import DecisionTreeRegressor  
  
tree_reg = DecisionTreeRegressor(max_depth=2)  
tree_reg.fit(X, y)
```



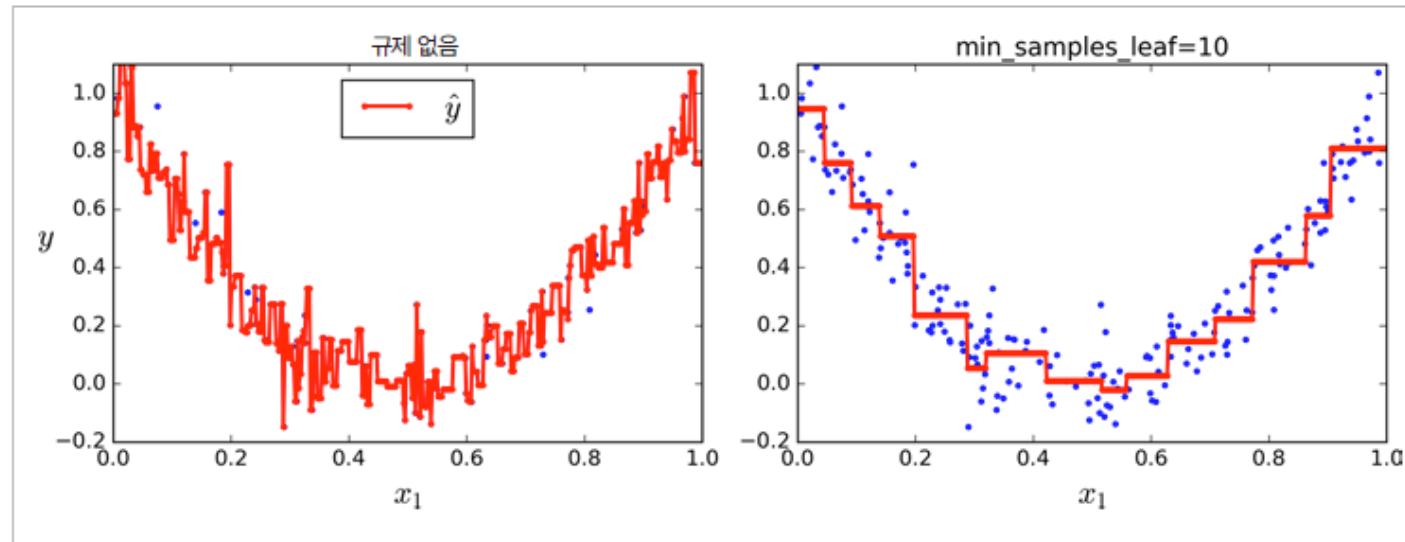
예, $x_1=0.6$ 의 타겟값 예측
value=0.0111인 리프노드에 도착
해당 노드에 속한 샘플의 수: 110
해당 노드의 평균제곱오차(mse) : 0.015



결정트리 회귀 DecisionTreeRegressor 와 과적합

분류에서와 같이 회귀 작업에서도 결정 트리가 과대적합되기 쉬움

규제가 없다면(즉, 기본 매개변수를 사용하면) 왼쪽 그림과 같이 훈련 세트에 아주 크게 과대적합된 예측 모델이 되었음. $\text{min_samples_leaf}=10$ 으로 지정하면 오른쪽 그래프처럼 훨씬 그럴싸한 모델이 만들어짐



결정 트리 회귀 모델의 규제

※ 출처: 핸즈온 머신러닝, 한빛출판사, p249

결정트리의 장단점

- 장점
 - 결정 트리는 이해하고 해석하기 쉬우며, 사용이 편하고, 여러 용도로 사용할 수 있으며, 성능도 뛰어남
- 단점 또는 제한 사항
 - 결정 트리는 계단 모양의 결정 경계를 만듦(모든 분할은 축에 수직).
 - 그래서 훈련 세트의 회전에 민감: 문제를 해결하는 한 가지 방법은 훈련 데이터를 더 좋은 방향으로 회전시키는 PCA 기법을 사용하는 것
 - 훈련 데이터에 있는 작은 변화에도 매우 민감

※ 랜덤포레스트 모델의 등장

